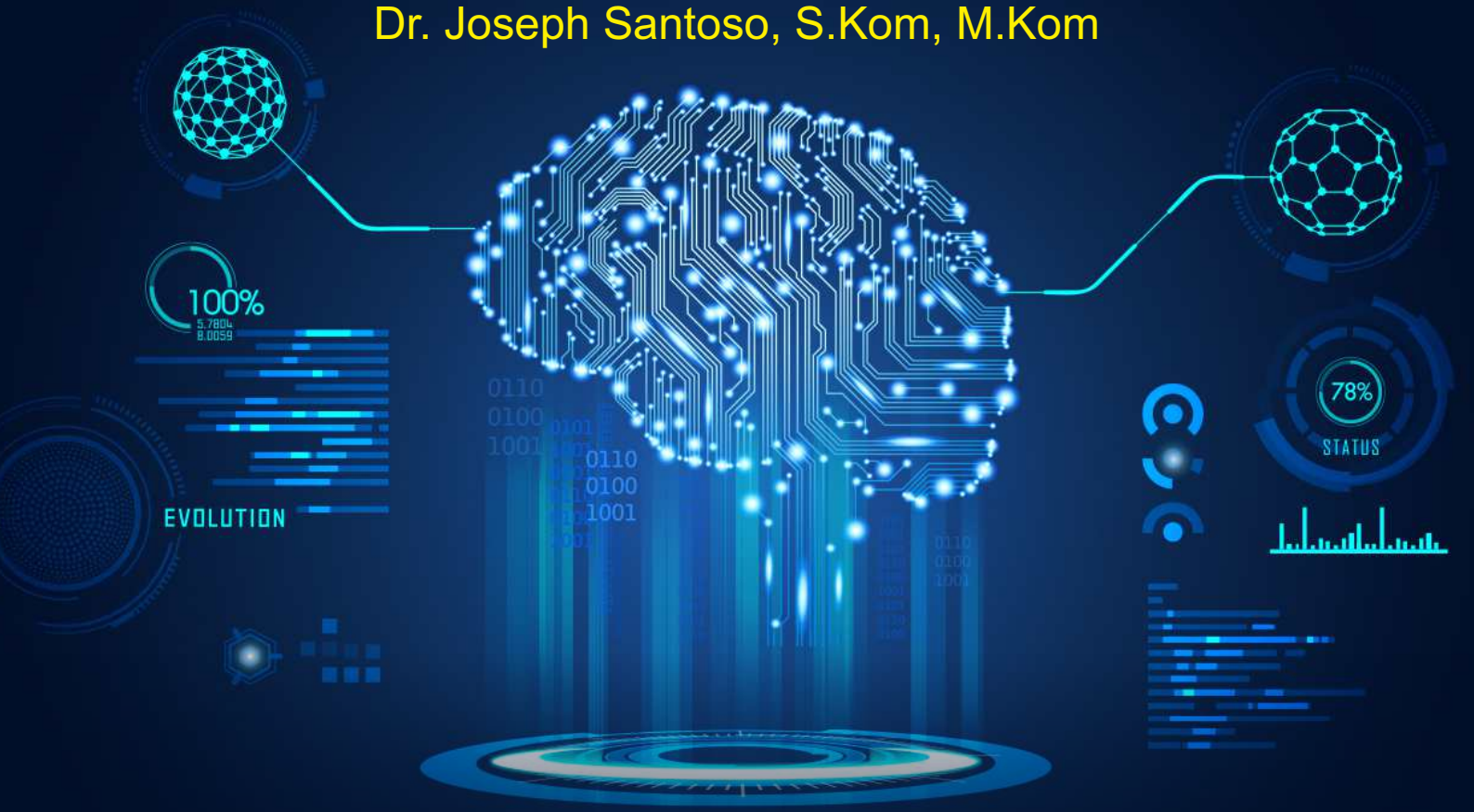
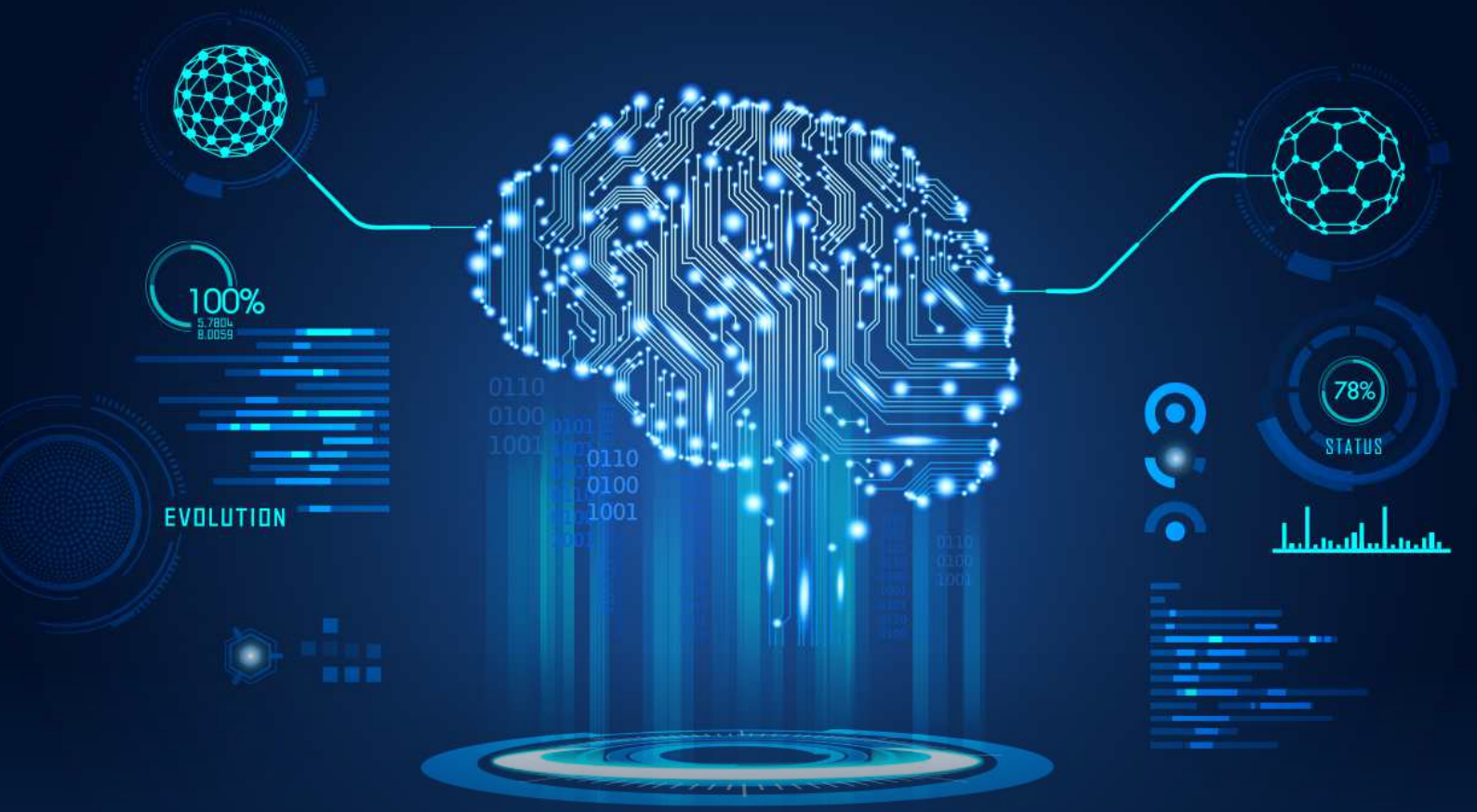
Oleh:
Dr. Joseph Santoso, S.Kom, M.Kom



ILMU DATA

(Data Science)

YAYASAN PRIMA AGUS TEKNIK



ILMU DATA
(Data Science)

Oleh:
Dr. Joseph Santoso, S.Kom, M.Kom

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-50-5 (PDF)
9 786238 120505

ILMU DATA (Data Science)

Penulis :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom

ISBN : 9 786238 120505

Editor :
Muhammad Sholikan, M.Kom

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur penulis panjatkan atas terselesaikannya buku yang berjudul "Ilmu data (Data Science) dapat terselesaikan dengan baik. Ilmu data merujuk pada bidang interdisipliner yang melibatkan pengumpulan, analisis, interpretasi, pengolahan, dan pemahaman data untuk mengambil wawasan dan mendukung pengambilan keputusan. Ilmu data menggabungkan konsep-konsep dari berbagai disiplin ilmu seperti statistik, ilmu komputer, matematika, dan domain pengetahuan spesifik. Langkah-langkah ilmu data yang dijelaskan dalam buku ini mencakup pengumpulan data, pembersihan data, eksplorasi data, permodelan, pengembangan, visualisasi data hingga interpretasi dan pengambilan keputusan. Ilmu data merubah kita beralih ke mesin untuk melakukan sebagian pekerjaan untuk kita: mengenali pola, membuat koneksi, dan memberi kita jawaban atas banyak pertanyaan kita. Mengandalkan komputer untuk melakukan sebagian pekerjaan bagi kita.

Buku ini dibagi dalam 9 bab. Bab 1 dan 2 menawarkan latar belakang dan kerangka teori umum yang diperlukan untuk memahami sisa buku ini, bab 1 adalah pengantar ilmu data dan data besar, diakhiri dengan contoh praktis Hadoop. Bab 2 membahas tentang proses ilmu data, mencakup langkah-langkah yang ada di hampir setiap proyek ilmu data.

Di bab 3 hingga 5, kami menerapkan pembelajaran mesin pada set data yang semakin besar. Bab 3 membuatnya tetap kecil. Data masih mudah masuk ke dalam memori komputer rata-rata. Bab 4 menambah tantangan dengan melihat "data besar." Data ini cocok dengan mesin Anda, tetapi sulit untuk memasukkannya ke dalam RAM, menjadikannya tantangan untuk diproses tanpa cluster komputasi. Bab 5 akhirnya melihat data besar. Untuk ini buku ini tidak dapat bekerja dengan banyak komputer. Bab 6 hingga 9 membahas beberapa materi menarik dalam ilmu data dalam masalah yang kurang lebih independent, bab 6 membahas tentang NoSQL dan perbedaannya dengan database relasional. Bab 7 menerapkan ilmu data untuk mengalirkan data. Di sini masalah utamanya bukanlah ukuran, melainkan kecepatan pembuatan data dan data lama menjadi usang.

Bab 8 membahas tentang penambangan teks. Tidak semua data dimulai sebagai angka. Text mining dan text analytics menjadi penting ketika data dalam format tekstual seperti email, blog, website, dan sebagainya. Bab 9 berfokus pada bagian terakhir dari proses ilmu data serta visualisasi data dan pembuatan aplikasi prototipe. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Agustus 2023
Penulis

Dr. Joseph Santoso, S.Kom, M.Kom

DAFTAR ISI

BAB 1
ILMU DATA DALAM DATA BESAR

Dalam bab ini diharapkan mahasiswa dapat:

- Mendefinisikan ilmu data dan data besar
- Mengenali berbagai jenis data
- Memperoleh wawasan tentang proses ilmu data
- Memperkenalkan bidang ilmu data dan data besar
- Bekerja melalui contoh-contoh Hadoop

Data besar adalah istilah selimut untuk setiap kumpulan kumpulan data yang begitu besar atau kompleks sehingga menjadi sulit untuk memprosesnya menggunakan teknik manajemen data tradisional seperti, misalnya, RDBMS (sistem manajemen basis data relasional). RDBMS yang diadopsi secara luas telah lama dianggap sebagai solusi satu ukuran untuk semua, tetapi tuntutan penanganan data besar menunjukkan sebaliknya. Ilmu data melibatkan penggunaan metode untuk menganalisis sejumlah besar data dan mengekstraksi pengetahuan yang dikandungnya. Anda dapat menganggap hubungan antara data besar dan ilmu data seperti hubungan antara minyak mentah dan kilang minyak. Ilmu data dan data besar berevolusi dari statistik dan manajemen data tradisional tetapi sekarang dianggap sebagai disiplin ilmu yang berbeda.

Ciri-ciri big data sering disebut sebagai tiga Vs:

- Volume—Berapa banyak data yang ada?
- Variasi—Seberapa beragam jenis data yang berbeda?
- Velocity/Kecepatan—Pada kecepatan berapa data baru dihasilkan?

Seringkali karakteristik ini dilengkapi dengan V keempat, Verity/kejujuran: Seberapa akurat datanya? Keempat properti ini membuat big data berbeda dari data yang ditemukan di alat manajemen data tradisional. Konsekuensinya, tantangan yang mereka bawa dapat dirasakan di hampir setiap aspek: pengambilan data, kurasi, penyimpanan, pencarian, berbagi, transfer, dan visualisasi. Selain itu, big data membutuhkan teknik khusus untuk mengekstrak wawasan.

Ilmu data adalah perpanjangan evolusioner dari statistik yang mampu menangani sejumlah besar data yang dihasilkan saat ini. Itu menambahkan metode dari ilmu komputer ke repertoar statistik. Dalam catatan penelitian dari Laney and Kart, Emerging Role of the Data Scientist dan Art of Data Science, penulis menyaring ratusan deskripsi pekerjaan untuk ilmuwan data, ahli statistik, dan analis BI (*Business Intelligence*) untuk mendeteksi perbedaan antara judul-judul tersebut. Hal utama yang membedakan ilmuwan data dari ahli statistik adalah kemampuan untuk bekerja dengan data besar dan pengalaman dalam pembelajaran mesin, komputasi, dan pembuatan algoritme. Alat mereka juga cenderung berbeda, dengan deskripsi pekerjaan ilmuwan data lebih sering menyebutkan kemampuan untuk menggunakan antara lain Hadoop, Pig, Spark, R, Python, dan Java. Jangan khawatir jika Anda merasa terintimidasi oleh daftar ini; sebagian besar akan diperkenalkan secara bertahap dalam buku ini, meskipun kami akan fokus pada Python. Python adalah bahasa yang bagus untuk ilmu data

karena memiliki banyak pustaka ilmu data yang tersedia, dan didukung secara luas oleh perangkat lunak khusus. Misalnya, hampir setiap database NoSQL populer memiliki API khusus Python. Karena fitur-fitur ini dan kemampuan untuk membuat prototipe dengan cepat dengan Python sambil mempertahankan kinerja yang dapat diterima, pengaruhnya terus berkembang di dunia ilmu data.

Karena jumlah data terus bertambah dan kebutuhan untuk memanfaatkannya menjadi semakin penting, setiap data scientist akan menemukan proyek data besar sepanjang karier mereka.

1.1 MANFAAT DAN KEGUNAAN DATA SCIENCE DAN BIG DATA

Ilmu data dan data besar digunakan hampir di semua tempat baik dalam pengaturan komersial maupun non-komersial. Jumlah kasus penggunaan sangat banyak, dan contoh yang akan kami berikan di seluruh buku ini hanya menggores permukaan kemungkinan.

Perusahaan komersial di hampir setiap industri menggunakan ilmu data dan data besar untuk mendapatkan wawasan tentang pelanggan, proses, staf, penyelesaian, dan produk mereka. Banyak perusahaan menggunakan ilmu data untuk menawarkan pengalaman pengguna yang lebih baik kepada pelanggan, serta untuk menjual silang, menjual lebih tinggi, dan mempersonalisasi penawaran mereka. Contoh bagusnya adalah Google AdSense, yang mengumpulkan data dari pengguna internet sehingga pesan komersial yang relevan dapat dicocokkan dengan orang yang menjelajahi internet. MaxPoint (http://maxpoint.com/us) adalah contoh lain dari iklan hasil personalisasi waktu nyata. Profesional sumber daya manusia juga menggunakan analitik orang dan penambangan teks untuk menyaring kandidat, memantau suasana hati karyawan, dan mempelajari jaringan informal di antara rekan kerja. People analytics adalah tema sentral dalam buku Moneyball: The Art of Winning an Unfair Game. Dalam buku (dan film) kami melihat bahwa proses scouting tradisional untuk bisbol Amerika dilakukan secara acak, dan menggantinya dengan sinyal yang berkorelasi mengubah segalanya. Mengandalkan statistik memungkinkan mereka untuk mempekerjakan pemain yang tepat dan mengadu domba mereka dengan lawan di mana mereka akan memiliki keuntungan terbesar. Lembaga keuangan menggunakan ilmu data untuk memprediksi pasar saham, menentukan risiko meminjamkan uang, dan mempelajari cara menarik klien baru untuk layanan mereka. Pada saat penulisan buku ini, setidaknya 50% perdagangan di seluruh dunia dilakukan secara otomatis oleh mesin berdasarkan algoritme yang dikembangkan oleh quants, sebagaimana ilmuwan data yang sering disebut bekerja pada algoritme perdagangan, dengan bantuan data besar dan teknik sains data.

Organisasi pemerintah juga menyadari nilai data. Banyak organisasi pemerintah tidak hanya mengandalkan ilmuwan data internal untuk menemukan informasi yang berharga, tetapi juga membagikan data mereka kepada publik. Anda dapat menggunakan data ini untuk mendapatkan wawasan atau membangun aplikasi berbasis data. Data.gov hanyalah satu contoh; itu adalah rumah dari data terbuka Pemerintah AS. Seorang ilmuwan data di organisasi pemerintah dapat mengerjakan berbagai proyek seperti mendeteksi penipuan dan aktivitas kriminal lainnya atau mengoptimalkan pendanaan proyek. Contoh terkenal diberikan oleh Edward Snowden, yang membocorkan dokumen internal Badan Keamanan Nasional

Amerika dan Markas Besar Komunikasi Pemerintah Inggris yang menunjukkan dengan jelas bagaimana mereka menggunakan ilmu data dan data besar untuk memantau jutaan individu. Organisasi tersebut mengumpulkan 5 miliar catatan data dari aplikasi yang tersebar luas seperti Google Maps, Angry Birds, email, dan pesan teks, di antara banyak sumber data lainnya. Kemudian mereka menerapkan teknik ilmu data untuk menyaring informasi.

Lembaga swadaya masyarakat (LSM) juga sudah tidak asing lagi dalam menggunakan data. Mereka menggunakannya untuk mengumpulkan uang dan membela tujuan mereka. *World Wildlife Fund* (WWF), misalnya, mempekerjakan ilmuwan data untuk meningkatkan efektivitas upaya penggalangan dana mereka. Banyak ilmuwan data mengabdikan sebagian waktunya untuk membantu LSM, karena LSM seringkali kekurangan sumber daya untuk mengumpulkan data dan mempekerjakan ilmuwan data. DataKind adalah salah satu kelompok ilmuwan data yang mengabdikan waktunya untuk kepentingan umat manusia.

Universitas menggunakan ilmu data dalam penelitian mereka tetapi juga untuk meningkatkan pengalaman belajar siswa mereka. Munculnya kursus online terbuka besar-besaran (MOOC) menghasilkan banyak data, yang memungkinkan universitas mempelajari bagaimana jenis pembelajaran ini dapat melengkapi kelas tradisional. MOOC adalah aset yang tak ternilai jika Anda ingin menjadi ilmuwan data dan profesional data besar, jadi lihatlah beberapa yang lebih terkenal: Coursera, Udacity, dan edX. Lanskap data besar dan ilmu data berubah dengan cepat, dan MOOC memungkinkan Anda untuk tetap mendapatkan informasi terbaru dengan mengikuti kursus dari universitas terkemuka. Jika Anda belum mengenal mereka, luangkan waktu untuk melakukannya sekarang; Anda akan mencintai mereka seperti kami.

1.2 ASPEK DATA

Dalam ilmu data dan data besar, Anda akan menemukan berbagai jenis data, dan masing-masing cenderung memerlukan alat dan teknik yang berbeda. Kategori utama data adalah sebagai berikut:

- Terstruktur
- Tidak terstruktur
- Bahasa alami
- Dihasilkan oleh mesin
- Berbasis grafik
- Audio, video, dan gambar
- Streaming

Mari jelajahi semua tipe data yang menarik ini.

1.2.1 Data terstruktur

Data terstruktur adalah data yang bergantung pada model data dan berada di bidang tetap dalam catatan. Dengan demikian, seringkali mudah untuk menyimpan data terstruktur dalam tabel di dalam database atau file Excel (gambar 1.1). SQL, atau Structured Query Language, adalah cara yang lebih disukai untuk mengelola dan meminta data yang berada di database. Anda mungkin juga menemukan data terstruktur yang mungkin menyulitkan Anda untuk menyimpannya dalam database relasional tradisional. Data hierarkis seperti silsilah

keluarga adalah salah satu contohnya. Namun, dunia tidak terdiri dari data terstruktur; itu dipaksakan oleh manusia dan mesin. Lebih sering, data datang tidak terstruktur.

	Indicator ID	Dimension List	Timeframe	Numeric Value	Missing Value Flag	Confidence Int
2	214390830	Total (Age-adjusted)	2008	74.6%		73.8%
3	214390833	Aged 18-44 years	2008	59.4%		58.0%
4	214390831	Aged 18-24 years	2008	37.4%		34.6%
5	214390832	Aged 25-44 years	2008	66.9%		65.5%
6	214390836	Aged 45-64 years	2008	88.6%		87.7%
7	214390834	Aged 45-54 years	2008	86.3%		85.1%
8	214390835	Aged 55-64 years	2008	91.5%		90.4%
9	214390840	Aged 65 years and over	2008	94.6%		93.8%
10	214390837	Aged 65-74 years	2008	93.6%		92.4%
11	214390838	Aged 75-84 years	2008	95.6%		94.4%
12	214390839	Aged 85 years and over	2008	96.0%		94.0%
13	214390841	Male (Age-adjusted)	2008	72.2%		71.1%
14	214390842	Female (Age-adjusted)	2008	76.8%		75.9%
15	214390843	White only (Age-adjusted)	2008	73.8%		72.9%
16	214390844	Black or African American only (Age-adjusted)	2008	77.0%		75.0%
17	214390845	American Indian or Alaska Native only (Age-adjusted)	2008	66.5%		57.1%
18	214390846	Asian only (Age-adjusted)	2008	80.5%		77.7%
19	214390847	Native Hawaiian or Other Pacific Islander only (Age-adjusted)	2008	DSU		
20	214390848	2 or more races (Age-adjusted)	2008	75.6%		69.6%

Gambar 1.1 Tabel Excel adalah contoh data terstruktur.

Gambar 1.2 Email sekaligus merupakan contoh data tidak terstruktur dan data bahasa alami.

1.2.2 Data tidak terstruktur

Data tidak terstruktur adalah data yang tidak mudah disesuaikan dengan model data karena kontennya spesifik konteks atau beragam. Salah satu contoh data tidak terstruktur adalah email biasa Anda (gambar 1.2). Meskipun email berisi elemen terstruktur seperti pengirim, judul, dan teks isi, merupakan tantangan untuk menemukan jumlah orang yang telah menulis keluhan email tentang karyawan tertentu karena ada begitu banyak cara untuk merujuk seseorang, misalnya. Ribuan bahasa dan dialek yang berbeda di luar sana semakin memperumit hal ini. Email yang ditulis manusia, seperti yang ditunjukkan pada gambar 1.2, juga merupakan contoh sempurna dari data bahasa alami.

1.2.3 Bahasa alami

Bahasa alami adalah tipe khusus dari data tidak terstruktur; itu menantang untuk diproses karena membutuhkan pengetahuan tentang teknik dan linguistik ilmu data tertentu. Komunitas pemrosesan bahasa alami telah berhasil dalam pengenalan entitas, pengenalan topik, peringkasan, penyelesaian teks, dan analisis sentimen, tetapi model yang dilatih dalam satu domain tidak digeneralisasikan dengan baik ke domain lain. Bahkan teknik canggih pun tidak mampu menguraikan makna setiap bagian teks. Ini seharusnya tidak mengejutkan: manusia juga berjuang dengan bahasa alami. Sifatnya ambigu. Konsep makna itu sendiri dipertanyakan di sini. Mintalah dua orang mendengarkan percakapan yang sama. Akankah mereka mendapatkan arti yang sama? Arti dari kata yang sama bisa berbeda-beda ketika datang dari seseorang yang kesal atau gembira.

1.2.4 Data yang dihasilkan mesin

Data yang dihasilkan mesin adalah informasi yang dibuat secara otomatis oleh komputer, proses, aplikasi, atau mesin lain tanpa campur tangan manusia.

```
CSIPERF:TXCOMMIT;313236
2014-11-28 11:36:13, Info                  CSI    00000153 Creating NT transaction (seq
69), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000154 Created NT transaction (seq 69)
result 0x00000000, handle @0x4e54
2014-11-28 11:36:13, Info                  CSI    00000155@2014/11/28:10:36:13.471
Beginning NT transaction commit...
2014-11-28 11:36:13, Info                  CSI    00000156@2014/11/28:10:36:13.705 CSI perf
trace:
CSIPERF:TXCOMMIT;273983
2014-11-28 11:36:13, Info                  CSI    00000157 Creating NT transaction (seq
70), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000158 Created NT transaction (seq 70)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:13, Info                  CSI    00000159@2014/11/28:10:36:13.764
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015a@2014/11/28:10:36:14.094 CSI perf
trace:
CSIPERF:TXCOMMIT;386259
2014-11-28 11:36:14, Info                  CSI    0000015b Creating NT transaction (seq
71), objectname [6]"(null)"
2014-11-28 11:36:14, Info                  CSI    0000015c Created NT transaction (seq 71)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:14, Info                  CSI    0000015d@2014/11/28:10:36:14.106
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015e@2014/11/28:10:36:14.428 CSI perf
trace:
CSIPERF:TXCOMMIT;375581
```

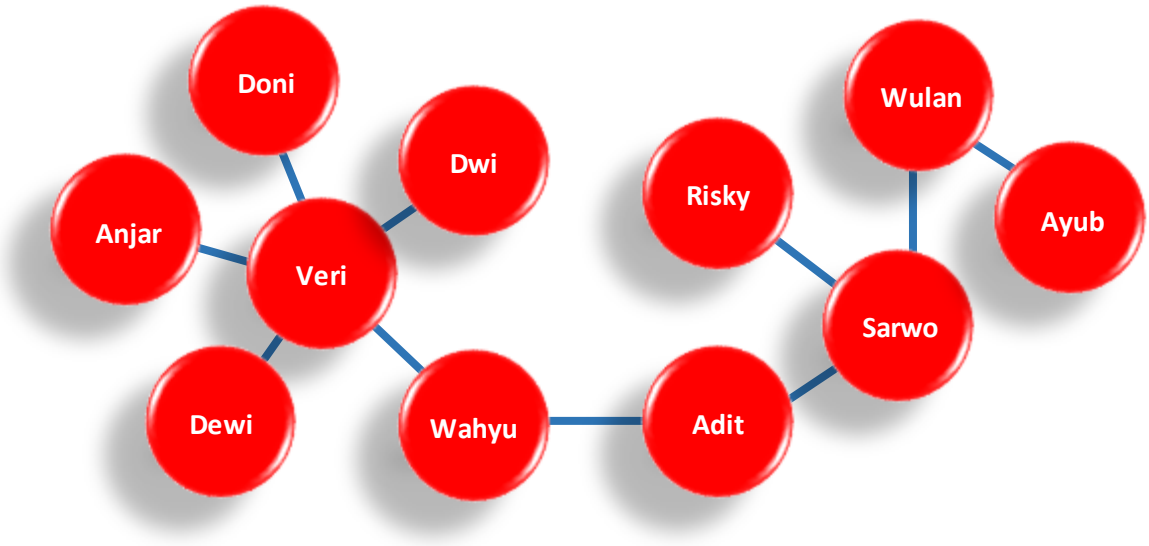
Gambar 1.3 Contoh data yang dihasilkan mesin

Data yang dihasilkan mesin menjadi sumber data utama dan akan terus demikian. Wikibon memperkirakan bahwa nilai pasar industri Internet (istilah yang diciptakan oleh Frost & Sullivan untuk merujuk pada integrasi mesin fisik kompleks dengan sensor jaringan dan perangkat lunak) akan menjadi sekitar Rp. 8 Triliun pada tahun 2020. IDC (International Data Corporation) memperkirakan akan ada 26 kali lebih banyak hal yang terhubung daripada orang pada tahun 2020. Jaringan ini biasa disebut sebagai internet of things.

Analisis data mesin bergantung pada alat yang sangat dapat diskalakan, karena volume dan kecepatannya yang tinggi. Contoh data mesin adalah log server web, catatan detail panggilan, log peristiwa jaringan, dan telemetri (gambar 1.3). Data mesin yang ditunjukkan pada gambar 1.3 akan cocok dengan database terstruktur tabel klasik. Ini bukan pendekatan terbaik untuk data yang sangat saling terhubung atau "berjaringan", di mana hubungan antar entitas memiliki peran yang berharga.

1.2.5 Data berbasis grafik atau jaringan

"Data grafik" bisa menjadi istilah yang membingungkan karena data apa pun dapat ditampilkan dalam grafik. "Grafik" dalam hal ini menunjuk pada teori grafik matematika. Dalam teori graf, graf adalah struktur matematis untuk memodelkan hubungan berpasangan antar objek. Singkatnya, data grafik atau jaringan adalah data yang berfokus pada hubungan atau kedekatan objek. Struktur grafik menggunakan node, tepi, dan properti untuk mewakili dan menyimpan data grafis. Data berbasis grafik adalah cara alami untuk merepresentasikan jejaring sosial, dan strukturnya memungkinkan Anda menghitung metrik tertentu seperti pengaruh seseorang dan jalur terpendek antara dua orang.



Gambar 1.4 Teman di jejaring sosial adalah contoh data berbasis grafik.

Contoh data berbasis grafik dapat ditemukan di banyak situs media sosial (gambar 1.4). Misalnya, di LinkedIn Anda dapat melihat siapa yang Anda kenal di perusahaan mana. Daftar pengikut Anda di Twitter adalah contoh lain dari data berbasis grafik. Kekuatan dan kecanggihan berasal dari beberapa grafik yang tumpang tindih dari node yang sama. Misalnya,

bayangkan tepi penghubung di sini untuk menunjukkan “teman” di Facebook. Bayangkan grafik lain dengan orang yang sama yang menghubungkan rekan bisnis melalui LinkedIn. Bayangkan grafik ketiga berdasarkan minat film di Netflix. Tumpang tindih tiga grafik yang tampak berbeda memungkinkan pertanyaan yang lebih menarik.

Database grafik digunakan untuk menyimpan data berbasis grafik dan ditanyakan dengan bahasa permintaan khusus seperti SPARQL. Data grafik menimbulkan tantangannya, tetapi untuk komputer yang menginterpretasikan data aditif dan gambar, itu bisa menjadi lebih sulit.

1.2.6 Audio, gambar, dan video

Audio, gambar, dan video adalah tipe data yang menimbulkan tantangan khusus bagi ilmuwan data. Tugas-tugas yang dianggap remeh oleh manusia, seperti mengenali objek dalam gambar, ternyata menjadi tantangan tersendiri bagi komputer. MLBAM (Major League Baseball Advanced Media) mengumumkan pada tahun 2014 bahwa mereka akan meningkatkan perekaman video hingga sekitar 7 TB per game untuk tujuan analitik langsung dalam game. Kamera berkecepatan tinggi di stadion akan menangkap gerakan bola dan atlet untuk dihitung secara real time, misalnya jalur yang diambil oleh bek relatif terhadap dua garis dasar.

Baru-baru ini sebuah perusahaan bernama DeepMind berhasil membuat algoritme yang mampu mempelajari cara bermain video game. Algoritme ini menggunakan layar video sebagai input dan belajar menginterpretasikan semuanya melalui proses pembelajaran mendalam yang kompleks. Ini adalah prestasi luar biasa yang mendorong Google untuk membeli perusahaan untuk rencana pengembangan Kecerdasan Buatan (AI) mereka sendiri. Algoritme pembelajaran mengambil data seperti yang dihasilkan oleh permainan komputer; itu streaming data.

1.2.7 Streaming data

Sementara data streaming dapat mengambil hampir semua bentuk sebelumnya, ia memiliki properti tambahan. Data mengalir ke dalam sistem ketika suatu peristiwa terjadi alih-alih dimuat ke dalam penyimpanan data secara berkelompok. Meskipun ini bukan jenis data yang berbeda, kami memperlakukannya di sini karena Anda perlu menyesuaikan proses Anda untuk menangani jenis informasi ini. Contohnya adalah "Apa yang sedang tren" di Twitter, acara olahraga atau musik langsung, dan pasar saham.

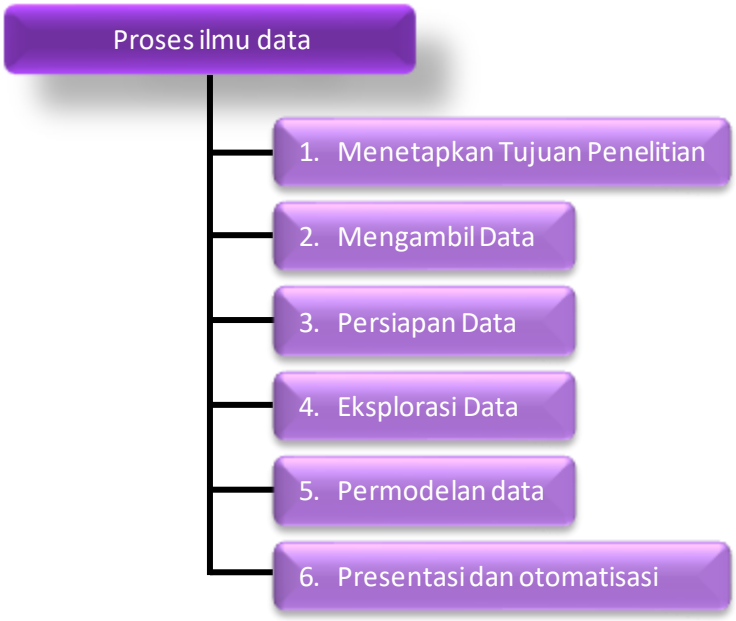
1.3 PROSES ILMU DATA

Proses ilmu data biasanya terdiri dari enam langkah, seperti yang dapat Anda lihat di peta pikiran pada gambar 1.5. Kami akan memperkenalkannya secara singkat di sini dan menanganinya secara lebih rinci di bab 2.

1.3.1 Menetapkan tujuan penelitian

Ilmu data sebagian besar diterapkan dalam konteks organisasi. Saat bisnis meminta Anda untuk melakukan proyek ilmu data, pertama-tama Anda akan menyiapkan piagam proyek. Piagam ini berisi informasi seperti apa yang akan Anda teliti, bagaimana perusahaan mendapat manfaat dari itu, data dan sumber daya apa yang Anda butuhkan, jadwal, dan kiriman. Di sepanjang buku ini, proses ilmu data akan diterapkan pada studi kasus yang lebih

besar dan Anda akan mendapatkan gambaran tentang kemungkinan tujuan penelitian yang berbeda.



Gambar 1.5 Proses ilmu data

1.3.2 Mengambil data

Langkah kedua adalah mengumpulkan data. Anda telah menyatakan dalam piagam proyek data mana yang Anda butuhkan dan di mana Anda dapat menemukannya. Pada langkah ini Anda memastikan bahwa Anda dapat menggunakan data dalam program Anda, yang berarti memeriksa keberadaan, kualitas, dan akses ke data. Data juga dapat dikirimkan oleh perusahaan pihak ketiga dan mengambil banyak bentuk mulai dari spreadsheet Excel hingga berbagai jenis database.

1.3.3 Persiapan data

Pengumpulan data adalah proses yang rawan kesalahan; pada fase ini Anda meningkatkan kualitas data dan menyiapkannya untuk digunakan pada langkah selanjutnya. Fase ini terdiri dari tiga subfase: pembersihan data menghilangkan nilai-nilai palsu dari sumber data dan ketidakkonsistenan di seluruh sumber data, integrasi data memperkaya sumber data dengan menggabungkan informasi dari beberapa sumber data, dan transformasi data memastikan bahwa data berada dalam kondisi yang sesuai. format untuk digunakan dalam model Anda.

1.3.4 Eksplorasi data

Eksplorasi data berkaitan dengan membangun pemahaman yang lebih dalam tentang data Anda. Anda mencoba memahami bagaimana variabel berinteraksi satu sama lain, distribusi data, dan apakah ada outlier. Untuk mencapai ini, Anda terutama menggunakan statistik deskriptif, teknik visual, dan pemodelan sederhana. Langkah ini sering menggunakan singkatan EDA, untuk Exploratory Data Analysis.

1.3.5 Pemodelan data atau pembuatan model

Pada fase ini Anda menggunakan model, pengetahuan domain, dan wawasan tentang data yang Anda temukan di langkah sebelumnya untuk menjawab pertanyaan penelitian. Anda memilih teknik dari bidang statistik, pembelajaran mesin, riset operasi, dan sebagainya. Membangun model adalah proses iteratif yang melibatkan pemilihan variabel untuk model, pelaksanaan model, dan diagnostik model.

1.3.6 Presentasi dan otomatisasi

Terakhir, Anda mempresentasikan hasilnya ke bisnis Anda. Hasil ini bisa bermacam-macam bentuknya, mulai dari presentasi hingga laporan penelitian. Terkadang Anda perlu mengotomatiskan pelaksanaan proses karena bisnis ingin menggunakan wawasan yang Anda peroleh di proyek lain atau mengaktifkan proses operasional untuk menggunakan hasil dari model Anda.

Proses Iterative

Uraian sebelumnya tentang proses ilmu data memberi Anda kesan bahwa Anda menjalani proses ini secara linier, tetapi pada kenyataannya Anda sering kali harus mundur dan mengerjakan ulang temuan tertentu. Misalnya, Anda mungkin menemukan outlier dalam fase eksplorasi data yang mengarah ke kesalahan impor data. Sebagai bagian dari proses ilmu data, Anda mendapatkan wawasan tambahan, yang dapat menimbulkan pertanyaan baru. Untuk mencegah pengerjaan ulang, pastikan Anda membatasi pertanyaan bisnis dengan jelas dan menyeluruh di awal. Sekarang kita memiliki pemahaman yang lebih baik tentang prosesnya, mari kita lihat teknologinya.

1.4 EKOSISTEM BIG DATA DAN ILMU DATA

Saat ini ada banyak alat dan kerangka data besar, dan mudah hilang karena teknologi baru muncul dengan cepat. Jauh lebih mudah setelah Anda menyadari bahwa ekosistem data besar dapat dikelompokkan ke dalam teknologi yang memiliki tujuan dan fungsi serupa, yang akan kita bahas di bagian ini. Ilmuwan data menggunakan banyak teknologi berbeda, tetapi tidak semuanya; kami akan mendedikasikan bab terpisah untuk kelas teknologi sains data yang paling penting. Peta pikiran pada gambar 1.6 menunjukkan komponen ekosistem data besar dan di mana berbagai teknologi berada. Mari kita lihat kelompok alat yang berbeda dalam diagram ini dan lihat fungsinya masing-masing. Kami akan mulai dengan sistem file terdistribusi.

1.4.1 Sistem file terdistribusi

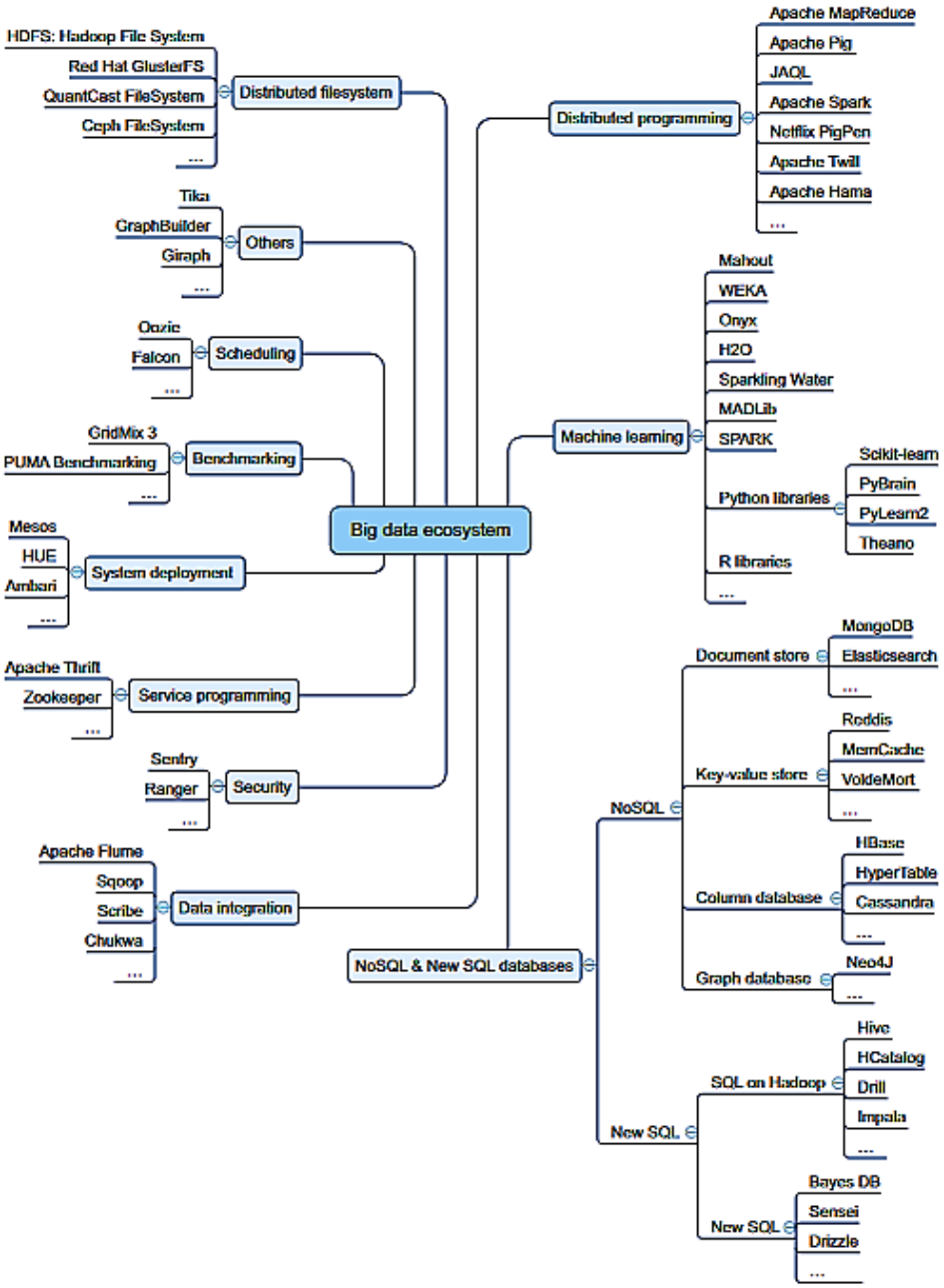
Sistem file terdistribusi mirip dengan sistem file normal, kecuali sistem ini berjalan di banyak server sekaligus. Karena ini adalah sistem file, Anda dapat melakukan hampir semua hal yang sama seperti yang Anda lakukan pada sistem file biasa. Tindakan seperti menyimpan, membaca, dan menghapus file serta menambahkan keamanan ke file merupakan inti dari setiap sistem file, termasuk yang terdistribusi. Sistem file terdistribusi memiliki keuntungan yang signifikan:

- Mereka dapat menyimpan file yang lebih besar dari satu disk komputer mana pun.
- File secara otomatis direplikasi di beberapa server untuk redundansi atau operasi paralel sambil menyembunyikan kerumitan melakukannya dari pengguna.

~ Sistem dapat diskalakan dengan mudah: Anda tidak lagi terikat oleh batasan memori atau penyimpanan dari satu server.

Di masa lalu, skala ditingkatkan dengan memindahkan semuanya ke server dengan lebih banyak memori, penyimpanan, dan CPU yang lebih baik (penskalaan vertikal). Saat ini Anda dapat menambahkan server kecil lainnya (penskalaan horizontal). Prinsip ini membuat potensi penskalaan hampir tidak terbatas.

Sistem file terdistribusi yang paling terkenal saat ini adalah Hadoop File System (HDFS). Ini adalah implementasi sumber terbuka dari Sistem File Google. Dalam buku ini kami fokus pada Sistem File Hadoop karena ini yang paling umum digunakan. Namun, ada banyak sistem file terdistribusi lainnya: Sistem File Red Hat Cluster, Sistem File Ceph, dan Sistem File Tachyon, untuk menyebutkan tiga.



Gambar 1.6 Teknologi big data dapat diklasifikasikan menjadi beberapa komponen utama.

1.4.2 Kerangka pemrograman terdistribusi

Setelah Anda menyimpan data di sistem file terdistribusi, Anda ingin mengeksploitasinya. Salah satu aspek penting dalam bekerja pada hard disk terdistribusi adalah Anda tidak akan memindahkan data ke program, melainkan memindahkan program ke data. Saat Anda memulai dari awal dengan bahasa pemrograman tujuan umum yang normal seperti C, Python, atau Java, Anda harus menghadapi kerumitan yang menyertai pemrograman terdistribusi, seperti memulai ulang pekerjaan yang gagal, melacak hasil dari berbagai subproses, dan seterusnya. Untungnya, komunitas sumber terbuka telah mengembangkan banyak kerangka kerja untuk menangani hal ini untuk Anda, dan ini memberi Anda pengalaman yang jauh lebih baik bekerja dengan data terdistribusi dan menangani banyak tantangan yang dibawanya.

1.4.3 Kerangka integrasi data

Setelah Anda memiliki sistem file terdistribusi, Anda perlu menambahkan data. Anda perlu memindahkan data dari satu sumber ke sumber lain, dan di sinilah kerangka kerja integrasi data seperti Apache Sqoop dan Apache Flume unggul. Prosesnya mirip dengan proses ekstrak, transformasi, dan pemuatan di gudang data tradisional.

1.4.4 Kerangka pembelajaran mesin

Saat Anda memiliki data, saatnya untuk mengekstrak wawasan yang didambakan. Di sinilah Anda mengandalkan bidang pembelajaran mesin, statistik, dan matematika terapan. Sebelum Perang Dunia II, semuanya harus dihitung dengan tangan, yang sangat membatasi kemungkinan analisis data. Setelah Perang Dunia II komputer dan komputasi ilmiah dikembangkan. Satu komputer dapat melakukan semua penghitungan dan penghitungan dan dunia peluang terbuka. Sejak terobosan ini, orang hanya perlu menurunkan rumus matematika, menuliskannya dalam algoritme, dan memuat datanya.

Dengan banyaknya data yang tersedia saat ini, satu komputer tidak dapat lagi menangani beban kerja sendirian. Nyatanya, beberapa algoritme yang dikembangkan pada milenium sebelumnya tidak akan pernah berhenti sebelum akhir alam semesta, bahkan jika Anda dapat menggunakan setiap komputer yang tersedia di Bumi. Ini ada hubungannya dengan kompleksitas waktu.

Contohnya adalah mencoba memecahkan kata sandi dengan menguji setiap kemungkinan kombinasi. Salah satu masalah terbesar dengan algoritme lama adalah algoritme tersebut tidak dapat diskalakan dengan baik. Dengan jumlah data yang perlu kita analisis hari ini, hal ini menjadi masalah, dan kerangka kerja serta pustaka khusus diperlukan untuk menangani jumlah data ini. Pustaka pembelajaran mesin yang paling populer untuk Python adalah Scikit-learn. Ini adalah kotak alat pembelajaran mesin yang hebat, dan kami akan menggunakannya nanti di buku ini. Ada, tentu saja, perpustakaan Python lainnya:

- **PyBrain untuk jaringan saraf**—Jaringan saraf adalah algoritme pembelajaran yang meniru otak manusia dalam mempelajari mekanika dan kompleksitas. Neural network sering dianggap sebagai advanced dan black box.
- **NLTK atau Natural Language Toolkit**—Seperti namanya, fokusnya bekerja dengan bahasa alami. Ini adalah perpustakaan ekstensif yang dibundel dengan sejumlah korpus teks untuk membantu Anda membuat model data Anda sendiri.

- **Pylearn2—**Toolbox pembelajaran mesin lain tetapi sedikit kurang matang dibandingkan Scikit-learn.
- **TensorFlow—**Library Python untuk pembelajaran mendalam yang disediakan oleh Google.

Lanskap tidak berakhir dengan perpustakaan Python, tentu saja. Spark adalah mesin pembelajaran mesin berlisensi Apache baru, yang berspesialisasi dalam pembelajaran mesin waktu-nyata. Layak untuk dilihat dan Anda dapat membaca lebih lanjut tentangnya di http://spark.apache.org/.

1.4.5 Database NoSQL

Jika Anda perlu menyimpan data dalam jumlah besar, Anda memerlukan perangkat lunak yang khusus mengelola dan membuat kueri data ini. Secara tradisional ini telah menjadi arena permainan database relasional seperti Oracle SQL, MySQL, Sybase IQ, dan lain-lain. Meskipun mereka masih menjadi teknologi masuk untuk banyak kasus penggunaan, jenis database baru telah muncul di bawah pengelompokan database NoSQL.

Nama grup ini bisa menyesatkan, karena "Tidak" dalam konteks ini berarti "Tidak Hanya". Kurangnya fungsionalitas dalam SQL bukanlah alasan terbesar untuk pergeseran paradigma, dan banyak database NoSQL telah mengimplementasikan versi SQL itu sendiri. Tetapi basis data tradisional memiliki kekurangan yang tidak memungkinkannya untuk berkembang dengan baik. Dengan memecahkan beberapa masalah database tradisional, database NoSQL memungkinkan pertumbuhan data yang hampir tak ada habisnya. Kekurangan ini terkait dengan setiap properti data besar: penyimpanan atau kekuatan pemrosesannya tidak dapat diskalakan melebihi satu node dan mereka tidak memiliki cara untuk menangani streaming, grafik, atau bentuk data yang tidak terstruktur.

Berbagai jenis database telah muncul, tetapi mereka dapat dikategorikan ke dalam jenis berikut:

- **Database kolom—**Data disimpan dalam kolom, yang memungkinkan algoritme untuk melakukan kueri lebih cepat. Teknologi yang lebih baru menggunakan penyimpanan berdasarkan sel. Struktur seperti meja masih penting.
- **Penyimpanan dokumen**—Penyimpanan dokumen tidak lagi menggunakan tabel, tetapi menyimpan setiap pengamatan dalam dokumen. Ini memungkinkan skema data yang jauh lebih fleksibel.
- **Streaming data—**Data dikumpulkan, diubah, dan diagregasi tidak dalam batch tetapi secara real time. Meskipun kami telah mengategorikannya di sini sebagai database untuk membantu Anda dalam pemilihan alat, ini lebih merupakan jenis masalah khusus yang mendorong terciptanya teknologi seperti Storm.
- **Penyimpanan nilai kunci—**Data tidak disimpan dalam tabel; alih-alih Anda menetapkan kunci untuk setiap nilai, seperti org.marketing.sales.2015: 20000. Ini menskalakan dengan baik tetapi menempatkan hampir semua implementasi pada pengembang.
- **SQL di Hadoop—**Kueri batch di Hadoop menggunakan bahasa mirip SQL yang menggunakan kerangka kerja pengurangan peta di latar belakang.
- **SQL Baru**—Kelas ini menggabungkan skalabilitas database NoSQL dengan keunggulan database relasional. Mereka semua memiliki antarmuka SQL dan model data relasional.

- **Database grafik—**Tidak semua masalah sebaiknya disimpan dalam tabel. Masalah tertentu lebih alami diterjemahkan ke dalam teori grafik dan disimpan dalam database grafik. Contoh klasik dari ini adalah jejaring sosial.

1.4.6 Alat penjadwalan

Alat penjadwalan membantu Anda mengotomatiskan tugas berulang dan memicu pekerjaan berdasarkan kejadian seperti menambahkan file baru ke folder. Ini mirip dengan alat seperti CRON di Linux tetapi secara khusus dikembangkan untuk data besar. Anda dapat menggunakannya, misalnya, untuk memulai tugas MapReduce setiap kali kumpulan data baru tersedia di direktori.

1.4.7 Alat pembandingan

Kelas alat ini dikembangkan untuk mengoptimalkan penginstalan data besar Anda dengan menyediakan rangkaian profil standar. Suite pembuatan profil diambil dari sekumpulan pekerjaan data besar yang representatif. Tolok ukur dan pengoptimalan infrastruktur dan konfigurasi data besar seringkali bukan pekerjaan untuk ilmuwan data itu sendiri, tetapi untuk profesional yang berspesialisasi dalam menyiapkan infrastruktur TI; jadi mereka tidak tercakup dalam buku ini. Menggunakan infrastruktur yang dioptimalkan dapat membuat perbedaan biaya yang besar. Misalnya, jika Anda bisa mendapatkan 10% pada sekelompok 100 server, Anda menghemat biaya 10 server.

1.4.8 Penyebaran sistem

Menyiapkan infrastruktur big data bukanlah tugas yang mudah dan membantu para insinyur dalam menerapkan aplikasi baru ke dalam klaster big data adalah tempat alat penerapan sistem bersinar. Mereka sebagian besar mengotomatiskan instalasi dan konfigurasi komponen data besar. Ini bukan tugas inti seorang ilmuwan data.

1.4.9 Pemrograman layanan

Misalkan Anda telah membuat aplikasi prediksi bola kelas dunia di Hadoop, dan Anda ingin mengizinkan orang lain untuk menggunakan prediksi yang dibuat oleh aplikasi Anda. Namun, Anda tidak tahu arsitektur atau teknologi semua orang yang tertarik menggunakan prediksi Anda. Alat layanan unggul di sini dengan memaparkan aplikasi data besar ke aplikasi lain sebagai layanan. Ilmuwan data terkadang perlu mengekspos model mereka melalui layanan. Contoh paling terkenal adalah layanan REST; REST adalah singkatan dari transfer negara representasional. Ini sering digunakan untuk memberi makan situs web dengan data.

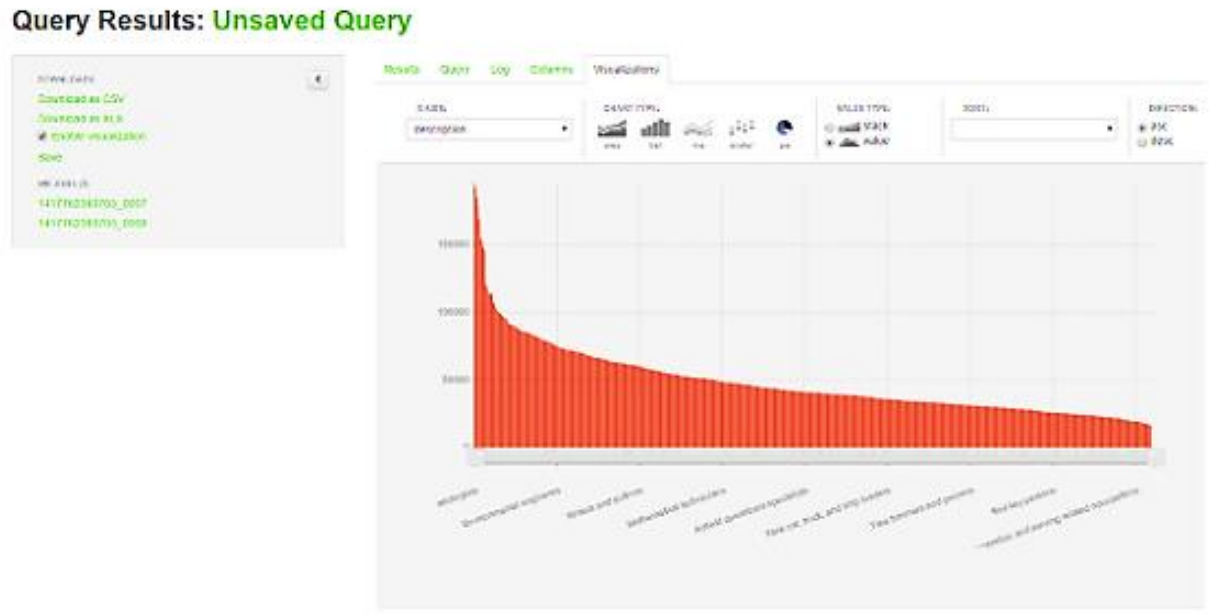
1.4.10 Keamanan

Apakah Anda ingin semua orang memiliki akses ke semua data Anda? Anda mungkin perlu memiliki kontrol yang baik atas akses ke data tetapi tidak ingin mengelolanya berdasarkan aplikasi per aplikasi. Alat keamanan data besar memungkinkan Anda memiliki kontrol terpusat dan menyeluruh atas akses ke data. Keamanan data besar telah menjadi topik tersendiri, dan ilmuwan data biasanya hanya dihadapkan dengannya sebagai konsumen data; jarang mereka menerapkan keamanan itu sendiri. Dalam buku ini kami tidak menjelaskan cara mengatur keamanan pada data besar karena ini adalah pekerjaan ahli keamanan.

1.5 CONTOH KERJA PENGANTAR HADOOP

Kami akan mengakhiri bab ini dengan aplikasi kecil dalam konteks data besar. Untuk ini kami akan menggunakan gambar Hortonworks Sandbox. Ini adalah mesin virtual yang dibuat oleh Hortonworks untuk mencoba beberapa aplikasi big data di mesin lokal. Nanti di buku ini Anda akan melihat bagaimana Juju memudahkan pemasangan Hadoop di banyak mesin.

Kami akan menggunakan kumpulan data kecil dari data gaji pekerjaan untuk menjalankan sampel pertama kami, tetapi membuat kueri kumpulan data besar yang terdiri dari miliaran baris akan sama mudahnya. Bahasa kueri akan tampak seperti SQL, tetapi di belakang layar pekerjaan MapReduce akan berjalan dan menghasilkan tabel hasil langsung, yang kemudian dapat diubah menjadi grafik batang. Hasil akhir dari latihan ini terlihat seperti gambar 1.7.



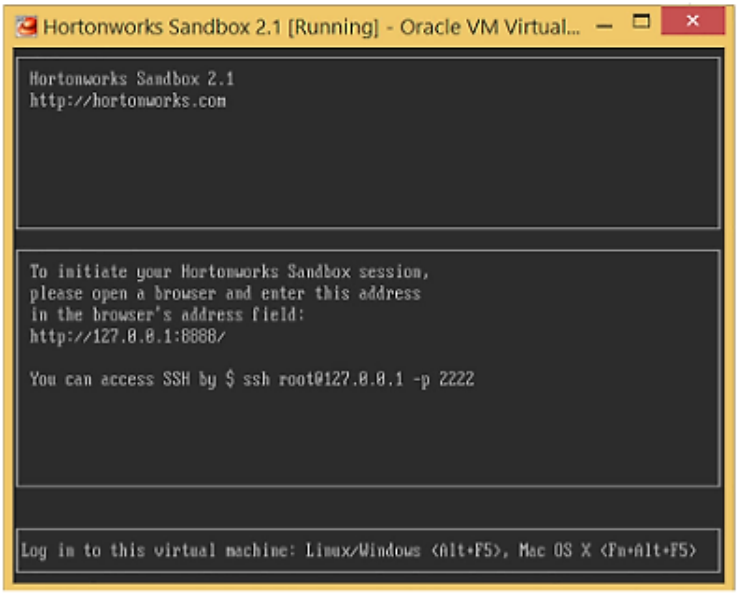
Gambar 1.7 Hasil akhir: gaji rata-rata berdasarkan deskripsi pekerjaan

Untuk bangun dan berjalan secepat mungkin, kami menggunakan Hortonworks Sandbox di dalam Virtual-Box. VirtualBox adalah alat virtualisasi yang memungkinkan Anda menjalankan sistem operasi lain di dalam sistem operasi Anda sendiri. Dalam hal ini Anda dapat menjalankan CentOS dengan instalasi Hadoop yang ada di dalam sistem operasi yang Anda instal.

Beberapa langkah diperlukan untuk mengaktifkan dan menjalankan kotak pasir di VirtualBox. Perhatian, langkah-langkah berikut berlaku pada saat bab ini ditulis (Februari 2015):

1. Unduh gambar virtual dari http://hortonworks.com/products/hortonworks-sandbox/#install.
2. Jalankan host mesin virtual Anda. VirtualBox dapat diunduh dari https://www.virtualbox.org/wiki/Downloads.
3. Tekan CTRL+I dan pilih gambar virtual dari Hortonworks.
4. Klik Berikutnya.

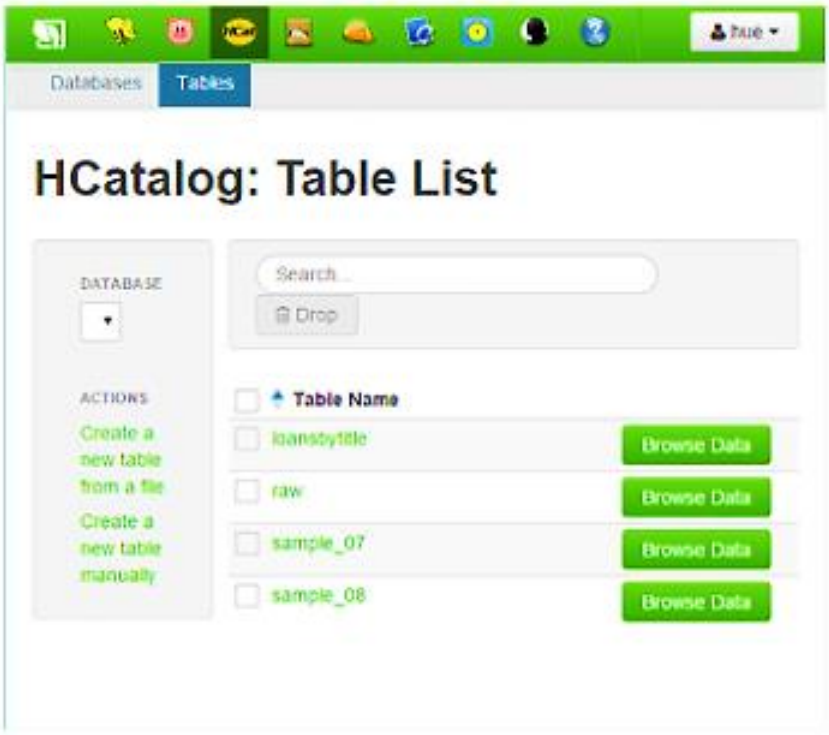
5. Klik Impor; setelah beberapa saat gambar Anda akan diimpor.
6. Sekarang pilih mesin virtual Anda dan klik Jalankan.
7. Berikan sedikit waktu untuk memulai distribusi CentOS dengan instalasi Hadoop berjalan, seperti yang ditunjukkan pada gambar 1.8. Perhatikan versi Sandbox di sini adalah 2.1. Dengan versi lain, segalanya bisa sedikit berbeda.



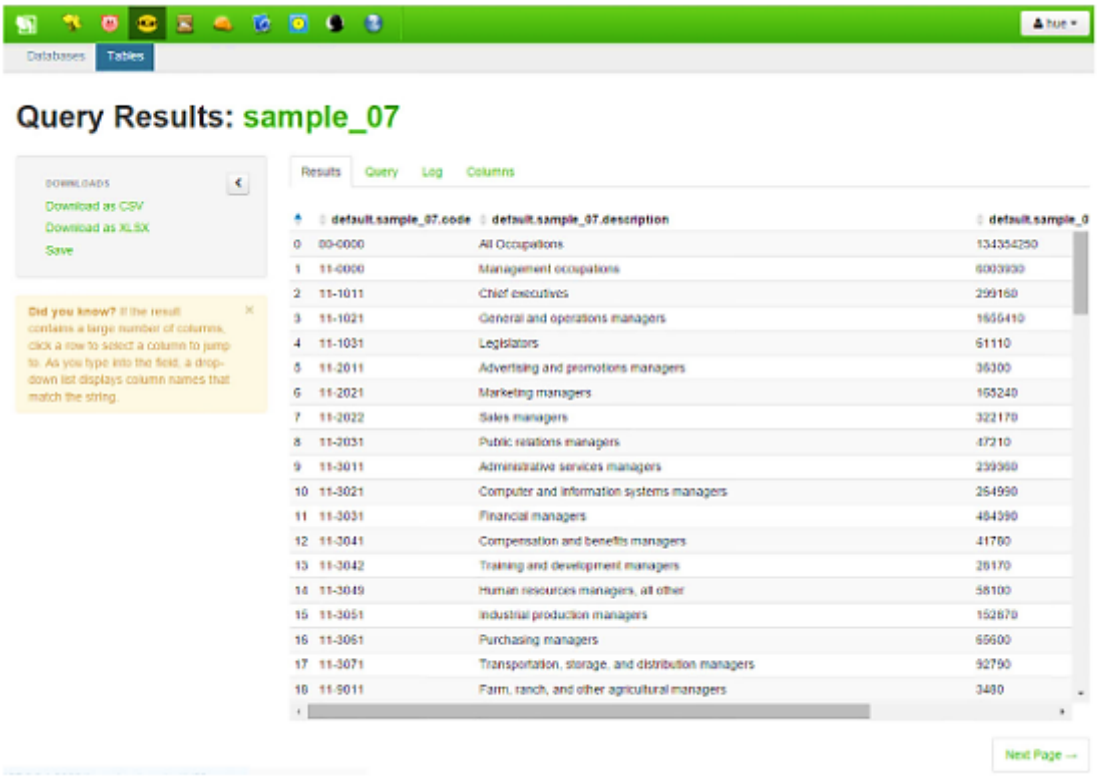
Gambar 1.8 Hortonworks Sandbox berjalan di dalam VirtualBox

Anda dapat langsung masuk ke mesin atau menggunakan SSH untuk masuk. Untuk aplikasi ini Anda akan menggunakan antarmuka web. Arahkan browser Anda ke alamat http://127.0.0.1:8000 dan Anda akan disambut dengan layar yang ditunjukkan pada gambar 1.9. Hortonworks telah mengunggah dua set sampel, yang dapat Anda lihat di HCatalog. Cukup klik tombol HCat di layar dan Anda akan melihat tabel yang tersedia untuk Anda (gambar 1.10).

Gambar 1.9 Layar selamat datang Hortonworks Sandbox



Gambar 1.10 Daftar tabel yang tersedia di Hcatalog



Gambar 1.11 Isi tabel

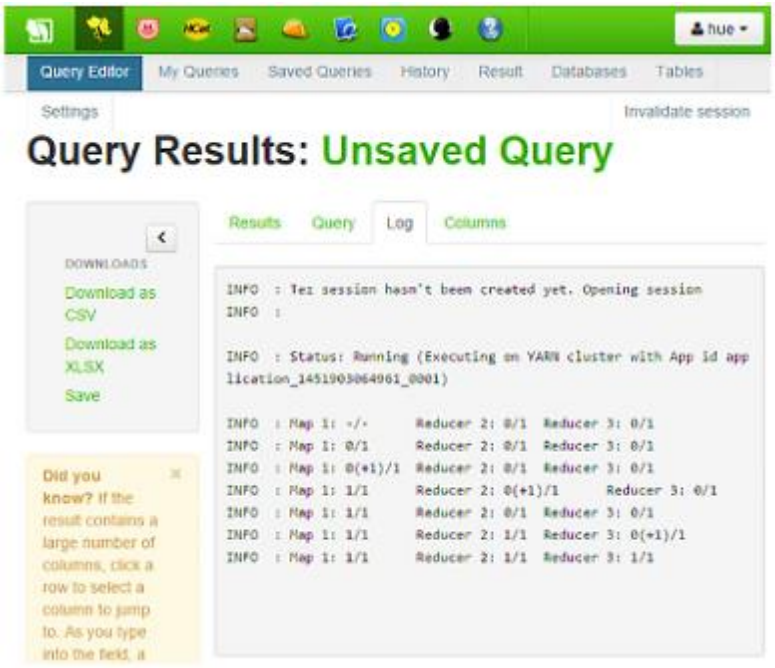
Gambar 1.12 Anda dapat menjalankan perintah HiveQL di editor Beeswax HiveQL. Di belakang layar itu diterjemahkan ke dalam pekerjaan MapReduce.

Untuk melihat isi data, klik tombol Browse Data di sebelah entri sample_07 untuk mendapatkan layar berikutnya (gambar 1.11). Ini terlihat seperti tabel biasa, dan Hive adalah alat yang memungkinkan Anda mendekatinya seperti database biasa dengan SQL. Itu benar: di Hive Anda mendapatkan hasil menggunakan HiveQL, dialek SQL lama. Untuk membuka editor Beeswax HiveQL, klik tombol Beeswax pada menu (gambar 1.12).

Untuk mendapatkan hasil Anda, jalankan kueri berikut:

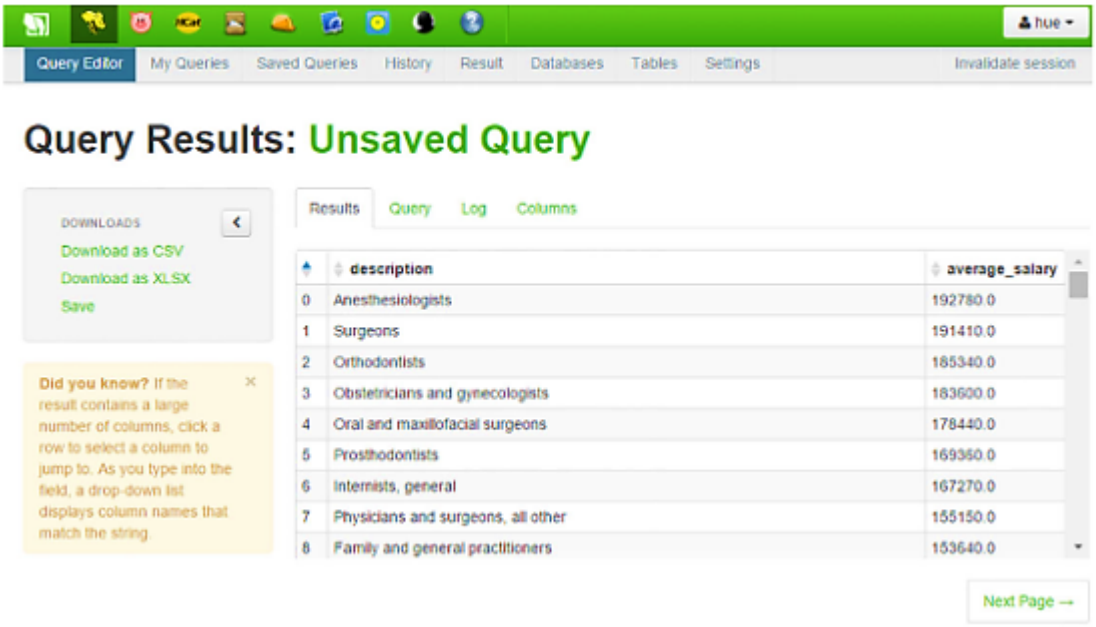
```
Select description, avg(salary) as average_salary from sample_07 group by
description order by average_salary desc.
```

Klik tombol Jalankan. Hive menerjemahkan HiveQL Anda menjadi pekerjaan MapReduce dan menjalankannya di lingkungan Hadoop Anda, seperti yang Anda lihat pada gambar 1.13. Namun sebaiknya hindari membaca jendela log untuk saat ini. Pada titik ini, itu menyesatkan. Jika ini kueri pertama Anda, mungkin diperlukan waktu 30 detik. Hadoop terkenal dengan periode pemanasannya.



Gambar 1.13 Pencatatan menunjukkan bahwa HiveQL Anda diterjemahkan ke dalam pekerjaan MapReduce. Catatan: Log ini berasal dari HDP versi Februari 2015, jadi versi saat ini mungkin terlihat sedikit berbeda.

Setelah beberapa saat hasilnya muncul. Kerja bagus! Kesimpulan dari hal ini, seperti yang ditunjukkan pada gambar 1.14, adalah pergi ke sekolah kedokteran adalah investasi yang bagus. Dengan tabel ini kami menyimpulkan tutorial pengantar Hadoop kami. Meskipun bab ini hanyalah permulaan, terkadang terasa sedikit berlebihan. Disarankan untuk membiarkannya sekarang dan kembali ke sini lagi ketika semua konsep telah dijelaskan secara menyeluruh. Ilmu data adalah bidang yang luas sehingga dilengkapi dengan kosakata yang luas. Kami berharap dapat memberi Anda gambaran sekilas tentang sebagian besar waktu kita bersama. Setelah itu, Anda memilih dan mengasah keterampilan Anda ke arah mana pun yang paling Anda minati. Itulah yang dimaksud dengan "Memperkenalkan Ilmu Data" dan kami harap Anda akan menikmati perjalanan bersama kami.



Gambar 1.14 Hasil akhir: ikhtisar gaji rata-rata berdasarkan profesi

1.6 RINGKASAN

Dalam bab ini Anda mempelajari hal-hal berikut:

- Data besar adalah istilah umum untuk setiap kumpulan kumpulan data yang begitu besar atau kompleks sehingga menjadi sulit untuk memprosesnya menggunakan teknik pengelolaan data tradisional. Mereka dicirikan oleh empat V: kecepatan, variasi, volume, dan kebenaran.
- Ilmu data melibatkan penggunaan metode untuk menganalisis kumpulan data kecil hingga kumpulan data besar yang sangat besar.
- Meskipun proses ilmu data tidak linier, proses ini dapat dibagi menjadi beberapa langkah:
  1. Menetapkan tujuan penelitian
  2. Mengumpulkan data
  3. Persiapan data
  4. Eksplorasi data
  5. Pemodelan
  6. Presentasi dan otomatisasi
- Lanskap big data lebih dari sekadar Hadoop. Ini terdiri dari banyak teknologi berbeda yang dapat dikategorikan sebagai berikut:
  - Berkas sistem
  - Kerangka pemrograman terdistribusi
  - Integrasi data
  - Database
  - Pembelajaran mesin
  - Keamanan
  - Penjadwalan
  - Tolok ukur

  - Penyebaran sistem
  - Pemrograman layanan
- Tidak semua kategori data besar banyak digunakan oleh ilmuwan data. Mereka berfokus terutama pada sistem file, kerangka kerja pemrograman terdistribusi, basis data, dan pembelajaran mesin. Mereka bersentuhan dengan komponen lain, tetapi ini adalah domain dari profesi lain.
- Data bisa datang dalam berbagai bentuk. Bentuk utamanya adalah
  - Data terstruktur
  - Data tidak terstruktur
  - Data bahasa alami
  - Data mesin
  - Data berbasis grafik
  - Streaming data

BAB 2
PROSES ILMU DATA

Dalam bab ini mahasiswa diharapakan mampu:

- Memahami alur proses ilmu data
- Mendiskusikan langkah-langkah dalam proses ilmu data

Tujuan dari bab ini adalah untuk memberikan gambaran tentang proses ilmu data tanpa menyelami data besar. Anda akan mempelajari cara bekerja dengan kumpulan data besar, data streaming, dan data teks di bab berikutnya.

2.1 TINJAUAN PROSES ILMU DATA

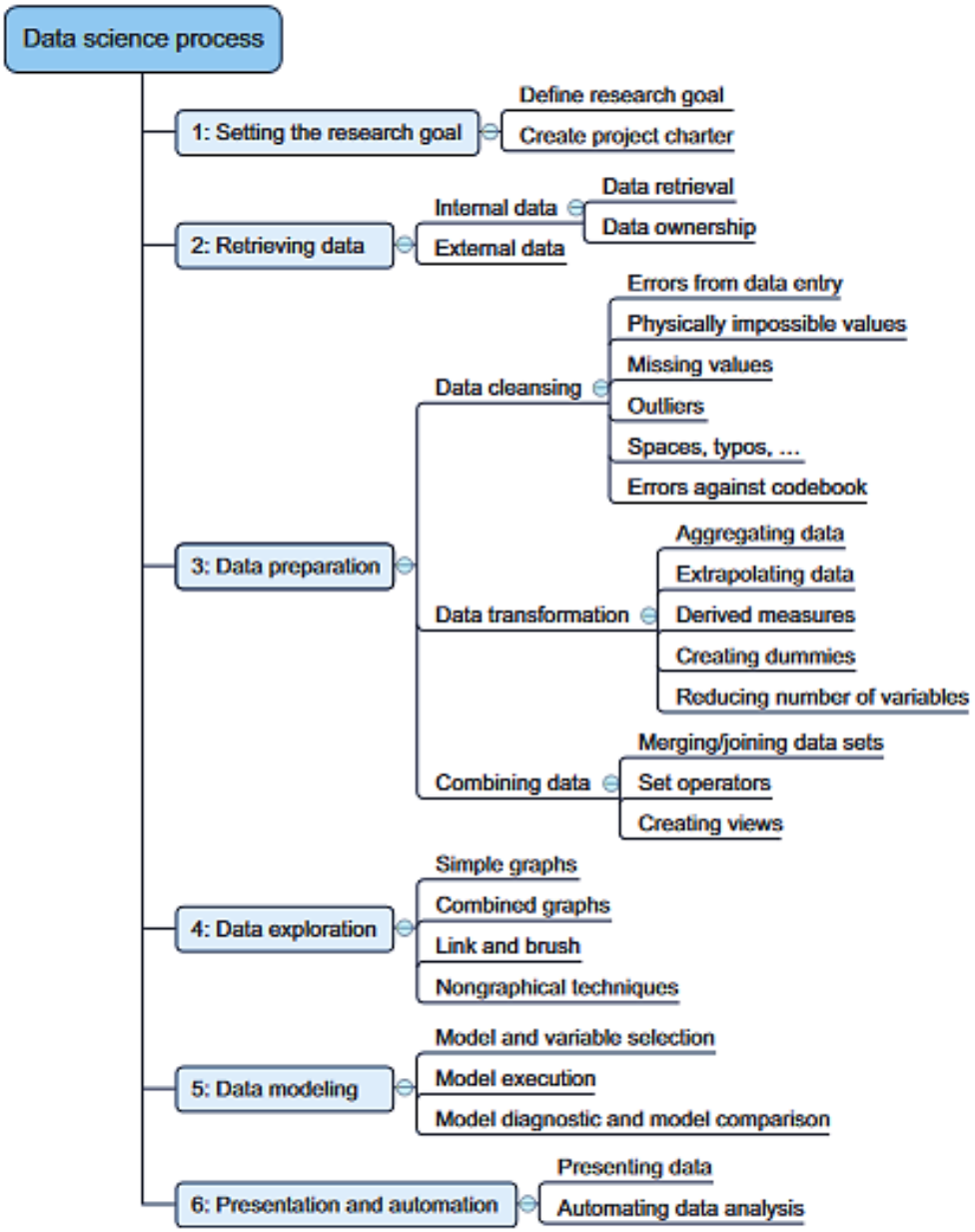
Mengikuti pendekatan terstruktur untuk ilmu data membantu Anda memaksimalkan peluang sukses dalam proyek ilmu data dengan biaya terendah. Itu juga memungkinkan untuk mengambil proyek sebagai sebuah tim, dengan setiap anggota tim berfokus pada apa yang mereka lakukan terbaik. Namun berhati-hatilah: pendekatan ini mungkin tidak cocok untuk setiap jenis proyek atau menjadi satu-satunya cara untuk melakukan ilmu data yang baik. Proses ilmu data tipikal terdiri dari enam langkah yang akan Anda lakukan iterasi, seperti yang ditunjukkan pada gambar 2.1.

Gambar 2.1 merangkum proses ilmu data dan menunjukkan langkah dan tindakan utama yang akan Anda ambil selama proyek. Daftar berikut adalah pengantar singkat; setiap langkah akan dibahas secara lebih mendalam di sepanjang bab ini.

1. Langkah pertama dari proses ini adalah menetapkan tujuan penelitian. Tujuan utama di sini adalah memastikan semua pemangku kepentingan memahami apa, bagaimana, dan mengapa proyek tersebut. Dalam setiap proyek serius ini akan menghasilkan piagam proyek.
2. Tahap kedua adalah pengambilan data. Anda ingin data tersedia untuk dianalisis, jadi langkah ini termasuk menemukan data yang cocok dan mendapatkan akses ke data dari pemilik data. Hasilnya adalah data dalam bentuk mentahnya, yang mungkin perlu dipoles dan diubah sebelum dapat digunakan.
3. Sekarang setelah Anda memiliki data mentah, saatnya menyiapkannya. Ini termasuk mengubah data dari bentuk mentah menjadi data yang langsung dapat digunakan dalam model Anda. Untuk mencapai ini, Anda akan mendeteksi dan mengoreksi berbagai jenis kesalahan dalam data, menggabungkan data dari sumber data yang berbeda, dan mengubahnya. Jika Anda telah berhasil menyelesaikan langkah ini, Anda dapat melanjutkan ke visualisasi dan pemodelan data.
4. Langkah keempat adalah eksplorasi data. Tujuan dari langkah ini adalah untuk mendapatkan pemahaman yang mendalam tentang data. Anda akan mencari pola, korelasi, dan penyimpangan berdasarkan teknik visual dan deskriptif. Wawasan yang Anda peroleh dari fase ini akan memungkinkan Anda untuk memulai pemodelan.
5. Akhirnya, kita sampai pada bagian terseksi: pembuatan model (sering disebut sebagai "pemodelan data" di seluruh buku ini). Sekarang Anda mencoba untuk mendapatkan

wawasan atau membuat prediksi yang dinyatakan dalam piagam proyek Anda. Sekarang saatnya mengeluarkan senjata berat, tetapi ingat penelitian telah mengajarkan kita bahwa seringkali (namun tidak selalu) kombinasi model sederhana cenderung mengungguli satu model rumit. Jika Anda telah melakukan fase ini dengan benar, Anda hampir selesai.

6. Langkah terakhir dari model ilmu data adalah mempresentasikan hasil Anda dan mengotomatiskan analisis, jika diperlukan. Salah satu tujuan proyek adalah mengubah proses dan/atau membuat keputusan yang lebih baik. Anda mungkin masih perlu meyakinkan bisnis bahwa temuan Anda memang akan mengubah proses bisnis seperti yang diharapkan. Di sinilah Anda bisa bersinar dalam peran influencer Anda. Pentingnya langkah ini lebih terlihat dalam proyek-proyek di tingkat strategis dan taktis. Proyek tertentu mengharuskan Anda untuk melakukan proses bisnis berulang kali, sehingga mengotomatiskan proyek akan menghemat waktu.



Gambar 2.1 Enam langkah proses data science

Pada kenyataannya Anda tidak akan maju secara linier dari langkah 1 ke langkah 6. Sering kali Anda akan mundur dan beralih di antara fase yang berbeda. Mengikuti enam langkah ini terbayar dalam hal rasio keberhasilan proyek yang lebih tinggi dan peningkatan dampak hasil penelitian. Proses ini memastikan Anda memiliki rencana penelitian yang terdefinisi dengan baik, pemahaman yang baik tentang pertanyaan bisnis, dan pengiriman yang jelas bahkan sebelum Anda mulai melihat data. Langkah pertama dari proses Anda berfokus pada mendapatkan data berkualitas tinggi sebagai input untuk model Anda. Dengan cara ini model Anda akan bekerja lebih baik di kemudian hari. Dalam ilmu data ada pepatah terkenal: Sampah masuk sama dengan sampah keluar. Manfaat lain dari mengikuti pendekatan terstruktur adalah Anda bekerja lebih banyak dalam mode prototipe saat Anda mencari model terbaik. Saat membuat prototipe, Anda mungkin akan mencoba beberapa model dan tidak terlalu berfokus pada masalah seperti kecepatan program atau penulisan kode yang bertentangan dengan standar. Hal ini memungkinkan Anda untuk fokus membawa nilai bisnis. Tidak setiap proyek diprakarsai oleh bisnis itu sendiri. Wawasan yang dipelajari selama analisis atau kedatangan data baru dapat menelurkan proyek baru. Saat tim ilmu data menghasilkan ide, pekerjaan telah dilakukan untuk membuat proposisi dan menemukan sponsor bisnis.

Membagi proyek menjadi tahapan yang lebih kecil juga memungkinkan karyawan untuk bekerja sama sebagai satu tim. Tidak mungkin menjadi spesialis dalam segala hal. Anda perlu mengetahui cara mengunggah semua data ke semua basis data yang berbeda, menemukan skema data optimal yang berfungsi tidak hanya untuk aplikasi Anda tetapi juga untuk proyek lain di dalam perusahaan Anda, dan kemudian melacak semua statistik dan data-teknik penambangan, sekaligus ahli dalam alat presentasi dan politik bisnis. Itu tugas yang sulit, dan itulah sebabnya semakin banyak perusahaan mengandalkan tim spesialis daripada mencoba menemukan satu orang yang dapat melakukan semuanya.

Proses yang kami jelaskan di bagian ini paling cocok untuk proyek ilmu data yang hanya berisi beberapa model. Itu tidak cocok untuk setiap jenis proyek. Misalnya, proyek yang berisi jutaan model real-time memerlukan pendekatan yang berbeda dari alur yang kami jelaskan di sini. Namun, seorang ilmuwan data pemula harus mengikuti cara kerja ini.

2.1.1 Jangan menjadi budak proses

Tidak setiap proyek akan mengikuti cetak biru ini, karena proses Anda tunduk pada preferensi ilmuwan data, perusahaan, dan sifat proyek yang Anda kerjakan. Beberapa perusahaan mungkin meminta Anda untuk mengikuti protokol yang ketat, sedangkan yang lain memiliki cara kerja yang lebih informal. Secara umum, Anda memerlukan pendekatan terstruktur saat mengerjakan proyek yang kompleks atau saat banyak orang atau sumber daya terlibat.

Model proyek tangkas adalah alternatif dari proses berurutan dengan iterasi. Karena metodologi ini memenangkan lebih banyak tempat di departemen TI dan di seluruh perusahaan, metodologi ini juga diadopsi oleh komunitas ilmu data. Meskipun metodologi tangkas cocok untuk proyek ilmu data, banyak kebijakan perusahaan akan mendukung pendekatan yang lebih kaku terhadap ilmu data. Merencanakan setiap detail proses ilmu data di muka tidak selalu memungkinkan, dan lebih sering daripada tidak Anda akan beralih di

antara langkah-langkah proses yang berbeda. Misalnya, setelah pengarahan Anda memulai aliran normal Anda hingga Anda berada dalam fase analisis data eksplorasi. Grafik Anda menunjukkan perbedaan perilaku antara dua kelompok—pria dan wanita mungkin? Anda tidak yakin karena Anda tidak memiliki variabel yang menunjukkan apakah pelanggan pria atau wanita. Anda perlu mengambil kumpulan data tambahan untuk mengonfirmasi ini. Untuk ini, Anda harus melalui proses persetujuan, yang menunjukkan bahwa Anda (atau bisnis) perlu memberikan semacam piagam proyek. Di perusahaan besar, mendapatkan semua data yang Anda butuhkan untuk menyelesaikan proyek Anda bisa menjadi siksaan.

2.2 MENENTUKAN TUJUAN PENELITIAN DAN MEMBUAT PIAGAM PROYEK

Sebuah proyek dimulai dengan memahami apa, mengapa, dan bagaimana proyek Anda (gambar 2.2). Apa yang perusahaan harapkan dari Anda? Dan mengapa manajemen sangat menghargai penelitian Anda? Apakah itu bagian dari gambaran strategis yang lebih besar atau proyek "serigala tunggal" yang berasal dari peluang yang terdeteksi seseorang? Menjawab ketiga pertanyaan ini (apa, mengapa, bagaimana) adalah tujuan dari fase pertama, sehingga semua orang tahu apa yang harus dilakukan dan dapat menyepakati tindakan terbaik.

Hasilnya harus berupa tujuan penelitian yang jelas, pemahaman yang baik tentang konteks, penyampaian yang terdefinisi dengan baik, dan rencana tindakan dengan jadwal. Informasi ini kemudian paling baik ditempatkan dalam piagam proyek. Panjang dan formalitas dapat, tentu saja, berbeda antara proyek dan perusahaan. Pada fase awal proyek ini, keterampilan orang dan ketajaman bisnis lebih penting daripada kecakapan teknis yang hebat, itulah sebabnya bagian ini sering kali dipandu oleh personel yang lebih senior.

Gambar 2.2 Langkah 1: Menetapkan tujuan penelitian

2.2.1 Luangkan waktu untuk memahami tujuan dan konteks penelitian Anda

Hasil penting adalah tujuan penelitian yang menyatakan tujuan tugas Anda secara jelas dan terfokus. Memahami tujuan dan konteks bisnis sangat penting untuk keberhasilan proyek. Lanjutkan mengajukan pertanyaan dan merancang contoh sampai Anda memahami harapan bisnis yang tepat, mengidentifikasi bagaimana proyek Anda sesuai dengan gambaran yang lebih besar, hargai bagaimana penelitian Anda akan mengubah bisnis, dan pahami bagaimana mereka akan menggunakan hasil Anda. Tidak ada yang lebih membuat frustrasi daripada menghabiskan waktu berbulan-bulan untuk meneliti sesuatu sampai Anda memiliki satu momen kecemerlangan dan menyelesaikan masalah, tetapi ketika Anda melaporkan temuan Anda kembali ke organisasi, semua orang segera menyadari bahwa Anda salah memahami pertanyaan mereka. Jangan mengabaikan fase ini dengan enteng. Banyak ilmuwan data gagal di sini: terlepas dari kecerdasan matematis dan kecemerlangan ilmiah mereka, mereka tampaknya tidak pernah memahami tujuan dan konteks bisnis.

2.2.2 Membuat piagam proyek

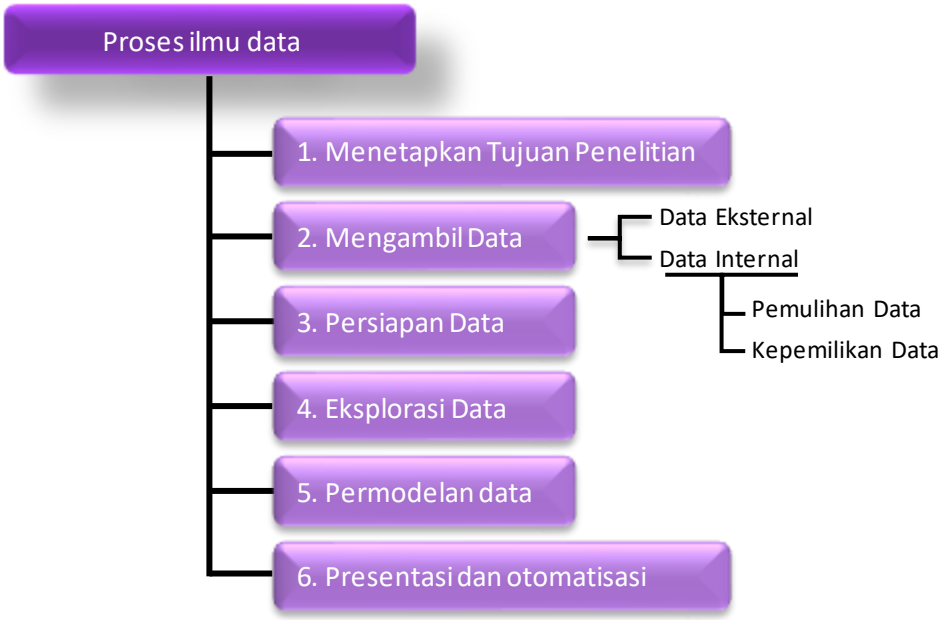
Klien ingin mengetahui di awal apa yang mereka bayar, jadi setelah Anda memiliki pemahaman yang baik tentang masalah bisnis, cobalah untuk mendapatkan kesepakatan formal tentang hasil kerja. Semua informasi ini sebaiknya dikumpulkan dalam piagam proyek. Untuk proyek penting apa pun, ini wajib. Piagam proyek membutuhkan kerja tim, dan masukan Anda setidaknya mencakup hal-hal berikut:

- Tujuan penelitian yang jelas
- Misi dan konteks proyek
- Bagaimana Anda akan melakukan analisis Anda
- Sumber daya apa yang ingin Anda gunakan
- Bukti bahwa ini adalah proyek yang dapat dicapai, atau bukti konsep
- Hasil dan ukuran keberhasilan
- Garis waktu

Klien Anda dapat menggunakan informasi ini untuk memperkirakan biaya proyek dan data serta orang yang diperlukan agar proyek Anda berhasil.

2.3 PENGAMBILAN DATA

Langkah selanjutnya dalam ilmu data adalah mengambil data yang diperlukan (gambar 2.3). Terkadang Anda perlu masuk ke lapangan dan merancang sendiri proses pengumpulan data, tetapi seringkali Anda tidak akan terlibat dalam langkah ini. Banyak perusahaan telah mengumpulkan dan menyimpan data untuk Anda, dan apa yang tidak mereka miliki seringkali dapat dibeli dari pihak ketiga. Jangan takut untuk mencari data di luar organisasi Anda, karena semakin banyak organisasi yang membuat bahkan data berkualitas tinggi tersedia secara gratis untuk penggunaan publik dan komersial.



Gambar 2.3 Langkah 2: Mengambil data

Data dapat disimpan dalam berbagai bentuk, mulai dari file teks sederhana hingga tabel dalam database. Tujuannya sekarang adalah memperoleh semua data yang Anda butuhkan. Ini mungkin sulit, dan bahkan jika Anda berhasil, data sering kali seperti berlian dalam bentuk kasar: perlu dipoles agar bisa berguna bagi Anda.

2.3.1 Mulailah dengan data yang disimpan di dalam perusahaan

Tindakan pertama Anda adalah menilai relevansi dan kualitas data yang tersedia di perusahaan Anda. Sebagian besar perusahaan memiliki program untuk memelihara data kunci, sehingga sebagian besar pekerjaan pembersihan mungkin sudah dilakukan. Data ini dapat disimpan dalam repositori data resmi seperti database, data mart, gudang data, dan data lake yang dikelola oleh tim profesional TI. Tujuan utama database adalah penyimpanan data, sementara gudang data dirancang untuk membaca dan menganalisis data tersebut. Data mart adalah bagian dari gudang data dan diarahkan untuk melayani unit bisnis tertentu. Sementara data warehouse dan data mart adalah rumah bagi data yang diproses sebelumnya, data lake berisi data dalam format natural atau mentah. Namun ada kemungkinan bahwa data Anda masih berada di file Excel di desktop pakar domain.

Menemukan data bahkan di dalam perusahaan Anda sendiri terkadang bisa menjadi tantangan. Seiring pertumbuhan perusahaan, data mereka tersebar di banyak tempat. Pengetahuan tentang data dapat tersebar saat orang berganti posisi dan meninggalkan perusahaan. Dokumentasi dan metadata tidak selalu menjadi prioritas utama manajer pengiriman, jadi mungkin Anda perlu mengembangkan beberapa keterampilan seperti Sherlock Holmes untuk menemukan semua bagian yang hilang.

Mendapatkan akses ke data adalah tugas sulit lainnya. Organisasi memahami nilai dan kepekaan data dan seringkali memiliki kebijakan sehingga setiap orang memiliki akses ke apa yang mereka butuhkan dan tidak lebih. Kebijakan ini diterjemahkan menjadi penghalang fisik dan digital yang disebut tembok Cina. “Dinding” ini bersifat wajib dan diatur dengan baik

untuk data pelanggan di sebagian besar negara. Ini juga untuk alasan yang bagus; bayangkan semua orang di perusahaan kartu kredit memiliki akses ke kebiasaan belanja Anda. Mendapatkan akses ke data mungkin memerlukan waktu dan melibatkan politik perusahaan.

2.3.2 Jangan takut untuk berbelanja

Jika data tidak tersedia di dalam organisasi Anda, lihat di luar dinding organisasi Anda. Banyak perusahaan berspesialisasi dalam mengumpulkan informasi berharga. Misalnya, Nielsen dan GFK terkenal akan hal ini di industri retail. Perusahaan lain menyediakan data sehingga Anda, pada gilirannya, dapat memperkaya layanan dan ekosistem mereka. Seperti halnya dengan Twitter, LinkedIn, dan Facebook.

Meskipun data dianggap sebagai aset yang lebih berharga daripada minyak oleh perusahaan tertentu, semakin banyak pemerintah dan organisasi membagikan data mereka secara gratis kepada dunia. Data ini bisa berkualitas sangat baik; itu tergantung pada institusi yang membuat dan mengelolanya. Informasi yang mereka bagikan mencakup berbagai topik seperti jumlah kecelakaan atau jumlah penyalahgunaan narkoba di wilayah tertentu dan demografinya. Data ini berguna saat Anda ingin memperkaya data hak milik, tetapi juga nyaman saat melatih keterampilan ilmu data Anda di rumah. Tabel 2.1 hanya menunjukkan sedikit pilihan dari semakin banyak penyedia data terbuka.

Tabel 2.1 Daftar penyedia data terbuka yang sebaiknya Anda gunakan untuk memulai

Buka situs data	**Keterangan**
Data.gov	Rumah data terbuka Pemerintah AS
https://open-data.europa.eu/	Pusat data terbuka Komisi Eropa
Freebase.org	Basis data terbuka yang mengambil informasinya dari situs-situs seperti Wikipedia, MusicBrains, dan arsip SEC
Data.worldbank.org	Inisiatif data terbuka dari Bank Dunia
Aiddata.org	Data terbuka untuk pembangunan internasional
Open.fda.gov	Buka data dari Food and Drug Administration AS

2.3.3 Lakukan pemeriksaan kualitas data sekarang untuk mencegah masalah di kemudian hari

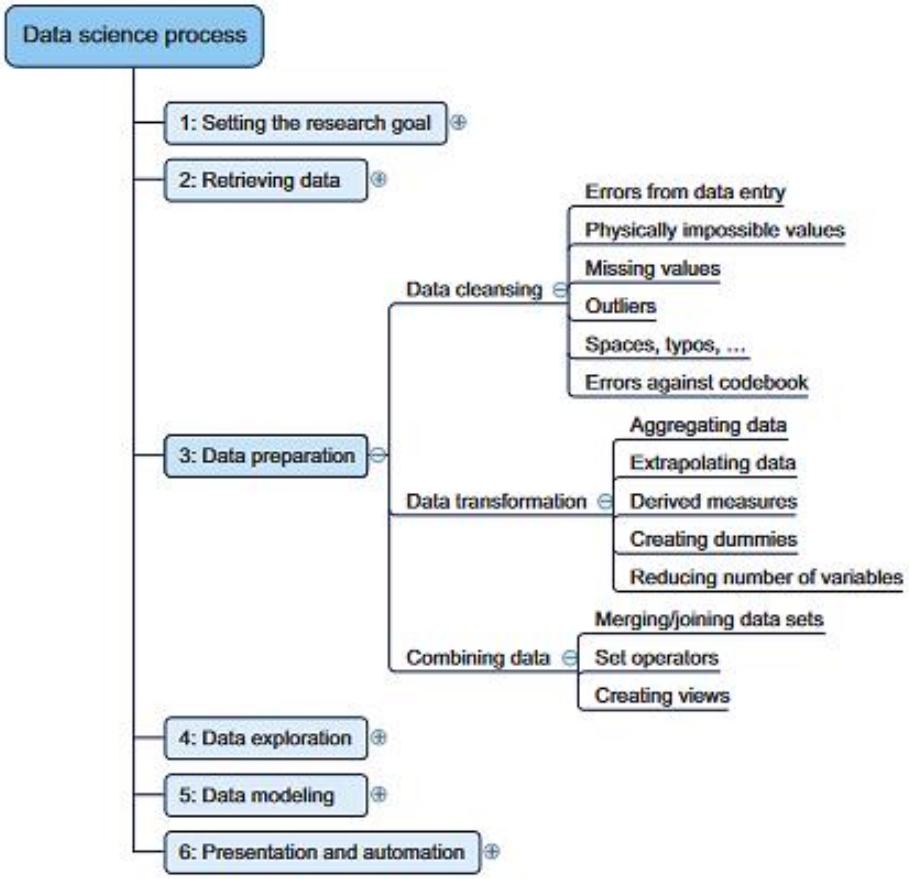
Berharap untuk menghabiskan sebagian besar waktu proyek Anda melakukan koreksi dan pembersihan data, terkadang hingga 80%. Pengambilan data adalah pertama kalinya Anda akan memeriksa data dalam proses ilmu data. Sebagian besar kesalahan yang akan Anda temui selama fase pengumpulan data mudah dikenali, tetapi terlalu ceroboh akan membuat Anda menghabiskan waktu berjam-jam untuk menyelesaikan masalah data yang seharusnya dapat dicegah selama impor data.

Anda akan menyelidiki data selama fase impor, persiapan data, dan eksplorasi. Perbedaannya terletak pada tujuan dan kedalaman penyelidikan. Selama pengambilan data, Anda memeriksa apakah data sama dengan data di dokumen sumber dan melihat apakah

Anda memiliki tipe data yang tepat. Ini seharusnya tidak memakan waktu terlalu lama; ketika Anda memiliki cukup bukti bahwa data tersebut serupa dengan data yang Anda temukan di dokumen sumber, Anda berhenti. Dengan persiapan data, Anda melakukan pemeriksaan yang lebih rumit. Jika Anda melakukan pekerjaan dengan baik selama fase sebelumnya, kesalahan yang Anda temukan sekarang juga ada di dokumen sumber. Fokusnya adalah pada konten variabel: Anda ingin menghilangkan kesalahan ketik dan kesalahan entri data lainnya serta membawa data ke standar umum di antara kumpulan data. Misalnya, Anda dapat mengoreksi USQ ke AS dan Inggris ke Inggris. Selama fase eksplorasi, fokus Anda beralih ke apa yang dapat Anda pelajari dari data. Sekarang Anda menganggap data bersih dan melihat properti statistik seperti distribusi, korelasi, dan outlier. Anda akan sering mengulangi fase ini. Misalnya, ketika Anda menemukan outlier dalam fase eksplorasi, mereka dapat menunjukkan kesalahan entri data. Sekarang setelah Anda memahami bagaimana kualitas data ditingkatkan selama proses berlangsung, kita akan melihat lebih dalam ke langkah persiapan data.

2.4 MEMBERSIHKAN, MENGINTEGRASIKAN, DAN MENGUBAH DATA

Data yang diterima dari fase pengambilan data kemungkinan akan menjadi "berlian yang masih kasar". Tugas Anda sekarang adalah membersihkan dan menyiapkannya untuk digunakan dalam fase pemodelan dan pelaporan. Melakukannya sangatlah penting karena model Anda akan bekerja lebih baik dan Anda akan kehilangan lebih sedikit waktu untuk mencoba memperbaiki keluaran yang aneh. Tidak cukup sering disebutkan: sampah masuk sama dengan sampah keluar. Model Anda memerlukan data dalam format tertentu, sehingga transformasi data akan selalu berperan. Merupakan kebiasaan yang baik untuk memperbaiki kesalahan data sedini mungkin dalam proses. Namun, ini tidak selalu memungkinkan dalam pengaturan yang realistis, jadi Anda harus mengambil tindakan korektif dalam program Anda. Gambar 2.4 menunjukkan tindakan yang paling umum dilakukan selama fase pembersihan data, integrasi, dan transformasi. Peta pikiran ini mungkin terlihat agak abstrak untuk saat ini, tetapi kami akan menangani semua poin ini secara lebih mendetail di bagian selanjutnya. Anda akan melihat kesamaan besar di antara semua tindakan ini.



Gambar 2.4 Langkah 3: Persiapan data

2.4.1 Membersihkan data

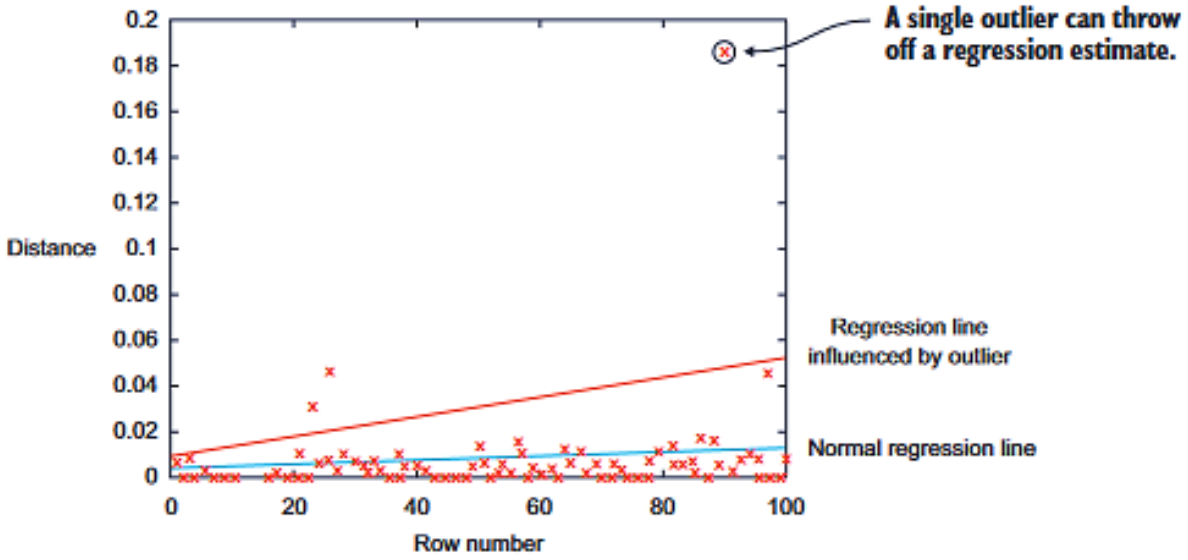
Pembersihan data adalah subproses dari proses ilmu data yang berfokus pada menghilangkan kesalahan pada data Anda sehingga data Anda menjadi representasi yang benar dan konsisten dari proses asalnya.

Dengan "representasi yang benar dan konsisten" kami menyiratkan bahwa setidaknya ada dua jenis kesalahan. Jenis pertama adalah kesalahan interpretasi, seperti ketika Anda menerima begitu saja nilai dalam data Anda, seperti mengatakan bahwa usia seseorang lebih dari 300 tahun. Jenis kesalahan kedua menunjukkan ketidakkonsistenan antara sumber data atau dengan nilai standar perusahaan Anda. Contoh dari kelas kesalahan ini adalah menempatkan "Wanita" di satu tabel dan "F" di tabel lain ketika mewakili hal yang sama: bahwa orang tersebut adalah wanita. Contoh lain adalah Anda menggunakan Pounds di satu meja dan Dolar di meja lain. Terlalu banyak kemungkinan kesalahan yang ada untuk membuat daftar ini lengkap, tetapi tabel 2.2 menunjukkan ikhtisar jenis kesalahan yang dapat dideteksi dengan pemeriksaan mudah—seolah-olah "hasil yang menggantung rendah".

Tabel 2.2 Ikhtisar kesalahan umum

Solusi umum	
Coba perbaiki masalah di awal rantai akuisisi data atau perbaiki di program.	
Deskripsi kesalahan	**Solusi yang mungkin**

Kesalahan yang menunjuk ke nilai salah dalam satu kumpulan data	
Kesalahan saat memasukkan data	Aturan manual
Ruang putih berlebihan	Gunakan fungsi string
Nilai-nilai yang mustahil	Aturan manual
Nilai yang hilang	Hapus pengamatan atau nilai
Penyimpangan	Validasi dan, jika salah, perlakukan sebagai nilai yang hilang (hapus atau sisipkan)
Kesalahan yang menunjukkan ketidakkonsistenan antara kumpulan data	
Penyimpangan dari buku kode	Cocokkan dengan tombol atau gunakan pengaturan manual
Satuan pengukuran yang berbeda	Hitung ulang
Berbagai tingkat agregasi	Bawa ke tingkat pengukuran yang sama dengan agregasi atau ekstrapolasi



Gambar 2.5 Titik yang dilingkari sangat memengaruhi model dan perlu diselidiki karena dapat menunjuk ke wilayah di mana Anda tidak memiliki cukup data atau mungkin menunjukkan kesalahan dalam data, tetapi juga dapat menjadi titik data yang valid.

Terkadang Anda akan menggunakan metode yang lebih canggih, seperti pemodelan sederhana, untuk menemukan dan mengidentifikasi kesalahan data; plot diagnostik bisa sangat berwawasan. Misalnya, pada gambar 2.5 kami menggunakan ukuran untuk mengidentifikasi titik data yang tampaknya tidak pada tempatnya. Kami melakukan regresi untuk mengenal data dan mendeteksi pengaruh pengamatan individu pada garis regresi. Ketika satu pengamatan memiliki terlalu banyak pengaruh, ini bisa menunjukkan kesalahan dalam data, tapi juga bisa menjadi poin yang valid. Namun, pada tahap pembersihan data, metode canggih ini jarang diterapkan dan sering dianggap oleh ilmuwan data tertentu sebagai sesuatu yang berlebihan. Sekarang setelah kami memberikan ikhtisar, saatnya menjelaskan kesalahan ini secara lebih mendetail.

Kesalahan Input Data

Pengumpulan data dan entri data adalah proses yang rawan kesalahan. Mereka sering membutuhkan campur tangan manusia, dan karena manusia hanya manusia, mereka membuat kesalahan ketik atau kehilangan konsentrasi sesaat dan memasukkan kesalahan ke dalam rantai. Tetapi data yang dikumpulkan oleh mesin atau komputer juga tidak lepas dari kesalahan. Kesalahan dapat timbul dari kecerobohan manusia, sedangkan yang lain disebabkan oleh kegagalan mesin atau perangkat keras. Contoh error yang berasal dari mesin adalah kesalahan transmisi atau bug pada fase extract, transform, dan load (ETL).

Untuk kumpulan data kecil, Anda dapat memeriksa setiap nilai dengan tangan. Mendeteksi kesalahan data ketika variabel yang Anda pelajari tidak memiliki banyak kelas dapat dilakukan dengan tabulasi data dengan hitungan. Ketika Anda memiliki variabel yang hanya dapat mengambil dua nilai: "Baik" dan "Buruk", Anda dapat membuat tabel frekuensi dan melihat apakah hanya itu dua nilai yang ada. Pada tabel 2.3, nilai “Godo” dan “Bade” menunjukkan ada yang tidak beres setidaknya dalam 16 kasus.

Tabel 2.3 Mendeteksi outlier pada variabel sederhana dengan tabel frekuensi

Nilai	Menghitung
Bagus	1598647
Buruk	1354468
Godo	15
Bade	1

Sebagian besar kesalahan jenis ini mudah diperbaiki dengan pernyataan penugasan sederhana dan aturan if-then-else:

```
if x == "Godo";
   x = "Good"
if x == "Bade";
   x == "Bad"
```

Ruang Putih Redundan

Spasi putih cenderung sulit dideteksi tetapi menyebabkan kesalahan seperti karakter redundan lainnya. Siapa yang tidak kehilangan beberapa hari dalam sebuah proyek karena bug yang disebabkan oleh spasi putih di akhir string? Anda meminta program untuk menggabungkan dua kunci dan melihat bahwa pengamatan hilang dari file keluaran. Setelah mencari berhari-hari melalui kode, akhirnya Anda menemukan bug tersebut. Kemudian tibalah bagian tersulit: menjelaskan keterlambatan kepada pemangku kepentingan proyek. Pembersihan selama fase ETL tidak dijalankan dengan baik, dan kunci dalam satu tabel berisi spasi kosong di akhir string. Hal ini menyebabkan ketidakcocokan kunci seperti “FR ” – “FR”, menghilangkan pengamatan yang tidak dapat dicocokkan.

Jika Anda tahu untuk berhati-hati terhadapnya, untungnya memperbaiki spasi putih yang berlebihan cukup mudah di sebagian besar bahasa pemrograman. Semuanya menyediakan fungsi string yang akan menghapus spasi kosong di depan dan di belakang.

Misalnya, di Python Anda dapat menggunakan fungsi strip() untuk menghapus spasi awal dan akhir.

Memperbaiki Kesalahan Huruf Kapital

Ketidaksesuaian huruf kapital sering terjadi. Sebagian besar bahasa pemrograman membedakan antara "Brasil" dan "Brasil". Dalam hal ini Anda dapat menyelesaikan masalah dengan menerapkan fungsi yang mengembalikan kedua string dalam huruf kecil, seperti `.lower()` dengan Python. `"Brazil".lower() == "brazil".lower()` harus menghasilkan true.

Nilai-Nilai Mustahil dan Pemeriksaan Sanitasi

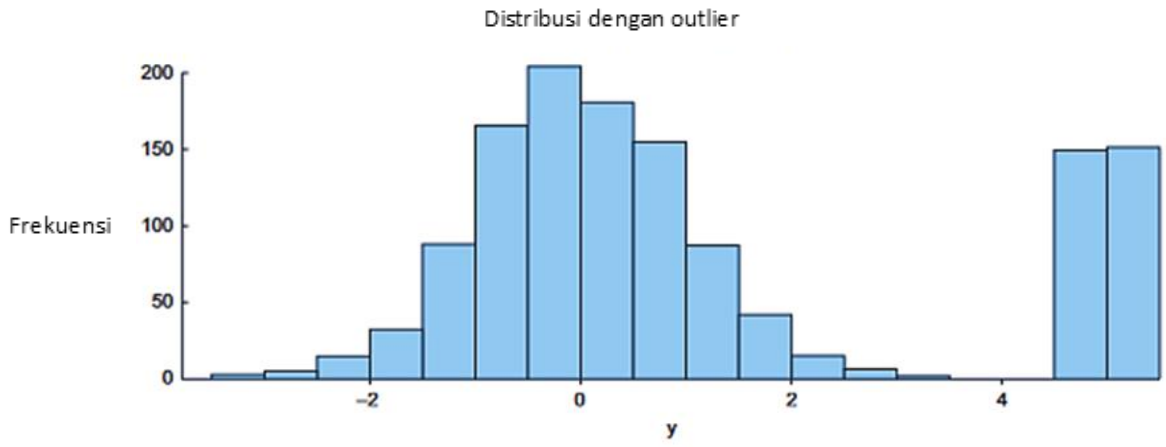
Pemeriksaan kewarasan adalah jenis pemeriksaan data lain yang berharga. Di sini Anda membandingkan nilai dengan nilai yang tidak mungkin secara fisik atau teoritis seperti orang yang lebih tinggi dari 3 meter atau seseorang dengan usia 299 tahun. Pemeriksaan kewarasan dapat langsung dinyatakan dengan aturan:

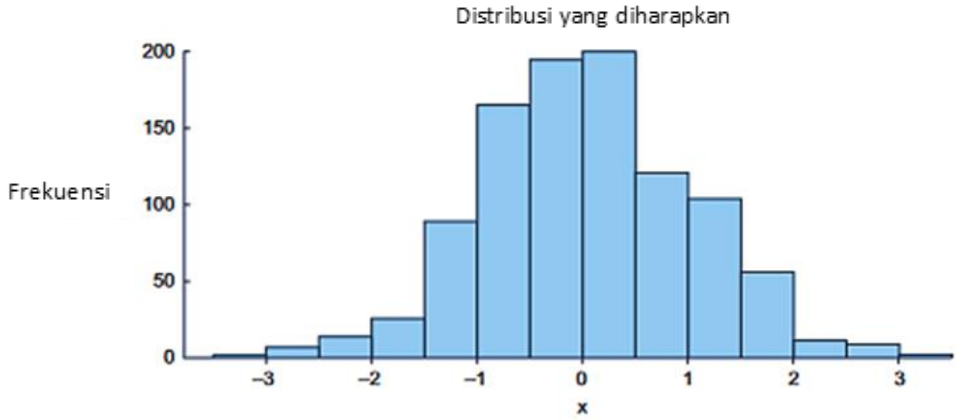
```
Check = 0 <= age <= 120
```

Outlier

Outlier adalah pengamatan yang tampak jauh dari pengamatan lain atau, lebih khusus lagi, satu pengamatan yang mengikuti logika atau proses generatif yang berbeda dari pengamatan lainnya. Cara termudah untuk menemukan outlier adalah dengan menggunakan plot atau tabel dengan nilai minimum dan maksimum. Contohnya ditunjukkan pada gambar 2.6. Plot di atas tidak menunjukkan outlier, sedangkan plot di bawah menunjukkan kemungkinan outlier di sisi atas ketika diharapkan distribusi normal. Distribusi normal, atau distribusi Gaussian, adalah distribusi paling umum dalam ilmu alam.

Ini menunjukkan sebagian besar kasus terjadi di sekitar rata-rata distribusi dan kejadiannya menurun ketika semakin jauh darinya. Nilai tinggi di grafik bawah dapat menunjukkan outlier saat mengasumsikan distribusi normal. Seperti yang kita lihat sebelumnya dengan contoh regresi, outlier dapat sangat memengaruhi pemodelan data Anda, jadi selidiki terlebih dahulu.





Gambar 2.6 Plot distribusi sangat membantu dalam mendeteksi outlier dan membantu Anda memahami variabelnya.

Menghadapi Nilai-Nilai Yang Hilang

Nilai yang hilang tidak selalu salah, tetapi Anda tetap harus menanganinya secara terpisah; teknik pemodelan tertentu tidak dapat menangani nilai yang hilang. Mereka mungkin menjadi indikator bahwa ada yang tidak beres dalam pengumpulan data Anda atau bahwa terjadi kesalahan dalam proses ETL. Teknik umum yang digunakan ilmuwan data tercantum dalam tabel 2.4.

Tabel 2.4 Gambaran teknik untuk menangani data yang hilang

Teknik	Keuntungan	Kerugian
Abaikan nilai-nilainya	Mudah dilakukan	Anda kehilangan informasi dari pengamatan
Tetapkan nilai ke nol	Mudah dilakukan	Tidak semua teknik pemodelan dan/atau implementasi dapat menangani nilai nol
Hitung nilai statis seperti 0 atau rata-rata	Mudah dilakukan Anda tidak kehilangan informasi dari variabel lain dalam pengamatan	Dapat menyebabkan estimasi palsu dari model
Hitung nilai dari perkiraan atau distribusi teoretis	Tidak terlalu mengganggu model	Lebih sulit untuk dieksekusi Anda membuat asumsi data
Memodelkan nilai (tidak tergantung)	Tidak terlalu mengganggu model	Dapat menyebabkan terlalu banyak kepercayaan pada model Dapat secara artifisial meningkatkan ketergantungan antar variabel Lebih sulit untuk dieksekusi Anda membuat asumsi data

Teknik mana yang digunakan pada jam berapa tergantung pada kasus khusus Anda. Jika, misalnya, Anda tidak memiliki pengamatan untuk dicadangkan, menghilangkan

pengamatan mungkin bukanlah suatu pilihan. Jika variabel dapat dijelaskan dengan distribusi yang stabil, Anda dapat menghitung berdasarkan ini. Namun, mungkinkah nilai yang hilang sebenarnya berarti "nol"? Ini dapat terjadi dalam penjualan misalnya: jika tidak ada promosi yang diterapkan pada keranjang pelanggan, promo pelanggan tersebut hilang, tetapi kemungkinan besar juga 0, tidak ada potongan harga.

Penyimpangan Dari Buku Kode

Mendeteksi kesalahan dalam kumpulan data yang lebih besar terhadap buku kode atau terhadap nilai standar dapat dilakukan dengan bantuan operasi himpunan. Buku kode adalah deskripsi data Anda, suatu bentuk metadata. Ini berisi hal-hal seperti jumlah variabel per pengamatan, jumlah pengamatan, dan apa arti setiap pengkodean dalam variabel. (Misalnya "0" sama dengan "negatif", "5" adalah singkatan dari "sangat positif".) Buku kode juga memberi tahu jenis data yang Anda lihat: apakah hierarkis, grafik, atau hal lain?

Anda melihat nilai-nilai yang ada di set A tetapi tidak di set B. Ini adalah nilai yang harus diperbaiki. Bukan kebetulan bahwa set adalah struktur data yang akan kita gunakan saat bekerja dalam kode. Merupakan kebiasaan yang baik untuk memberikan pemikiran tambahan pada struktur data Anda; itu dapat menghemat pekerjaan dan meningkatkan kinerja program Anda.

Jika Anda memiliki beberapa nilai untuk diperiksa, lebih baik memasukkannya dari buku kode ke dalam tabel dan menggunakan operator perbedaan untuk memeriksa perbedaan antara kedua tabel. Dengan cara ini, Anda bisa mendapatkan keuntungan dari kekuatan database secara langsung. Lebih lanjut tentang ini di bab 5.

Unit Pengukuran Yang Berbeda

Saat mengintegrasikan dua set data, Anda harus memperhatikan unit pengukurannya masing-masing. Contohnya adalah ketika Anda mempelajari harga bensin di dunia. Untuk melakukan ini, Anda mengumpulkan data dari berbagai penyedia data. Kumpulan data dapat berisi harga per galon dan lainnya dapat berisi harga per liter. Konversi sederhana akan berhasil dalam kasus ini.

Tingkat Agregasi yang Berbeda

Memiliki tingkat agregasi yang berbeda serupa dengan memiliki jenis pengukuran yang berbeda. Contohnya adalah kumpulan data yang berisi data per minggu versus kumpulan data yang berisi data per minggu kerja. Jenis kesalahan ini umumnya mudah dideteksi, dan meringkas (atau kebalikannya, memperluas) kumpulan data akan memperbaikinya. Setelah membersihkan kesalahan data, Anda menggabungkan informasi dari sumber data yang berbeda. Namun sebelum kita membahas topik ini, kita akan mengambil sedikit jalan memutar dan menekankan pentingnya membersihkan data sedini mungkin.

2.4.2 Perbaiki kesalahan sedini mungkin

Praktik yang baik adalah memediasi kesalahan data sedini mungkin dalam rantai pengumpulan data dan memperbaiki sesedikit mungkin di dalam program Anda sambil memperbaiki asal mula masalahnya. Mengambil data adalah tugas yang sulit, dan organisasi menghabiskan jutaan dolar untuk itu dengan harapan dapat membuat keputusan yang lebih baik. Proses pengumpulan data rawan kesalahan, dan dalam organisasi besar melibatkan banyak langkah dan tim.

Data harus dibersihkan saat diperoleh karena berbagai alasan:

- Tidak semua orang menemukan anomali data. Pengambil keputusan mungkin membuat kesalahan yang mahal pada informasi berdasarkan data yang salah dari aplikasi yang gagal mengoreksi data yang salah.
- Jika kesalahan tidak diperbaiki di awal proses, pembersihan harus dilakukan untuk setiap proyek yang menggunakan data tersebut.
- Kesalahan data mungkin mengarah ke proses bisnis yang tidak berfungsi seperti yang dirancang. Misalnya, kedua penulis bekerja di pengecer di masa lalu, dan mereka merancang sistem kupon untuk menarik lebih banyak orang dan menghasilkan keuntungan lebih tinggi. Selama proyek ilmu data, kami menemukan klien yang menyalahgunakan sistem kupon dan mendapatkan uang saat membeli bahan makanan. Tujuan dari sistem kupon adalah untuk merangsang penjualan silang, bukan untuk memberikan produk secara gratis. Cacat ini merugikan uang perusahaan dan tidak ada seorang pun di perusahaan yang menyadarinya. Dalam hal ini datanya tidak salah secara teknis tetapi datang dengan hasil yang tidak diharapkan.
- Kesalahan data mungkin menunjukkan peralatan yang rusak, seperti saluran transmisi yang rusak dan sensor yang rusak.
- Kesalahan data dapat menunjukkan bug dalam perangkat lunak atau dalam integrasi perangkat lunak yang mungkin penting bagi perusahaan. Saat melakukan proyek kecil di bank kami menemukan bahwa dua aplikasi perangkat lunak menggunakan pengaturan lokal yang berbeda. Ini menyebabkan masalah dengan angka lebih dari 1.000. Untuk satu aplikasi, angka 1.000 berarti satu, dan untuk aplikasi lainnya berarti seribu.

Memperbaiki data segera setelah diambil itu bagus di dunia yang sempurna. Sayangnya, seorang ilmuwan data tidak selalu memiliki suara dalam pengumpulan data dan hanya memberi tahu departemen TI untuk memperbaiki hal-hal tertentu mungkin tidak akan berhasil. Jika Anda tidak dapat memperbaiki data di sumbernya, Anda harus menanganinya di dalam kode Anda. Manipulasi data tidak berakhir dengan mengoreksi kesalahan; Anda masih perlu menggabungkan data yang masuk.

Sebagai catatan terakhir: selalu simpan salinan data asli Anda (jika memungkinkan). Kadang-kadang Anda mulai membersihkan data tetapi Anda akan membuat kesalahan: menghubungkan variabel dengan cara yang salah, menghapus outlier yang memiliki informasi tambahan yang menarik, atau mengubah data sebagai akibat dari salah tafsir awal. Jika Anda menyimpan salinannya, Anda dapat mencoba lagi. Untuk "mengalirkan data" yang dimanipulasi pada saat kedatangan, ini tidak selalu memungkinkan dan Anda akan menerima periode penyesuaian sebelum Anda dapat menggunakan data yang Anda tangkap. Salah satu hal yang lebih sulit bukanlah pembersihan data dari kumpulan data individual, tetapi menggabungkan berbagai sumber menjadi satu kesatuan yang lebih masuk akal.

2.4.3 Menggabungkan data dari sumber data yang berbeda

Data Anda berasal dari beberapa tempat yang berbeda, dan dalam sublangkah ini kami berfokus pada pengintegrasian sumber-sumber yang berbeda ini. Data bervariasi dalam ukuran, jenis, dan struktur, mulai dari database dan file Excel hingga dokumen teks.

Kami fokus pada data dalam struktur tabel dalam bab ini demi singkatnya. Sangat mudah untuk mengisi seluruh buku tentang topik ini saja, dan kami memilih untuk fokus pada proses ilmu data daripada menyajikan skenario untuk setiap jenis data. Namun perlu diingat bahwa ada jenis sumber data lain, seperti penyimpanan nilai kunci, penyimpanan dokumen, dan sebagainya, yang akan kami tangani di tempat yang lebih sesuai di buku ini.

Cara Yang Berbeda Untuk Menggabungkan Data

Anda dapat melakukan dua operasi untuk menggabungkan informasi dari set data yang berbeda. Operasi pertama adalah bergabung: memperkaya pengamatan dari satu tabel dengan informasi dari tabel lain. Operasi kedua adalah menambahkan atau menumpuk: menambahkan pengamatan dari satu tabel ke tabel lain.

Saat Anda menggabungkan data, Anda memiliki opsi untuk membuat tabel fisik baru atau tabel virtual dengan membuat tampilan. Keuntungan dari tampilan adalah tidak menghabiskan lebih banyak ruang disk. Mari kita uraikan sedikit tentang metode ini.

Tabel Bergabung

Menggabungkan tabel memungkinkan Anda menggabungkan informasi dari satu pengamatan yang ditemukan di satu tabel dengan informasi yang Anda temukan di tabel lain. Fokusnya adalah pada memperkaya pengamatan tunggal. Katakanlah tabel pertama berisi informasi tentang pembelian pelanggan dan tabel lainnya berisi informasi tentang wilayah tempat tinggal pelanggan Anda. Menggabungkan tabel memungkinkan Anda menggabungkan informasi sehingga Anda dapat menggunakannya untuk model Anda, seperti yang ditunjukkan pada gambar 2.7.

Untuk menggabungkan tabel, Anda menggunakan variabel yang mewakili objek yang sama di kedua tabel, seperti tanggal, nama negara, atau nomor Jaminan Sosial. Bidang umum ini dikenal sebagai kunci. Ketika kunci ini juga secara unik menentukan catatan dalam tabel, mereka disebut kunci primer. Satu tabel mungkin memiliki perilaku pembelian dan tabel lainnya mungkin memiliki informasi demografis seseorang. Pada gambar 2.7 kedua tabel berisi nama klien, dan ini memudahkan untuk memperkaya pengeluaran klien dengan wilayah klien. Orang yang terbiasa dengan Excel akan melihat kesamaan dengan menggunakan fungsi pencarian.

Gambar 2.7 Menggabungkan dua tabel pada kunci Item dan Region

Jumlah baris yang dihasilkan dalam tabel keluaran tergantung pada jenis gabungan yang tepat yang Anda gunakan. Kami memperkenalkan berbagai jenis gabungan nanti di buku ini.

Tabel Penyimpanan

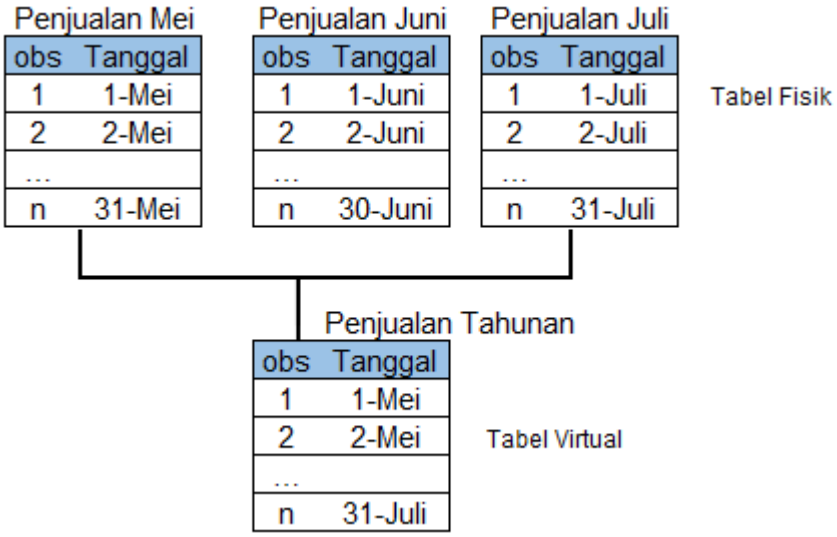
Menambahkan atau menumpuk tabel secara efektif menambahkan pengamatan dari satu tabel ke tabel lainnya. Gambar 2.8 menunjukkan contoh penambahan tabel. Satu tabel berisi observasi dari bulan Januari dan tabel kedua berisi observasi dari bulan Februari. Hasil menambahkan tabel ini lebih besar dengan pengamatan dari Januari serta Februari.

Gambar 2.8 Menambahkan data dari tabel adalah operasi umum tetapi membutuhkan struktur yang sama dalam tabel yang ditambahkan.

Operasi yang setara dalam teori himpunan akan menjadi penyatuan, dan ini juga merupakan perintah dalam SQL, bahasa umum dari basis data relasional. Operator himpunan lain juga digunakan dalam ilmu data, seperti perbedaan himpunan dan irisan.

Menggunakan Tampilan Untuk Simulasi Data Gabung Dan Tambahkan

Untuk menghindari duplikasi data, Anda secara virtual menggabungkan data dengan tampilan. Pada contoh sebelumnya kami mengambil data bulanan dan menggabungkannya dalam tabel fisik baru. Masalahnya adalah kami menggandakan data dan karena itu membutuhkan lebih banyak ruang penyimpanan. Dalam contoh yang sedang kita kerjakan, hal itu mungkin tidak menimbulkan masalah, tetapi bayangkan bahwa setiap tabel terdiri dari terabyte data; maka menjadi masalah untuk menggandakan data. Untuk alasan ini, konsep tampilan diciptakan. Tampilan berperilaku seolah-olah Anda sedang mengerjakan tabel, tetapi tabel ini hanyalah lapisan virtual yang menggabungkan tabel untuk Anda. Gambar 2.9 menunjukkan bagaimana data penjualan dari bulan yang berbeda digabungkan secara virtual menjadi tabel penjualan tahunan alih-alih menduplikasi data. Namun, tampilan memang memiliki kelemahan. Meskipun gabungan tabel hanya dilakukan sekali, gabungan yang membuat tampilan dibuat ulang setiap kali kueri, menggunakan lebih banyak daya pemrosesan daripada yang dimiliki tabel yang telah dihitung sebelumnya.



Gambar 2.9 Tampilan membantu Anda menggabungkan data tanpa replikasi.

Pengayaan Tindakan Agregat

Pengayaan data juga dapat dilakukan dengan menambahkan informasi yang dihitung ke dalam tabel, seperti jumlah total penjualan atau berapa persentase total stok yang telah terjual di suatu wilayah tertentu (gambar 2.10).

Tindakan ekstra seperti ini dapat menambah perspektif. Melihat gambar 2.10, kami sekarang memiliki kumpulan data agregat, yang pada gilirannya dapat digunakan untuk menghitung partisipasi setiap produk dalam kategorinya. Ini bisa berguna selama eksplorasi data, tetapi lebih bermanfaat saat membuat model data. Seperti biasa ini tergantung pada kasus yang tepat, tetapi dari model pengalaman kami dengan "ukuran relatif" seperti% penjualan (kuantitas produk yang dijual/jumlah total terjual) cenderung mengungguli model yang menggunakan angka mentah (kuantitas terjual) sebagai masukan.

Jenis	Produk	Penjualan dalam Rp X.000	Penjualan t-1 dalam Rp. X.000	Pertumbuhan	Penjualan Per Jenis	Peringkat Penjualan
A	B	X	Y	(X-Y)/Y	AX	NX
Olah Raga	Olah Raga 1	95	98	-3.06%	215	2
Olah Raga	Olah Raga 2	120	132	-9.09%	215	1
Sepatu	Sepatu 1	10	6	66.67%	10	3

Gambar 2.10 Pertumbuhan, penjualan berdasarkan kelas produk, dan peringkat penjualan adalah contoh pengukuran turunan dan agregat.

2.4.4 Mengubah data

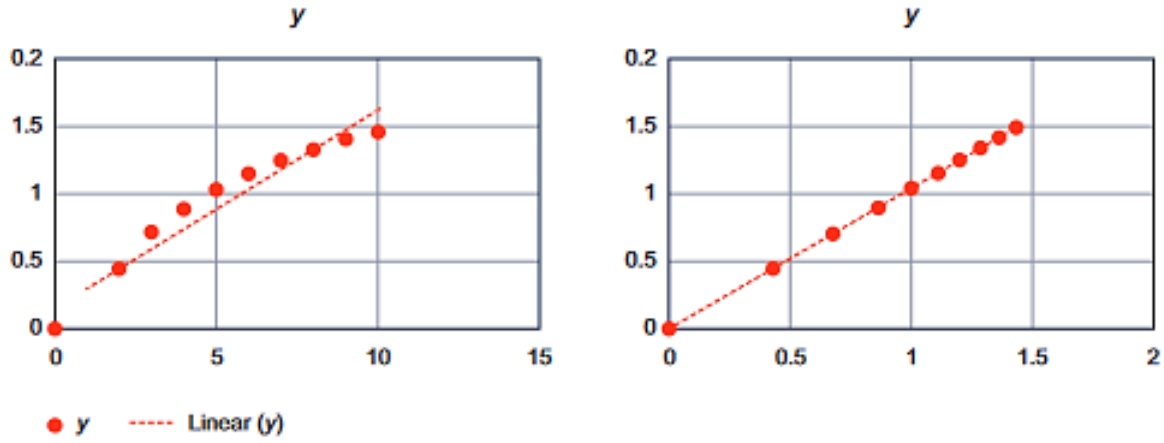
Model tertentu memerlukan datanya dalam bentuk tertentu. Sekarang setelah Anda membersihkan dan mengintegrasikan data, inilah tugas berikutnya yang akan Anda lakukan: mengubah data Anda sehingga menjadi bentuk yang sesuai untuk pemodelan data.

Mengubah Data

Hubungan antara variabel input dan variabel output tidak selalu linier. Ambil, misalnya, hubungan dalam bentuk y = aebx. Mengambil log dari variabel independen menyederhanakan

masalah estimasi secara dramatis. Gambar 2.11 menunjukkan bagaimana mengubah variabel input sangat menyederhanakan masalah estimasi. Lain kali Anda mungkin ingin menggabungkan dua variabel menjadi variabel baru.

x	1	2	3	4	5	6	7	8	9	10
log(x)	0.00	0.43	0.68	0.86	1.00	1.11	1.21	1.29	1.37	1.43
y	0.00	0.44	0.69	0.87	1.02	1.11	1.24	1.32	1.38	1.46



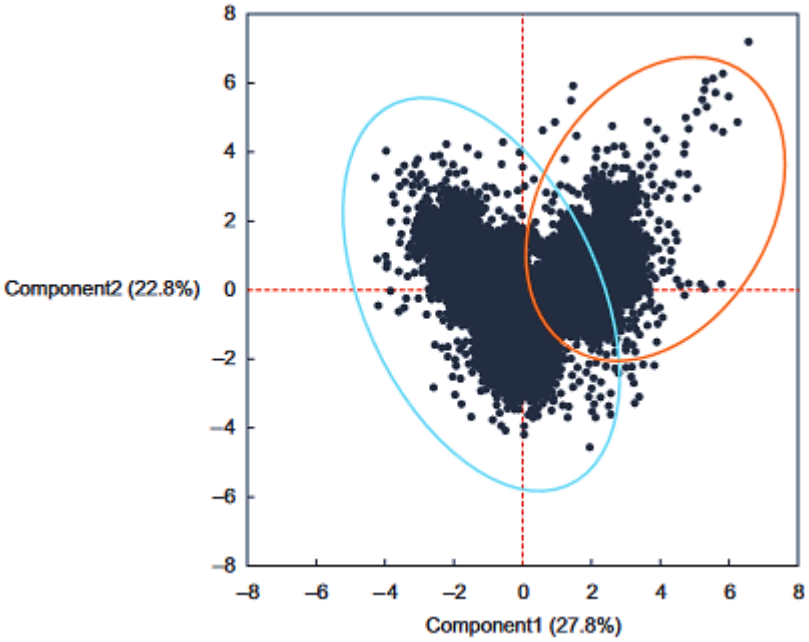
Gambar 2.11 Transformasi x ke log x membuat hubungan antara x dan y linier (kanan), dibandingkan dengan non-log x (kiri).

Mengurangi Jumlah Variabel

Terkadang Anda memiliki terlalu banyak variabel dan perlu mengurangi jumlahnya karena variabel tersebut tidak menambahkan informasi baru ke dalam model. Memiliki terlalu banyak variabel dalam model Anda membuat model sulit untuk ditangani, dan teknik tertentu tidak bekerja dengan baik saat Anda membebani mereka dengan terlalu banyak variabel masukan. Misalnya, semua teknik yang didasarkan pada jarak Euclidean bekerja dengan baik hanya sampai 10 variabel.

Jarak Euclidean

Jarak euklidean atau jarak "biasa" merupakan perluasan dari salah satu hal pertama yang dipelajari seseorang dalam matematika tentang segitiga (trigonometri): teorema kaki Pythagoras. Jika Anda mengetahui panjang kedua sisi di samping sudut 90° dari segitiga siku-siku, Anda dapat dengan mudah menurunkan panjang sisi yang tersisa (sisi miring). Rumusnya adalah $Sisi\ miring = \sqrt{(side1 + side2)^2}$. Jarak Euclidean antara dua titik dalam bidang dua dimensi dihitung menggunakan rumus serupa: $Jarak\ = \sqrt{((x1 - x2)^2 + (y1 - y2)^2)}$. Jika Anda ingin memperluas penghitungan jarak ini ke lebih banyak dimensi, tambahkan koordinat titik dalam dimensi yang lebih tinggi tersebut ke dalam rumus. Untuk tiga dimensi diperoleh $Jarak\ = \sqrt{((x1 - x2)^2 + (y1 - y2)^2 + (z1 - z2)^2)}$



Gambar 2.12 Pengurangan variabel memungkinkan Anda mengurangi jumlah variabel sambil mempertahankan informasi sebanyak mungkin.

Ilmuwan data menggunakan metode khusus untuk mengurangi jumlah variabel tetapi mempertahankan jumlah data maksimum. Kami akan membahas beberapa metode ini di bab 3. Gambar 2.12 menunjukkan bagaimana pengurangan jumlah variabel memudahkan untuk memahami nilai kunci. Ini juga menunjukkan bagaimana dua variabel menyumbang 50,6% dari variasi dalam kumpulan data (komponen1 = 27,8% + komponen2 = 22,8%). Variabel-variabel ini, yang disebut "component1" dan "component2", keduanya merupakan kombinasi dari variabel asli. Mereka adalah komponen utama dari struktur data yang mendasarinya. Jika saat ini belum begitu jelas, jangan khawatir, analisis komponen utama (PCA) akan dijelaskan lebih menyeluruh di bab 3. Yang juga bisa Anda lihat adalah adanya variabel ketiga (tidak diketahui) yang memisahkan kelompok pengamatan menjadi dua.

Mengubah Variabel Menjadi Dummies

Variabel dapat diubah menjadi variabel dummy (gambar 2.13). Variabel dummy hanya dapat mengambil dua nilai: true(1) atau false(0). Mereka digunakan untuk menunjukkan tidak adanya efek kategoris yang dapat menjelaskan pengamatan. Dalam hal ini Anda akan membuat kolom terpisah untuk kelas yang disimpan dalam satu variabel dan menandainya dengan 1 jika kelas tersebut ada dan 0 sebaliknya. Contohnya adalah mengubah satu kolom bernama Hari Kerja menjadi kolom Senin hingga Minggu. Anda menggunakan indikator untuk menunjukkan apakah pengamatan dilakukan pada hari Senin; Anda menempatkan 1 pada hari Senin dan 0 di tempat lain. Mengubah variabel menjadi boneka adalah teknik yang digunakan dalam pemodelan dan populer, tetapi tidak eksklusif bagi, ekonom.

Di bagian ini, kami memperkenalkan langkah ketiga dalam proses ilmu data—membersihkan, mengubah, dan mengintegrasikan data—yang mengubah data mentah Anda menjadi input yang dapat digunakan untuk fase pemodelan. Langkah selanjutnya dalam

proses ilmu data adalah mendapatkan pemahaman yang lebih baik tentang isi data dan hubungan antara variabel dan pengamatan; kami mengeksplorasi ini di bagian selanjutnya.

Pelanggan	Tahun	Jenis kelamin	Penjualan
1	2015	P	10
2	2015	L	8
1	2016	P	11
3	2016	L	12
4	2017	P	14
3	2017	L	13

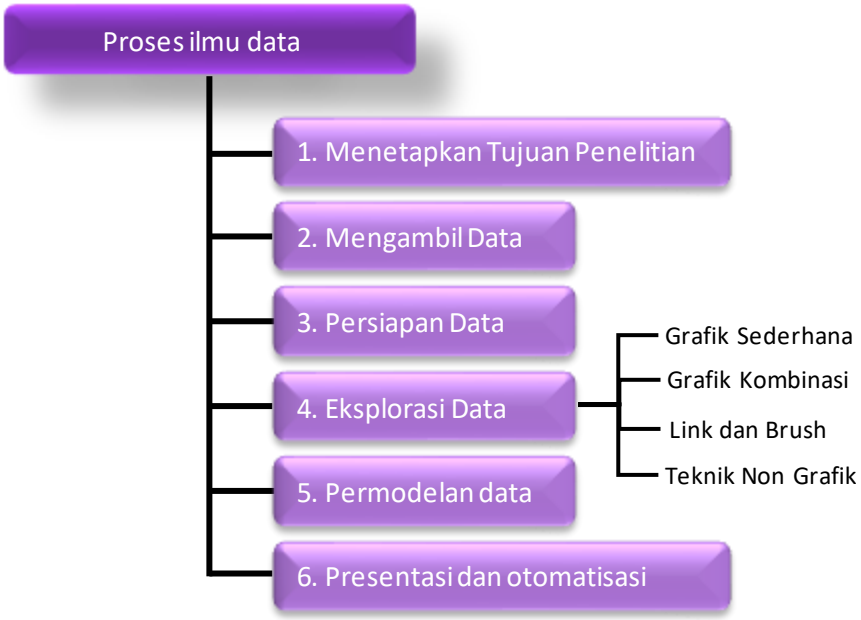
L P

Pelanggan	Tahun	Penjualan	Laki - Laki	Perempuan
1	2015	10	0	1
1	2016	11	0	1
2	2015	8	1	0
3	2016	12	1	0
3	2017	13	1	0
4	2017	14	0	1

Gambar 2.13 Mengubah variabel menjadi boneka adalah transformasi data yang memecah variabel yang memiliki banyak kelas menjadi beberapa variabel, masing-masing hanya memiliki dua kemungkinan nilai: 0 atau 1.

2.5 EKSPLORASI ANALISIS DATA

Selama analisis data eksplorasi, Anda mendalami data tersebut (lihat gambar 2.14). Informasi menjadi lebih mudah untuk dipahami saat ditampilkan dalam gambar, oleh karena itu Anda terutama menggunakan teknik grafis untuk mendapatkan pemahaman tentang data Anda dan interaksi antar variabel. Fase ini adalah tentang mengeksplorasi data, jadi menjaga agar pikiran Anda tetap terbuka dan mata Anda terbuka sangat penting selama fase analisis data eksplorasi. Tujuannya bukan untuk membersihkan data, tetapi biasanya Anda masih menemukan anomali yang Anda lewatkan sebelumnya, memaksa Anda untuk mengambil langkah mundur dan memperbaikinya.



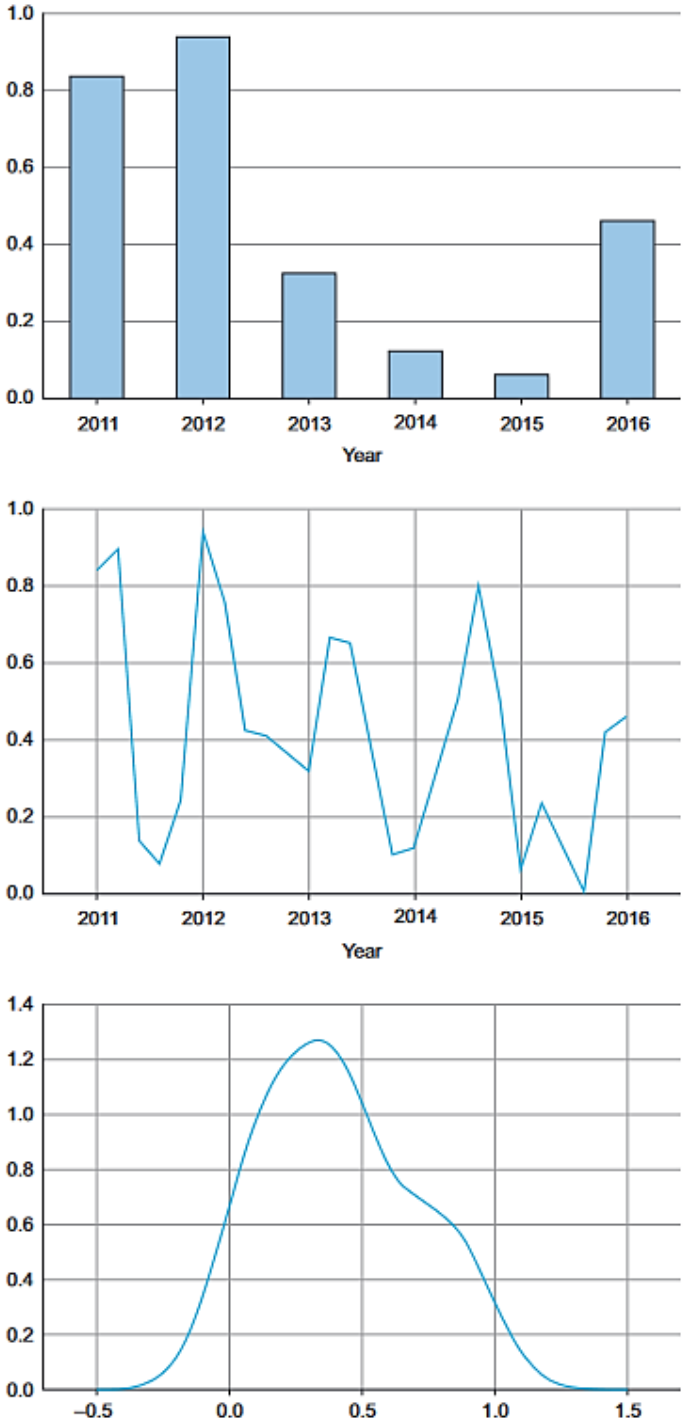
Gambar 2.14 Langkah 4: Eksplorasi data

Teknik visualisasi yang Anda gunakan dalam fase ini berkisar dari grafik garis sederhana atau histogram, seperti yang ditunjukkan pada gambar 2.15, hingga diagram yang lebih kompleks seperti Sankey dan grafik jaringan. Terkadang berguna untuk membuat grafik komposit dari grafik sederhana untuk mendapatkan lebih banyak wawasan tentang data. Di lain waktu grafik dapat dianimasikan atau dibuat interaktif untuk membuatnya lebih mudah dan, mari kita akui, jauh lebih menyenangkan.

Mike Bostock memiliki contoh interaktif dari hampir semua jenis grafik. Layak menghabiskan waktu di situs webnya, meskipun sebagian besar contohnya lebih berguna untuk presentasi data daripada eksplorasi data.

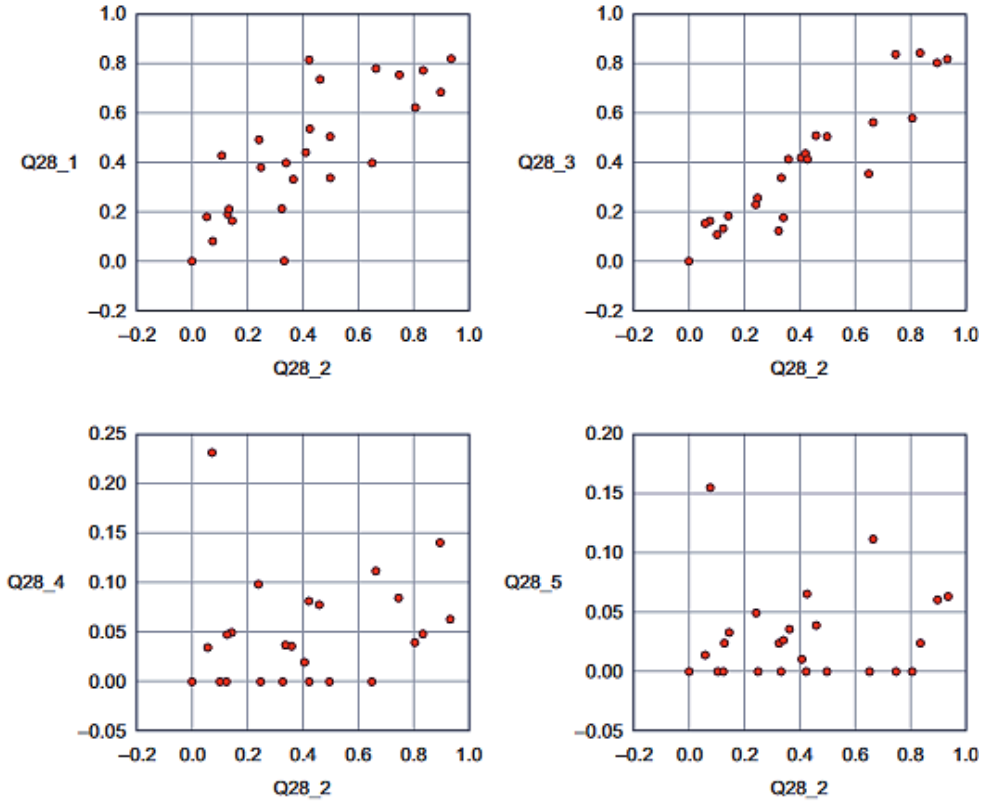
Plot ini dapat digabungkan untuk memberikan lebih banyak wawasan, seperti yang ditunjukkan pada gambar 2.16. Melapisi beberapa plot adalah praktik umum. Pada gambar 2.17 kita menggabungkan graf sederhana menjadi diagram Pareto, atau diagram 80-20.

Gambar 2.18 menunjukkan teknik lain: brushing dan linking. Dengan menyikat dan menautkan Anda menggabungkan dan menautkan grafik dan tabel (atau tampilan) yang berbeda sehingga perubahan dalam satu grafik secara otomatis ditransfer ke grafik lainnya. Contoh terperinci dari hal ini dapat ditemukan di bab 9. Eksplorasi data yang interaktif ini memfasilitasi penemuan wawasan baru.

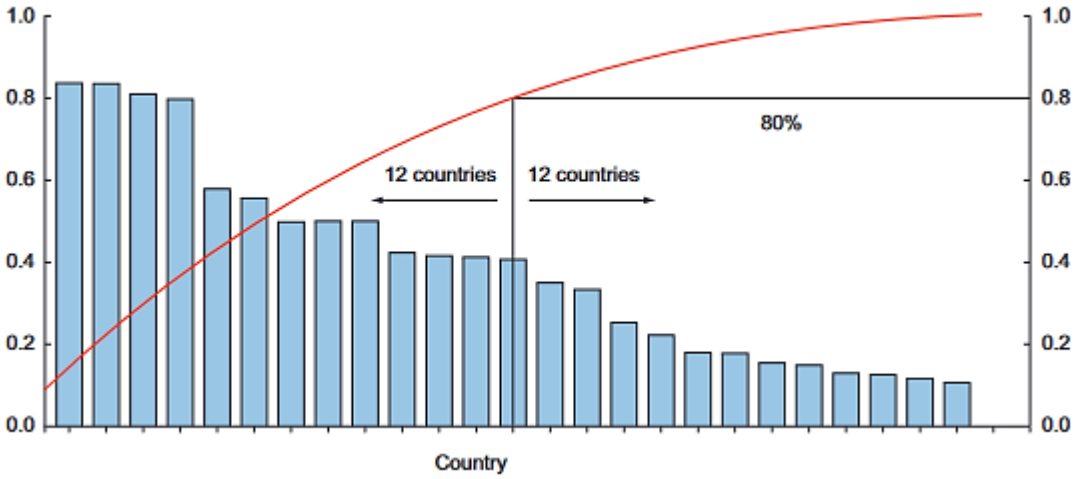


Gambar 2.15 Dari atas ke bawah, diagram batang, plot garis, dan distribusi adalah beberapa grafik yang digunakan dalam analisis eksplorasi.

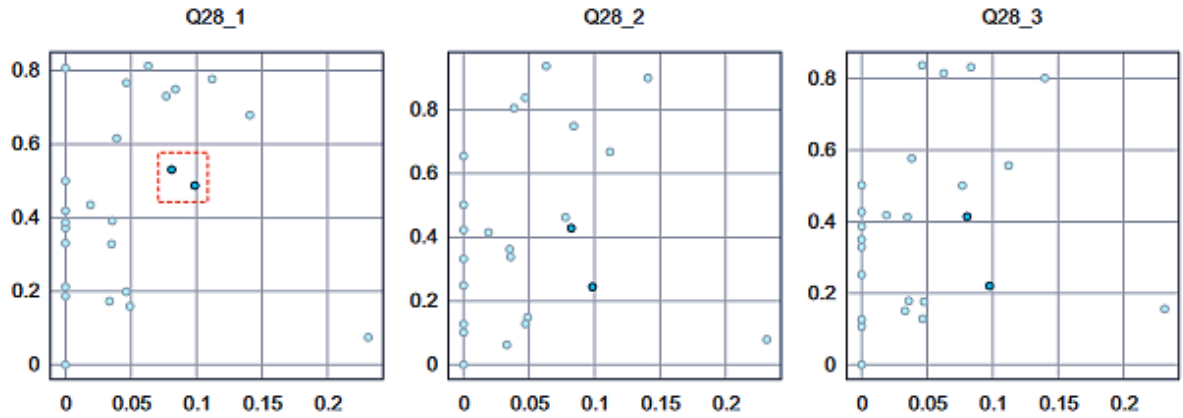
Gambar 2.18 menunjukkan skor rata-rata per negara untuk pertanyaan. Hal ini tidak hanya menunjukkan korelasi yang tinggi antara jawaban, tetapi mudah untuk melihat bahwa saat Anda memilih beberapa titik pada subplot, titik tersebut akan sesuai dengan titik serupa pada grafik lainnya. Dalam hal ini titik-titik yang dipilih pada grafik kiri sesuai dengan titik-titik pada grafik tengah dan kanan, meskipun keduanya lebih sesuai pada grafik tengah dan kanan.



Gambar 2.16 Menggambar beberapa plot secara bersamaan dapat membantu Anda memahami struktur data Anda pada beberapa variabel.

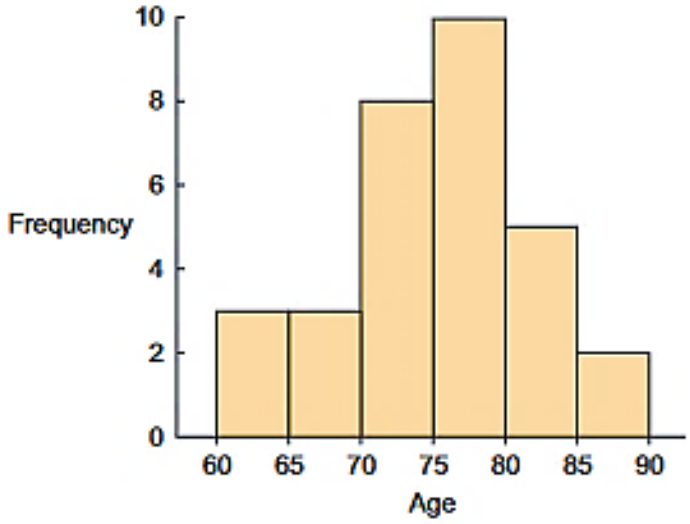


Gambar 2.17 Diagram Pareto merupakan kombinasi dari nilai dan distribusi kumulatif. Sangat mudah untuk melihat dari diagram ini bahwa 50% negara pertama mengandung sedikit kurang dari 80% dari jumlah total. Jika grafik ini mewakili daya beli pelanggan dan kami menjual produk mahal, kami mungkin tidak perlu menghabiskan anggaran pemasaran kami di setiap negara; kita bisa mulai dengan 50% pertama.



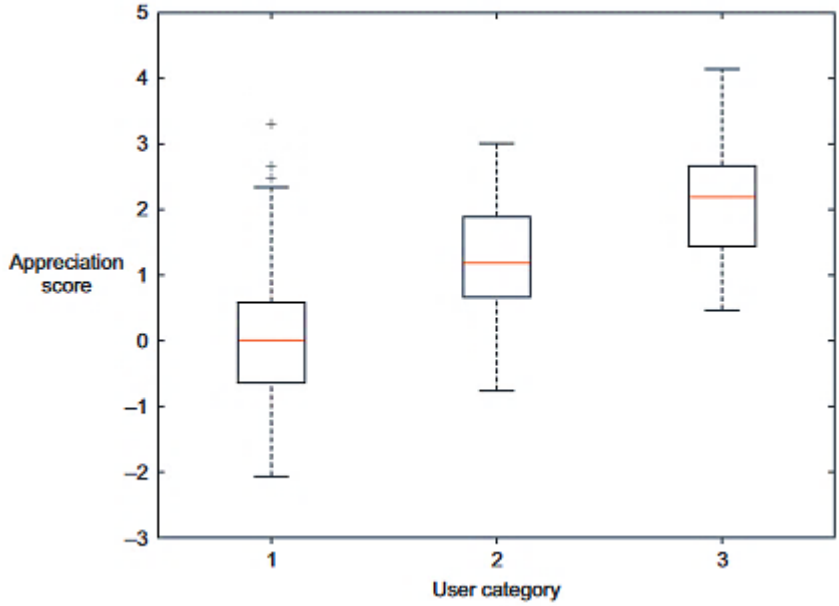
Gambar 2.18 Tautan dan kuas memungkinkan Anda untuk memilih pengamatan di satu plot dan menyorot pengamatan yang sama di plot lainnya.

Dua grafik penting lainnya adalah histogram yang ditunjukkan pada gambar 2.19 dan boxplot yang ditunjukkan pada gambar 2.20. Dalam histogram, sebuah variabel dipotong menjadi kategori-kategori diskrit dan jumlah kejadian dalam setiap kategori dirangkum dan ditampilkan dalam grafik. Boxplot, di sisi lain, tidak menunjukkan berapa banyak pengamatan yang ada tetapi menawarkan kesan distribusi dalam kategori. Itu dapat menunjukkan ukuran maksimum, minimum, median, dan karakterisasi lainnya pada saat yang bersamaan.



Gambar 2.19 Contoh histogram: jumlah penduduk pada kelompok umur dengan interval 5 tahun

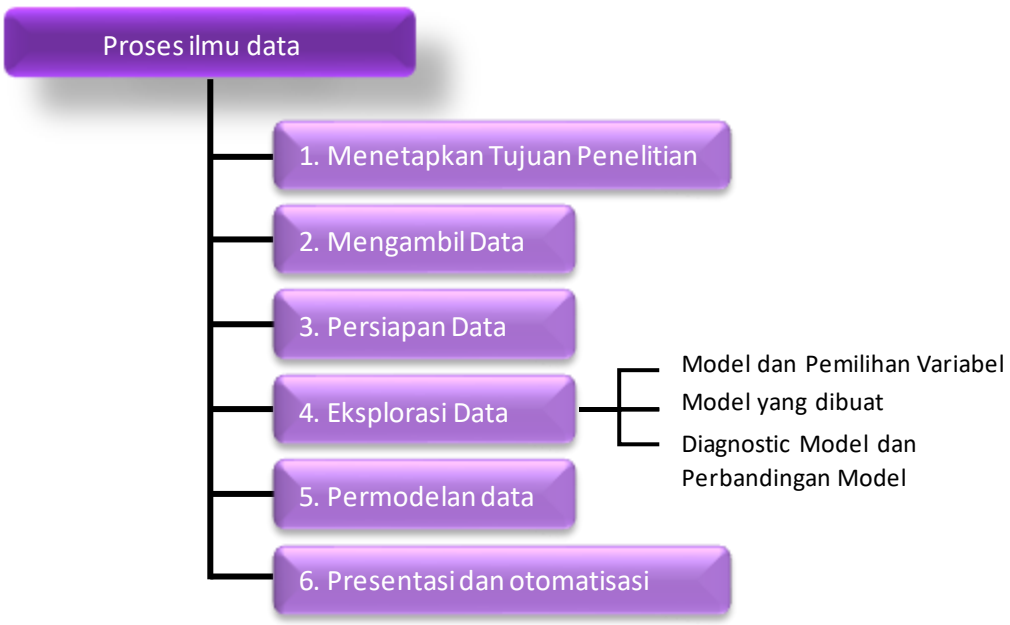
Teknik yang kami jelaskan dalam fase ini sebagian besar bersifat visual, tetapi dalam praktiknya tentu saja tidak terbatas pada teknik visualisasi. Tabulasi, pengelompokan, dan teknik pemodelan lainnya juga dapat menjadi bagian dari analisis eksplorasi. Bahkan membuat model sederhana pun bisa menjadi bagian dari langkah ini. Sekarang setelah Anda menyelesaikan fase eksplorasi data dan Anda telah memahami data Anda dengan baik, saatnya untuk beralih ke fase berikutnya: membuat model.



Gambar 2.20 Contoh boxplot: setiap kategori pengguna memiliki distribusi apresiasi masing-masing untuk gambar tertentu di situs web fotografi.

2.6 MEMBANGUN MODEL

Dengan data yang bersih dan pemahaman konten yang baik, Anda siap membangun model dengan tujuan membuat prediksi yang lebih baik, mengklasifikasikan objek, atau mendapatkan pemahaman tentang sistem yang Anda modelkan. Fase ini jauh lebih fokus daripada langkah analisis eksplorasi, karena Anda tahu apa yang Anda cari dan hasil yang Anda inginkan. Gambar 2.21 menunjukkan komponen bangunan model.



Gambar 2.21 Langkah 5: Pemodelan data

Teknik yang akan Anda gunakan sekarang dipinjam dari bidang pembelajaran mesin, penambangan data, dan/atau statistik. Dalam bab ini kita hanya mengeksplorasi puncak gunung es dari teknik yang ada, sedangkan bab 3 memperkenalkannya dengan benar. Ini di luar cakupan buku ini untuk memberi Anda lebih dari sekadar pengantar konseptual, tetapi itu cukup untuk membantu Anda memulai; 20% dari teknik akan membantu Anda dalam 80% kasus karena teknik tumpang tindih dalam apa yang ingin mereka capai. Mereka sering mencapai tujuan mereka dengan cara yang serupa tetapi sedikit berbeda.

Membangun model adalah proses berulang. Cara Anda membuat model bergantung pada apakah Anda menggunakan statistik klasik atau sekolah pembelajaran mesin yang lebih baru, dan jenis teknik yang ingin Anda gunakan. Either way, sebagian besar model terdiri dari langkah-langkah utama berikut:

1. Pemilihan teknik pemodelan dan variabel yang akan dimasukkan dalam model
2. Eksekusi model
3. Diagnosis dan perbandingan model

2.6.1 Pemilihan model dan variabel

Anda harus memilih variabel yang ingin Anda sertakan dalam model dan teknik pemodelan. Temuan Anda dari analisis eksplorasi seharusnya sudah memberikan gambaran yang adil tentang variabel apa yang akan membantu Anda membangun model yang baik. Banyak teknik pemodelan tersedia, dan memilih model yang tepat untuk suatu masalah memerlukan penilaian Anda. Anda harus mempertimbangkan kinerja model dan apakah proyek Anda memenuhi semua persyaratan untuk menggunakan model Anda, serta faktor lainnya:

- Haruskah model dipindahkan ke lingkungan produksi dan, jika demikian, apakah akan mudah diimplementasikan?
- Seberapa sulit pemeliharaan model: berapa lama akan tetap relevan jika tidak disentuh?
- Apakah model harus mudah dijelaskan? Ketika pemikiran selesai, saatnya untuk bertindak.

2.6.2 Eksekusi model

Setelah memilih model, Anda harus menerapkannya dalam kode. Untungnya, sebagian besar bahasa pemrograman, seperti Python, sudah memiliki pustaka seperti StatsModels atau Scikit-learn. Paket-paket ini menggunakan beberapa teknik yang paling populer. Pengkodean model ad alah tugas yang tidak sepele dalam banyak kasus, sehingga ketersediaan pustaka ini dapat mempercepat prosesnya. Seperti yang Anda lihat pada kode berikut, cukup mudah untuk menggunakan regresi linier (gambar 2.22) dengan StatsModels atau Scikit-learn. Melakukan ini sendiri akan membutuhkan lebih banyak usaha bahkan untuk teknik sederhana. Daftar berikut menunjukkan pelaksanaan model prediksi linier.

Listing 2.1 Executing a linear prediction model on semi-random data

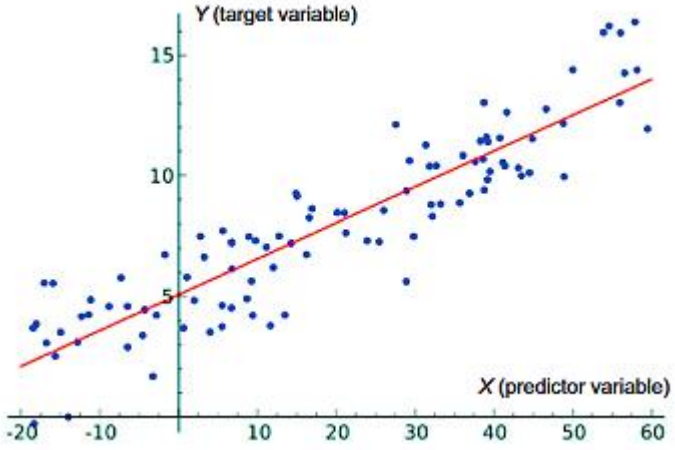
```
import statsmodels.api as sm
import numpy as np
predictors = np.random.random(1000).reshape(500,2)
target = predictors.dot(np.array([0.4, 0.6])) + np.random.random(500)
lmRegModel = sm.OLS(target,predictors)
result = lmRegModel.fit()
result.summary()
```

Imports required Python modules.

Creates random data for predictors (x-values) and semi-random data for the target (y-values) of the model. We use predictors as input to create the target so we infer a correlation here.

Fits linear regression on data.

Shows model fit statistics.



Gambar 2.22 Regresi linier mencoba menyesuaikan garis sambil meminimalkan jarak ke setiap titik

Dep. Variable:	y	R-squared:	0.893
Model:	OLS	Adj. R-squared:	0.893
Method:	Least Squares	F-statistic:	2088.
Date:	Fri, 30 Oct 2015	Prob (F-statistic):	7.13e-243
Time:	12:44:31	Log-Likelihood:	-176.74
No. Observations:	500	AIC:	357.5
Df Residuals:	498	BIC:	365.9
Df Model:	2		
Covariance Type:	nonrobust		

Model fit: higher is better but too high is suspicious.

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
x1	0.7658	0.040	19.130	0.000	0.687 0.844
x2	1.1252	0.039	28.603	0.000	1.048 1.202

p-value to show whether a predictor variable has a significant influence on the target. Lower is better and <0.05 is often considered "significant."

Omnibus:	34.269	Durbin-Watson:	1.943
Prob(Omnibus):	0.000	Jarque-Bera (JB):	13.480
Skew:	-0.125	Prob(JB):	0.00118
Kurtosis:	2.235	Cond. No.	2.51

Linear equation coefficients.
y = 0.7658xl + 1.1252x2.

Gambar 2.23 Keluaran informasi model regresi linier

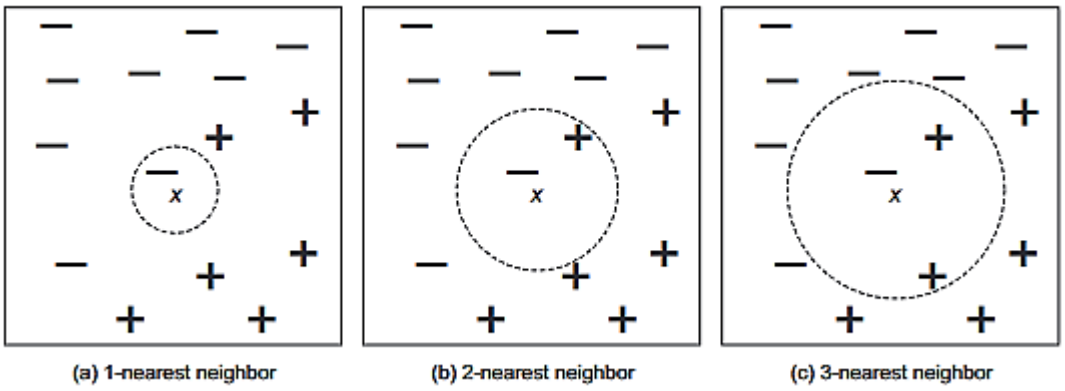
Dakam buku ini menciptakan nilai prediktor yang dimaksudkan untuk memprediksi bagaimana perilaku variabel target. Untuk regresi linier, sebuah "hubungan linier" antara masing-masing variabel x (prediktor) dan y (target) diasumsikan, seperti yang ditunjukkan pada gambar 2.22. Bagaimanapun, membuat variabel target, berdasarkan prediktor dengan menambahkan sedikit keacakan. Seharusnya tidak mengejutkan bahwa ini memberi kita model yang pas. results.summary() menampilkan tabel pada gambar 2.23. Pikiran Anda, hasil yang tepat tergantung pada variabel acak yang Anda dapatkan.

Mari abaikan sebagian besar keluaran yang kita dapatkan di sini dan fokus pada bagian terpenting:

- **Kecocokan model—**Untuk ini digunakan R-kuadrat atau R-kuadrat yang disesuaikan. Ukuran ini merupakan indikasi jumlah variasi data yang ditangkap oleh model. Perbedaan antara R-kuadrat yang disesuaikan dan R-kuadrat minimal di sini karena yang disesuaikan adalah yang normal + penalti untuk kompleksitas model. Sebuah model menjadi kompleks ketika banyak variabel (atau fitur) diperkenalkan. Anda tidak memerlukan model yang kompleks jika model sederhana tersedia, sehingga R-squared yang disesuaikan menghukum Anda karena terlalu rumit. Bagaimanapun, 0,893 tinggi, dan itu seharusnya karena kita curang. Ada aturan praktis, tetapi untuk model dalam bisnis, model di atas 0,85 sering dianggap baik. Jika Anda ingin memenangkan kompetisi yang Anda butuhkan di tahun 90-an. Namun untuk penelitian, sering ditemukan model yang sangat rendah (<0,2 bahkan). Yang lebih penting adalah pengaruh variabel prediktor yang diperkenalkan.
- **Variabel prediktor memiliki koefisien—**Untuk model linier, ini mudah untuk ditafsirkan. Dalam contoh kami jika Anda menambahkan "1" ke x1, itu akan mengubah y menjadi "0,7658". Sangat mudah untuk melihat bagaimana menemukan prediktor yang baik dapat menjadi jalan Anda menuju Hadiah Nobel meskipun model Anda secara keseluruhan adalah sampah. Jika, misalnya, Anda menentukan bahwa gen tertentu signifikan sebagai penyebab kanker, ini adalah pengetahuan penting, bahkan jika gen itu sendiri tidak menentukan apakah seseorang akan terkena kanker. Contoh di sini adalah klasifikasi, bukan regresi, tetapi intinya tetap sama: mendeteksi pengaruh lebih penting dalam studi ilmiah daripada model yang pas (belum lagi lebih realistis). Tapi kapan kita tahu gen memiliki dampak itu? Ini disebut signifikansi.
- **Signifikansi prediktor —** Koefisiennya besar, tetapi terkadang tidak cukup bukti untuk menunjukkan bahwa pengaruh itu ada. Inilah yang dimaksud dengan nilai-p. Penjelasan panjang tentang kesalahan tipe 1 dan tipe 2 dimungkinkan di sini tetapi penjelasan singkatnya adalah: jika nilai p lebih rendah dari 0,05, variabel tersebut dianggap signifikan bagi kebanyakan orang. Sebenarnya, ini adalah nomor arbitrer. Artinya ada 5% kemungkinan prediktor tidak memiliki pengaruh. Apakah Anda menerima peluang 5% ini untuk salah? Terserah Anda. Beberapa orang memperkenalkan ambang batas yang sangat signifikan (p<0,01) dan sedikit signifikan (p<0,1).

Regresi linier berfungsi jika Anda ingin memprediksi suatu nilai, tetapi bagaimana jika Anda ingin mengklasifikasikan sesuatu? Kemudian Anda pergi ke model klasifikasi, yang paling terkenal di antara mereka adalah k-tetangga terdekat. Seperti yang ditunjukkan pada gambar

2.24, k-nearest neighbor melihat titik berlabel di dekat titik tak berlabel dan, berdasarkan ini, membuat prediksi seperti apa seharusnya label itu.



Gambar 2.24 Teknik K-nearest neighbor melihat titik k-nearest untuk melakukan prediksi.

Mari kita coba dengan kode Python menggunakan pustaka belajar Scikit, seperti pada daftar berikut ini.

Listing 2.2 Executing k-nearest neighbor classification on semi-random data

```
from sklearn import neighbors
predictors = np.random.random(1000).reshape(500,2)
target = np.around(predictors.dot(np.array([0.4, 0.6])) +
         np.random.random(500))
clf = neighbors.KNeighborsClassifier(n_neighbors=10)
knn = clf.fit(predictors,target)
knn.score(predictors, target)
```

Imports modules.

Creates random predictor data and semi-random target data based on predictor data.

Fits 10-nearest neighbors model.

Gets model fit score: what percent of the classification was correct?

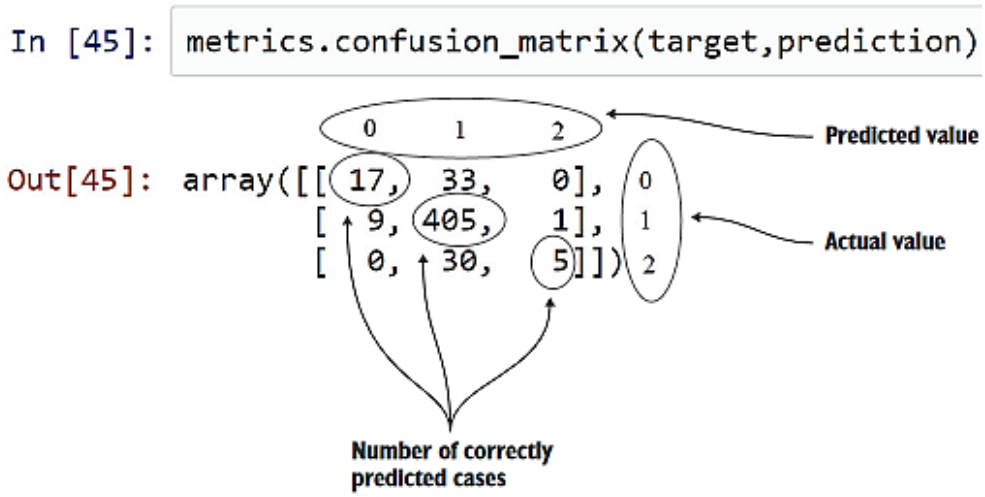
Seperti sebelumnya, kami membuat data berkorelasi acak dan mengejutkan, mengejutkan kami mendapatkan 85% kasus yang diklasifikasikan dengan benar. Jika kita ingin melihat secara mendalam, kita perlu menilai modelnya. Jangan biarkan knn.score() membodohi Anda; itu mengembalikan akurasi model, tetapi dengan "menskor model" kita sering bermaksud menerapkannya pada data untuk membuat prediksi.

```
prediction = knn.predict(predictors)
```

Sekarang kita bisa menggunakan prediksi dan membandingkannya dengan yang sebenarnya menggunakan matriks kebingungan.

```
metrics.confusion_matrix(target,prediction)
```

Kami mendapatkan matriks 3-kali-3 seperti yang ditunjukkan pada gambar 2.25.



Gambar 2.25 Confusion matrix: menunjukkan berapa banyak kasus yang diklasifikasikan dengan benar dan salah diklasifikasikan dengan membandingkan prediksi dengan nilai sebenarnya. Keterangan: kelas (0,1,2) ditambahkan pada gambar untuk klarifikasi.

Matriks kebingungan menunjukkan bahwa kami telah memprediksi dengan benar 17+405+5 kasus, jadi itu bagus. Tapi apakah ini benar-benar kejutan? Tidak, karena alasan berikut:

- Pertama, pengklasifikasi hanya memiliki tiga pilihan; menandai perbedaan dengan terakhir kali np.around() akan membulatkan data ke bilangan bulat terdekat. Dalam hal ini adalah 0, 1, atau 2. Dengan hanya 3 opsi, Anda tidak dapat melakukan lebih buruk dari 33% benar pada 500 tebakan, bahkan untuk distribusi acak nyata seperti melempar koin.
- Kedua, kami curang lagi, mengkorelasikan variabel respon dengan prediktor. Karena cara kami melakukan ini, kami mendapatkan sebagian besar pengamatan sebagai "1". Dengan menebak “1” untuk setiap kasus, kita sudah mendapatkan hasil yang serupa.
- Kami membandingkan prediksi dengan nilai sebenarnya, benar, tetapi kami tidak pernah memprediksi berdasarkan data baru. Prediksi dilakukan dengan menggunakan data yang sama dengan data yang digunakan untuk membangun model. Ini semua bagus dan keren untuk membuat diri Anda merasa nyaman, tetapi ini tidak memberi Anda indikasi apakah model Anda akan berfungsi saat menemukan data yang benar-benar baru. Untuk ini, kami memerlukan sampel penolakan, seperti yang akan dibahas di bagian selanjutnya.

Mengetik kode ini tidak akan menghasilkan keajaiban dengan sendirinya. Mungkin perlu beberapa saat untuk mendapatkan bagian pemodelan dan semua parameternya dengan benar. Hanya segelintir teknik yang siap diimplementasikan dalam Python. Tapi cukup mudah untuk menggunakan model yang tersedia di R dalam Python dengan bantuan pustaka RPy. RPy menyediakan antarmuka dari Python ke R.R adalah lingkungan perangkat lunak bebas, banyak digunakan untuk komputasi statistik. Jika Anda belum melakukannya, setidaknya patut untuk dilihat, karena pada tahun 2014 masih menjadi salah satu yang paling populer

2.6.3 Diagnosis model dan perbandingan model

Anda akan membangun beberapa model yang kemudian Anda pilih yang terbaik berdasarkan beberapa kriteria. Bekerja dengan sampel ketidakhadiran membantu Anda memilih model

berperforma terbaik. Sampel holdout adalah bagian dari data yang Anda keluarkan dari pembuatan model sehingga dapat digunakan untuk mengevaluasi model sesudahnya. Prinsipnya di sini sederhana: model harus bekerja pada data yang tidak terlihat. Anda hanya menggunakan sebagian kecil dari data Anda untuk memperkirakan model dan bagian lainnya, sampel ketidaksesuaian, dijauhkan dari persamaan. Model tersebut kemudian dilepaskan pada data yang tidak terlihat dan ukuran kesalahan dihitung untuk mengevaluasinya. Berbagai ukuran kesalahan tersedia, dan pada gambar 2.26 kami menunjukkan gagasan umum tentang membandingkan model. Ukuran kesalahan yang digunakan dalam contoh adalah mean square error.

$$\text{MSE} = \frac{1}{n}\sum_{i-1}^{n}\left(\hat{Y}_i - Y_i\right)^2$$

Gambar 2.26 Formula mean square error

Kesalahan kuadrat rata-rata adalah ukuran sederhana: periksa setiap prediksi seberapa jauh itu dari kebenaran, kuadratkan kesalahan ini, dan jumlahkan kesalahan setiap prediksi. Gambar 2.27 membandingkan kinerja dua model untuk memprediksi ukuran pesanan dari harga. Model pertama adalah ukuran = 3 * harga dan model kedua adalah ukuran = 10. Untuk memperkirakan model, kami menggunakan 800 pengamatan yang dipilih secara acak dari 1.000 (atau 80%), tanpa menunjukkan 20% data lainnya ke model. Setelah model dilatih, kami memprediksi nilai untuk 20% variabel lainnya berdasarkan nilai sebenarnya yang sudah kami ketahui, dan menghitung kesalahan model dengan ukuran kesalahan. Kemudian kami memilih model dengan kesalahan terendah. Pada contoh ini kami memilih model 1 karena memiliki total error terendah.

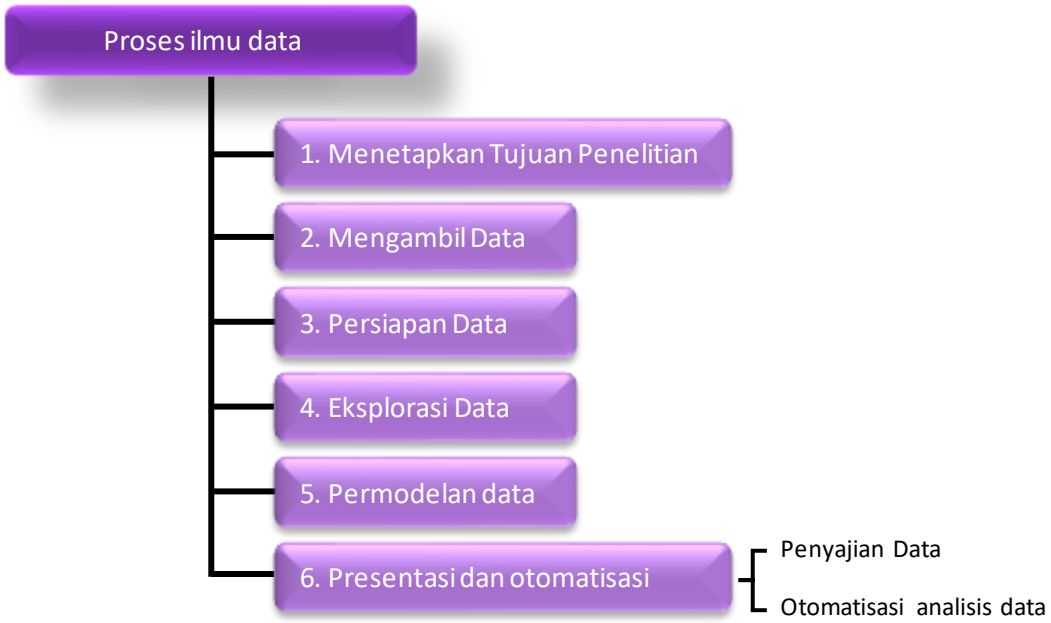
Banyak model membuat asumsi yang kuat, seperti independensi masukan, dan Anda harus memverifikasi bahwa asumsi ini benar-benar terpenuhi. Ini disebut diagnostik model. Bagian ini memberikan pengantar singkat tentang langkah-langkah yang diperlukan untuk membangun model yang valid. Setelah Anda memiliki model kerja, Anda siap untuk melanjutkan ke langkah terakhir.

	n	Ukuran	Harga	Prediksi Model 1	Prediksi Model 2	Model Error 1	Model Error 2
	1	10	3				
	2	15	5				
	3	18	6				
	4	14	5				
	...	...					
	800	9	3				
80% Pelatihan	801	12	4	12	10	0	2
	802	13	4	12	10	1	3
	...						
	999	21	7	21	10	0	11
20% Pengujian	1000	10	4	12	10	-2	0
				Total		5861	110225

Gambar 2.27 Sampel holdout membantu Anda membandingkan model dan memastikan bahwa Anda dapat menggeneralisasikan hasil ke data yang belum dilihat oleh model.

2.7 MENYAJIKAN TEMUAN DAN MEMBUAT APLIKASI

Setelah Anda berhasil menganalisis data dan membuat model yang berkinerja baik, Anda siap untuk mempresentasikan temuan Anda kepada dunia (gambar 2.28). Ini adalah bagian yang menarik; semua jam kerja keras Anda terbayar dan Anda dapat menjelaskan apa yang Anda temukan kepada para pemangku kepentingan.



Gambar 2.28 Langkah 6: Presentasi dan otomatisasi

Terkadang orang sangat bersemangat dengan pekerjaan Anda sehingga Anda perlu mengulanginya lagi dan lagi karena mereka menghargai prediksi model Anda atau wawasan yang Anda hasilkan. Untuk alasan ini, Anda perlu mengotomatiskan model Anda. Ini tidak selalu berarti bahwa Anda harus mengulang semua analisis Anda setiap saat. Terkadang cukup hanya menerapkan penilaian model; di lain waktu Anda mungkin membuat aplikasi yang secara otomatis memperbarui laporan, spreadsheet Excel, atau presentasi PowerPoint. Tahap terakhir dari proses ilmu data adalah di mana soft skill Anda akan sangat berguna, dan ya, itu sangat penting. Nyatanya, kami menyarankan Anda menemukan buku khusus dan informasi lain tentang subjek tersebut dan mengerjakannya, karena mengapa repot-repot melakukan semua kerja keras ini jika tidak ada yang mendengarkan apa yang Anda katakan?

Jika Anda melakukannya dengan benar, Anda sekarang memiliki model kerja dan pemangku kepentingan yang puas, jadi kita dapat menyimpulkan bab ini di sini.

2.8 RINGKASAN

Dalam bab ini Anda mempelajari proses ilmu data yang terdiri dari enam langkah:

- **Menetapkan tujuan penelitian**—Mendefinisikan apa, mengapa, dan bagaimana proyek Anda dalam piagam proyek.

- **Mengambil data—**Menemukan dan mendapatkan akses ke data yang diperlukan dalam proyek Anda. Data ini dapat ditemukan di dalam perusahaan atau diambil dari pihak ketiga.
- **Persiapan data—**Memeriksa dan memulihkan kesalahan data, memperkaya data dengan data dari sumber data lain, dan mengubahnya menjadi format yang sesuai untuk model Anda.
- **Eksplorasi data—**Mendalami data Anda menggunakan statistik deskriptif dan teknik visual.
- **Pemodelan data—**Menggunakan pembelajaran mesin dan teknik statistik untuk mencapai tujuan proyek Anda.
- **Presentasi dan otomatisasi—**Mempresentasikan hasil Anda kepada pemangku kepentingan dan mengindustrialisasikan proses analisis Anda untuk penggunaan ulang berulang dan integrasi dengan alat lain.

BAB 3
PEMBELAJARAN MESIN

Dalam bab ini, mahasiswa diharapkan mampu:

- Memahami mengapa ilmuwan data menggunakan pembelajaran mesin
- Mengidentifikasi pustaka Python terpenting untuk pembelajaran mesin
- Mendiskusikan proses pembuatan model
- Menggunakan teknik pembelajaran mesin
- Mendapatkan pengalaman langsung dengan pembelajaran mesin

Tahukah Anda bagaimana komputer belajar melindungi Anda dari orang jahat? Komputer menyaring lebih dari 60% email Anda dan dapat belajar untuk melakukan pekerjaan yang lebih baik dalam melindungi Anda dari waktu ke waktu. Bisakah Anda secara eksplisit mengajarkan komputer untuk mengenali orang dalam gambar? Mungkin tetapi tidak praktis untuk menyandikan semua cara yang mungkin untuk mengenali seseorang, tetapi Anda akan segera melihat bahwa kemungkinannya hampir tidak terbatas. Agar berhasil, Anda perlu menambahkan keterampilan baru ke perangkat Anda, pembelajaran mesin, yang merupakan topik bab ini.

3.1 APA ITU PEMBELAJARAN MESIN?

"Pembelajaran mesin adalah bidang studi yang memberi komputer kemampuan untuk belajar tanpa diprogram secara eksplisit."

—Arthur Samuel

Definisi pembelajaran mesin yang diciptakan oleh Arthur Samuel sering dikutip dan jenius dalam keluasannya, tetapi menimbulkan pertanyaan tentang bagaimana komputer belajar. Untuk mencapai pembelajaran mesin, para ahli mengembangkan algoritma tujuan umum yang dapat digunakan pada kelas besar masalah pembelajaran. Saat Anda ingin menyelesaikan tugas tertentu, Anda hanya perlu memberi algoritme data yang lebih spesifik. Di satu sisi, Anda memprogram dengan contoh. Dalam kebanyakan kasus komputer akan menggunakan data sebagai sumber informasi dan membandingkan keluarannya dengan keluaran yang diinginkan dan kemudian memperbaikinya. Semakin banyak data atau "pengalaman" yang didapat komputer, semakin baik pekerjaan yang ditunjuknya, seperti yang dilakukan manusia.

Ketika pembelajaran mesin dilihat sebagai sebuah proses, definisi berikut ini sangat berguna:

"Pembelajaran mesin adalah proses di mana komputer dapat bekerja lebih akurat saat mengumpulkan dan belajar dari data yang diberikan."

—Mike Roberts

Misalnya, saat pengguna menulis lebih banyak pesan teks di ponsel, ponsel akan belajar lebih banyak tentang kosa kata umum pesan tersebut dan dapat memprediksi (pelengkapan otomatis) kata-kata mereka dengan lebih cepat dan lebih akurat.

Dalam bidang sains yang lebih luas, pembelajaran mesin adalah subbidang kecerdasan buatan dan terkait erat dengan matematika terapan dan statistik. Semua ini mungkin terdengar agak abstrak, tetapi pembelajaran mesin memiliki banyak penerapan dalam kehidupan sehari-hari.

3.1.1 Aplikasi pembelajaran mesin dalam ilmu data

Regresi dan klasifikasi sangat penting bagi seorang ilmuwan data. Untuk mencapai tujuan tersebut, salah satu alat utama yang digunakan ilmuwan data adalah pembelajaran mesin. Penggunaan regresi dan klasifikasi otomatis sangat luas, seperti berikut ini:

- Menemukan ladang minyak, tambang emas, atau situs arkeologi berdasarkan situs yang ada (klasifikasi dan regresi)
- Menemukan nama tempat atau orang dalam teks (klasifikasi)
- Mengidentifikasi orang berdasarkan gambar atau rekaman suara (klasifikasi)
- Mengenali burung berdasarkan siulannya (klasifikasi)
- Mengidentifikasi pelanggan yang menguntungkan (regresi dan klasifikasi)
- Secara proaktif mengidentifikasi bagian-bagian mobil yang cenderung rusak (regresi)
- Mengidentifikasi tumor dan penyakit (klasifikasi)
- Memprediksi jumlah uang yang akan dikeluarkan seseorang untuk produk X (regresi)
- Memprediksi jumlah letusan gunung berapi dalam satu periode (regresi)
- Memprediksi pendapatan tahunan perusahaan Anda (regresi)
- Memprediksi tim mana yang akan memenangkan Liga Champions dalam sepak bola (klasifikasi)

Kadang-kadang ilmuwan data membangun model (abstraksi realitas) yang memberikan wawasan tentang proses yang mendasari suatu fenomena. Ketika tujuan model bukanlah prediksi tetapi interpretasi, itu disebut analisis akar penyebab. Berikut beberapa contohnya:

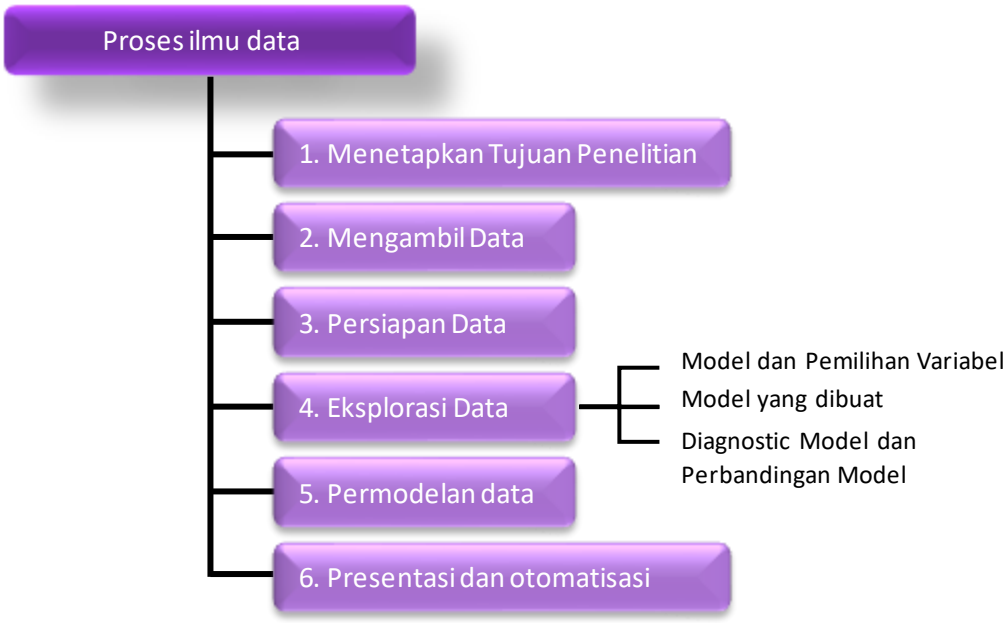
- Memahami dan mengoptimalkan proses bisnis, seperti menentukan produk mana yang menambah nilai pada lini produk
- Menemukan apa yang menyebabkan diabetes
- Menentukan penyebab kemacetan lalu lintas

Daftar aplikasi pembelajaran mesin ini hanya dapat dilihat sebagai hidangan pembuka karena ada di mana-mana dalam ilmu data. Regresi dan klasifikasi adalah dua teknik penting, tetapi repertoar dan aplikasinya tidak berakhir, dengan pengelompokan sebagai salah satu contoh lain dari teknik yang berharga. Teknik pembelajaran mesin dapat digunakan selama proses ilmu data, seperti yang akan kita bahas di bagian selanjutnya.

3.1.2 Di mana pembelajaran mesin digunakan dalam proses ilmu data

Meskipun pembelajaran mesin terutama terkait dengan langkah pemodelan data dari proses sains data, ini dapat digunakan di hampir setiap langkah. Untuk menyegarkan ingatan Anda dari bab-bab sebelumnya, proses data science ditunjukkan pada gambar 3.1.

Fase pemodelan data tidak dapat dimulai sampai Anda memiliki data mentah kualitatif yang dapat Anda pahami. Namun sebelumnya, fase persiapan data dapat memanfaatkan penggunaan pembelajaran mesin. Contohnya adalah membersihkan daftar string teks; pembelajaran mesin dapat mengelompokkan string serupa sehingga menjadi lebih mudah untuk memperbaiki kesalahan ejaan.

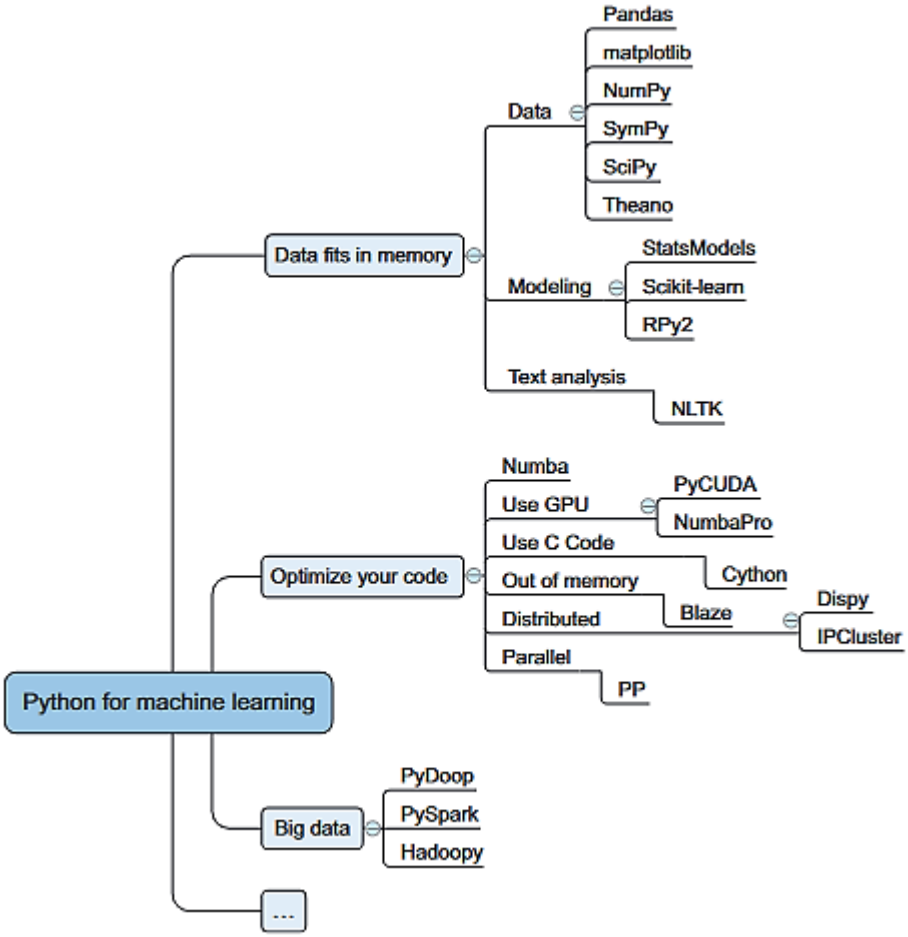


Gambar 3.1 Proses ilmu data

Pembelajaran mesin juga berguna saat menjelajahi data. Algoritma dapat membasmi pola dasar dalam data yang sulit ditemukan hanya dengan bagan. Mengingat bahwa pembelajaran mesin berguna selama proses ilmu data, tidak mengejutkan bahwa sejumlah besar pustaka Python dikembangkan untuk membuat hidup Anda sedikit lebih mudah.

3.1.3 Alat Python yang digunakan dalam pembelajaran mesin

Python memiliki banyak sekali paket yang dapat digunakan dalam pengaturan pembelajaran mesin. Ekosistem pembelajaran mesin Python dapat dibagi menjadi tiga jenis paket utama, seperti yang ditunjukkan pada gambar 3.2.



Gambar 3.2 Ringkasan paket Python yang digunakan selama fase pembelajaran mesin

Jenis paket pertama yang ditunjukkan pada gambar 3.2 terutama digunakan dalam tugas-tugas sederhana dan saat data masuk ke dalam memori. Jenis kedua digunakan untuk mengoptimalkan kode Anda ketika Anda telah menyelesaikan pembuatan prototipe dan mengalami masalah kecepatan atau memori. Tipe ketiga khusus untuk menggunakan Python dengan teknologi data besar.

Paket Untuk Bekerja Dengan Data Di Memory

Saat membuat prototipe, paket berikut dapat membantu Anda memulai dengan menyediakan fungsionalitas tingkat lanjut dengan beberapa baris kode:

- SciPy adalah perpustakaan yang mengintegrasikan paket dasar yang sering digunakan dalam komputasi ilmiah seperti NumPy, matplotlib, Pandas, dan SymPy.
- NumPy memberi Anda akses ke fungsi array yang kuat dan fungsi aljabar linier.
- Matplotlib adalah paket perencanaan 2D yang populer dengan beberapa fungsionalitas 3D.
- Panda adalah paket data-wrangling yang berkinerja tinggi, tetapi mudah digunakan. Ini memperkenalkan kerangka data ke Python, sejenis tabel data dalam memori. Ini adalah konsep yang terdengar akrab bagi pengguna reguler R.
- SymPy adalah paket yang digunakan untuk matematika simbolik dan aljabar komputer.
- StatsModels adalah paket untuk metode dan algoritme statistik.
- Scikit-learn adalah perpustakaan yang berisi algoritma pembelajaran mesin.

- RPy2 memungkinkan Anda memanggil fungsi R dari dalam Python. R adalah program statistik open source yang populer.
- NLTK (Natural Language Toolkit) adalah toolkit Python dengan fokus pada analitik teks.

Pustaka ini bagus untuk memulai, tetapi begitu Anda membuat keputusan untuk menjalankan program Python tertentu pada interval yang sering, kinerja akan berperan.

Mengoptimalkan Operasi

Setelah aplikasi Anda beralih ke produksi, pustaka yang tercantum di sini dapat membantu Anda memberikan kecepatan yang Anda perlukan. Terkadang hal ini melibatkan koneksi ke infrastruktur big data seperti Hadoop dan Spark.

- **Numba dan NumbaPro —** Ini menggunakan kompilasi just-in-time untuk mempercepat aplikasi yang ditulis langsung dengan Python dan beberapa anotasi. NumbaPro juga memungkinkan Anda menggunakan kekuatan unit prosesor grafis (GPU) Anda.
- **PyCUDA** —Ini memungkinkan Anda untuk menulis kode yang akan dijalankan pada GPU alih-alih CPU Anda dan karenanya ideal untuk aplikasi yang berat perhitungan. Ini bekerja paling baik dengan masalah yang cenderung diparalelkan dan membutuhkan sedikit input dibandingkan dengan jumlah siklus komputasi yang diperlukan. Contohnya adalah mempelajari kekokohan prediksi Anda dengan menghitung ribuan hasil berbeda berdasarkan satu status awal.
- **Cython, atau C untuk Python** — Ini membawa bahasa pemrograman C ke Python. C adalah bahasa tingkat rendah, jadi kodenya lebih dekat dengan apa yang akhirnya digunakan komputer (bytecode). Semakin dekat kode dengan bit dan byte, semakin cepat dijalankan. Komputer juga lebih cepat ketika mengetahui jenis variabel (disebut pengetikan statis). Python tidak dirancang untuk melakukan ini, dan Cython membantu Anda mengatasi kekurangan ini.
- **Blaze** —Blaze memberi Anda struktur data yang bisa lebih besar dari memori utama komputer Anda, memungkinkan Anda bekerja dengan kumpulan data besar.
- **Dispy dan IPCluster** —Paket ini memungkinkan Anda untuk menulis kode yang dapat didistribusikan melalui sekumpulan komputer.
- **PP** —Python dijalankan sebagai satu proses secara default. Dengan bantuan PP Anda dapat memparalelkan perhitungan pada satu mesin atau lebih dari cluster.
- **Pydoop dan Hadoopy**—Ini menghubungkan Python ke Hadoop, kerangka data besar yang umum.
- **PySpark**—Ini menghubungkan Python dan Spark, kerangka data besar dalam memori.

Sekarang setelah Anda melihat ikhtisar pustaka yang tersedia, mari kita lihat proses pemodelan itu sendiri.

3.2 PROSES PEMODELAN

Fase pemodelan terdiri dari empat langkah:

1. Rekayasa fitur dan pemilihan model
2. Melatih model
3. Validasi dan pemilihan model

4. Menerapkan model terlatih ke data yang tidak terlihat

Sebelum Anda menemukan model yang bagus, Anda mungkin akan mengulang di antara tiga langkah pertama.

Langkah terakhir tidak selalu ada karena terkadang tujuannya bukan prediksi tetapi penjelasan (analisis akar penyebab). Misalnya, Anda mungkin ingin mengetahui penyebab kepunahan spesies tetapi tidak perlu memprediksi spesies mana yang akan meninggalkan planet kita selanjutnya.

Dimungkinkan untuk merangkai atau menggabungkan beberapa teknik. Saat Anda merangkai beberapa model, output dari model pertama menjadi input untuk model kedua. Saat Anda menggabungkan beberapa model, Anda melatihnya secara mandiri dan menggabungkan hasilnya. Teknik terakhir ini juga dikenal sebagai pembelajaran ensemble.

Sebuah model terdiri dari konstruksi informasi yang disebut fitur atau prediktor dan target atau variabel respon. Sasaran model Anda adalah memprediksi variabel target, misalnya suhu tinggi besok. Variabel yang membantu Anda melakukan ini dan (biasanya) diketahui oleh Anda adalah fitur atau variabel prediktor seperti suhu hari ini, pergerakan awan, kecepatan angin saat ini, dan sebagainya. Model terbaik adalah model yang secara akurat mewakili kenyataan, sebaiknya tetap ringkas dan dapat ditafsirkan. Untuk mencapai hal ini, rekayasa fitur adalah bagian pemodelan yang paling penting dan paling menarik. Misalnya, ciri penting dalam model yang mencoba menjelaskan kepunahan hewan darat besar dalam 60.000 tahun terakhir di Australia ternyata adalah jumlah populasi dan persebaran manusia.

3.2.1 Fitur rekayasa dan pemilihan model

Dengan fitur teknik, Anda harus membuat dan membuat prediktor yang memungkinkan untuk model tersebut. Ini adalah salah satu langkah terpenting dalam proses karena model menggabungkan kembali fitur-fitur ini untuk mencapai prediksinya. Seringkali Anda mungkin perlu berkonsultasi dengan seorang ahli atau literatur yang sesuai untuk menghasilkan fitur yang bermakna.

Fitur tertentu adalah variabel yang Anda dapatkan dari kumpulan data, seperti halnya dengan kumpulan data yang disediakan dalam latihan kami dan di sebagian besar latihan sekolah. Dalam praktiknya, Anda harus menemukan fiturnya sendiri, yang mungkin tersebar di kumpulan data yang berbeda. Dalam beberapa proyek kami harus mengumpulkan lebih dari 20 sumber data yang berbeda sebelum kami memiliki data mentah yang kami butuhkan. Sering kali Anda harus menerapkan transformasi ke input sebelum menjadi prediktor yang baik atau menggabungkan beberapa input. Contoh menggabungkan beberapa input adalah variabel interaksi: dampak dari salah satu variabel rendah, tetapi jika keduanya ada, dampaknya menjadi sangat besar. Hal ini terutama benar dalam lingkungan kimia dan medis. Misalnya, meskipun cuka dan pemutih sendiri merupakan produk rumah tangga biasa yang tidak berbahaya, mencampurkannya menghasilkan gas klorin yang beracun, gas yang membunuh ribuan orang selama Perang Dunia I.

Dalam kedokteran, farmasi klinis adalah disiplin yang didedikasikan untuk meneliti efek interaksi obat-obatan. Ini adalah pekerjaan yang penting, dan bahkan tidak harus melibatkan dua obat untuk menghasilkan hasil yang berpotensi berbahaya. Misalnya,

mencampurkan obat antijamur seperti Sporanox dengan jeruk bali memiliki efek samping yang serius.

Kadang-kadang Anda harus menggunakan teknik pemodelan untuk mendapatkan fitur: keluaran model menjadi bagian dari model lain. Ini tidak biasa, terutama dalam penambangan teks. Dokumen dapat diberi anotasi terlebih dahulu untuk mengklasifikasikan konten ke dalam kategori, atau Anda dapat menghitung jumlah tempat geografis atau orang dalam teks. Penghitungan ini seringkali lebih sulit daripada kedengarannya; model pertama kali diterapkan untuk mengenali kata-kata tertentu sebagai orang atau tempat. Semua informasi baru ini kemudian dituangkan ke dalam model yang ingin Anda bangun. Salah satu kesalahan terbesar dalam konstruksi model adalah bias ketersediaan: fitur Anda hanyalah fitur yang dapat Anda peroleh dengan mudah dan model Anda secara konsekuen mewakili "kebenaran" sepihak ini. Model yang menderita bias ketersediaan sering gagal saat divalidasi karena menjadi jelas bahwa model tersebut bukan representasi kebenaran yang valid.

Dalam Perang Dunia II, setelah pengeboman terjadi di wilayah Jerman, banyak pesawat Inggris kembali dengan lubang peluru di sayap, di sekitar hidung, dan di dekat ekor pesawat. Hampir tidak ada dari mereka yang memiliki lubang peluru di kokpit, kemudi ekor, atau blok mesin, jadi teknisi memutuskan pelapisan lapis baja tambahan harus ditambahkan ke sayap. Ini tampak seperti ide yang bagus sampai seorang matematikawan bernama Abraham Wald menjelaskan kesalahan mereka yang jelas: mereka hanya memperhitungkan pesawat yang kembali. Lubang peluru di sayap sebenarnya tidak menjadi perhatian mereka, karena setidaknya pesawat dengan kerusakan seperti ini bisa pulang untuk diperbaiki. Fortifikasi pesawat karenanya ditingkatkan di tempat-tempat yang tidak terluka saat pesawat kembali. Alasan awal mengalami bias ketersediaan: para insinyur mengabaikan bagian penting dari data karena lebih sulit diperoleh. Dalam hal ini mereka beruntung, karena penalarannya bisa dibalik untuk mendapatkan hasil yang diinginkan tanpa mendapatkan data dari pesawat yang jatuh. Saat fitur awal dibuat, model dapat dilatih ke data.

3.2.2 Melatih model Anda

Dengan prediktor yang tepat dan teknik pemodelan, Anda dapat melanjutkan ke pelatihan model. Pada fase ini Anda mempresentasikan data model Anda yang dapat dipelajari.

Teknik pemodelan yang paling umum memiliki implementasi siap industri di hampir setiap bahasa pemrograman, termasuk Python. Ini memungkinkan Anda untuk melatih model Anda dengan menjalankan beberapa baris kode. Untuk teknik ilmu data yang lebih canggih, Anda mungkin akan melakukan perhitungan matematis yang berat dan menerapkannya dengan teknik ilmu komputer modern. Setelah model dilatih, saatnya untuk menguji apakah model tersebut dapat diekstrapolasi menjadi kenyataan: validasi model.

3.2.3 Memvalidasi model

Ilmu data memiliki banyak teknik pemodelan, dan pertanyaannya adalah mana yang tepat untuk digunakan. Model yang baik memiliki dua sifat: ia memiliki daya prediksi yang baik dan menggeneralisasi dengan baik data yang belum dilihatnya. Untuk mencapai ini, Anda menentukan ukuran kesalahan (seberapa salah modelnya) dan strategi validasi.

Dua ukuran kesalahan umum dalam pembelajaran mesin adalah tingkat kesalahan klasifikasi untuk masalah klasifikasi dan kesalahan kuadrat rata-rata untuk masalah regresi. Tingkat kesalahan klasifikasi adalah persentase pengamatan dalam kumpulan data uji yang diberi label salah oleh model Anda; lebih rendah lebih baik. Kesalahan kuadrat rata-rata mengukur seberapa besar kesalahan rata-rata prediksi Anda. Mengkuadratkan kesalahan rata-rata memiliki dua konsekuensi: Anda tidak dapat membatalkan prediksi yang salah di satu arah dengan prediksi yang salah di arah yang lain. Misalnya, melebih-lebihkan omset masa depan untuk bulan depan sebesar 5.000 tidak membatalkan perkiraan yang terlalu rendah sebesar 5.000 untuk bulan berikutnya. Sebagai konsekuensi kedua dari pengkuadratan, kesalahan yang lebih besar mendapatkan bobot yang lebih besar daripada yang seharusnya. Kesalahan kecil tetap kecil atau bahkan bisa menyusut (jika <1), sedangkan kesalahan besar diperbesar dan pasti akan menarik perhatian Anda.

Ada banyak strategi validasi, termasuk yang umum berikut ini:

- Membagi data Anda ke dalam kumpulan pelatihan dengan X% dari pengamatan dan menyimpan sisanya sebagai kumpulan data penyimpanan (kumpulan data yang tidak pernah digunakan untuk pembuatan model)—Ini adalah teknik yang paling umum.
- Validasi silang K-folds—Strategi ini membagi kumpulan data menjadi k bagian dan menggunakan setiap bagian satu kali sebagai kumpulan data pengujian sambil menggunakan yang lain sebagai kumpulan data pelatihan. Ini memiliki keuntungan bahwa Anda menggunakan semua data yang tersedia di kumpulan data.
- Abaikan-1 —Pendekatan ini sama dengan lipatan-k tetapi dengan k=1. Anda selalu meninggalkan satu pengamatan dan melatih data lainnya. Ini hanya digunakan pada kumpulan data kecil, jadi lebih berharga bagi orang yang mengevaluasi eksperimen laboratorium daripada analis data besar.

Istilah populer lainnya dalam pembelajaran mesin adalah regularisasi. Saat menerapkan regularisasi, Anda dikenai penalti untuk setiap variabel tambahan yang digunakan untuk membangun model. Dengan regularisasi L1, Anda meminta model dengan prediktor sesedikit mungkin. Ini penting untuk kekokohan model: solusi sederhana cenderung berlaku dalam lebih banyak situasi. Regularisasi L2 bertujuan untuk menjaga varian antara koefisien prediktor sekecil mungkin. Varian yang tumpang tindih antara prediktor membuat sulit untuk mengetahui dampak sebenarnya dari masing-masing prediktor. Menjaga varians mereka dari tumpang tindih akan meningkatkan interpretabilitas. Sederhananya: regularisasi terutama digunakan untuk menghentikan model menggunakan terlalu banyak fitur dan dengan demikian mencegah pemasangan yang berlebihan.

Validasi sangat penting karena menentukan apakah model Anda berfungsi dalam kondisi kehidupan nyata. Terus terang, apakah model Anda bernilai sepeser pun. Meski begitu, kadang-kadang orang mengirim makalah ke jurnal ilmiah terkemuka (dan kadang-kadang bahkan berhasil menerbitkannya) dengan validasi yang salah. Akibatnya mereka ditolak atau perlu mencabut kertas karena semuanya salah. Situasi seperti ini berdampak buruk bagi kesehatan mental Anda, jadi ingatlah selalu hal ini: uji model Anda pada data yang belum pernah dilihat oleh model yang dibangun dan pastikan data ini adalah representasi sebenarnya dari apa yang akan ditemuinya ketika diterapkan pada pengamatan baru oleh

orang lain. Untuk model klasifikasi, instrumen seperti matriks kebingungan (diperkenalkan pada bab 2 tetapi dijelaskan secara menyeluruh nanti pada bab ini) adalah emas; merangkul mereka. Setelah Anda membuat model yang bagus, Anda dapat (opsional) menggunakannya untuk memprediksi masa depan.

3.2.4 Memprediksi pengamatan baru

Jika Anda telah berhasil menerapkan tiga langkah pertama, Anda sekarang memiliki model performa yang digeneralisasikan ke data yang tidak terlihat. Proses menerapkan model Anda ke data baru disebut penilaian model. Faktanya, penilaian model adalah sesuatu yang Anda lakukan secara implisit selama validasi, hanya sekarang Anda tidak mengetahui hasil yang benar. Sekarang Anda harus cukup memercayai model Anda untuk menggunakannya secara nyata.

Penilaian model melibatkan dua langkah. Pertama, Anda menyiapkan kumpulan data yang memiliki fitur persis seperti yang ditentukan oleh model Anda. Ini bermuara pada pengulangan persiapan data yang Anda lakukan pada langkah pertama proses pemodelan tetapi untuk kumpulan data baru. Kemudian Anda menerapkan model pada kumpulan data baru ini, dan ini menghasilkan prediksi. Sekarang mari kita lihat berbagai jenis teknik pembelajaran mesin: masalah yang berbeda memerlukan pendekatan yang berbeda.

3.3 JENIS PEMBELAJARAN MESIN

Secara umum, kita dapat membagi berbagai pendekatan untuk pembelajaran mesin dengan jumlah upaya manusia yang diperlukan untuk mengoordinasikannya dan cara mereka menggunakan data berlabel—data dengan kategori atau nomor nilai nyata yang ditetapkan padanya yang mewakili hasil pengamatan sebelumnya.

- Teknik pembelajaran terawasi mencoba membedakan hasil dan belajar dengan mencoba menemukan pola dalam kumpulan data berlabel. Interaksi manusia diperlukan untuk melabeli data.
- Teknik pembelajaran tanpa pengawasan tidak bergantung pada data berlabel dan berupaya menemukan pola dalam kumpulan data tanpa interaksi manusia.
- Teknik pembelajaran semi-diawasi memerlukan data berlabel, dan oleh karena itu interaksi manusia, untuk menemukan pola dalam kumpulan data, tetapi teknik tersebut masih dapat berkembang menuju hasil dan belajar meskipun melewati data yang tidak berlabel juga.

Di bagian ini, kita akan melihat ketiga pendekatan, melihat tugas apa yang lebih sesuai untuk masing-masing, dan menggunakan satu atau dua pustaka Python yang disebutkan sebelumnya untuk memberi Anda gambaran tentang kode dan menyelesaikan tugas. Dalam setiap contoh ini, kami akan bekerja dengan kumpulan data yang dapat diunduh yang telah dibersihkan, jadi kami akan langsung melompat ke langkah pemodelan data dari proses ilmu data, seperti yang dibahas sebelumnya di bab ini.

3.3.1 Pembelajaran yang diawasi

Seperti disebutkan sebelumnya, pembelajaran terbimbing adalah teknik pembelajaran yang hanya dapat diterapkan pada data berlabel. Contoh penerapannya adalah membedakan angka dari gambar. Mari selami studi kasus tentang pengenalan angka.

Studi Kasus: Membeda Digit Dari Gambar

Salah satu dari banyak pendekatan umum di web untuk menghentikan komputer agar tidak meretas akun pengguna adalah pemeriksaan Captcha—gambar teks dan angka yang harus diuraikan oleh pengguna manusia dan dimasukkan ke dalam bidang formulir sebelum mengirim formulir kembali ke web server. Sesuatu seperti gambar 3.3 seharusnya terlihat familier.

Gambar 3.3 Kontrol Captcha sederhana dapat digunakan untuk mencegah spam otomatis dikirim melalui formulir web online.

Dengan bantuan pengklasifikasi Naïve Bayes, algoritme sederhana namun kuat untuk mengkategorikan pengamatan ke dalam kelas yang dijelaskan lebih detail di sidebar, Anda dapat mengenali digit dari gambar tekstual. Gambar-gambar ini tidak berbeda dengan pemeriksaan Captcha yang dilakukan banyak situs web untuk memastikan Anda bukan komputer yang mencoba meretas akun pengguna. Mari kita lihat betapa sulitnya membiarkan komputer mengenali gambar angka. Tujuan penelitian kami adalah membiarkan komputer mengenali gambar angka (langkah pertama dari proses ilmu data). Data yang akan kami kerjakan adalah kumpulan data MNIST, yang sering digunakan dalam literatur sains data untuk pengajaran dan pembandingan.

Memperkenalkan pengklasifikasi Naïve Bayes dalam konteks filter spam

Tidak setiap email yang Anda terima memiliki niat jujur. Kotak masuk Anda dapat berisi email komersial atau massal yang tidak diminta, alias spam. Tidak hanya mengganggu, spam juga sering digunakan dalam penipuan dan sebagai pembawa virus. Kaspersky3 memperkirakan bahwa lebih dari 60% email di dunia adalah spam. Untuk melindungi pengguna dari spam, sebagian besar klien email menjalankan program di latar belakang yang mengklasifikasikan email sebagai spam atau aman.

Teknik populer dalam pemfilteran spam menggunakan pengklasifikasi yang menggunakan kata-kata di dalam email sebagai prediktor. Ini menampilkan kemungkinan bahwa email tertentu adalah spam mengingat kata-kata yang menyusunnya (dalam istilah matematika, P(spam|kata-kata)). Untuk mencapai kesimpulan ini menggunakan tiga perhitungan:

- P(spam)—Tingkat rata-rata spam tanpa mengetahui kata-katanya. Menurut Kaspersky, 60% email adalah spam.
- P(kata)—Seberapa sering kombinasi kata ini digunakan terlepas dari spam.

- P(words | spam)—Seberapa sering kata-kata ini terlihat saat email pelatihan diberi label sebagai spam.

Untuk menentukan kemungkinan email baru adalah spam, Anda akan menggunakan rumus berikut: P(spam|words) = P(spam)P(words|spam) / P(words)

Ini adalah penerapan aturan P(B|A) = P(B) P(A|B) / P(A), yang dikenal sebagai aturan Bayes dan yang meminjamkan namanya ke classifier ini. Bagian "naif" berasal dari asumsi pengklasifikasi bahwa keberadaan satu fitur tidak memberi tahu Anda apa pun tentang fitur lain (independensi fitur, juga disebut tidak adanya multikolinearitas). Pada kenyataannya, fitur seringkali terkait, terutama dalam teks. Misalnya kata “beli” akan sering diikuti dengan “sekarang”. Terlepas dari asumsi yang tidak realistis, pengklasifikasi naif bekerja dengan sangat baik dalam praktiknya.

Dengan sedikit teori di sidebar, Anda siap melakukan pemodelan itu sendiri. Pastikan untuk menjalankan semua kode yang akan datang dalam lingkup yang sama karena setiap bagian membutuhkan kode sebelumnya. File IPython dapat diunduh untuk bab ini dari halaman unduhan Manning buku ini. Gambar MNIST dapat ditemukan di paket kumpulan data Scikit-learn dan sudah dinormalisasi untuk Anda (semua diskalakan ke ukuran yang sama: 64x64 piksel), jadi kami tidak memerlukan banyak persiapan data (langkah ketiga dari proses ilmu data). Tapi pertama-tama mari kita ambil data kita sebagai langkah kedua dari proses ilmu data, dengan daftar berikut.

Listing 3.1 Step 2 of the data science process: fetching the digital image data

```
from sklearn.datasets import load_digits
import pylab as pl
digits = load_digits()
```

Imports digits database.

Loads digits.

Bekerja dengan gambar tidak jauh berbeda dengan bekerja dengan kumpulan data lainnya. Dalam kasus gambar abu-abu, Anda memberi nilai di setiap entri matriks yang menggambarkan nilai abu-abu yang akan ditampilkan. Kode berikut menunjukkan proses ini dan merupakan langkah keempat dari proses ilmu data: eksplorasi data.

Listing 3.2 Step 4 of the data science process: using Scikit-learn

```
pl.gray()
pl.matshow(digits.images[0])
pl.show()
digits.images[0]
```

Turns image into gray-scale values.

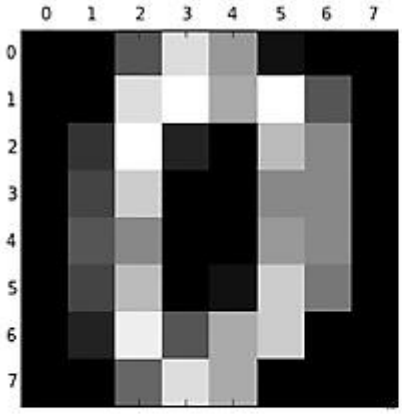
Shows first images.

Shows the corresponding matrix.

Gambar 3.4 menunjukkan bagaimana gambar “0” buram diterjemahkan menjadi matriks data. Gambar 3.4 menunjukkan keluaran kode yang sebenarnya, tetapi mungkin gambar 3.5 dapat sedikit memperjelas hal ini, karena gambar ini menunjukkan bagaimana

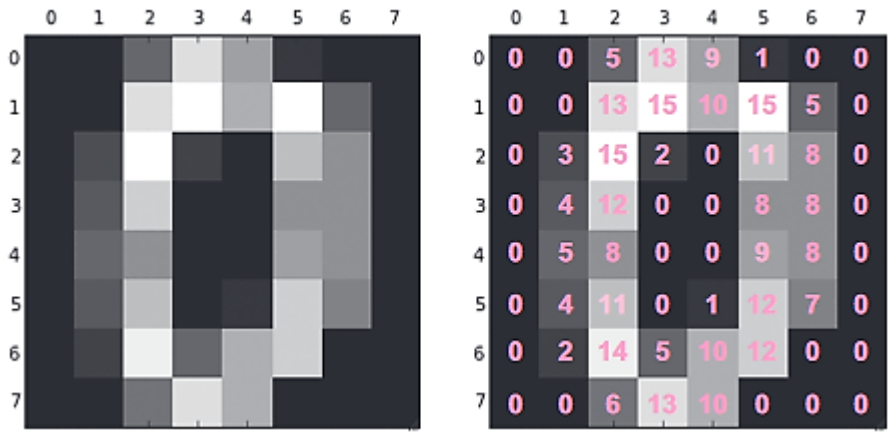
setiap elemen dalam vektor adalah bagian dari gambar. Sejauh ini mudah, bukan? Ada, tentu saja, sedikit lebih banyak pekerjaan yang harus dilakukan. Pengklasifikasi Naïve Bayes mengharapkan daftar nilai, tetapi pl.matshow() mengembalikan larik dua dimensi (matriks) yang mencerminkan bentuk gambar. Untuk meratakannya ke dalam daftar, kita perlu memanggil reshape() pada digits.images. Hasil bersihnya akan menjadi array satu dimensi yang terlihat seperti ini:

```
array([[ 0.,  0.,  5.,  13.,  9.,  1.,  0.,  0.,  0.,  0.,  13.,  15.,  10.,  15.,  5.,  0.,
0.,  3.,  15.,  2.,  0.,  11.,  8.,  0.,  0.,  4.,  12.,  0.,  0.,  8.,  8.,  0.,
0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.,  0.,  4.,  11.,  0.,  1.,  12.,  7.,  0.,
0.,  2.,  14.,  5.,  10.,  12.,  0.,  0.,  0.,  0.,  6.,  13.,  10.,  0.,  0.,  0.]])
```



```
rray([[  0.,   0.,   5.,  13.,   9.,   1.,   0.,   0.],
       [  0.,   0.,  13.,  15.,  10.,  15.,   5.,   0.],
       [  0.,   3.,  15.,   2.,   0.,  11.,   8.,   0.],
       [  0.,   4.,  12.,   0.,   0.,   8.,   8.,   0.],
       [  0.,   5.,   8.,   0.,   0.,   9.,   8.,   0.],
       [  0.,   4.,  11.,   0.,   1.,  12.,   7.,   0.],
       [  0.,   2.,  14.,   5.,  10.,  12.,   0.,   0.],
       [  0.,   0.,   6.,  13.,  10.,   0.,   0.,   0.]])
```

Gambar 3.4 Representasi abu-abu kabur dari angka 0 dengan matriks yang sesuai. Semakin tinggi angkanya, semakin mendekati putih; semakin rendah angkanya, semakin dekat ke hitam.



Gambar 3.5 Kami akan mengubah gambar menjadi sesuatu yang dapat digunakan oleh pengklasifikasi Naïve Bayes dengan mendapatkan nilai skala abu-abu untuk setiap pikselnya (ditampilkan di sebelah kanan) dan memasukkan nilai tersebut ke dalam daftar.

Cuplikan kode sebelumnya menunjukkan matriks gambar 3.5 diratakan (jumlah dimensi dikurangi dari dua menjadi satu) ke daftar Python. Mulai saat ini, ini adalah masalah klasifikasi standar, yang membawa kita ke langkah kelima dari proses ilmu data: pembuatan model. Sekarang kita memiliki cara untuk meneruskan konten gambar ke dalam pengklasifikasi, kita perlu mengirimkannya kumpulan data pelatihan sehingga dapat mulai mempelajari cara memprediksi angka dalam gambar. Kami sebutkan sebelumnya bahwa Scikit-learn berisi subset dari database MNIST (1.800 gambar), jadi kami akan menggunakannya. Setiap gambar juga diberi label dengan nomor yang sebenarnya ditampilkan. Ini akan membangun model probabilistik dalam memori dari digit yang paling mungkin ditampilkan dalam gambar dengan nilai skala abu-abunya. Setelah program melewati set pelatihan dan membangun model, kami kemudian dapat memberikannya set data uji untuk melihat seberapa baik program telah belajar menginterpretasikan gambar menggunakan model. Daftar berikut menunjukkan bagaimana menerapkan langkah-langkah ini dalam kode.

Listing 3.3 Image data classification problem on images of digits

```
from sklearn.cross_validation import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
import pylab as plt

y = digits.target

n_samples = len(digits.images)
X= digits.images.reshape((n_samples, -1))

print X

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

gnb = GaussianNB()
fit = gnb.fit(X_train,y_train)
predicted = fit.predict(X_test)
confusion_matrix(y_test, predicted)
```

Step 1: Select target variable.

Step 2: Prepare data. Reshape adapts the matrix form. This method could, for instance, turn a 10x10 matrix into 100 vectors.

Step 3: Split into test set and training set.

Step 4: Select a Naïve Bayes classifier; use a Gaussian distribution to estimate probability.

Step 5: Fit data.

Step 6: Predict data for unseen data.

Step 7: Create confusion matrix.

Hasil akhir dari kode ini disebut matriks konfusi, seperti yang ditunjukkan pada gambar 3.6. Dikembalikan sebagai larik dua dimensi, ini menunjukkan seberapa sering angka yang diprediksi adalah angka yang benar pada diagonal utama dan juga di entri matriks (i,j), di mana j diprediksi tetapi gambar menunjukkan i. Melihat gambar 3.6 kita dapat melihat bahwa model

memprediksi angka 2 dengan benar sebanyak 17 kali (pada koordinat 3,3), tetapi model juga memprediksi angka 8 sebanyak 15 kali padahal sebenarnya angka 2 pada gambar (pada 9, 3).

```
array([[37,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 39,  0,  0,  0,  0,  1,  0,  3,  0],
       [ 0,  9, 17,  3,  0,  0,  0,  0, 15,  0],
       [ 0,  0,  0, 38,  0,  0,  0,  2,  5,  0],
       [ 0,  1,  0,  0, 27,  0,  2,  8,  0,  0],
       [ 0,  1,  0,  1,  0, 43,  0,  3,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 52,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0, 47,  0,  0],
       [ 0,  5,  0,  1,  0,  1,  0,  4, 37,  0],
       [ 0,  2,  0,  7,  1,  0,  0,  3,  7, 27]])
```

Gambar 3.6 Confusion matrix dihasilkan dengan memprediksi angka apa yang digambarkan oleh gambar buram

Matriks konfusi

Matriks konfusi adalah matriks yang menunjukkan seberapa salah (atau benar) prediksi model, seberapa banyak model tersebut menjadi "bingung". Dalam bentuknya yang paling sederhana, ini akan menjadi tabel 2x2 untuk model yang mencoba mengklasifikasikan pengamatan sebagai A atau B. Katakanlah kita memiliki model klasifikasi yang memprediksi apakah seseorang akan membeli produk terbaru kita: puding ceri goreng. Kami dapat memprediksi: "Ya, orang ini akan membeli" atau "Tidak, pelanggan ini tidak akan membeli". Setelah kami membuat prediksi untuk 100 orang, kami dapat membandingkannya dengan perilaku mereka yang sebenarnya, menunjukkan kepada kami berapa kali kami melakukannya dengan benar. Contohnya ditunjukkan pada tabel 3.1.

Tabel 3.1 Contoh matriks konfusi

Matriks konfusi	Prediksi "Orang akan membeli"	Prediksi "Orang tidak akan membeli"
Seseorang membeli puding ceri goreng	35 (benar-benar positif)	10 (negatif palsu)
Orang tidak membeli puding ceri goreng	15 (positif palsu)	40 (negatif sebenarnya)

Model benar dalam (35+40) 75 kasus dan salah dalam (15+10) 25 kasus, menghasilkan akurasi 75% (75 benar/100 total pengamatan).

Semua pengamatan yang diklasifikasikan dengan benar dijumlahkan pada diagonal (35+40) sementara yang lainnya (15+10) salah diklasifikasikan. Ketika model hanya memprediksi dua kelas (biner), tebakan kita yang benar adalah dua kelompok: benar positif (diprediksi untuk membeli dan melakukannya) dan benar negatif (memprediksi mereka tidak akan membeli dan tidak). Tebakan kami yang salah dibagi menjadi dua kelompok: positif palsu (diprediksi mereka akan membeli tetapi ternyata tidak) dan negatif palsu (diprediksi tidak membeli tetapi mereka melakukannya). Matriks berguna untuk melihat model mana yang

paling bermasalah. Dalam hal ini kita cenderung terlalu percaya diri dengan produk kita dan terlalu mudah mengklasifikasikan pelanggan sebagai calon pembeli (false positive).

Dari matriks konfusi, kita dapat menyimpulkan bahwa untuk sebagian besar gambar, prediksinya cukup akurat. Dalam model yang baik, Anda akan mengharapkan jumlah angka pada diagonal utama matriks (juga dikenal sebagai jejak matriks) menjadi sangat tinggi dibandingkan dengan jumlah semua entri matriks, yang menunjukkan bahwa prediksi paling benar untuk sebagian besar bagian.

Anggaplah kita ingin memamerkan hasil kita dengan cara yang lebih mudah dimengerti atau kita ingin memeriksa beberapa gambar dan prediksi yang telah dibuat oleh program kita: kita dapat menggunakan kode berikut untuk menampilkan satu di samping yang lain. Kemudian kita dapat melihat di mana program tersebut salah dan membutuhkan lebih banyak pelatihan. Jika kita puas dengan hasilnya, pembuatan model berakhir di sini dan kita sampai pada langkah enam: menyajikan hasilnya.

Listing 3.4 Inspecting predictions vs actual numbers

```
images_and_predictions = list(zip(digits.images, fit.predict(X)))
for index, (image, prediction) in enumerate(images_and_predictions[:6]):
    plt.subplot(6, 3 ,index + 5)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    plt.title('Prediction: %i' % prediction)
plt.show()
```

Adds an extra subplot on a 6x3 plot grid. This code could be simplified as: plt.subplot (3, 2 ,index) but this looks visually more appealing.

Stores number image matrix and its prediction (as a number) together in array.

Loops through first 7 images.

Doesn't show an axis.

Shows image in grayscale.

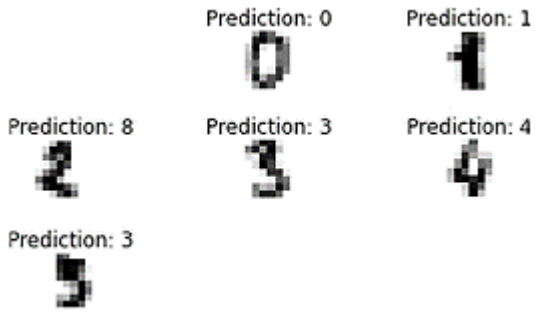
Shows the predicted value as the title to the shown image.

Shows the full plot that is now populated with 6 subplots.

Gambar 3.7 menunjukkan bagaimana semua prediksi tampaknya benar kecuali untuk digit angka 2, yang diberi label sebagai 8. Kita harus memaafkan kesalahan ini karena angka 2 ini memiliki kesamaan visual dengan 8. Angka kiri bawah bersifat ambigu, bahkan untuk manusia; ini 5 atau 3? Ini masih bisa diperdebatkan, tetapi algoritme menganggapnya 3.

Dengan membedakan gambar mana yang disalahartikan, kita dapat melatih model lebih lanjut dengan memberi label dengan nomor yang benar yang ditampilkan dan memasukkannya kembali ke dalam model sebagai set pelatihan baru (langkah 5 dari proses ilmu data). Ini akan membuat model lebih akurat, sehingga siklus belajar, memprediksi, mengoreksi terus berlanjut dan prediksi menjadi lebih akurat. Ini adalah kumpulan data terkontrol yang kami gunakan sebagai contoh. Semua contoh memiliki ukuran yang sama dan semuanya dalam 16 warna abu-abu. Perluas hingga gambar ukuran variabel string panjang variabel warna variabel karakter alfanumerik yang ditampilkan di kontrol Captcha, dan Anda dapat menghargai mengapa model yang cukup akurat untuk memprediksi gambar Captcha apa pun belum ada.

Dalam contoh pembelajaran yang diawasi ini, jelas bahwa tanpa label yang terkait dengan setiap gambar yang memberi tahu program angka yang ditunjukkan gambar tersebut, model tidak dapat dibuat dan prediksi tidak dapat dibuat. Sebaliknya, pendekatan pembelajaran tanpa pengawasan tidak memerlukan datanya untuk diberi label dan dapat digunakan untuk memberi struktur pada kumpulan data yang tidak terstruktur.



Gambar 3.7 Untuk setiap gambar buram, sebuah angka diprediksi; hanya angka 2 yang disalahartikan sebagai 8. Kemudian angka yang ambigu diprediksi menjadi 3 tetapi bisa juga menjadi 5; bahkan bagi mata manusia ini tidak jelas.

3.3.2 Pembelajaran tanpa pengawasan

Secara umum benar bahwa sebagian besar kumpulan data besar tidak memiliki label pada datanya, jadi kecuali Anda menyortir semuanya dan memberinya label, pendekatan pembelajaran terawasi terhadap data tidak akan berfungsi. Sebaliknya, kita harus mengambil pendekatan yang akan berhasil dengan data ini karena

- Kita dapat mempelajari distribusi data dan menyimpulkan kebenaran tentang data di berbagai bagian distribusi.
- Kita dapat mempelajari struktur dan nilai dalam data dan menyimpulkan data dan struktur baru yang lebih bermakna darinya.

Ada banyak teknik untuk masing-masing pendekatan pembelajaran tanpa pengawasan ini. Namun, di dunia nyata Anda selalu bekerja menuju tujuan penelitian yang ditentukan pada fase pertama proses ilmu data, jadi Anda mungkin perlu menggabungkan atau mencoba teknik yang berbeda sebelum kumpulan data dapat diberi label, memungkinkan pembelajaran yang diawasi teknik, mungkin, atau bahkan tujuan itu sendiri tercapai.

Membedakan Struktur Latent Yang Sederhana Dari Data Anda

Tidak semua hal bisa diukur. Saat Anda bertemu seseorang untuk pertama kalinya, Anda mungkin mencoba menebak apakah dia menyukai Anda berdasarkan perilaku dan responsnya. Tetapi bagaimana jika mereka mengalami hari yang buruk sampai sekarang? Mungkin kucing mereka tertabrak atau mereka masih belum menghadiri pemakaman seminggu sebelumnya? Intinya adalah bahwa variabel tertentu dapat segera tersedia sementara yang lain hanya dapat disimpulkan dan karena itu hilang dari kumpulan data Anda. Jenis variabel pertama dikenal sebagai variabel yang dapat diamati dan jenis kedua dikenal sebagai variabel laten. Dalam contoh kita, keadaan emosional teman baru Anda adalah

variabel laten. Itu pasti memengaruhi penilaian mereka terhadap Anda tetapi nilainya tidak jelas.

Memperoleh atau menyimpulkan variabel laten dan nilainya berdasarkan konten sebenarnya dari kumpulan data adalah keterampilan yang berharga untuk dimiliki karena

- Variabel laten dapat menggantikan beberapa variabel yang sudah ada dalam kumpulan data.
- Dengan mengurangi jumlah variabel dalam kumpulan data, kumpulan data menjadi lebih mudah dikelola, setiap algoritme lebih lanjut yang berjalan di atasnya akan bekerja lebih cepat, dan prediksi dapat menjadi lebih akurat.
- Karena variabel laten dirancang atau ditargetkan untuk tujuan penelitian yang ditentukan, Anda kehilangan sedikit informasi penting dengan menggunakannya.

Jika kita dapat mengurangi kumpulan data dari 14 variabel yang dapat diamati per baris menjadi 5 atau 6 variabel laten, misalnya, kita memiliki kesempatan yang lebih baik untuk mencapai tujuan penelitian kita karena struktur kumpulan data yang disederhanakan. Seperti yang akan Anda lihat dari contoh di bawah, ini bukan kasus pengurangan set data yang ada menjadi sesedikit mungkin variabel laten. Anda harus menemukan sweet spot di mana jumlah variabel laten yang dihasilkan menghasilkan nilai terbanyak. Mari kita praktikkan ini dengan studi kasus kecil.

Studi Kasus: Menemukan Variabel Laten Dalam Set Data Kualitas Anggur

Dalam studi kasus singkat ini, Anda akan menggunakan teknik yang dikenal sebagai Principal Component Analysis (PCA) untuk menemukan variabel laten dalam kumpulan data yang menggambarkan kualitas anggur. Kemudian Anda akan membandingkan seberapa baik serangkaian variabel laten bekerja dalam memprediksi kualitas anggur dengan rangkaian asli yang dapat diamati. Anda akan belajar

1. Bagaimana mengidentifikasi dan menurunkan variabel laten tersebut.
2. Bagaimana menganalisis di mana sweet spot—berapa banyak variabel baru yang mengembalikan utilitas paling banyak—dengan menghasilkan dan menginterpretasikan plot scree yang dihasilkan oleh PCA. (Kita akan melihat plot scree sebentar lagi.)

Mari kita lihat komponen utama dari contoh ini.

- **Kumpulan data—**Universitas California, Irvine (UCI) memiliki repositori online berisi 325 kumpulan data untuk latihan pembelajaran mesin di http://archive.ics.uci.edu/ml/. Kami akan menggunakan Kumpulan Data Kualitas Anggur untuk anggur merah yang dibuat oleh P. Cortez, A. Cerdeira, F. Almeida, T. Matos, dan J. Reis4. Panjangnya 1.600 baris dan memiliki 11 variabel per baris, seperti yang ditunjukkan pada tabel 3.2.
- **Analisis Komponen Utama—**Teknik untuk menemukan variabel laten dalam kumpulan data Anda sambil mempertahankan informasi sebanyak mungkin.
- **Scikit-learn—**Kami menggunakan perpustakaan ini karena sudah mengimplementasikan PCA untuk kami dan merupakan cara untuk menghasilkan plot scree.

Bagian pertama dari proses ilmu data adalah untuk menetapkan tujuan penelitian kami: Kami ingin menjelaskan umpan balik subyektif "kualitas anggur" menggunakan properti anggur yang berbeda.

Pekerjaan pertama kami kemudian adalah mengunduh kumpulan data (langkah kedua: memperoleh data), seperti yang ditunjukkan dalam daftar berikut, dan menyiapkannya untuk analisis (langkah ketiga: persiapan data). Kemudian kami dapat menjalankan algoritme PCA dan melihat hasilnya untuk melihat opsi kami.

Tabel 3.2 Tiga baris pertama Kumpulan Data Kualitas Anggur Merah

Keasaman tetap	Keasaman yang mudah menguap	Asam sitrat	Gula sisa	Klorida	Sulfur dioksida bebas	Sulfur dioksida total	Kepadatan	pH	Sulfat	Alkohol	Kualitas
7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	9.4	5
7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	9.8	5
7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	9.8	5

Listing 3.5 Data acquisition and variable standardization

X is a matrix of predictor variables. These variables are wine properties such as density and alcohol presence.

```
import pandas as pd
from sklearn import preprocessing
from sklearn.decomposition import PCA
import pylab as plt
from sklearn import preprocessing

url = http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
     winequality-red.csv
data = pd.read_csv(url, sep= ";")
X = data[[u'fixed acidity', u'volatile acidity', u'citric acid',
      u'residual sugar', u'chlorides', u'free sulfur dioxide',
      u'total sulfur dioxide', u'density', u'pH', u'sulphates',
      u'alcohol']]
y = data.quality
X= preprocessing.StandardScaler().fit(X).transform(X)
```

Downloads location of wine-quality data set.

Reads in the CSV data. It's separated by a semi-colon.

y is a vector and represents the dependent variable (target variable). y is the perceived wine quality.

When standardizing data, the following formula is applied to every data point: $z = (x-\mu)/\sigma$, where z is the new observation value, x the old one, μ is the mean, and σ the standard deviation. The PCA of a data matrix is easier to interpret when the columns have first been centered by their means.

Dengan persiapan data awal di belakang Anda, Anda dapat menjalankan PCA. Plot scree yang dihasilkan (yang akan segera dijelaskan) ditunjukkan pada gambar 3.8. Karena PCA adalah teknik eksploratif, sekarang kita sampai pada langkah keempat dari proses ilmu data: eksplorasi data, seperti yang ditunjukkan pada daftar berikut.

Listing 3.6 Executing the principal component analysis

```
model = PCA()
results = model.fit(X)
Z = results.transform(X)
plt.plot(results.explained_variance_)
plt.show()
```

Creates instance of principal component analysis class

Applies PCA on predictor variables to see if they can be compacted into fewer variables

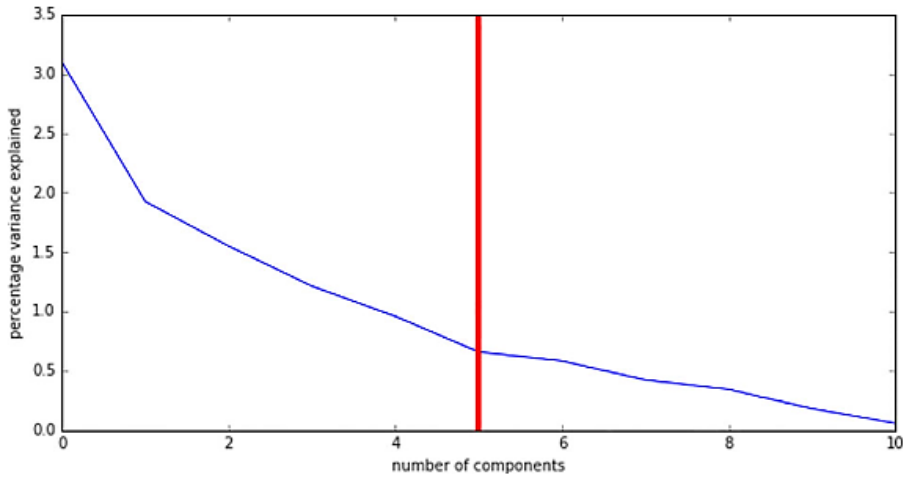
Turns result into array so we can use newly created data

Plots explained variance in variables; this plot is a scree plot

Shows plot

Sekarang mari kita lihat scree plot pada gambar 3.8.

Plot yang dihasilkan dari kumpulan data anggur ditunjukkan pada gambar 3.8. Yang Anda harapkan adalah bentuk siku atau tongkat hoki di plot. Ini menunjukkan bahwa beberapa variabel dapat mewakili sebagian besar informasi dalam kumpulan data sementara sisanya hanya menambahkan sedikit lagi. Dalam plot kami, PCA memberi tahu kami bahwa mengurangi set menjadi satu variabel dapat menangkap sekitar 28% dari total informasi dalam set (plot berbasis nol, jadi variabel satu berada di posisi nol pada sumbu x), dua variabel akan menangkap sekitar 17% lebih atau total 45%, dan seterusnya. Tabel 3.3 menunjukkan kepada Anda pembacaan lengkap.



Gambar 3.8 Plot scree PCA menunjukkan jumlah marginal informasi dari setiap variabel baru yang dapat dibuat PCA. Variabel pertama menjelaskan sekitar 28% varian dalam data, variabel kedua menjelaskan 17% lainnya, variabel ketiga sekitar 15%, dan seterusnya.

Tabel 3.3 Temuan PCA

Jumlah variabel	*Informasi tambahan ditangkap*	*Jumlah data yang diambil*
1	28%	28%

2	17%	45%
3	14%	59%
4	10%	69%
5	8%	77%
6	7%	84%
7	5%	89%
8 - 11	…	100%

Bentuk siku-siku dalam plot menunjukkan bahwa lima variabel dapat menampung sebagian besar informasi yang ditemukan di dalam data. Sebagai gantinya, Anda dapat memperdebatkan batasan pada enam atau tujuh variabel, tetapi kami akan memilih kumpulan data yang lebih sederhana dibandingkan dengan varian data yang lebih sedikit dibandingkan kumpulan data asli.

Pada titik ini, kita dapat melanjutkan dan melihat apakah kumpulan data asli yang dikodekan ulang dengan lima variabel laten cukup baik untuk memprediksi kualitas anggur secara akurat, tetapi sebelum kita melakukannya, kita akan melihat bagaimana kita dapat mengidentifikasi apa yang diwakilinya.

Menafsirkan Variabel Baru

Dengan keputusan awal dibuat untuk mengurangi kumpulan data dari 11 variabel asli menjadi 5 variabel laten, kita dapat memeriksa untuk melihat apakah mungkin untuk menafsirkan atau menamainya berdasarkan hubungannya dengan aslinya. Nama sebenarnya lebih mudah digunakan daripada kode seperti lv1, lv2, dan seterusnya. Kita dapat menambahkan baris kode di daftar berikut untuk menghasilkan tabel yang menunjukkan bagaimana dua set variabel berkorelasi.

Listing 3.7 Showing PCA components in a Pandas data frame

```
pd.DataFrame(results.components_, columns=list(
➥  [u'fixed acidity', u'volatile acidity', u'citric acid', u'residual sugar',
➥   u'chlorides', u'free sulfur dioxide',  u'total sulfur dioxide', u'density',
➥   u'pH', u'sulphates',  u'alcohol']))
```

Baris-baris pada tabel hasil (tabel 3.4) menunjukkan korelasi matematis. Atau, dalam bahasa Inggris, variabel laten pertama lv1, yang menangkap kira-kira 28% dari total informasi dalam himpunan, memiliki rumus berikut.

```
Lv1 = (fixed acidity * 0.489314) + (volatile acidity * -0.238584) + … +
(alcohol * -0.113232)
```

Tabel 3.4 Bagaimana PCA menghitung korelasi 11 variabel asli dengan 5 variabel laten

	Keasaman tetap	Keasaman yang mudah menguap	Asam sitrat	Gula sisa	Klorida	Sulfur dioksida bebas	Sulfur dioksida total	Kepadatan	pH	Sulfat	Alkohol
0	0.4893	- 0.2385	0.4636	0.1461	0.2122	- 0.0361	0.0236	0.3953	-0.4385	0.2429	- 0.1132
1	- 0.1105	0.2749	- 0.1518	0.2720	0.1480	0.5135	0.5695	0.2336	0.0067	- 0.0375	- 0.3861
2	0.1233	0.4499	- 0.2382	- 0.1012	0.0926	- 0.4287	-0.3224	0.3389	- 0.0576	- 0.2797	- 0.4716
3	- 0.2296	0.0789	- 0.0794	- 0.3727	0.6662	- 0.0435	- 0.0345	- 0.1745	-0.0038	0.5508	- 0.1221
4	0.0826	- 0.2187	0.0586	- 0.7321	- 0.2465	0.1591	0.2224	- 0.1571	- 0.2675	- 0.2259	- 0.3506

Memberikan nama yang bisa digunakan untuk setiap variabel baru sedikit lebih rumit dan mungkin memerlukan konsultasi dengan pakar anggur yang sebenarnya untuk keakuratannya. Namun, karena kami tidak memiliki ahli anggur, kami akan memanggil mereka sebagai berikut (tabel 3.5).

Tabel 3.5 Interpretasi variabel pembuatan PCA kualitas anggur

Variabel laten	Kemungkinan interpretasi
0	Keasaman yang terus-menerus
1	Sulfida
2	Keasaman yang mudah menguap
3	Klorida
4	Kurangnya sisa gula

Kita sekarang dapat mengode ulang kumpulan data asli hanya dengan lima variabel laten. Melakukan ini adalah persiapan data lagi, jadi kami meninjau kembali langkah ketiga dari proses ilmu data: persiapan data. Seperti disebutkan dalam bab 2, proses ilmu data bersifat rekursif dan ini terutama terjadi antara langkah ketiga: persiapan data dan langkah 4: eksplorasi data. Tabel 3.6 menunjukkan tiga baris pertama dengan ini selesai.

Tabel 3.6 Tiga baris pertama dari Kumpulan Data Kualitas Anggur Merah dikode ulang dalam lima variabel laten

	Keasaman yang terus-menerus	Sulfida	Keasaman yang mudah menguap	Klorida	Kurangnya sisa gula
0	-1.619530	0.450950	**1.774454**	0.043740	-0.067014
1	-0.799170	1.856553	0.911690	0.548066	0.018392
2	**2.357673**	-0.269976	-0.243489	-0.928450	**1.499149**

Kita sudah dapat melihat nilai tinggi untuk anggur 0 dalam keasaman yang mudah menguap, sedangkan anggur 2 sangat tinggi dalam keasaman yang persisten. Sama sekali tidak terdengar seperti anggur yang enak!

Membandingkan Akurasi Set Data Asli Dengan Variabel Laten

Sekarang setelah kami memutuskan kumpulan data kami harus dikodekan ulang menjadi 5 variabel laten daripada 11 variabel asli, saatnya untuk melihat seberapa baik kumpulan data baru bekerja untuk memprediksi kualitas anggur jika dibandingkan dengan aslinya. Kami akan menggunakan algoritme Naïve Bayes Classifier yang kami lihat di contoh sebelumnya untuk pembelajaran yang diawasi untuk membantu. Mari kita mulai dengan melihat seberapa baik 11 variabel asli dapat memprediksi skor kualitas wine. Daftar berikut menyajikan kode untuk melakukan ini.

Listing 3.8 Wine score prediction before principal component analysis

```
from sklearn.cross_validation import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
import pylab as plt

gnb = GaussianNB()
fit = gnb.fit(X,y)
pred = fit.predict(X)
print confusion_matrix(pred,y)
print confusion_matrix(pred,y).trace()
```

Use Gaussian distribution Naïve Bayes classifier for estimation.

Fit data.

Predict data for unseen data.

Study confusion matrix.

Count of all correctly classified cases: all counts on trace or diagonal summed up after analyzing confusion matrix. We can see the Naïve Bayes classifier scores 897 correct predictions out of 1599.

Sekarang kita akan menjalankan tes prediksi yang sama, tetapi dimulai dengan hanya 1 variabel laten, bukan 11 variabel asli. Kemudian kita akan menambahkan yang lain, melihat hasilnya, menambahkan yang lain, dan seterusnya untuk melihat bagaimana performa prediksi meningkat. Daftar berikut menunjukkan bagaimana hal ini dilakukan.

Listing 3.9 Wine score prediction with increasing number of principal components

```
predicted_correct = []
for i in range(1,10):
    model = PCA(n_components = i)
    results = model.fit(X)
    Z = results.transform(X)
    fit = gnb.fit(Z,y)
    pred = fit.predict(Z)
    predicted_correct.append(confusion_matrix(pred,y).trace())
    print predicted_correct
plt.plot(predicted_correct)
plt.show()
```

Array will be filled with correctly predicted observations

Loops through first 10 detected principal components

Instantiate PCA model with 1 component (first iteration) up to 10 components (in 10th iteration)

Fit PCA model on x-variables (features)

Z is result in matrix form (actually an array filled with arrays)

Use Gaussian distribution Naïve Bayes classifier for estimation

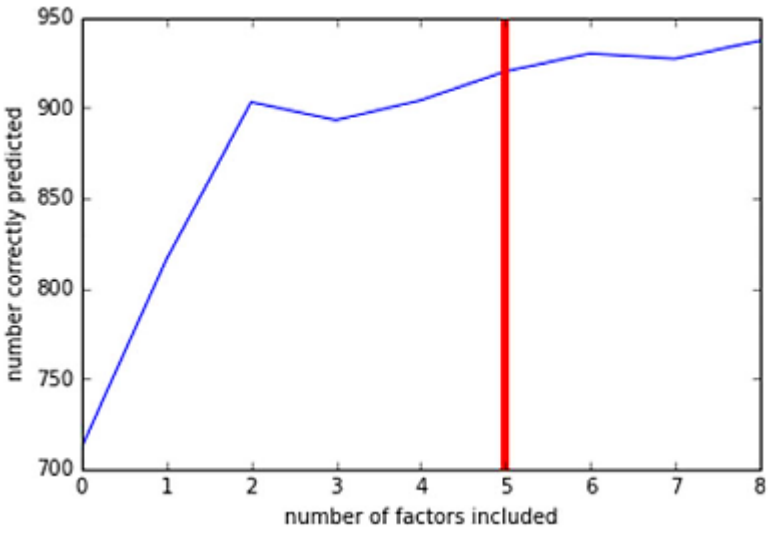
The actual prediction itself using the fitted model

At end of each iteration we append number of correctly classified observations

Printing this array we can see how after each iteration, new count of correctly classified observations is appended

Easier to see when array plotted

Plot shown



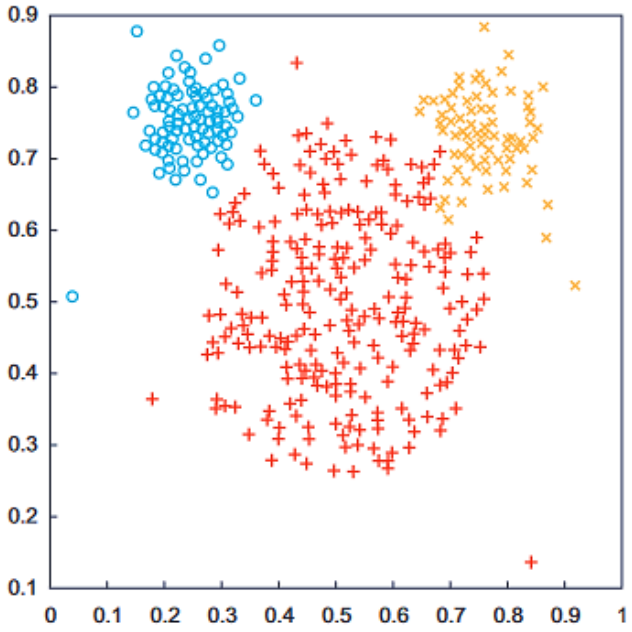
Gambar 3.9 Plot hasil menunjukkan bahwa menambahkan lebih banyak variabel laten ke model (sumbu x) sangat meningkatkan daya prediksi (sumbu y) hingga titik tertentu tetapi kemudian berkurang. Keuntungan dalam kekuatan prediktif dari penambahan variabel pada akhirnya akan hilang.

Plot yang dihasilkan ditunjukkan pada gambar 3.9. Plot pada gambar 3.9 menunjukkan bahwa dengan hanya 3 variabel laten, pengklasifikasi melakukan pekerjaan yang lebih baik dalam memprediksi kualitas anggur dibandingkan dengan 11 variabel asli. Selain itu, menambahkan lebih banyak variabel laten di luar 5 tidak menambah kekuatan prediksi sebanyak 5 variabel pertama. Ini menunjukkan bahwa pilihan kami untuk memotong 5 variabel adalah pilihan yang bagus, seperti yang kami harapkan. Kami melihat cara mengelompokkan variabel serupa, tetapi juga memungkinkan untuk mengelompokkan pengamatan.

Mengelompokkan Pengamatan Serupa Untuk Mendapatkan Wawasan Dari Distribusi Data Anda

Misalkan sejenak Anda sedang membangun situs web yang merekomendasikan film kepada pengguna berdasarkan preferensi yang telah mereka masukkan dan film yang telah mereka tonton. Kemungkinan besar jika mereka menonton banyak film horor, mereka cenderung ingin tahu tentang film horor baru dan bukan tentang film roman remaja baru. Dengan mengelompokkan pengguna yang telah menonton film yang kurang lebih sama dan menetapkan preferensi yang kurang lebih sama, Anda dapat memperoleh banyak wawasan tentang apa lagi yang mungkin ingin mereka rekomendasikan.

Teknik umum yang kami jelaskan di sini dikenal sebagai pengelompokan. Dalam proses ini, kami mencoba untuk membagi kumpulan data kami menjadi subset pengamatan, atau kelompok, di mana pengamatan harus serupa dengan yang ada di kelompok yang sama tetapi sangat berbeda dari pengamatan di kelompok lain. Gambar 3.10 memberi Anda gambaran visual tentang apa yang ingin dicapai oleh pengelompokan. Lingkaran di kiri atas gambar terlihat jelas berdekatan satu sama lain namun lebih jauh dari yang lain. Hal yang sama berlaku untuk salib di kanan atas.



Gambar 3.10 Tujuan pengelompokan adalah untuk membagi kumpulan data menjadi himpunan bagian yang "cukup berbeda". Dalam plot ini misalnya, pengamatan telah dibagi menjadi tiga kelompok.

Scikit-learn mengimplementasikan beberapa algoritma umum untuk pengelompokan data dalam modul sklearn.cluster-nya, termasuk algoritma k-means, propagasi afinitas, dan pengelompokan spektral. Masing-masing memiliki satu atau dua kasus penggunaan yang lebih cocok,5 meskipun k-means adalah algoritme tujuan umum yang baik untuk memulai. Namun, seperti semua algoritme pengelompokan, Anda perlu menentukan jumlah klaster yang diinginkan terlebih dahulu, yang tentunya menghasilkan proses coba-coba sebelum mencapai kesimpulan yang layak. Ini juga mengandaikan bahwa semua data yang diperlukan untuk analisis sudah tersedia. Bagaimana jika tidak?

Mari kita lihat kasus sebenarnya dari pengelompokan iris (bunga) berdasarkan sifat-sifatnya (panjang dan lebar sepal, panjang dan lebar petal, dan sebagainya). Dalam contoh ini kita akan menggunakan algoritma k-means. Ini adalah algoritme yang bagus untuk mendapatkan impresi data tetapi peka terhadap nilai awal, sehingga Anda dapat berakhir dengan cluster yang berbeda setiap kali Anda menjalankan algoritme kecuali jika Anda secara manual menentukan nilai awal dengan menentukan seed (konstanta untuk generator nilai awal). Jika Anda perlu mendeteksi hierarki, lebih baik Anda menggunakan algoritme dari kelas teknik pengelompokan hierarki.

Salah satu kelemahan lainnya adalah kebutuhan untuk menentukan jumlah cluster yang diinginkan terlebih dahulu. Ini sering menghasilkan proses coba-coba sebelum sampai pada kesimpulan yang memuaskan.

Mengeksekusi kodenya cukup sederhana. Ini mengikuti struktur yang sama dengan semua analisis lainnya kecuali Anda tidak harus melewati variabel target. Terserah algoritme untuk mempelajari pola yang menarik. Daftar berikut menggunakan kumpulan data iris untuk melihat apakah algoritma dapat mengelompokkan berbagai jenis iris.

Listing 3.10 Iris classification example

Print first 5 observations of data frame to screen; now we can clearly see 4 variables: sepal length, sepal width, petal length, and petal width.

Add another variable called "cluster" to data frame. This indicates the cluster membership of every flower in data set.

Load in iris (flowers) data of Scikit-learn.

Fit model to data. All variables are considered independent variables; unsupervised learning has no target variable (y).

Transform iris data into Pandas data frame.

```
import sklearn
from sklearn import cluster
import pandas as pd

data = sklearn.datasets.load_iris()
X = pd.DataFrame(data.data, columns = list(data.feature_names))
print X[:5]
model   = cluster.KMeans(n_clusters=3, random_state=25)
results = model.fit(X)
X["cluster"] = results.predict(X)
X["target"] = data.target
X["c"] = "lookatmeIamimportant"
print X[:5]
classification_result = X[["cluster",
     "target","c"]].groupby(["cluster","target"]).agg("count")
print(classification_result)
```

Let's finally add a target variable (y) to the data frame.

Adding a variable c is just a little trick we use to do a count later. The value here is arbitrary because we need a column to count the rows.

Three parts to this code. First we select the cluster, target, and c columns. Then we group by the cluster and target columns. Finally, we aggregate the row of the group with a simple count aggregation.

Initialize a k-means cluster model with 3 clusters. The random_state is a random seed; if you don't put it in, the seed will also be random. We opt for 3 clusters because we saw in the last listing this might be a good compromise between complexity and performance.

The matrix this classification result represents gives us an indication of whether our clustering was successful. For cluster 0, we're spot on. On clusters 1 and 2 there has been a slight mix-up, but in total we only get 16 (14+2) misclassifications out of 150.

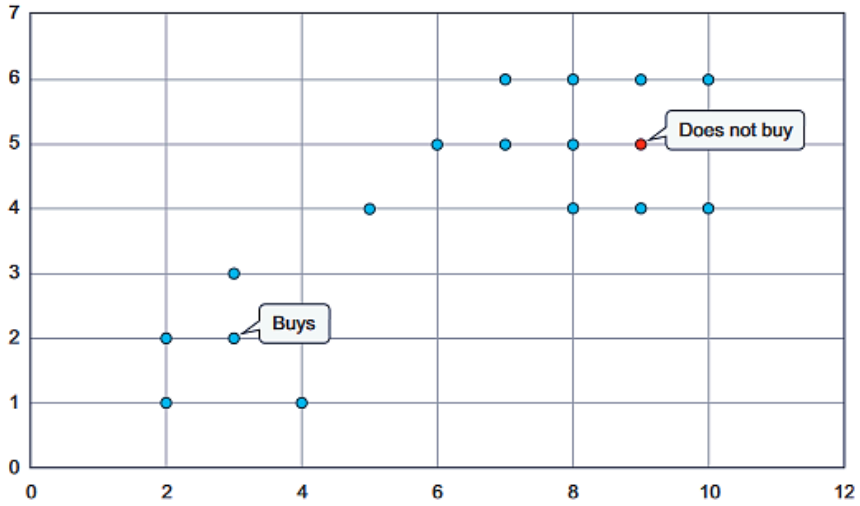
		c
cluster	target	
0	0	50
1	1	48
	2	14
2	1	2
	2	36

Gambar 3.11 Output dari klasifikasi iris

Gambar 3.11 menunjukkan output dari klasifikasi iris. Gambar ini menunjukkan bahwa bahkan tanpa menggunakan label, Anda akan menemukan klaster yang mirip dengan klasifikasi iris resmi dengan hasil 134 (50+48+36) klasifikasi yang benar dari 150. Anda tidak selalu harus memilih antara diawasi dan tidak diawasi; terkadang menggabungkannya adalah sebuah pilihan.

3.4 PEMBELAJARAN SEMI-DIAWASI

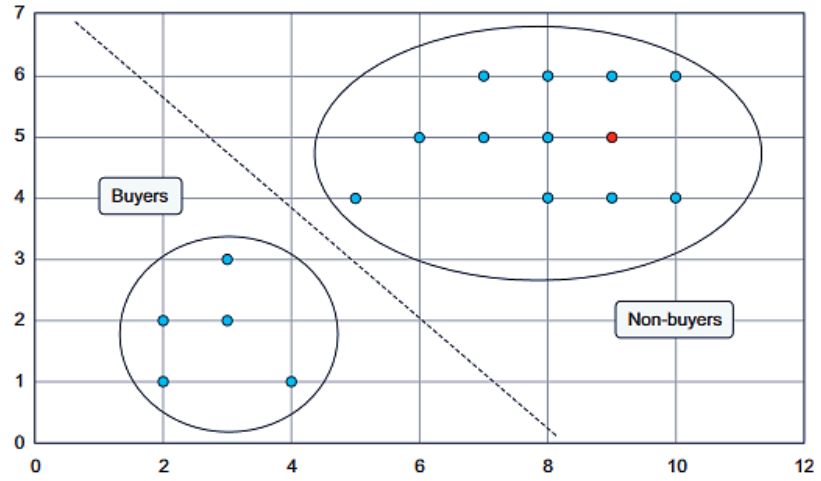
Anda tidak perlu heran jika mengetahui bahwa meskipun kami ingin semua data kami diberi label sehingga kami dapat menggunakan teknik pembelajaran mesin yang diawasi yang lebih canggih, pada kenyataannya kami sering memulai hanya dengan data berlabel minimal, jika memang diberi label sama sekali. Kami dapat menggunakan teknik pembelajaran mesin tanpa pengawasan kami untuk menganalisis apa yang kami miliki dan mungkin menambahkan label ke kumpulan data, tetapi akan sangat mahal untuk memberi label semuanya. Tujuan kami kemudian adalah untuk melatih model prediktor kami dengan data berlabel sesedikit mungkin. Di sinilah teknik pembelajaran semi-diawasi masuk—hibrida dari dua pendekatan yang telah kita lihat. Ambil contoh plot pada gambar 3.12. Dalam hal ini, data hanya memiliki dua pengamatan berlabel; biasanya ini terlalu sedikit untuk membuat prediksi yang valid.



Gambar 3.12 Plot ini hanya memiliki dua observasi berlabel—terlalu sedikit untuk observasi terawasi, tetapi cukup untuk memulai dengan pendekatan tak terawasi atau semi terawasi.

Teknik pembelajaran semi-diawasi yang umum adalah propagasi label. Dalam teknik ini, Anda mulai dengan kumpulan data berlabel dan memberikan label yang sama ke titik data yang serupa. Ini mirip dengan menjalankan algoritme pengelompokan pada kumpulan data dan memberi label pada setiap kluster berdasarkan label yang dikandungnya. Jika kita menerapkan pendekatan ini pada kumpulan data pada gambar 3.12, kita mungkin akan mendapatkan sesuatu seperti gambar 3.13.

Salah satu pendekatan khusus untuk pembelajaran semi-diawasi yang perlu disebutkan di sini adalah pembelajaran aktif. Dalam pembelajaran aktif, program menunjukkan pengamatan yang ingin dilihatnya diberi label untuk putaran pembelajaran berikutnya berdasarkan beberapa kriteria yang telah Anda tentukan. Misalnya, Anda dapat mengaturnya untuk mencoba dan memberi label pengamatan yang paling tidak pasti tentang algoritme, atau Anda dapat menggunakan beberapa model untuk membuat prediksi dan memilih titik di mana model paling tidak setuju.



Gambar 3.13 Gambar sebelumnya menunjukkan bahwa data hanya memiliki dua pengamatan berlabel, terlalu sedikit untuk pembelajaran yang diawasi.

Gambar ini menunjukkan bagaimana Anda dapat mengeksploitasi struktur kumpulan data yang mendasarinya untuk mempelajari pengklasifikasi yang lebih baik daripada dari data berlabel saja. Data dibagi menjadi dua cluster dengan teknik clustering; kami hanya memiliki dua nilai berlabel, tetapi jika kami berani, kami dapat menganggap orang lain di dalam kluster tersebut memiliki label yang sama (pembeli atau non-pembeli), seperti yang digambarkan di sini. Teknik ini tidak sempurna; lebih baik mendapatkan label yang sebenarnya jika Anda bisa.

Dengan dasar-dasar pembelajaran mesin yang Anda inginkan, bab selanjutnya membahas penggunaan pembelajaran mesin dalam batasan satu komputer. Ini cenderung menantang ketika kumpulan data terlalu besar untuk dimuat seluruhnya ke dalam memori.

3.5 RINGKASAN

Di bab ini, Anda mempelajarinya

- Ilmuwan data sangat bergantung pada teknik dari statistik dan pembelajaran mesin untuk melakukan pemodelan mereka. Ada banyak aplikasi kehidupan nyata untuk pembelajaran mesin, mulai dari mengklasifikasikan siulan burung hingga memprediksi letusan gunung berapi.
- Proses pemodelan terdiri dari empat fase:
    1. Rekayasa fitur, persiapan data, dan parameterisasi model —Kami menentukan parameter input dan variabel untuk model kami.
    2. Pelatihan model—Model diisi dengan data dan mempelajari pola yang tersembunyi di dalam data.
    3. Pemilihan dan validasi model—Sebuah model dapat bekerja dengan baik atau buruk; berdasarkan kinerjanya, kami memilih model yang paling masuk akal.
    4. Penskoran model—Jika model kami dapat dipercaya, model tersebut akan merilis data baru. Jika kita melakukan pekerjaan kita dengan baik, itu akan memberi kita wawasan tambahan atau memberi kita prediksi yang baik tentang apa yang akan terjadi di masa depan.

- Dua jenis besar teknik pembelajaran mesin
  1. Diawasi—Pembelajaran yang memerlukan data berlabel.
  2. Unsupervised —Pembelajaran yang tidak memerlukan data berlabel tetapi biasanya kurang akurat atau dapat diandalkan dibandingkan pembelajaran yang diawasi.
- Pembelajaran semi-diawasi berada di antara teknik-teknik tersebut dan digunakan ketika hanya sebagian kecil dari data yang diberi label.
- Dua studi kasus masing-masing mendemonstrasikan pembelajaran yang diawasi dan tidak diawasi:
  1. Studi kasus pertama kami memanfaatkan pengklasifikasi Naïve Bayes untuk mengklasifikasikan gambar angka sebagai angka yang diwakilinya. Kami juga melihat matriks kebingungan sebagai sarana untuk menentukan seberapa baik kinerja model klasifikasi kami.
  2. Studi kasus kami tentang teknik tanpa pengawasan menunjukkan bagaimana kami dapat menggunakan analisis komponen utama untuk mengurangi variabel input untuk pembuatan model lebih lanjut sambil mempertahankan sebagian besar informasi.

BAB 4
MENANGANI DATA BESAR DALAM SATU KOMPUTER

Dalam Bab ini, diharapkan mahasiswa mampu memahami:

- Bekerja dengan kumpulan data besar di satu komputer
- Bekerja dengan pustaka Python yang cocok untuk kumpulan data yang lebih besar
- Memahami pentingnya memilih algoritma dan struktur data yang tepat
- Memahami bagaimana Anda dapat mengadaptasi algoritme untuk bekerja di dalam basis data

Bagaimana jika Anda memiliki begitu banyak data yang tampaknya melebihi Anda, dan teknik Anda tampaknya tidak lagi memadai? Apa yang Anda lakukan, menyerah atau beradaptasi? Untungnya Anda memilih untuk beradaptasi, karena Anda masih membaca. Bab ini memperkenalkan Anda pada teknik dan alat untuk menangani kumpulan data yang lebih besar yang masih dapat dikelola oleh satu komputer jika Anda mengadopsi teknik yang tepat.

Bab ini memberi Anda alat untuk melakukan klasifikasi dan regresi ketika data tidak lagi sesuai dengan RAM (memori akses acak) komputer Anda, sedangkan bab 3 berfokus pada kumpulan data dalam memori. Bab 5 akan melangkah lebih jauh dan mengajari Anda cara menangani kumpulan data yang memerlukan banyak komputer untuk diproses. Yang kami maksud dengan data besar dalam bab ini adalah data yang menyebabkan masalah dalam hal memori atau kecepatan tetapi masih dapat ditangani oleh satu komputer.

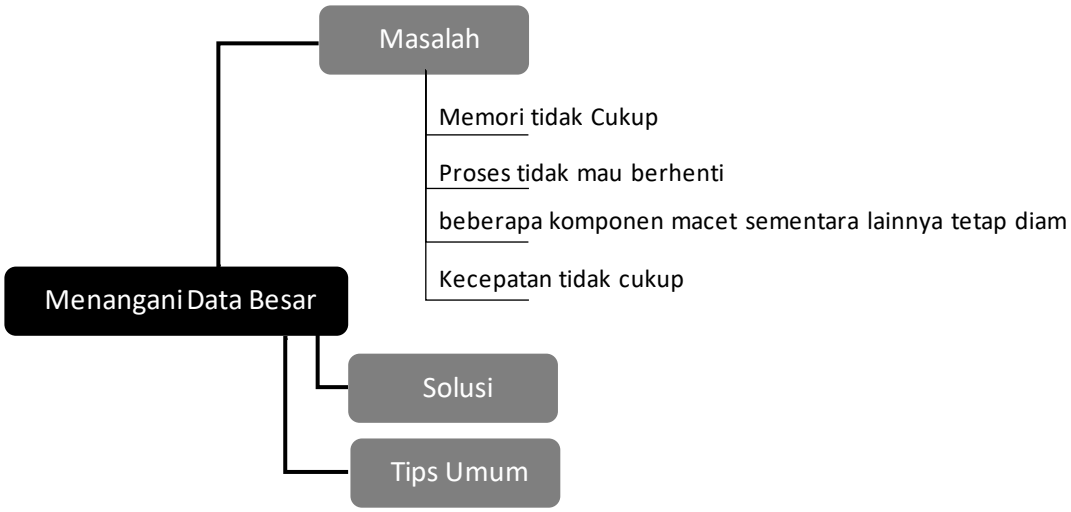
Kami memulai bab ini dengan ikhtisar masalah yang Anda hadapi saat menangani kumpulan data besar. Kemudian kami menawarkan tiga jenis solusi untuk mengatasi masalah ini: sesuaikan algoritme Anda, pilih struktur data yang tepat, dan pilih alat yang tepat. Ilmuwan data bukan satu-satunya yang harus berurusan dengan volume data yang besar, sehingga Anda dapat menerapkan praktik terbaik umum untuk mengatasi masalah data yang besar. Akhirnya, kami menerapkan pengetahuan ini pada dua studi kasus. Kasus pertama menunjukkan cara mendeteksi URL jahat, dan kasus kedua menunjukkan cara membuat mesin pemberi rekomendasi di dalam database.

4.1 MASALAH YANG ANDA HADAPI SAAT MENANGANI DATA BESAR

Volume data yang besar menimbulkan tantangan baru, seperti memori yang kelebihan beban dan algoritma yang tidak pernah berhenti berjalan. Ini memaksa Anda untuk beradaptasi dan memperluas repertoar teknik Anda. Tetapi meskipun Anda dapat melakukan analisis, Anda harus menangani masalah seperti I/O (input/output) dan kekurangan CPU, karena ini dapat menyebabkan masalah kecepatan. Gambar 4.1 menunjukkan peta pikiran yang secara bertahap akan terungkap saat kita melalui langkah-langkah: masalah, solusi, dan tips.

Komputer hanya memiliki jumlah RAM yang terbatas. Saat Anda mencoba memeras lebih banyak data ke dalam memori ini daripada yang sebenarnya muat, OS akan mulai menukar blok memori ke disk, yang jauh lebih tidak efisien daripada menyimpan semuanya di

memori. Tetapi hanya beberapa algoritme yang dirancang untuk menangani kumpulan data besar; kebanyakan dari mereka memuat seluruh kumpulan data ke dalam memori sekaligus, yang menyebabkan kesalahan kehabisan memori. Algoritme lain perlu menyimpan banyak salinan data dalam memori atau menyimpan hasil antara. Semua ini memperburuk masalah.



Gambar 4.1 Tinjauan masalah yang dihadapi saat bekerja dengan lebih banyak data daripada yang dapat ditampung dalam memori

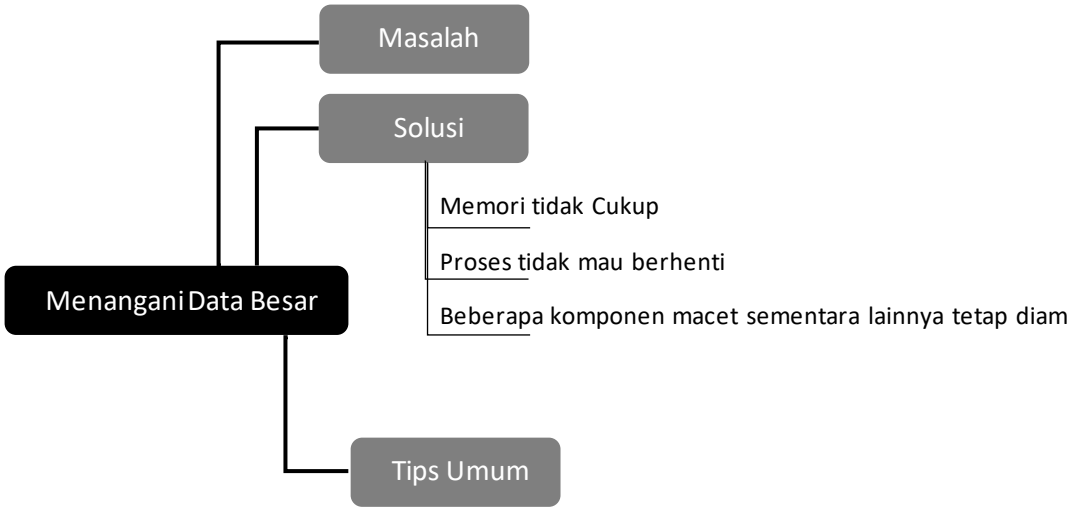
Bahkan saat Anda menyembuhkan masalah ingatan, Anda mungkin perlu berurusan dengan sumber daya terbatas lainnya: waktu. Meskipun komputer mungkin mengira Anda hidup selama jutaan tahun, pada kenyataannya Anda tidak akan melakukannya (kecuali jika Anda masuk ke cryostasis sampai PC Anda selesai). Algoritme tertentu tidak memperhitungkan waktu; mereka akan terus berjalan selamanya. Algoritme lain tidak dapat berakhir dalam jumlah waktu yang masuk akal ketika mereka hanya perlu memproses beberapa megabyte data.

Hal ketiga yang akan Anda amati ketika berhadapan dengan kumpulan data besar adalah bahwa komponen komputer Anda dapat mulai membentuk hambatan sementara membiarkan sistem lain menganggur. Meskipun ini tidak separah algoritme yang tidak pernah berakhir atau kesalahan kehabisan memori, ini masih menimbulkan biaya yang serius. Pikirkan penghematan biaya dalam hal hari orang dan infrastruktur komputasi untuk kelaparan CPU. Program tertentu tidak memasukkan data dengan cukup cepat ke prosesor karena harus membaca data dari hard drive, yang merupakan salah satu komponen paling lambat di komputer. Ini telah diatasi dengan diperkenalkannya solid state drive (SSD), tetapi SSD masih jauh lebih mahal daripada teknologi hard disk drive (HDD) yang lebih lambat dan lebih luas.

4.2 TEKNIK UMUM UNTUK MENANGANI VOLUME DATA BESAR

Algoritma tanpa akhir, kesalahan kehabisan memori, dan masalah kecepatan adalah tantangan paling umum yang Anda hadapi saat bekerja dengan data besar. Di bagian ini, kami akan menyelidiki solusi untuk mengatasi atau meringankan masalah ini. Solusinya dapat dibagi

menjadi tiga kategori: menggunakan algoritma yang tepat, memilih struktur data yang tepat, dan menggunakan alat yang tepat (gambar 4.2).

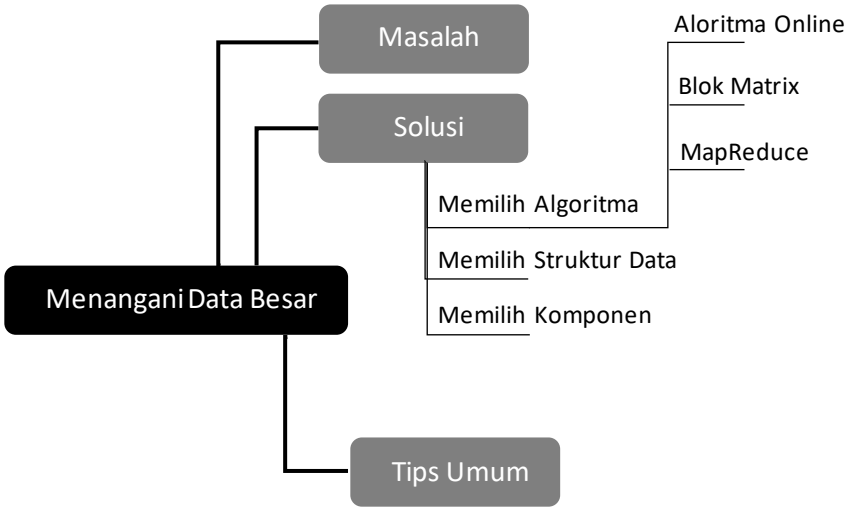


Gambar 4.2 Gambaran umum solusi untuk menangani kumpulan data besar

Tidak ada pemetaan satu-ke-satu yang jelas antara masalah dan solusi karena banyak solusi mengatasi kekurangan memori dan kinerja komputasi. Misalnya, kompresi kumpulan data akan membantu Anda mengatasi masalah memori karena kumpulan data menjadi lebih kecil. Tapi ini juga mempengaruhi kecepatan komputasi dengan pergeseran dari hard disk yang lambat ke CPU yang cepat. Berlawanan dengan RAM (memori akses acak), hard disk akan menyimpan semuanya bahkan setelah daya padam, tetapi menulis ke disk membutuhkan lebih banyak waktu daripada mengubah informasi dalam RAM yang cepat berlalu. Ketika terus-menerus mengubah informasi, RAM lebih disukai daripada hard disk (lebih tahan lama). Dengan kumpulan data yang belum dikemas, banyak operasi baca dan tulis (I/O) terjadi, tetapi CPU sebagian besar tetap menganggur, sedangkan dengan kumpulan data terkompresi, CPU mendapatkan bagian yang adil dari beban kerja. Ingatlah hal ini sementara kami menjelajahi beberapa solusi.

4.2.1 Memilih algoritma yang tepat

Memilih algoritma yang tepat dapat memecahkan lebih banyak masalah daripada menambahkan perangkat keras yang lebih banyak atau lebih baik. Algoritme yang cocok untuk menangani data besar tidak perlu memuat seluruh kumpulan data ke dalam memori untuk membuat prediksi. Idealnya, algoritma juga mendukung perhitungan paralel. Pada bagian ini kita akan menggali tiga jenis algoritme yang dapat melakukannya: algoritme online, algoritme blok, dan algoritme MapReduce, seperti yang ditunjukkan pada gambar 4.3.



Gambar 4.3 Tinjauan teknik untuk mengadaptasi algoritma ke set data besar

Algoritma Pembelajaran Online

Beberapa, tetapi tidak semua, algoritma pembelajaran mesin dapat dilatih menggunakan satu pengamatan pada satu waktu alih-alih memasukkan semua data ke dalam memori. Setelah titik data baru tiba, model dilatih dan observasi dapat dilupakan; efeknya sekarang dimasukkan ke dalam parameter model. Sebagai contoh, model yang digunakan untuk memprediksi cuaca dapat menggunakan parameter yang berbeda (seperti tekanan atmosfir atau suhu) di wilayah yang berbeda. Saat data dari satu wilayah dimuat ke dalam algoritme, algoritme akan melupakan data mentah ini dan beralih ke wilayah berikutnya. Cara kerja "gunakan dan lupakan" ini adalah solusi sempurna untuk masalah memori karena pengamatan tunggal tidak mungkin cukup besar untuk mengisi semua memori komputer modern.

Listing 4.1 menunjukkan bagaimana menerapkan prinsip ini pada perceptron dengan pembelajaran online. Perceptron adalah salah satu algoritma pembelajaran mesin paling kompleks yang digunakan untuk klasifikasi biner (0 atau 1); misalnya, apakah pelanggan akan membeli atau tidak?

Listing 4.1 Training a perceptron by observation

The learning rate of an algorithm is the adjustment it makes every time a new observation comes in. If this is high, the model will adjust quickly to new observations but might "overshoot" and never get precise. An oversimplified example: the optimal (and unknown) weight for an x-variable = 0.75. Current estimation is 0.4 with a learning rate of 0.5; the adjustment = 0.5 (learning rate) * 1(size of error) * 1 (value of x) = 0.5. 0.4 (current weight) + 0.5 (adjustment) = 0.9 (new weight), instead of 0.75. The adjustment was too big to get the correct result.

```
import numpy as np
class perceptron():
    def __init__(self, X,y, threshold = 0.5,
learning_rate = 0.1, max_epochs = 10):
        self.threshold = threshold
        self.learning_rate = learning_rate
        self.X = X
        self.y = y
        self.max_epochs = max_epochs
```

Sets up perceptron class.

The __init__ method of any Python class is always run when creating an instance of the class. Several default values are set here.

The threshold is an arbitrary cutoff between 0 and 1 to decide whether the prediction becomes a 0 or a 1. Often it's 0.5, right in the middle, but it depends on the use case.

X and y variables are assigned to the class.

One epoch is one run through all the data. We allow for a maximum of 10 runs until we stop the perceptron.

Each observation will end up with a weight. The initialize function sets these weights for each incoming observation. We allow for 2 options: all weights start at 0 or they are assigned a small (between 0 and 0.05) random weight.

```
    def initialize(self, init_type = 'zeros'):
        if init_type == 'random':
            self.weights = np.random.rand(len(self.X[0])) * 0.05
        if init_type == 'zeros':
            self.weights = np.zeros(len(self.X[0]))
```

The training function.

We start at the first epoch.

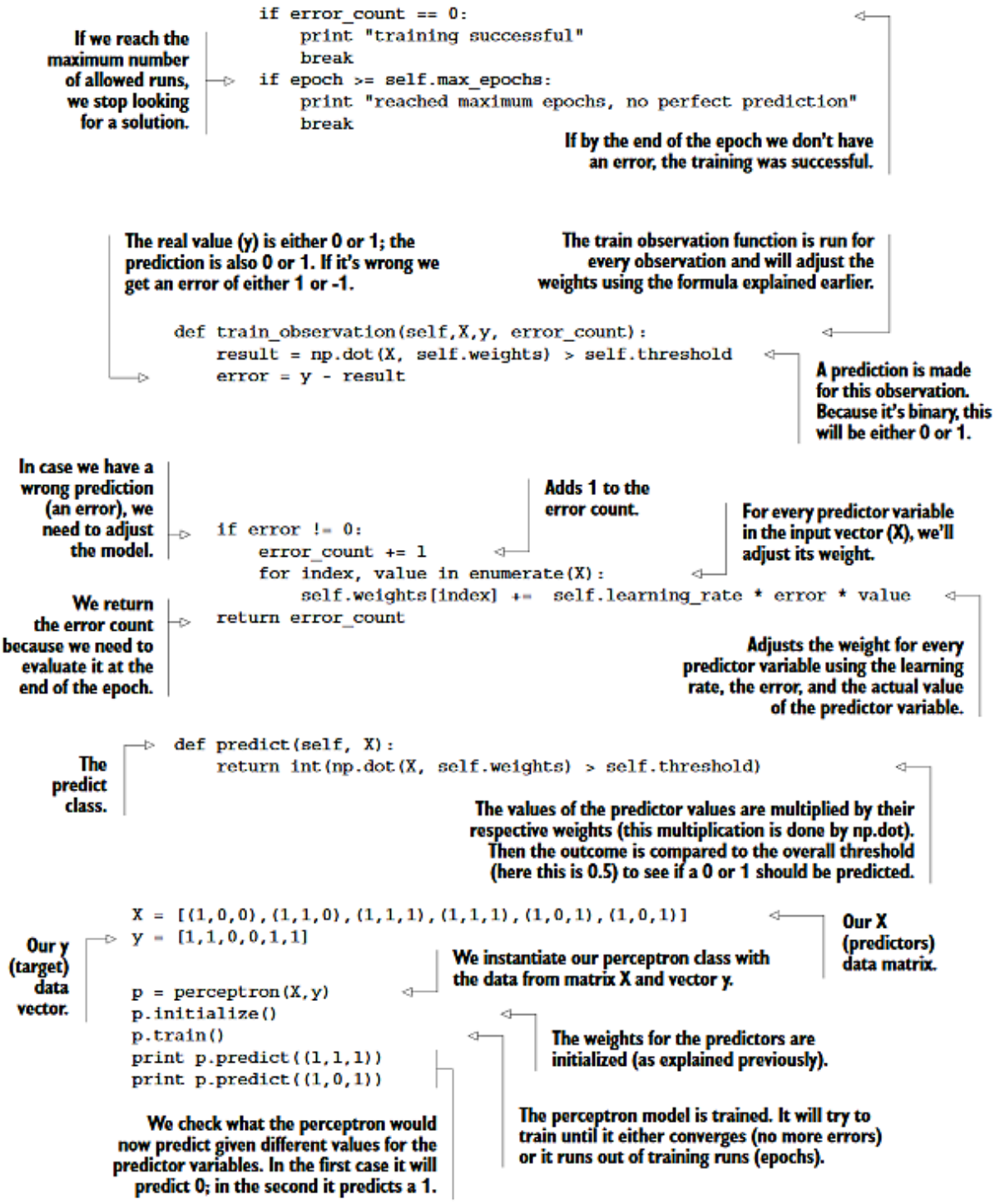
True is always true, so technically this is a never-ending loop, but we build in several stop (break) conditions.

```
    def train(self):
        epoch = 0
        while True:
            error_count = 0
            epoch += 1
            for (X,y) in zip(self.X, self.y):
                error_count += self.train_observation(X,y,error_count)
```

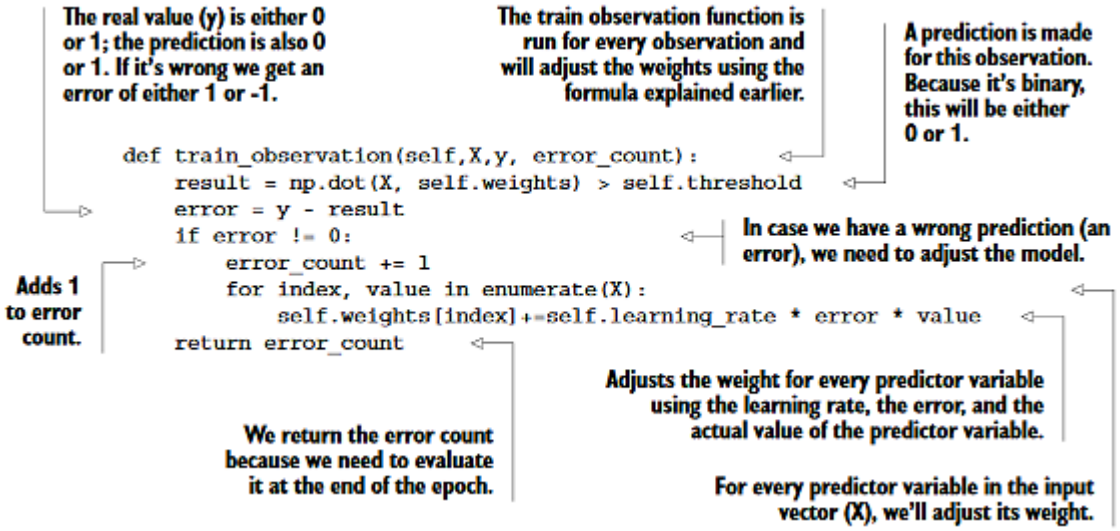
Adds one to the current number of epochs.

Initiates the number of encountered errors at 0 for each epoch. This is important; if an epoch ends without errors, the algorithm converged and we're done.

We loop through the data and feed it to the train observation function, one observation at a time.



Kami akan memperbesar bagian kode yang mungkin tidak begitu jelas untuk dipahami tanpa penjelasan lebih lanjut. Kami akan mulai dengan menjelaskan cara kerja fungsi train_observation(). Fungsi ini memiliki dua bagian besar. Yang pertama adalah menghitung prediksi dari suatu pengamatan dan membandingkannya dengan nilai sebenarnya. Bagian kedua adalah mengubah bobot jika prediksi tampaknya salah.



Prediksi (y) dihitung dengan mengalikan vektor masukan variabel bebas dengan bobot masing-masing dan menjumlahkan suku-sukunya (seperti dalam regresi linier). Kemudian nilai ini dibandingkan dengan ambang batas. Jika lebih besar dari ambang batas, algoritme akan memberikan 1 sebagai keluaran, dan jika lebih kecil dari ambang batas, algoritme akan memberikan 0 sebagai keluaran. Menetapkan ambang adalah hal yang subjektif dan bergantung pada kasus bisnis Anda. Katakanlah Anda memprediksi apakah seseorang memiliki penyakit mematikan tertentu, dengan 1 positif dan 0 negatif. Dalam hal ini, lebih baik memiliki ambang batas yang lebih rendah: tidak seburuk ditemukan positif dan melakukan penyelidikan kedua daripada mengabaikan penyakit dan membiarkan pasien meninggal. Kesalahan dihitung, yang akan memberikan arah perubahan bobot.

```
result = np.dot(X, self.weights) > self.threshold
error = y - result
```

Bobot diubah sesuai dengan tanda kesalahan. Pembaruan dilakukan dengan aturan pembelajaran untuk perceptrons. Untuk setiap bobot dalam vektor bobot, Anda memperbarui nilainya dengan aturan berikut:

$$\Delta w_i = \alpha \varepsilon x_i$$

dimana Δw_i adalah jumlah bobot yang perlu diubah, α adalah learning rate, ε adalah error, dan xi adalah nilai ke-i pada vektor input (variabel prediktor ke-i). Penghitungan kesalahan adalah variabel untuk melacak berapa banyak pengamatan yang salah diprediksi pada zaman ini dan dikembalikan ke fungsi pemanggilan. Anda menambahkan satu pengamatan ke penghitung kesalahan jika prediksi aslinya salah. Zaman adalah pelatihan tunggal yang dijalankan melalui semua pengamatan.

```
if error != 0:
        error_count += 1
            for index, value in enumerate(X):
                self.weights[index] +=  self.learning_rate * error * value
```

Fungsi kedua yang akan kita bahas lebih detail adalah fungsi train(). Fungsi ini memiliki loop internal yang terus melatih perceptron hingga dapat memprediksi dengan sempurna atau hingga mencapai sejumlah putaran pelatihan (zaman), seperti yang ditunjukkan pada daftar berikut.

Listing 4.2 Using `train` functions

```
def train(self):
        epoch = 0
        while True:
            error_count = 0
            epoch += 1
            for (X,y) in zip(self.X, self.y):
                error_count += self.train_observation(X,y,error_count)
            if error_count == 0:
                print "training succesfull"
                break
            if epoch >= self.max_epochs:
                print "reached maximum epochs, no perfect prediction"
                break an loop to train the data
```

Training function.

Starts at the first epoch.

True is always true so technically this is a never-ending loop, but we build in several stop (break) conditions.

Initiates the number of encountered errors at 0 for each epoch. This is important, because if an epoch ends without errors, the algorithm converged and we're done.

Adds one to the current number of epochs.

We loop through the data and feed it to the train observation function, one observation at a time.

If by the end of the epoch we don't have an error, training was successful.

If we reach the maximum number of allowed runs, we stop looking for a solution.

Sebagian besar algoritme online juga dapat menangani batch mini; dengan cara ini, Anda dapat memberi mereka kumpulan 10 hingga 1.000 pengamatan sekaligus sambil menggunakan jendela geser untuk memeriksa data Anda. Anda memiliki tiga opsi:

- Pembelajaran batch penuh (juga disebut pembelajaran statistik)—Masukkan algoritme semua data sekaligus. Inilah yang kami lakukan di bab 3.
- Pembelajaran mini-batch—Masukkan algoritme sesendok penuh (100, 1000, …, tergantung pada apa yang dapat ditangani perangkat keras Anda) pengamatan pada satu waktu.
- Pembelajaran online—Masukkan algoritme satu pengamatan pada satu waktu.

Teknik pembelajaran online terkait dengan algoritme streaming, di mana Anda melihat setiap titik data hanya sekali. Pikirkan tentang data Twitter yang masuk: data tersebut dimuat ke dalam algoritme, dan kemudian pengamatan (tweet) dibuang karena banyaknya data tweet yang masuk mungkin akan segera membanjiri perangkat keras. Algoritma pembelajaran online berbeda dari algoritma streaming karena mereka dapat melihat pengamatan yang sama berkali-kali. Benar, algoritma pembelajaran online dan algoritma streaming sama-sama bisa belajar dari pengamatan satu per satu. Perbedaannya adalah bahwa algoritme online juga

digunakan pada sumber data statis dan juga pada sumber data streaming dengan menyajikan data dalam batch kecil (sekecil pengamatan tunggal), yang memungkinkan Anda memeriksa data berkali-kali. Ini tidak terjadi pada algoritme streaming, di mana data mengalir ke sistem dan Anda biasanya perlu segera melakukan penghitungan. Mereka serupa karena mereka hanya menangani beberapa pada satu waktu.

Membagi Matriks Besar Menjadi Banyak Matriks Kecil

Sedangkan di bab sebelumnya kita hampir tidak perlu berurusan dengan seberapa tepatnya algoritma memperkirakan parameter, menyelami ini terkadang bisa membantu. Misalnya, dengan memotong tabel data besar menjadi matriks kecil, kita masih bisa melakukan regresi linier. Logika di balik pemisahan matriks ini dan bagaimana regresi linier dapat dihitung dengan matriks dapat ditemukan di sidebar. Cukup untuk mengetahui untuk saat ini bahwa pustaka Python yang akan kita gunakan akan menangani pemisahan matriks, dan bobot variabel regresi linier dapat dihitung menggunakan kalkulus matriks.

Matriks blok dan formula matriks estimasi koefisien regresi linier Algoritma tertentu dapat diterjemahkan ke dalam algoritma yang menggunakan blok matriks, bukan matriks penuh. Saat Anda mempartisi matriks menjadi matriks blok, Anda membagi seluruh matriks menjadi beberapa bagian dan bekerja dengan bagian yang lebih kecil daripada matriks penuh. Dalam hal ini Anda dapat memuat matriks yang lebih kecil ke dalam memori dan melakukan perhitungan, sehingga menghindari kesalahan kehabisan memori. Gambar 4.4 menunjukkan bagaimana Anda dapat menulis ulang penjumlahan matriks A + B menjadi submatriks.

Rumus pada gambar 4.4 menunjukkan bahwa tidak ada perbedaan antara menjumlahkan matriks A dan B sekaligus dalam satu langkah atau terlebih dahulu menjumlahkan bagian atas matriks dan kemudian menjumlahkan bagian bawah. Semua operasi matriks dan vektor umum, seperti perkalian, inversi, dan dekomposisi nilai singular (teknik pengurangan variabel seperti PCA), dapat ditulis dalam bentuk matriks blok.1 Operasi matriks blok menghemat memori dengan memecah masalah menjadi blok yang lebih kecil dan mudah diparalelkan.

$$A + B = \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} + \begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{bmatrix}$$

$$= \begin{bmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{j,1} & \cdots & a_{j,m} \\ \hline a_{j+1,1} & \cdots & a_{j+1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{bmatrix} + \begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{j,1} & \cdots & b_{j,m} \\ \hline b_{j+1,1} & \cdots & b_{j+1,m} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{bmatrix} = \begin{bmatrix} A_1 \\ \hline A_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ \hline B_2 \end{bmatrix}$$

Gambar 4.4 Matriks blok dapat digunakan untuk menghitung jumlah matriks A dan B.

Meskipun sebagian besar paket numerik memiliki kode yang sangat optimal, mereka hanya bekerja dengan matriks yang dapat masuk ke dalam memori dan akan menggunakan matriks blok dalam memori bila menguntungkan. Dengan matriks kehabisan memori, mereka tidak mengoptimalkan ini untuk Anda dan terserah Anda untuk mempartisi matriks menjadi matriks yang lebih kecil dan menerapkan versi matriks blok. Regresi linier adalah cara untuk memprediksi variabel kontinu dengan kombinasi linear dari prediktornya; salah satu cara paling dasar untuk melakukan perhitungan adalah dengan teknik yang disebut kuadrat terkecil biasa. Rumus dalam bentuk matriks adalah

$$\boldsymbol{\beta} = (\boldsymbol{X}^{-\boldsymbol{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y}$$

di mana β adalah koefisien yang ingin diambil, X adalah prediktor, dan y adalah variabel target. Alat Python yang kami miliki untuk menyelesaikan tugas kami adalah sebagai berikut:

- bcolz adalah pustaka Python yang dapat menyimpan larik data secara ringkas dan menggunakan hard drive saat larik tidak lagi muat ke dalam memori utama.
- Dask adalah pustaka yang memungkinkan Anda untuk mengoptimalkan aliran perhitungan dan membuat perhitungan secara paralel lebih mudah dilakukan. Itu tidak dikemas dengan pengaturan Anaconda default jadi pastikan untuk menggunakan conda install dask di lingkungan virtual Anda sebelum menjalankan kode di bawah ini. Catatan: beberapa kesalahan telah dilaporkan saat mengimpor Dask saat menggunakan Python 64bit. Dask bergantung pada beberapa pustaka lain (seperti toolz), tetapi ketergantungan tersebut harus ditangani secara otomatis oleh pip atau conda.

Daftar berikut menunjukkan perhitungan matriks blok dengan perpustakaan ini.

Listing 4.3 Block matrix calculations with bcolz and Dask libraries

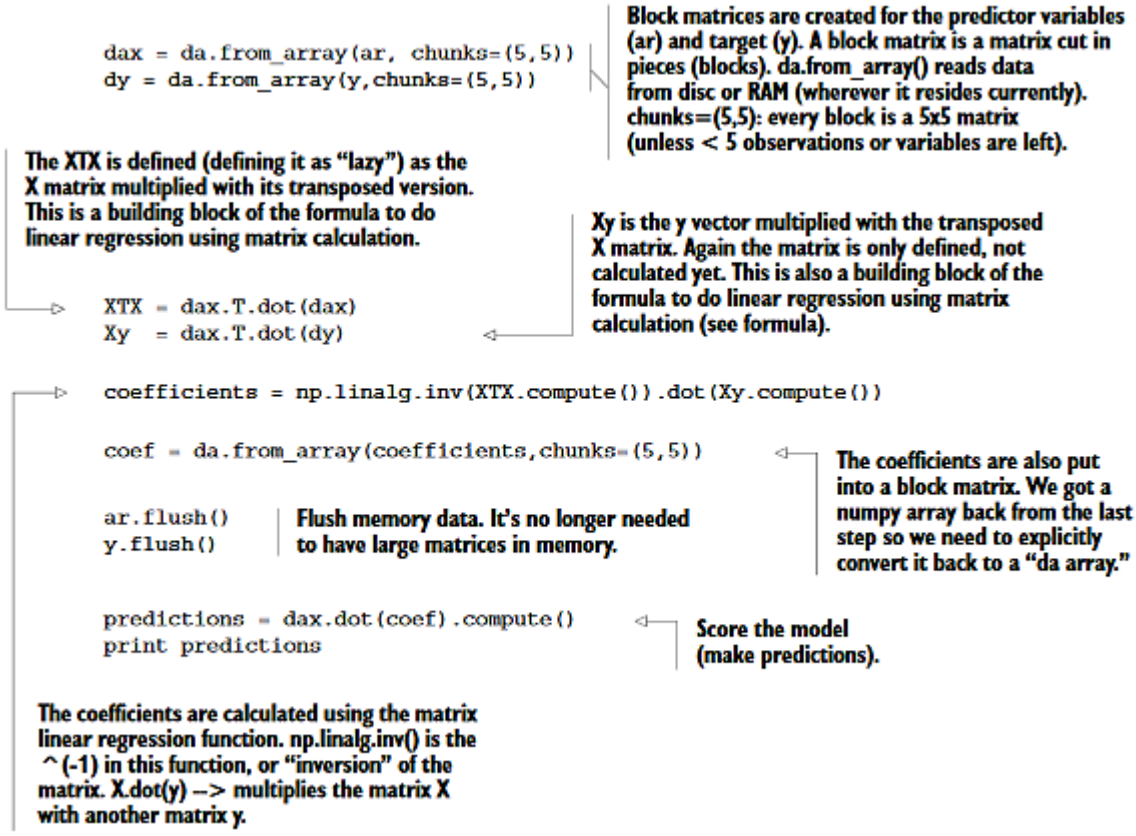
Number of observations (scientific notation). 1e4 = 10.000. Feel free to change this.

Creates fake data: np.arange(n).reshape(n/2,2) creates a matrix of 5000 by 2 (because we set n to 10.000). bc.carray = numpy is an array extension that can swap to disc. This is also stored in a compressed way. rootdir = 'ar.bcolz' –> creates a file on disc in case out of RAM. You can check this on your file system next to this ipython file or whatever location you ran this code from. mode = 'w' –> is the write mode. dtype = 'float64' –> is the storage type of the data (which is float numbers).

```
import dask.array as da
import bcolz as bc
import numpy as np
import dask

n = 1e4

ar = bc.carray(np.arange(n).reshape(n/2,2)  , dtype='float64',
     rootdir = 'ar.bcolz', mode = 'w')
y  = bc.carray(np.arange(n/2), dtype='float64', rootdir =
     'yy.bcolz', mode = 'w')
```



Perhatikan bahwa Anda tidak perlu menggunakan inversi matriks blok karena XTX adalah matriks persegi dengan ukuran nr. prediktor * nr. dari prediktor. Ini beruntung karena Dask belum mendukung inversi matriks blok.

Mapreduce

Algoritma MapReduce mudah dipahami dengan analogi: Bayangkan Anda diminta menghitung semua suara untuk pemilu nasional. Negara Anda memiliki 25 partai, 1.500 kantor pemungutan suara, dan 2 juta orang. Anda dapat memilih untuk mengumpulkan semua tiket pemungutan suara dari setiap kantor satu per satu dan menghitungnya secara terpusat, atau Anda dapat meminta kantor lokal untuk menghitung suara untuk 25 partai dan menyerahkan hasilnya kepada Anda, dan kemudian Anda dapat mengumpulkannya berdasarkan partai. Pengurang peta mengikuti proses yang mirip dengan cara kerja kedua. Mereka pertama-tama memetakan nilai ke kunci dan kemudian melakukan agregasi pada kunci tersebut selama fase pengurangan. Lihat kode semu daftar berikut untuk mendapatkan perasaan yang lebih baik untuk ini.

Listing 4.4 MapReduce pseudo code example

```
For each person in voting office:
    Yield (voted_party, 1)
For each vote in voting office:
    add_vote_to_party()
```

Salah satu keuntungan dari algoritme MapReduce adalah mudah untuk diparalelkan dan didistribusikan. Ini menjelaskan keberhasilan mereka dalam lingkungan terdistribusi seperti Hadoop, tetapi mereka juga dapat digunakan pada komputer individu. Kita akan melihatnya

lebih mendalam di bab berikutnya, dan contoh implementasi (JavaScript) juga disediakan di bab 9. Saat mengimplementasikan MapReduce dengan Python, Anda tidak perlu memulai dari awal. Sejumlah perpustakaan telah melakukan sebagian besar pekerjaan untuk Anda, seperti Hadoopy, Octopy, Disco, atau Dumbo.

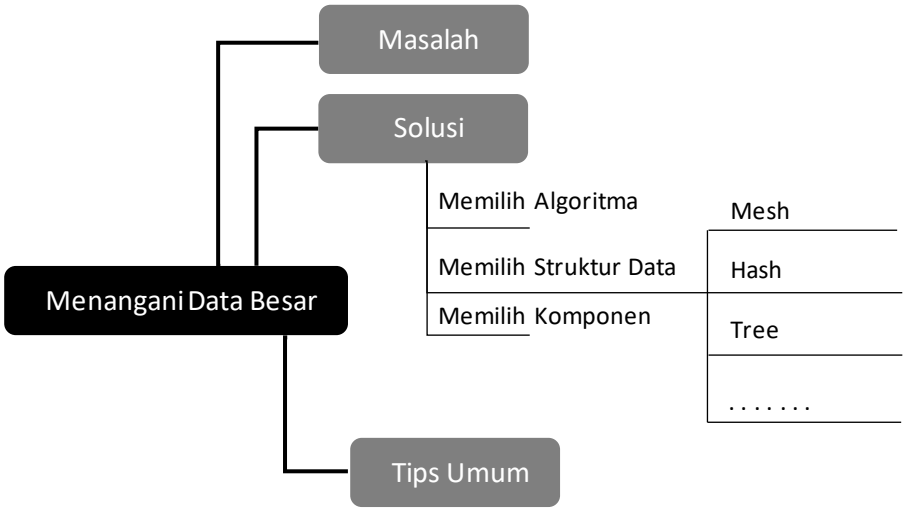
4.2.2 Memilih struktur data yang tepat

Algoritma dapat membuat atau menghancurkan program Anda, tetapi cara Anda menyimpan data sama pentingnya. Struktur data memiliki persyaratan penyimpanan yang berbeda, tetapi juga memengaruhi kinerja CRUD (buat, baca, perbarui, dan hapus) dan operasi lain pada kumpulan data. Gambar 4.5 menunjukkan bahwa Anda memiliki banyak struktur data yang berbeda untuk dipilih, tiga di antaranya akan kita bahas di sini: data jarang, data pohon, dan data hash. Pertama mari kita lihat kumpulan data jarang.

Data Jangkau

Kumpulan data jarang berisi informasi yang relatif sedikit dibandingkan dengan entri (pengamatan). Lihat gambar 4.6: hampir semuanya adalah "0" dengan hanya satu "1" yang ada pada pengamatan kedua pada variabel 9.

Data seperti ini mungkin terlihat konyol, tetapi sering kali ini yang Anda dapatkan saat mengonversi data tekstual menjadi data biner. Bayangkan satu set 100.000 tweet Twitter yang sama sekali tidak berhubungan. Sebagian besar dari mereka mungkin memiliki kurang dari 30 kata, tetapi bersama-sama mereka mungkin memiliki ratusan atau ribuan kata yang berbeda. Dalam bab tentang penambangan teks, kita akan melalui proses pemotongan dokumen teks menjadi kata-kata dan menyimpannya sebagai vektor. Tapi untuk saat ini bayangkan apa yang akan Anda dapatkan jika setiap kata diubah menjadi variabel biner, dengan "1" berarti "ada di tweet ini", dan "0" berarti "tidak ada di tweet ini". Ini memang akan menghasilkan data yang jarang. Matriks besar yang dihasilkan dapat menyebabkan masalah memori meskipun hanya berisi sedikit informasi.



Gambar 4.5 Sekilas tentang struktur data yang sering diterapkan dalam ilmu data saat bekerja dengan data berukuran besar

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 4.6 Contoh matriks jarang: hampir semuanya 0; nilai lain adalah pengecualian dalam matriks jarang

Untungnya, data seperti ini bisa disimpan secara kompak. Dalam kasus gambar 4.6 bisa terlihat seperti ini:

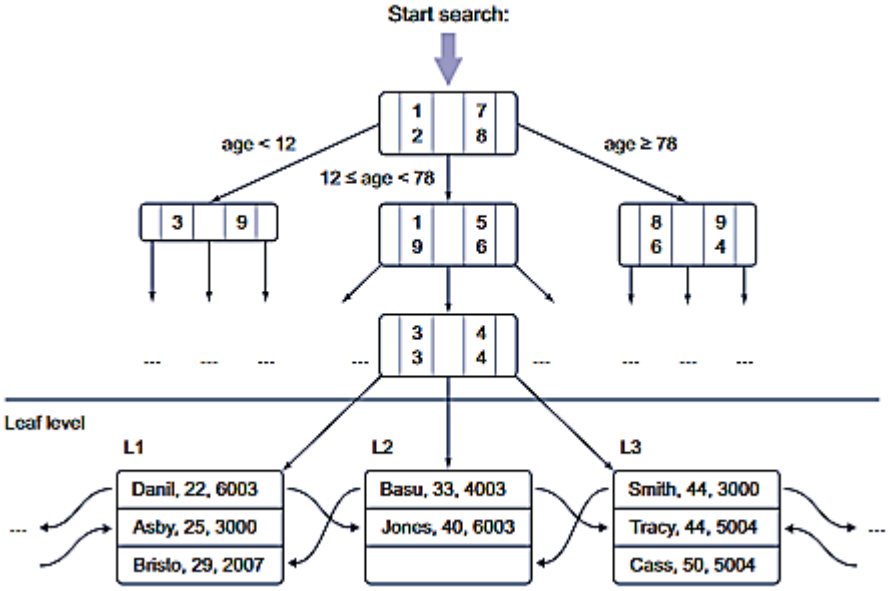
```
data = [(2,9,1)]
```

Baris 2, kolom 9 memuat nilai 1. Dukungan untuk bekerja dengan matriks jarang tumbuh di Python. Banyak algoritme sekarang mendukung atau menampilkan matriks renggang.

Struktur Pohon

Pohon adalah kelas struktur data yang memungkinkan Anda mengambil informasi lebih cepat daripada memindai melalui tabel. Sebuah pohon selalu memiliki nilai akar dan subpohon dari anak-anak, masing-masing dengan anak-anaknya, dan seterusnya. Contoh sederhananya adalah silsilah keluarga Anda sendiri atau silsilah biologis dan caranya membelah menjadi cabang, ranting, dan daun. Aturan keputusan sederhana memudahkan untuk menemukan pohon anak tempat data Anda berada. Lihat gambar 4.7 untuk melihat bagaimana struktur pohon memungkinkan Anda mendapatkan informasi yang relevan dengan cepat.

Pada gambar 4.7 Anda memulai pencarian di bagian atas dan memilih kategori usia terlebih dahulu, karena ternyata itulah faktor yang paling banyak memotong alternatif. Ini terus berlanjut sampai Anda mendapatkan apa yang Anda cari. Bagi siapa pun yang tidak mengenal Akinator, kami sarankan untuk mengunjungi http://en.akinator.com/. Akinator adalah jin dalam lampu ajaib yang mencoba menebak seseorang dalam pikiran Anda dengan menanyakan beberapa pertanyaan tentang dia. Cobalah dan kagumi. . . atau lihat bagaimana keajaiban ini adalah pencarian pohon.



Gambar 4.7 Contoh struktur data pohon: aturan keputusan seperti kategori usia dapat digunakan untuk menemukan seseorang dengan cepat dalam pohon keluarga

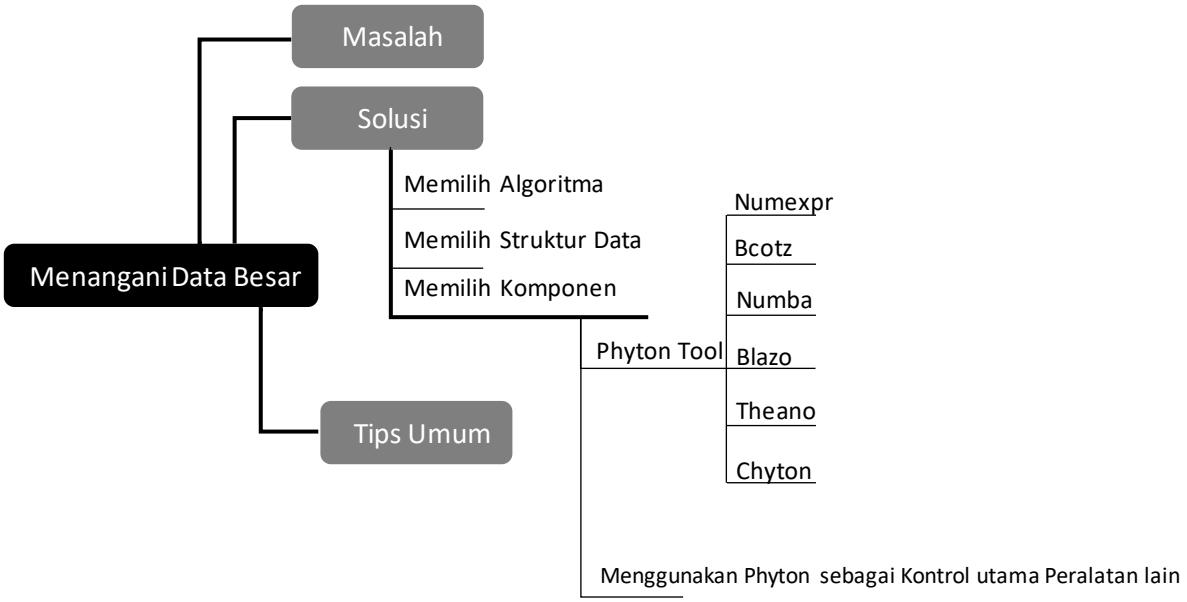
Pohon juga populer di database. Database memilih untuk tidak memindai tabel dari baris pertama hingga terakhir, tetapi menggunakan perangkat yang disebut indeks untuk menghindari hal ini. Indeks seringkali didasarkan pada struktur data seperti pohon dan tabel hash untuk menemukan pengamatan lebih cepat. Penggunaan indeks sangat mempercepat proses pencarian data. Mari kita lihat tabel hash ini.

Tabel Hash

Tabel hash adalah struktur data yang menghitung kunci untuk setiap nilai dalam data Anda dan memasukkan kunci ke dalam ember. Dengan cara ini Anda dapat dengan cepat mengambil informasi dengan melihat di keranjang yang tepat saat Anda menemukan data. Kamus di Python adalah implementasi tabel hash, dan mereka adalah kerabat dekat dari penyimpanan nilai kunci. Anda akan menemukannya di contoh terakhir bab ini saat Anda membangun sistem pemberi rekomendasi di dalam database. Tabel hash digunakan secara luas dalam database sebagai indeks untuk pengambilan informasi yang cepat.

4.2.3 Memilih alat yang tepat

Dengan kelas algoritma dan struktur data yang tepat, saatnya memilih alat yang tepat untuk pekerjaan itu. Alat yang tepat dapat berupa pustaka Python atau setidaknya alat yang dikendalikan dari Python, seperti yang ditunjukkan pada gambar 4.8. Jumlah alat bermanfaat yang tersedia sangat banyak, jadi kami hanya akan melihat beberapa di antaranya.



Gambar 4.8 Gambaran alat yang dapat digunakan saat bekerja dengan data besar

Alat Python

Python memiliki sejumlah pustaka yang dapat membantu Anda menangani data besar. Mulai dari struktur data yang lebih cerdas di atas pengoptimal kode hingga kompiler just-in-time. Berikut ini adalah daftar pustaka yang ingin kami gunakan saat berhadapan dengan data besar:

- **Cython—**Semakin dekat Anda dengan perangkat keras komputer yang sebenarnya, semakin penting bagi komputer untuk mengetahui jenis data apa yang harus diproses. Untuk komputer, menjumlahkan 1 + 1 berbeda dengan menjumlahkan 1,00 + 1,00. Contoh pertama terdiri dari bilangan bulat dan yang kedua terdiri dari float, dan perhitungan ini dilakukan oleh berbagai bagian CPU. Di Python Anda tidak perlu menentukan tipe data apa yang Anda gunakan, jadi kompiler Python harus menyimpulkannya. Tetapi menyimpulkan tipe data adalah operasi yang lambat dan sebagian mengapa Python bukan salah satu bahasa tercepat yang tersedia. Cython, superset dari Python, memecahkan masalah ini dengan memaksa pemrogram untuk menentukan tipe data saat mengembangkan program. Setelah kompiler memiliki informasi ini, ia menjalankan program lebih cepat.
- **Numexpr—**Numexpr adalah inti dari banyak paket big data, seperti halnya NumPy untuk paket dalam memori. Numexpr adalah evaluator ekspresi numerik untuk NumPy tetapi bisa berkali-kali lebih cepat daripada NumPy asli. Untuk mencapai ini, itu menulis ulang ekspresi Anda dan menggunakan kompiler internal (tepat waktu).
- **Numba—**Numba membantu Anda mencapai kecepatan yang lebih tinggi dengan mengompilasi kode tepat sebelum Anda mengeksekusinya, juga dikenal sebagai kompilasi just-in-time. Ini memberi Anda keuntungan menulis kode tingkat tinggi tetapi mencapai kecepatan yang mirip dengan kode C.
- **Bcolz—**Bcolz membantu Anda mengatasi masalah kehabisan memori yang dapat terjadi saat menggunakan NumPy. Itu dapat menyimpan dan bekerja dengan array dalam bentuk terkompresi yang optimal. Ini tidak hanya mengurangi kebutuhan data Anda tetapi juga

menggunakan Numexpr di latar belakang untuk mengurangi perhitungan yang diperlukan saat melakukan perhitungan dengan array bcolz.

- **Blaze—**Blaze sangat ideal jika Anda ingin menggunakan kekuatan backend database tetapi menyukai "cara Python" bekerja dengan data. Blaze akan menerjemahkan kode Python Anda ke dalam SQL tetapi dapat menangani lebih banyak penyimpanan data daripada database relasional seperti CSV, Spark, dan lainnya. Blaze memberikan cara terpadu untuk bekerja dengan banyak database dan pustaka data. Blaze masih dalam pengembangan, jadi banyak fitur yang belum diimplementasikan.
- **Theano—**Theano memungkinkan Anda untuk bekerja secara langsung dengan unit pemrosesan grafis (GPU) dan melakukan penyederhanaan simbolis jika memungkinkan, dan dilengkapi dengan kompiler just-in-time yang luar biasa. Selain itu, ini adalah perpustakaan yang bagus untuk menangani konsep matematika yang canggih namun berguna: tensor.
- **Dask—**Dask memungkinkan Anda mengoptimalkan aliran perhitungan dan menjalankannya secara efisien. Ini juga memungkinkan Anda untuk mendistribusikan perhitungan. Lihat http:// dask.pydata.org/en/latest/.

Pustaka ini sebagian besar tentang penggunaan Python itu sendiri untuk pemrosesan data (selain Blaze, yang juga terhubung ke database). Untuk mencapai kinerja kelas atas, Anda dapat menggunakan Python untuk berkomunikasi dengan semua jenis database atau perangkat lunak lain.

Gunakan Python Sebagai Master Untuk Mengendalikan Alat Lainnya

Sebagian besar produsen perangkat lunak dan alat mendukung antarmuka Python ke perangkat lunak mereka. Ini memungkinkan Anda memanfaatkan perangkat lunak khusus dengan kemudahan dan produktivitas yang disertakan dengan Python. Dengan cara ini Python membedakan dirinya dari bahasa sains data populer lainnya seperti R dan SAS. Anda harus memanfaatkan kemewahan ini dan memanfaatkan kekuatan alat khusus semaksimal mungkin. Bab 6 menampilkan studi kasus menggunakan Python untuk terhubung ke database NoSQL, seperti halnya bab 7 dengan data grafik. Sekarang mari kita lihat tip bermanfaat yang lebih umum saat menangani data besar.

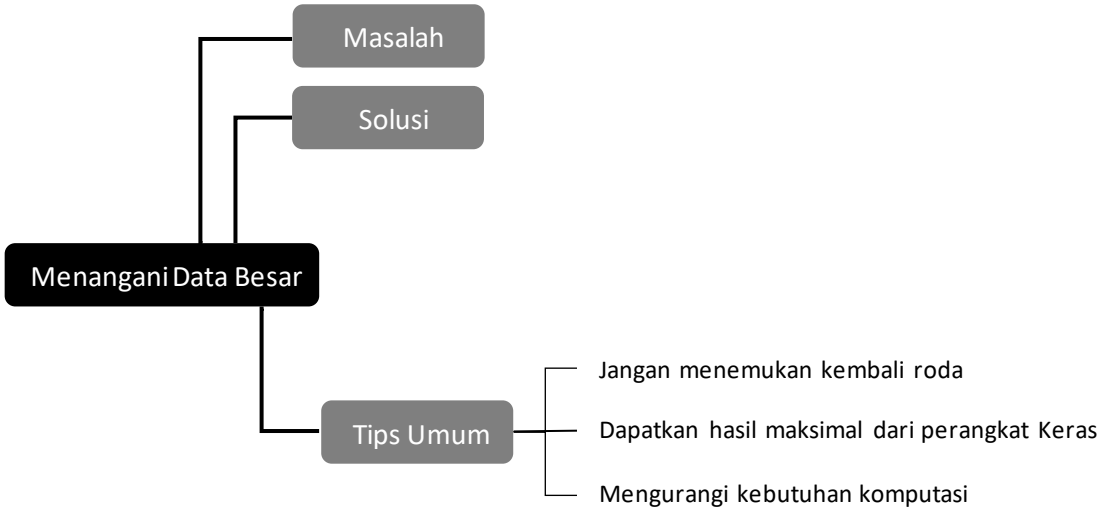
4.3 TIP PEMROGRAMAN UMUM UNTUK MENANGANI KUMPULAN DATA BESAR

Trik yang bekerja dalam konteks pemrograman umum masih berlaku untuk ilmu data. Beberapa kata mungkin sedikit berbeda, tetapi prinsip dasarnya sama untuk semua programmer. Bagian ini merangkum trik-trik yang penting dalam konteks ilmu data.

Anda dapat membagi trik umum menjadi tiga bagian, seperti yang ditunjukkan pada peta pikiran gambar 4.9:

- Jangan menemukan kembali roda. Gunakan alat dan perpustakaan yang dikembangkan oleh orang lain.
- Dapatkan hasil maksimal dari perangkat keras Anda. Mesin Anda tidak pernah digunakan secara maksimal; dengan adaptasi sederhana Anda dapat membuatnya bekerja lebih keras.

- Kurangi kebutuhan komputasi. Kurangi kebutuhan memori dan pemrosesan Anda sebanyak mungkin.



Gambar 4.9 Tinjauan praktik terbaik pemrograman umum saat bekerja dengan data besar

"Jangan menemukan kembali roda" lebih mudah diucapkan daripada dilakukan ketika dihadapkan dengan masalah tertentu, tetapi pikiran pertama Anda harus selalu, 'Orang lain pasti pernah menghadapi masalah yang sama ini sebelum saya.

4.3.1 Jangan menemukan kembali roda

"Jangan ulangi siapa pun" mungkin lebih baik daripada "jangan ulangi dirimu sendiri". Tambahkan nilai dengan tindakan Anda: pastikan itu penting. Memecahkan masalah yang sudah dipecahkan adalah buang-buang waktu. Sebagai ilmuwan data, Anda memiliki dua aturan besar yang dapat membantu Anda menangani data besar dan membuat Anda jauh lebih produktif, misalnya:

- Memanfaatkan kekuatan database. Reaksi pertama yang dimiliki sebagian besar ilmuwan data saat bekerja dengan kumpulan data besar adalah menyiapkan tabel dasar analitik mereka di dalam database. Cara ini bekerja dengan baik ketika fitur yang ingin Anda siapkan terbilang sederhana. Saat persiapan ini melibatkan pemodelan tingkat lanjut, cari tahu apakah mungkin menggunakan fungsi dan prosedur yang ditentukan pengguna. Contoh terakhir dari bab ini adalah tentang mengintegrasikan database ke dalam alur kerja Anda.
- Gunakan pustaka yang dioptimalkan. Membuat perpustakaan seperti Mahout, Weka, dan algoritme pembelajaran mesin lainnya membutuhkan waktu dan pengetahuan. Mereka sangat dioptimalkan dan menggabungkan praktik terbaik dan teknologi canggih. Habiskan waktu Anda untuk menyelesaikan sesuatu, bukan untuk menemukan kembali dan mengulangi upaya orang lain, kecuali demi memahami cara kerja sesuatu.

Maka Anda harus mempertimbangkan batasan perangkat keras Anda.

4.3.2 Dapatkan hasil maksimal dari perangkat keras Anda

Sumber daya di komputer dapat menganggur, sedangkan sumber daya lainnya digunakan secara berlebihan. Ini memperlambat program dan bahkan dapat membuatnya

gagal. Terkadang mungkin (dan perlu) untuk mengalihkan beban kerja dari sumber daya yang kelebihan beban ke sumber daya yang kurang dimanfaatkan dengan menggunakan teknik berikut:

- Beri makan data terkompresi CPU. Trik sederhana untuk menghindari kelaparan CPU adalah dengan memberi makan data terkompresi CPU alih-alih data (mentah) yang digelembungkan. Ini akan mengalihkan lebih banyak pekerjaan dari hard disk ke CPU, persis seperti yang ingin Anda lakukan, karena hard disk tidak dapat mengikuti CPU di sebagian besar arsitektur komputer modern.
- Manfaatkan GPU. Terkadang CPU Anda dan bukan memori Anda yang menjadi penghambat. Jika komputasi Anda dapat diparalelkan, Anda dapat memanfaatkan peralihan ke GPU. Ini memiliki throughput yang jauh lebih tinggi untuk perhitungan daripada CPU. GPU sangat efisien dalam pekerjaan yang dapat diparalelkan tetapi memiliki lebih sedikit cache daripada CPU. Tetapi tidak ada gunanya beralih ke GPU jika masalahnya adalah hard disk Anda. Beberapa paket Python, seperti Theano dan NumbaPro, akan menggunakan GPU tanpa banyak usaha pemrograman. Jika ini tidak cukup, Anda dapat menggunakan paket CUDA (Compute Unified Device Architecture) seperti PyCUDA. Ini juga merupakan trik terkenal dalam penambangan bitcoin, jika Anda tertarik untuk menghasilkan uang sendiri.
- Gunakan banyak utas. Masih mungkin untuk memparalelkan perhitungan pada CPU Anda. Anda dapat mencapai ini dengan utas Python normal.

4.3.3 Kurangi kebutuhan komputasi Anda

"Bekerja cerdas + keras = prestasi." Ini juga berlaku untuk program yang Anda tulis. Cara terbaik untuk menghindari masalah data yang besar adalah dengan menghapus sebanyak mungkin pekerjaan di depan dan membiarkan komputer bekerja hanya pada bagian yang tidak dapat dilewati. Daftar berikut berisi metode untuk membantu Anda mencapai hal ini:

- Buat profil kode Anda dan pulihkan potongan kode yang lambat. Tidak setiap bagian dari kode Anda perlu dioptimalkan; gunakan profiler untuk mendeteksi bagian yang lambat di dalam program Anda dan perbaiki bagian ini.
- Gunakan kode yang telah dikompilasi jika memungkinkan, terutama jika melibatkan loop. Kapan pun memungkinkan, gunakan fungsi dari paket yang dioptimalkan untuk komputasi numerik alih-alih mengimplementasikan semuanya sendiri. Kode dalam paket ini seringkali sangat dioptimalkan dan dikompilasi.
- Jika tidak, kompilasi sendiri kodenya. Jika Anda tidak dapat menggunakan paket yang sudah ada, gunakan kompiler just-in-time atau implementasikan bagian paling lambat dari kode Anda dalam bahasa tingkat rendah seperti C atau Fortran dan integrasikan ini dengan basis kode Anda. Jika Anda mengambil langkah ke bahasa tingkat rendah (bahasa yang lebih dekat dengan kode byte komputer universal), pelajari cara bekerja dengan pustaka komputasi seperti LAPACK, BLAST, Intel MKL, dan ATLAS. Ini sangat dioptimalkan, dan sulit untuk mencapai kinerja yang serupa dengannya.
- Hindari menarik data ke dalam memori. Saat Anda bekerja dengan data yang tidak sesuai dengan memori Anda, hindari memasukkan semuanya ke dalam memori. Cara sederhana untuk melakukan ini adalah dengan membaca data dalam potongan dan mem-parsing

data dengan cepat. Ini tidak akan berfungsi pada setiap algoritme tetapi memungkinkan penghitungan pada kumpulan data yang sangat besar.

- Gunakan generator untuk menghindari penyimpanan data perantara. Generator membantu Anda mengembalikan data per observasi, bukan dalam batch. Dengan cara ini Anda menghindari menyimpan hasil antara.
- Gunakan data sesedikit mungkin. Jika tidak ada algoritme skala besar yang tersedia dan Anda tidak ingin mengimplementasikan sendiri teknik seperti itu, Anda masih dapat melatih data Anda hanya pada sampel data asli.
- Gunakan keterampilan matematika Anda untuk menyederhanakan perhitungan sebanyak mungkin. Ambil persamaan berikut, misalnya: $(a + b)^2 = a^2 + 2ab + b^2$. Ruas kiri akan dihitung lebih cepat daripada ruas kanan persamaan; bahkan untuk contoh sepele ini, bisa membuat perbedaan ketika berbicara tentang potongan data yang besar.

4.4 STUDI KASUS 1: MEMPREDIKSI URL BERBAHAYA

Internet mungkin adalah salah satu penemuan terbesar di zaman modern. Ini telah mendorong perkembangan umat manusia, tetapi tidak semua orang menggunakan penemuan hebat ini dengan niat baik. Banyak perusahaan (Google, salah satunya) mencoba melindungi kami dari penipuan dengan mendeteksi situs web jahat untuk kami. Melakukannya bukanlah tugas yang mudah, karena internet memiliki miliaran halaman web untuk dipindai. Dalam studi kasus ini, kami akan menunjukkan cara bekerja dengan kumpulan data yang tidak lagi muat di memori.

Apa yang akan kita gunakan

- **Data**—Data dalam studi kasus ini tersedia sebagai bagian dari proyek penelitian. Proyek ini berisi data dari 120 hari, dan setiap pengamatan memiliki sekitar 3.200.000 fitur. Variabel target berisi 1 jika itu adalah situs web berbahaya dan -1 jika tidak. Untuk informasi lebih lanjut, silakan lihat "Melampaui Daftar Hitam: Belajar Mendeteksi Situs Web Berbahaya dari URL yang Mencurigakan."2
- **Pustaka Scikit-learn**—Pustaka ini harus sudah terpasang di lingkungan Python Anda saat ini, karena kita menggunakannya di bab sebelumnya.

Seperti yang Anda lihat, kami tidak membutuhkan banyak hal untuk kasus ini, jadi mari selami lebih dalam.

4.4.1 Langkah 1: Menentukan tujuan penelitian

Tujuan dari proyek kami adalah untuk mendeteksi apakah URL tertentu dapat dipercaya atau tidak. Karena datanya sangat besar, kami bertujuan melakukannya dengan cara yang ramah memori. Pada langkah berikutnya pertama-tama kita akan melihat apa yang terjadi jika kita tidak menyibukkan diri dengan masalah memori (RAM).

4.4.2 Langkah 2: Mendapatkan data URL

Mulailah dengan mengunduh data dari http://sysnet.ucsd.edu/projects/url/#datasets dan letakkan di folder. Pilih data dalam format SVMLight. SVMLight adalah format berbasis teks dengan satu pengamatan per baris. Untuk menghemat ruang, ini menghilangkan angka nol.

Daftar berikut dan gambar 4.10 menunjukkan apa yang terjadi ketika Anda mencoba membaca dalam 1 file dari 120 dan membuat matriks normal seperti yang diharapkan oleh sebagian besar algoritme. Metode todense() mengubah data dari format file khusus menjadi matriks normal di mana setiap entri berisi nilai.

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-532-d196c05088ce> in <module>()
      5 print "there are %d files" % len(files)
      6 X,y = load_svmlight_file(files[0] ,n_features=3500000)
----> 7 X.todense()
```

Gambar 4.10 Kesalahan memori saat mencoba mengambil kumpulan data besar ke dalam memori

Listing 4.5 Generating an out-of-memory error

```
import glob
from sklearn.datasets import load_svmlight_file
files = glob.glob('C:\Users\Gebruiker\Downloads\
url_svmlight.tar\url_svmlight\*.svm')
files = glob.glob('C:\Users\Gebruiker\Downloads\
url_svmlight\url_svmlight\*.svm')
print "there are %d files" % len(files)
X,y = load_svmlight_file(files[0],n_features=3231952)
X.todense()
```

Points to files (Linux).

Points to files (Windows: tar file needs to be untarred first).

Indication of number of files.

Loads files.

The data is a big, but sparse, matrix. By turning it into a dense matrix (every 0 is represented in the file), we create an out-of-memory error.

Kami mendapatkan kesalahan kehabisan memori. Artinya, kecuali Anda menjalankan kode ini di mesin besar. Setelah beberapa trik, Anda tidak akan lagi mengalami masalah memori ini dan akan mendeteksi 97% situs berbahaya.

Alat Dan Teknik

Kami mengalami kesalahan memori saat memuat satu file—masih 119 lagi. Untungnya, kami memiliki beberapa trik di lengan baju kami. Mari kita coba teknik ini selama studi kasus:

- Gunakan representasi data yang jarang.
- Beri makan data terkompresi algoritme alih-alih data mentah.
- Gunakan algoritme online untuk membuat prediksi.

Kami akan membahas lebih dalam setiap "trik" saat kami menggunakannya. Sekarang kita memiliki data kita secara lokal, mari kita akses. Langkah 3 dari proses ilmu data, penyiapan dan pembersihan data, tidak diperlukan dalam kasus ini karena URL telah dibersihkan sebelumnya. Namun, kami membutuhkan suatu bentuk eksplorasi sebelum melepaskan algoritme pembelajaran kami.

4.4.3 Langkah 4: Eksplorasi data

Untuk melihat apakah kita bahkan dapat menerapkan trik pertama kita (representasi jarang), kita perlu mencari tahu apakah data memang mengandung banyak angka nol. Kita dapat memeriksa ini dengan potongan kode berikut:

```
print "number of non-zero entries %2.6f"  % float((X.nnz)/(float(X.shape[0])
* float(X.shape[1])))
```

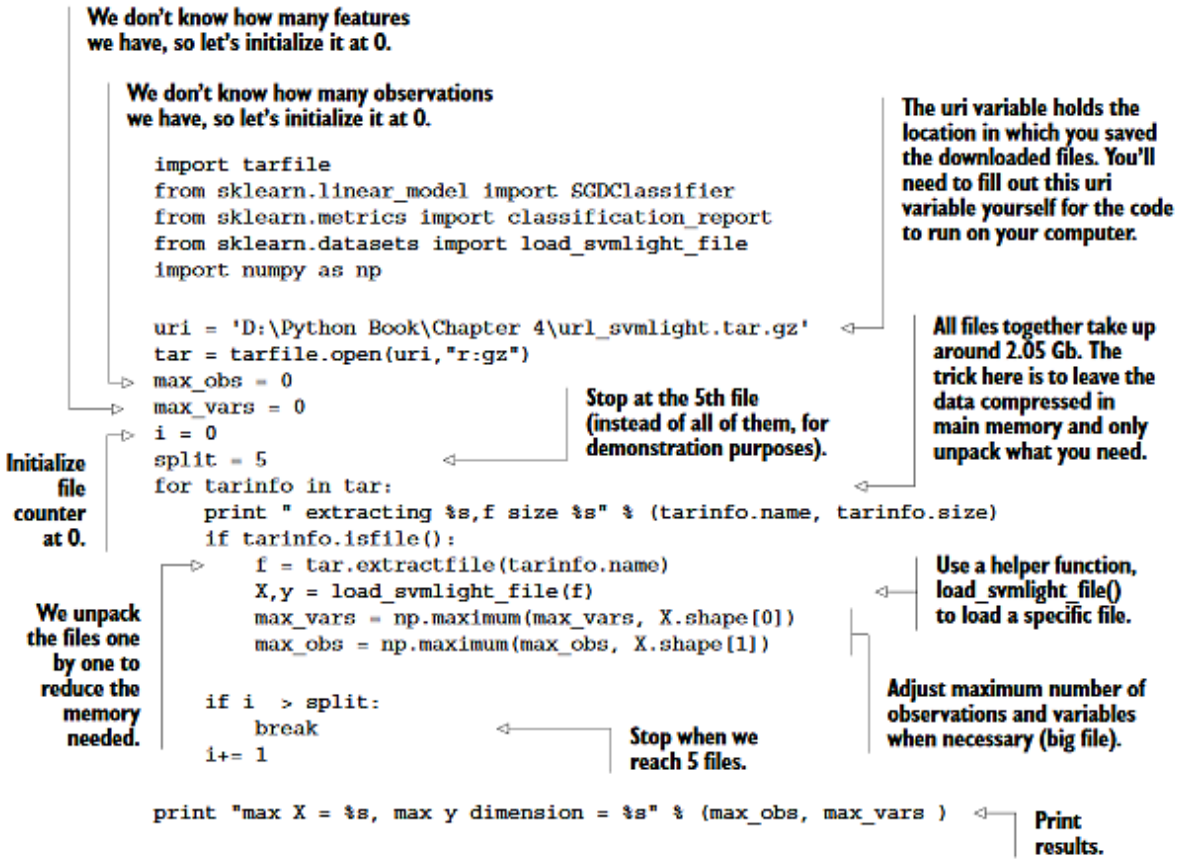
Ini menghasilkan yang berikut:

```
number of non-zero entries 0.000033
```

Data yang mengandung sedikit informasi dibandingkan dengan nol disebut data jarang. Ini dapat disimpan dengan lebih ringkas jika Anda menyimpan data sebagai [(0,0,1),(4,4,1)] alih-alih

```
[[1,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,1]]
```

Listing 4.6 Checking data size



Salah satu format file yang menerapkan ini adalah SVMLight, dan itulah alasan kami mengunduh data dalam format ini. Kami belum selesai, karena kami perlu merasakan dimensi dalam data. Untuk mendapatkan informasi ini kita sudah perlu menjaga agar data tetap terkompresi sambil memeriksa jumlah maksimum pengamatan dan variabel. Kita juga perlu membaca data file demi file. Dengan cara ini Anda mengkonsumsi lebih sedikit memori. Trik

kedua adalah memberi makan file terkompresi CPU. Dalam contoh kami, ini sudah dikemas dalam format tar.gz. Anda membongkar file hanya saat Anda membutuhkannya, tanpa menulisnya ke hard disk (bagian paling lambat dari komputer Anda). Untuk contoh kami, ditampilkan dalam daftar 4.6, kami hanya akan bekerja pada 5 file pertama, tetapi jangan ragu untuk menggunakan semuanya.

Bagian dari kode membutuhkan penjelasan tambahan. Dalam kode ini kita mengulang file svm di dalam arsip tar. Kami membongkar file satu per satu untuk mengurangi memori yang dibutuhkan. Karena file-file ini dalam format SVM, kami menggunakan helper, functionload_svmlight_file() untuk memuat file tertentu. Kemudian kita dapat melihat berapa banyak pengamatan dan variabel yang dimiliki file tersebut dengan memeriksa bentuk kumpulan data yang dihasilkan. Berbekal informasi ini, kita dapat beralih ke pembuatan model.

4.4.4 Langkah 5: Pembuatan model

Sekarang kita menyadari dimensi data kita, kita dapat menerapkan dua trik yang sama (representasi renggang dari file terkompresi) dan menambahkan yang ketiga (menggunakan algoritma online), dalam daftar berikut. Ayo temukan situs web berbahaya itu!

Listing 4.7 Creating a model to distinguish the malicious from the normal URLs

We know number of features from data exploration.

The target variable can be 1 or -1. "1": website safe to visit, "-1": website unsafe.

Set up stochastic gradient classifier.

```
classes = [-1,1]
sgd = SGDClassifier(loss="log")
n_features=3231952
split = 5
i = 0
for tarinfo in tar:
    if i  > split:
        break
    if tarinfo.isfile():
        f = tar.extractfile(tarinfo.name)
        X,y = load_svmlight_file(f,n_features=n_features)
        if i < split:
            sgd.partial_fit(X, y, classes=classes)
        if i == split:
            print classification_report(sgd.predict(X),y)
    i += 1
```

All files together take up around 2.05 Gb. The trick here is to leave data compressed in main memory and only unpack what you need.

We unpack the files one by one to reduce the memory needed.

Use a helper function, load_svmlight_file() to load a specific file.

Third important thing is online algorithm. It can be fed data points file by file (batches).

Initialize file counter at 0.

Stop when we reach 5 files and print results.

Stop at 5th file (instead of all of them, for demonstration purposes).

Kode dalam daftar sebelumnya terlihat cukup mirip dengan yang kita lakukan sebelumnya, selain dari pengklasifikasi penurunan gradien stokastik SGDClassifier(). Di sini, kami melatih algoritme secara iteratif dengan menampilkan pengamatan dalam satu file dengan fungsi partial_fit() . Mengulangi hanya 5 file pertama di sini memberikan hasil yang ditunjukkan pada tabel 4.1. Tabel menunjukkan tindakan diagnostik klasifikasi: presisi, recall, skor F1, dan acceptance.

Tabel 4.1 Klasifikasi masalah: Apakah sebuah website dapat dipercaya atau tidak?

	presisi	recall	skor-f1	acceptance
-1	0,97	0,99	0,98	14045
1	0,97	0,94	0,96	5955
rata-rata/total	0,97	0,97	0,97	20000

Hanya 3% (1 - 0,97) dari situs jahat yang tidak terdeteksi (presisi), dan 6% (1 - 0,94) situs yang terdeteksi salah dituduh (recall). Ini adalah hasil yang layak, jadi kami dapat menyimpulkan bahwa metodologinya berhasil. Jika kami menjalankan kembali analisis, hasilnya mungkin sedikit berbeda, karena algoritme dapat menyatu sedikit berbeda. Jika Anda tidak keberatan menunggu beberapa saat, Anda dapat menggunakan kumpulan data lengkap. Anda sekarang dapat menangani semua data tanpa masalah. Kami tidak akan memiliki langkah keenam (presentasi atau otomatisasi) dalam studi kasus ini.

Sekarang mari kita lihat aplikasi kedua dari teknik kita; kali ini Anda akan membuat sistem pemberi rekomendasi di dalam database. Untuk contoh sistem pemberi rekomendasi yang terkenal, kunjungi situs web Amazon. Saat menjelajah, Anda akan segera dihadapkan pada rekomendasi: “Orang yang membeli produk ini juga membeli…”

4.5 STUDI KASUS 2: MEMBANGUN SISTEM PEMBERI REKOMENDASI DATABASE

Pada kenyataannya sebagian besar data yang Anda kerjakan disimpan dalam basis data relasional, tetapi sebagian besar basis data tidak cocok untuk penambangan data. Namun seperti yang ditunjukkan dalam contoh ini, memungkinkan untuk mengadaptasi teknik kami sehingga Anda dapat melakukan sebagian besar analisis di dalam database itu sendiri, sehingga mendapat keuntungan dari pengoptimal kueri database, yang akan mengoptimalkan kode untuk Anda. Dalam contoh ini kita akan membahas cara menggunakan struktur data tabel hash dan cara menggunakan Python untuk mengontrol alat lain.

4.5.1 Alat dan teknik yang dibutuhkan

Sebelum masuk ke studi kasus, kita perlu melihat sekilas alat yang diperlukan dan latar belakang teoretis untuk apa yang akan kita lakukan di sini.

Peralatan

- **Database MySQL —** Membutuhkan database MySQL untuk bekerja dengannya. Jika Anda belum menginstal server komunitas MySQL, Anda dapat mengunduhnya dari www.mysql.com. Lampiran C: “Installing a MySQL server” menjelaskan cara mengaturnya.
- **Koneksi database MySQL Pustaka Python** — Untuk menyambung ke server ini dari Python, Anda juga perlu menginstal SQLAlchemy atau pustaka lain yang mampu berkomunikasi dengan MySQL. Kami menggunakan MySQLdb. Di Windows Anda tidak dapat langsung menggunakan Conda untuk menginstalnya. Pertama instal Binstar (layanan manajemen paket lain) dan cari paket mysql-python yang sesuai untuk pengaturan Python Anda.

```
conda install binstar
binstar search -t conda mysql-python
```

Perintah berikut yang dimasukkan ke dalam baris perintah Windows berfungsi untuk kami (setelah mengaktifkan lingkungan Python):

```
conda install --channel https://conda.binstar.org/krisvanneste mysql-python
```

Sekali lagi, jangan ragu untuk menggunakan perpustakaan SQLAlchemy jika itu adalah sesuatu yang membuat Anda lebih nyaman.

- Kita juga membutuhkan library pandas python, tetapi seharusnya sudah terinstal sekarang.

Dengan infrastruktur yang tersedia, mari selami beberapa tekniknya.

Teknik

Sistem pemberi rekomendasi sederhana akan mencari pelanggan yang telah menyewa film serupa seperti yang Anda miliki dan kemudian menyarankan film yang telah ditonton orang lain tetapi belum Anda tonton. Teknik ini disebut k-nearest neighbor dalam machine learning.

Pelanggan yang berperilaku serupa dengan Anda belum tentu merupakan pelanggan yang paling mirip. Anda akan menggunakan teknik untuk memastikan bahwa Anda dapat menemukan pelanggan serupa (optimal lokal) tanpa jaminan bahwa Anda telah menemukan pelanggan terbaik (optimal global). Sebuah teknik umum yang digunakan untuk memecahkan ini disebut Hashing Sensitif Lokalitas.

Gagasan di balik Pencirian Sensitif-Lokal sederhana: Bangun fungsi yang memetakan pelanggan serupa secara berdekatan (mereka dimasukkan ke dalam keranjang dengan label yang sama) dan pastikan bahwa objek yang berbeda diletakkan di keranjang yang berbeda.
Inti dari ide ini adalah fungsi yang melakukan pemetaan. Fungsi ini disebut fungsi hash: fungsi yang memetakan berbagai input ke output tetap. Fungsi hash paling sederhana menggabungkan nilai dari beberapa kolom acak. Tidak masalah berapa banyak kolom (input yang dapat diskalakan); itu membawanya kembali ke satu kolom (keluaran tetap).

Anda akan menyiapkan tiga fungsi hash untuk menemukan pelanggan serupa. Tiga fungsi mengambil nilai dari tiga film:

- Fungsi pertama mengambil nilai film 10, 15, dan 28.
- Fungsi kedua mengambil nilai film 7, 18, dan 22.
- Fungsi terakhir mengambil nilai film 16, 19, dan 30.

Ini akan memastikan bahwa pelanggan yang berada di keranjang yang sama berbagi setidaknya beberapa film. Tetapi pelanggan dalam satu keranjang mungkin masih berbeda pada film yang tidak termasuk dalam fungsi hashing. Untuk mengatasi ini, Anda masih perlu membandingkan pelanggan di dalam keranjang satu sama lain. Untuk ini, Anda perlu membuat ukuran jarak baru. Jarak yang akan Anda gunakan untuk membandingkan pelanggan disebut jarak hamming. Jarak hamming digunakan untuk menghitung berapa banyak dua

string berbeda. Jarak didefinisikan sebagai jumlah karakter yang berbeda dalam sebuah string. Tabel 4.2 menawarkan beberapa contoh jarak hamming.

Tabel 4.2 Contoh perhitungan jarak hamming

Tali 1	Tali 2	Jarak hamming
Topi	Kucing	1
Topi	Gila	2
Harimau	Harimau	2
Paris	Roma	5

Membandingkan banyak kolom adalah operasi yang mahal, jadi Anda memerlukan trik untuk mempercepatnya. Karena kolom berisi variabel biner (0 atau 1) untuk menunjukkan apakah pelanggan telah membeli film atau tidak, Anda dapat menggabungkan informasi tersebut sehingga informasi yang sama dimuat di kolom baru. Tabel 4.3 menunjukkan variabel "film" yang berisi informasi sebanyak semua kolom film digabungkan.

Tabel 4.3 Menggabungkan informasi dari berbagai kolom ke dalam kolom film. Ini juga cara kerja DNA: semua informasi dalam rangkaian panjang.

Kolom 1	Film 1	Film 2	Film 3	Film 4	Film
Pelanggan 1	1	0	1	1	1011
Pelanggan 2	0	0	0	1	0001

Ini memungkinkan Anda untuk menghitung jarak hamming jauh lebih efisien. Dengan menangani operator ini sedikit, Anda dapat mengeksploitasi operator XOR. Hasil dari operator XOR (^) adalah sebagai berikut:

```
1^1 = 0
1^0 = 1
0^1 = 1
0^0 = 0
```

Dengan ini, proses untuk menemukan pelanggan serupa menjadi sangat sederhana. Mari kita lihat dulu dalam kode semu:

Prapemrosesan:

1. Tentukan fungsi p (misalnya, 3) yang memilih entri k (misalnya, 3) dari vektor film. Disini kita mengambil 3 fungsi (p) yang masing-masing mengambil 3 (k) film.
2. Terapkan fungsi ini ke setiap titik dan simpan di kolom terpisah. (Dalam literatur setiap fungsi disebut fungsi hash dan setiap kolom akan menyimpan ember.)

Titik pertanyaan q:

1. Terapkan fungsi p yang sama ke titik (pengamatan) q yang ingin Anda kueri.
2. Ambil untuk setiap fungsi titik-titik yang sesuai dengan hasil di keranjang yang sesuai.

Hentikan ketika Anda telah mengambil semua poin dalam keranjang atau mencapai 2p poin (misalnya 10 jika Anda memiliki 5 fungsi).

3. Hitung jarak untuk setiap titik dan kembalikan titik dengan jarak minimum.

Mari kita lihat implementasi sebenarnya di Python untuk memperjelas semuanya.

4.5.2 Langkah 1: Pertanyaan penelitian

Katakanlah Anda bekerja di toko video dan manajer bertanya kepada Anda apakah mungkin menggunakan informasi tentang film yang disewa orang untuk memprediksi film lain yang mungkin mereka sukai. Bos Anda telah menyimpan data dalam database MySQL, dan terserah Anda untuk melakukan analisis. Yang dia maksud adalah sistem pemberi rekomendasi, sistem otomatis yang mempelajari preferensi orang dan merekomendasikan film dan produk lain yang belum dicoba oleh pelanggan. Tujuan dari studi kasus kami adalah untuk menciptakan sistem rekomendasi yang ramah memori. Kami akan mencapai ini menggunakan database dan beberapa trik tambahan. Kami akan membuat sendiri data untuk studi kasus ini sehingga kami dapat melewati langkah pengambilan data dan langsung beralih ke persiapan data. Dan setelah itu kita dapat melewatkan langkah eksplorasi data dan langsung menuju ke pembuatan model.

4.5.3 Langkah 2: Persiapan data

Data yang dikumpulkan bos Anda ditunjukkan pada tabel 4.4. Kami akan membuat data ini sendiri untuk kepentingan demonstrasi.

Tabel 4.4 Kutipan dari database klien dan film yang disewa pelanggan

Pelanggan	Film 1	Film 2	Film 3	...	Film 32
Jack Dani	1	0	0		**1**
Wilhelmson	1	1	0		**1**
…					
Jane Dane	0	0	1		**0**
Xi liu	0	0	0		**1**
Eros Mazo	1	1	0		**1**
…					

Untuk setiap pelanggan, Anda mendapatkan indikasi apakah mereka pernah menyewa film tersebut sebelumnya (1) atau tidak (0). Mari kita lihat apa lagi yang Anda perlukan agar Anda dapat memberi atasan Anda sistem pemberi rekomendasi yang diinginkannya.

Pertama mari kita hubungkan Python ke MySQL untuk membuat data kita. Buat koneksi ke MySQL menggunakan nama pengguna dan kata sandi Anda. Dalam daftar berikut kami menggunakan database yang disebut "test". Ganti nama pengguna, kata sandi, dan database dengan nilai yang sesuai untuk pengaturan Anda dan ambil koneksi dan kursor. Kursor database adalah struktur kontrol yang mengingat di mana Anda saat ini berada di database.

Listing 4.8 Creating customers in the database

```
import MySQLdb
import pandas as pd

user = '****'
password = '****'
database = 'test'
mc = MySQLdb.connect('localhost',user,password,database)
cursor = mc.cursor()

nr_customers = 100
colnames = ["movie%d" %i for i in range(1,33)]
pd.np.random.seed(2015)
generated_customers = pd.np.random.randint(0,2,32 *
     nr_customers).reshape(nr_customers,32)

data = pd.DataFrame(generated_customers, columns = list(colnames))
data.to_sql('cust',mc, flavor = 'mysql', index = True, if_exists =
     'replace', index_label = 'cust_id')
```

First we establish the connection; you'll need to fill out your own username, password, and schema-name (variable "database").

Next we simulate a database with customers and create a few observations.

Store the data inside a Pandas data frame and write the data frame in a MySQL table called "cust". If this table already exists, replace it.

Kami membuat 100 pelanggan dan menetapkan secara acak apakah mereka menonton film tertentu atau tidak, dan kami memiliki total 32 film. Data pertama kali dibuat dalam kerangka data Pandas tetapi kemudian diubah menjadi kode SQL. Catatan: Anda mungkin menemukan peringatan saat menjalankan kode ini. Peringatan menyatakan: Rasa "mysql" dengan koneksi DBAPI sudah tidak digunakan lagi dan akan dihapus di versi mendatang. MySQL akan didukung lebih lanjut dengan mesin SQLAlchemy. Jangan ragu untuk beralih ke SQLAlchemy atau perpustakaan lain. Kami akan menggunakan SQLAlchemy di bab lain, tetapi menggunakan MySQLdb di sini untuk memperluas contoh.

Untuk mengkueri database secara efisien nanti, kami memerlukan persiapan data tambahan, termasuk hal-hal berikut:

- Membuat string bit. String bit adalah versi terkompresi dari konten kolom (nilai 0 dan 1). Pertama, nilai biner ini digabungkan; kemudian string bit yang dihasilkan ditafsirkan kembali sebagai angka. Ini mungkin terdengar abstrak sekarang tetapi akan menjadi lebih jelas dalam kodenya.
- Mendefinisikan fungsi hash. Fungsi hash sebenarnya akan membuat string bit.
- Menambahkan indeks ke tabel, untuk mempercepat pengambilan data.

Membuat Bit String

Sekarang Anda membuat tabel perantara yang cocok untuk kueri, menerapkan fungsi hash, dan merepresentasikan urutan bit sebagai angka desimal. Akhirnya, Anda bisa menempatkannya di sebuah meja. Pertama, Anda perlu membuat string bit. Anda perlu mengonversi string "11111111" menjadi nilai biner atau numerik agar fungsi hamming berfungsi. Kami memilih representasi numerik, seperti yang ditunjukkan pada daftar berikutnya.

Listing 4.9 Creating bit strings

We represent the string as a numeric value. The string will be a concatenation of zeros (0) and ones (1) because these indicate whether someone has seen a certain movie or not. The strings are then regarded as bit code. For example: 0011 is the same as the number 3. What def createNum() does: takes in 8 values, concatenates these 8 column values and turns them into a string, then turns the byte code of the string into a number.

```
def createNum(x1,x2,x3,x4,x5,x6,x7,x8):
    return  [int('%d%d%d%d%d%d%d%d' % (i1,i2,i3,i4,i5,i6,i7,i8),2)
for (i1,i2,i3,i4,i5,i6,i7,i8) in zip(x1,x2,x3,x4,x5,x6,x7,x8)]

assert int('1111',2) == 15
assert int('1100',2) == 12
assert createNum([1,1],[1,1],[1,1],[1,1],[1,1],[1,1],[1,0],[1,0])
     == [255,252]

store = pd.DataFrame()
store['bit1'] = createNum(data.movie1,
     data.movie2,data.movie3,data.movie4,data.movie5,
data.movie6,data.movie7,data.movie8)
store['bit2'] = createNum(data.movie9,
     data.movie10,data.movie11,data.movie12,data.movie13,
data.movie14,data.movie15,data.movie16)
store['bit3'] = createNum(data.movie17,
     data.movie18,data.movie19,data.movie20,data.movie21,
data.movie22,data.movie23,data.movie24)
store['bit4'] = createNum(data.movie25,
     data.movie26,data.movie27,data.movie28,data.movie29,
data.movie30,data.movie31,data.movie32)
```

Translate the movie column to 4 bit strings in numeric form. Each bit string represents 8 movies. 4*8 = 32 movies. Note: you could use a 32-bit string instead of 4*8 to keep the code short.

Test if the function works correctly. Binary code 1111 is the same as 15 (=1*8+1*4+1*2+1*1). If the assert fails, it will raise an assert error; otherwise nothing will happen.

Dengan mengonversi informasi dari 32 kolom menjadi 4 angka, kami mengompresnya untuk pencarian nanti. Gambar 4.11 menunjukkan apa yang kami dapatkan saat meminta 2 pengamatan pertama (riwayat tontonan film pelanggan) dalam format baru ini.

```
store[0:2]
```

Langkah selanjutnya adalah membuat fungsi hash, karena mereka akan memungkinkan kita mengambil sampel data yang akan kita gunakan untuk menentukan apakah dua pelanggan memiliki perilaku yang sama.

	bit1	bit2	bit3	bit4
0	10	62	42	182
1	23	28	223	180

Gambar 4.11 2 informasi pelanggan pertama pada semua 32 film setelah string bit ke konversi numerik

Membuat Fungsi Hash

Fungsi hash yang kami buat mengambil nilai film untuk pelanggan. Kami memutuskan di bagian teori studi kasus ini untuk membuat 3 fungsi hash: fungsi pertama menggabungkan film 10, 5, dan 18; yang kedua menggabungkan film 7, 18, dan 22; dan yang ketiga menggabungkan 16, 19, dan 30. Terserah Anda jika ingin memilih yang lain; ini dapat dipilih secara acak. Daftar kode berikut menunjukkan bagaimana hal ini dilakukan.

Listing 4.10 Creating hash functions

Define hash function (it is exactly like the createNum() function without the final conversion to a number and for 3 columns instead of 8).

```
def hash_fn(x1,x2,x3):
    return [b'%d%d%d' % (i,j,k) for (i,j,k) in zip(x1,x2,x3)]

assert hash_fn([1,0],[1,1],[0,0]) == [b'110',b'010']

store['bucket1'] = hash_fn(data.movie10, data.movie15,data.movie28)
store['bucket2'] = hash_fn(data.movie7, data.movie18,data.movie22)
store['bucket3'] = hash_fn(data.movie16, data.movie19,data.movie30)
store.to_sql('movie_comparison',mc, flavor = 'mysql', index = True,
          index_label = 'cust_id', if_exists = 'replace')
```

Test if it works correctly (if no error is raised, it works). It's sampling on columns but all observations will be selected.

Store this information in database.

Create hash values from customer movies, respectively [10,15, 28], [7,18, 22], [16,19, 30].

Fungsi hash menggabungkan nilai dari film yang berbeda menjadi nilai biner seperti yang terjadi sebelumnya di fungsi createNum(), hanya saja kali ini kami tidak mengonversi ke angka dan kami hanya mengambil 3 film, bukan 8 sebagai masukan. Fungsi penegasan menunjukkan bagaimana menggabungkan 3 nilai untuk setiap pengamatan. Ketika klien telah membeli film 10 tetapi bukan film 15 dan 28, itu akan mengembalikan b'100' untuk keranjang 1. Ketika klien membeli film 7 dan 18, tetapi bukan 22, itu akan mengembalikan b'110' untuk ember 2. Jika kita melihat hasil saat ini kita melihat 4 variabel yang kita buat sebelumnya (bit1, bit2, bit3, bit4) dari 9 film yang dipilih sendiri (gambar 4.12).

	bit1	bit2	bit3	bit4	bucket1	bucket2	bucket3
0	10	62	42	182	011	100	011
1	23	28	223	180	001	111	001

Gambar 4.12 Informasi dari bit string compression dan 9 sample film

Trik terakhir yang akan kami terapkan adalah mengindeks tabel pelanggan sehingga pencarian terjadi lebih cepat.

Menambahkan Indeks Ke Tabel

Sekarang Anda harus menambahkan indeks untuk mempercepat pengambilan sesuai kebutuhan dalam sistem real-time. Ini ditunjukkan dalam daftar berikutnya.

Listing 4.11 Creating an index

```
def createIndex(column, cursor):
    sql = 'CREATE INDEX %s ON movie_comparison (%s);' % (column, column)
    cursor.execute(sql)

createIndex('bucket1',cursor)
createIndex('bucket2',cursor)
createIndex('bucket3',cursor)
```

Create function to easily create indices. Indices will quicken retrieval.

Put index on bit buckets.

Dengan data yang diindeks, kita sekarang dapat beralih ke "bagian pembuatan model". Dalam studi kasus ini tidak ada pembelajaran mesin aktual atau model statistik yang diterapkan. Sebagai gantinya, kami akan menggunakan teknik yang jauh lebih sederhana: perhitungan jarak string. Dua senar dapat dibandingkan dengan menggunakan jarak hamming seperti yang dijelaskan sebelumnya dalam pendahuluan teori studi kasus.

4.5.4 Langkah 5: Pembuatan model

Untuk menggunakan jarak hamming dalam database, kita perlu mendefinisikannya sebagai fungsi.

Menciptakan Fungsi Jarak Hamming

Kami menerapkan ini sebagai fungsi yang ditentukan pengguna. Fungsi ini dapat menghitung jarak untuk bilangan bulat 32-bit (sebenarnya 4*8), seperti yang ditunjukkan pada daftar berikut.

Listing 4.12 Creating the hamming distance

```
Sql = '''
CREATE FUNCTION HAMMINGDISTANCE(
  A0 BIGINT, A1 BIGINT, A2 BIGINT, A3 BIGINT,
  B0 BIGINT, B1 BIGINT, B2 BIGINT, B3 BIGINT
)

RETURNS INT DETERMINISTIC
RETURN
  BIT_COUNT(A0 ^ B0) +
  BIT_COUNT(A1 ^ B1) +
  BIT_COUNT(A2 ^ B2) +
  BIT_COUNT(A3 ^ B3); '''

cursor.execute(Sql)

Sql = '''Select hammingdistance(
    b'11111111',b'00000000',b'11011111',b'11111111'
,b'11111111',b'10001001',b'11011111',b'11111111'
)'''
pd.read_sql(Sql,mc)
```

The function is stored in a database. You can only do this once; running this code a second time will result in an error: OperationalError: (1304, 'FUNCTION HAMMINGDISTANCE already exists').

Define function. It takes 8 input arguments: 4 strings of length 8 for the first customer and another 4 strings of length 8 for the second customer. This way we can compare 2 customers side-by-side for 32 movies.

To check this function you can run this SQL statement with 8 fixed strings. Notice the "b" before each string, indicating that you're passing bit values. The outcome of this particular test should be 3, which indicates the series of strings differ in only 3 places.

This runs the query.

Jika semuanya baik-baik saja, output dari kode ini seharusnya 3. Sekarang setelah kita memiliki fungsi hamming distance, kita dapat menggunakannya untuk menemukan pelanggan yang mirip dengan pelanggan tertentu, dan inilah yang kita ingin aplikasi kita lakukan. Mari beralih ke bagian terakhir: memanfaatkan penyiapan kami sebagai semacam aplikasi.

4.5.5 Langkah 6: Presentasi dan otomatisasi

Sekarang setelah semuanya siap, aplikasi kita perlu melakukan dua langkah saat berhadapan dengan pelanggan tertentu:

- Cari pelanggan serupa.
- Menyarankan film yang belum ditonton oleh pelanggan berdasarkan apa yang telah dia tonton dan riwayat tontonan dari pelanggan serupa.

Hal pertama yang pertama: pilih sendiri pelanggan yang beruntung.

Menemukan Pelanggan yang Serupa

Saatnya melakukan kueri waktu nyata. Dalam daftar berikut, pelanggan 27 adalah orang yang bahagia yang akan memilihkan film berikutnya untuknya. Tapi pertama-tama kita perlu memilih pelanggan dengan riwayat menonton yang serupa.

Listing 4.13 Finding similar customers

We do two-step sampling. First sampling: index must be exactly the same as the one of the selected customer (is based on 9 movies). Selected people must have seen (or not seen) these 9 movies exactly like our customer did. Second sampling is a ranking based on the 4-bit strings. These take into account all the movies in the database.

Pick customer from database.

```
customer_id = 27
sql = "select * from movie_comparison where cust_id = %s" % customer_id
cust_data = pd.read_sql(sql,mc)
sql =  """ select cust_id,hammingdistance(bit1,
bit2,bit3,bit4,%s,%s,%s,%s) as distance
           from movie_comparison where bucket1 = '%s' or bucket2 ='%s'
or bucket3='%s' order by distance limit 3""" %
     (cust_data.bit1[0],cust_data.bit2[0],
cust_data.bit3[0], cust_data.bit4[0],
       cust_data.bucket1[0], cust_data.bucket2[0],cust_data.bucket3[0])
shortlist = pd.read_sql(sql,mc)
```

We show the 3 customers that most resemble customer 27. Customer 27 ends up first.

Tabel 4.5 menunjukkan pelanggan 2 dan 97 paling mirip dengan pelanggan 27. Jangan lupa bahwa data dihasilkan secara acak, sehingga siapa pun yang mereplikasi contoh ini mungkin mendapatkan hasil yang berbeda. Sekarang kami akhirnya dapat memilih film untuk ditonton pelanggan 27.

Tabel 4.5 Pelanggan yang paling mirip dengan pelanggan 27

	cust_id	jarak
0	27	0
1	2	8
2	97	9

Mencari Film Baru

Kami perlu melihat film yang pelanggan 27 belum pernah lihat, tetapi pelanggan terdekat memilikinya, seperti yang ditunjukkan pada daftar berikut. Ini juga merupakan

pemeriksaan yang baik untuk melihat apakah fungsi jarak Anda bekerja dengan benar. Meskipun ini mungkin bukan pelanggan terdekat, ini cocok dengan pelanggan 27. Dengan menggunakan indeks yang di-hash, Anda mendapatkan kecepatan luar biasa saat menanyakan database besar.

Listing 4.14 Finding an unseen movie

```
cust = pd.read_sql('select * from cust where cust_id in (27,2,97)',mc)
dif = cust.T
dif[dif[0] != dif[1]]
```

Select movies customer 27 didn't see yet.

Transpose for convenience.

Select movies customers 27, 2, 97 have seen.

Tabel 4.6 menunjukkan bahwa Anda dapat merekomendasikan film 12, 15, atau 31 berdasarkan perilaku pelanggan 2.

Tabel 4.6 Film dari pelanggan 2 dapat dijadikan saran untuk pelanggan 27.

	0	1	2
Cust_id	2	27	97
Film3	0	1	1
Film9	0	1	1
Film11	0	1	1
Film12	1	0	0
Film15	1	0	0
Film16	0	1	1
Film25	0	1	1
Film31	1	0	0

Pecandu film kami yang bahagia sekarang dapat memanjakan dirinya dengan film baru, yang disesuaikan dengan kesukaannya. Di bab berikutnya kita akan melihat data yang lebih besar dan melihat bagaimana kita dapat mengatasinya dengan menggunakan Horton Sandbox yang kita unduh di bab 1.

4.6 RINGKASAN

Bab ini membahas topik-topik berikut:

- Masalah utama yang dapat Anda hadapi saat bekerja dengan kumpulan data besar adalah sebagai berikut:
  – Tidak cukup memori
  – Program yang berjalan lama
  – Sumber daya yang membentuk kemacetan dan menyebabkan masalah kecepatan
- Ada tiga jenis solusi utama untuk masalah ini:

– Sesuaikan algoritme Anda.
– Gunakan struktur data yang berbeda.
– Andalkan alat dan perpustakaan.

- Tiga teknik utama dapat digunakan untuk mengadaptasi algoritme:
  – Sajikan data algoritme satu pengamatan pada satu waktu alih-alih memuat kumpulan data lengkap sekaligus.
  – Bagilah matriks menjadi matriks yang lebih kecil dan gunakan ini untuk membuat perhitungan Anda.
  – Menerapkan algoritma MapReduce (menggunakan pustaka Python seperti Hadoopy, Octopy, Disco, atau Dumbo).
- Tiga struktur data utama digunakan dalam ilmu data. Yang pertama adalah jenis matriks yang mengandung informasi yang relatif sedikit, yaitu matriks jarang. Yang kedua dan ketiga adalah struktur data yang memungkinkan Anda mengambil informasi dengan cepat dalam kumpulan data besar: fungsi hash dan struktur pohon.
- Python memiliki banyak alat yang dapat membantu Anda menangani kumpulan data yang besar. Beberapa alat akan membantu Anda dengan ukuran volume, yang lain akan membantu Anda memparalelkan perhitungan, dan yang lainnya mengatasi kecepatan Python yang relatif lambat itu sendiri. Juga mudah untuk menggunakan Python sebagai alat untuk mengontrol alat ilmu data lainnya karena Python sering dipilih sebagai bahasa untuk mengimplementasikan API.
- Praktik terbaik dari ilmu komputer juga berlaku dalam konteks ilmu data, sehingga penerapannya dapat membantu Anda mengatasi masalah yang Anda hadapi dalam konteks data besar.

BAB 5
MEMULAI BIG DATA

Dalam bab ini diharapkan mahasiswa mampu:

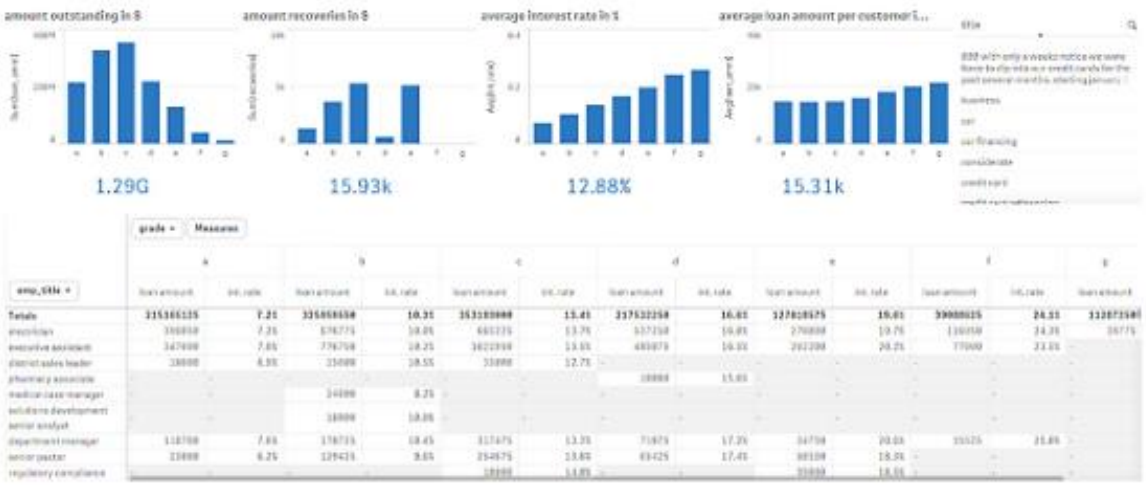
- Ambil langkah pertama Anda dengan dua aplikasi big data: Hadoop dan Spark
- Menggunakan Python untuk menulis pekerjaan big data
- Membangun dasbor interaktif yang terhubung ke data yang disimpan dalam database data besar

Selama dua bab terakhir, kami terus meningkatkan ukuran data. Di bab 3 kita bekerja dengan kumpulan data yang dapat masuk ke dalam memori utama sebuah komputer. Bab 4 memperkenalkan teknik untuk menangani set data yang terlalu besar untuk disimpan dalam memori tetapi masih dapat diproses di satu komputer. Dalam bab ini Anda akan belajar bekerja dengan teknologi yang dapat menangani data yang begitu besar sehingga satu node (komputer) tidak lagi mencukupi. Bahkan mungkin tidak muat di seratus komputer.

Kami akan tetap sedekat mungkin dengan cara kerja dari bab-bab sebelumnya; fokusnya adalah memberi Anda kepercayaan diri untuk bekerja di platform big data. Untuk melakukan ini, bagian utama bab ini adalah studi kasus. Anda akan membuat dasbor yang memungkinkan Anda menjelajahi data dari pemberi pinjaman bank. Di akhir bab ini, Anda akan melalui langkah-langkah berikut:

- Memuat data ke Hadoop, platform data besar yang paling umum.
- Ubah dan bersihkan data dengan Spark.
- Simpan ke dalam database data besar yang disebut Hive.
- Visualisasikan data ini secara interaktif dengan Qlik Sense, alat visualisasi.

Semua ini (terlepas dari visualisasi) akan dikoordinasikan dari dalam skrip Python. Hasil akhirnya adalah dashboard yang memungkinkan Anda menjelajahi data, seperti yang ditunjukkan pada gambar 5.1.



Gambar 5.1 Dasbor Qlik Interaktif

Ingatlah bahwa kita hanya akan menggores permukaan baik praktik maupun teori dalam bab pengantar tentang teknologi data besar ini. Studi kasus akan menyentuh tiga teknologi big data (Hadoop, Spark, dan Hive), tetapi hanya untuk manipulasi data, bukan pembuatan model. Terserah Anda untuk menggabungkan teknologi data besar yang Anda lihat di sini dengan teknik pembuatan model yang telah kita singgung di bab sebelumnya.

5.1 PENYIMPANAN DAN PEMROSESAN DATA DALAM KERANGKA KERJA

Teknologi data besar baru seperti Hadoop dan Spark membuatnya lebih mudah untuk bekerja dengan dan mengontrol sekelompok komputer. Hadoop dapat menskalakan hingga ribuan komputer, membuat klaster dengan penyimpanan berukuran petabyte. Hal ini memungkinkan bisnis untuk memahami nilai dari sejumlah besar data yang tersedia.

5.1.1 Hadoop: kerangka kerja untuk menyimpan dan memproses kumpulan data besar

Apache Hadoop adalah kerangka kerja yang menyederhanakan bekerja dengan sekelompok komputer. Ini bertujuan untuk menjadi semua hal berikut dan lebih banyak lagi:

- **Dapat Diandalkan—**Dengan secara otomatis membuat banyak salinan data dan menerapkan kembali logika pemrosesan jika terjadi kegagalan.
- **Toleransi kesalahan—**Mendeteksi kesalahan dan menerapkan pemulihan otomatis.
- Dapat diskalakan—Data dan pemrosesannya didistribusikan melalui kelompok komputer (penskalaan horizontal).
- **Portable—**Dapat diinstal pada semua jenis perangkat keras dan sistem operasi.

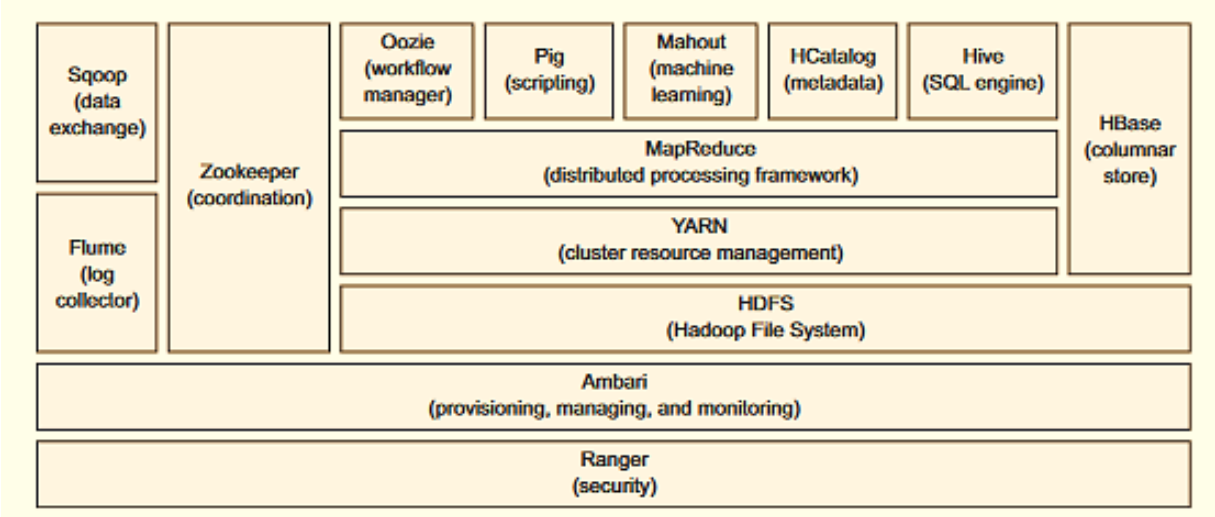
Kerangka inti terdiri dari sistem file terdistribusi, manajer sumber daya, dan sistem untuk menjalankan program terdistribusi. Dalam prakteknya memungkinkan Anda untuk bekerja dengan sistem file terdistribusi hampir semudah dengan sistem file lokal komputer di rumah Anda. Namun di latar belakang, data bisa tersebar di antara ribuan server.

Komponen Hadoop Yang Berbeda

Di jantung Hadoop kami temukan

- Sistem file terdistribusi (HDFS)
- Sebuah metode untuk menjalankan program dalam skala besar (MapReduce)
- Sebuah sistem untuk mengelola sumber daya klaster (BENANG)

Selain itu, muncul ekosistem aplikasi (gambar 5.2), seperti database Hive dan HBase dan kerangka kerja untuk pembelajaran mesin seperti Mahout. Kami akan menggunakan Hive di bab ini. Hive memiliki bahasa berdasarkan SQL yang banyak digunakan untuk berinteraksi dengan data yang disimpan di dalam database.

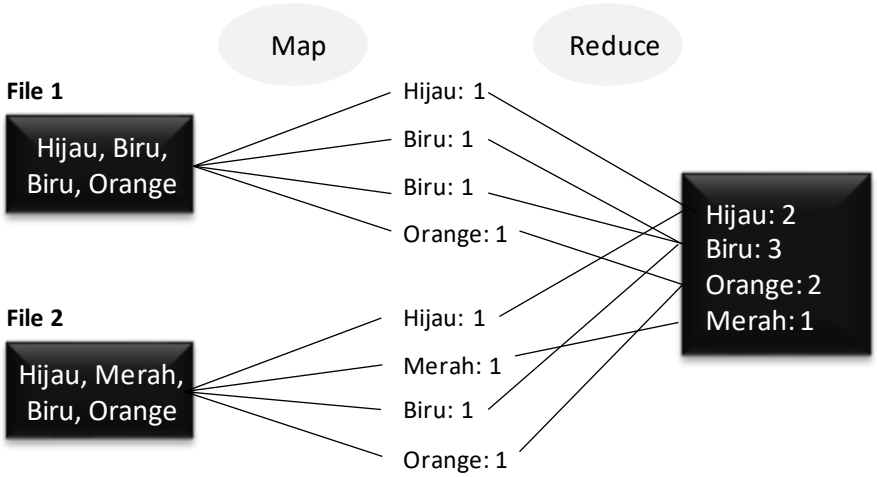


Gambar 5.2 Contoh dari ekosistem aplikasi yang muncul di sekitar Hadoop Core Framework

Anda dapat menggunakan alat populer Impala untuk mengkueri data Hive hingga 100 kali lebih cepat. Kami tidak akan membahas Impala dalam buku ini, tetapi informasi lebih lanjut dapat ditemukan di http:// impala.io/. Kita sudah memiliki pengantar singkat tentang MapReduce di bab 4, tetapi mari kita uraikan sedikit di sini karena ini adalah bagian yang sangat penting dari Hadoop.

Mapreduce: Bagaimana Hadoop Mencapai Parallelisme

Hadoop menggunakan metode pemrograman yang disebut MapReduce untuk mencapai paralelisme. Algoritma MapReduce membagi data, memprosesnya secara paralel, dan kemudian mengurutkan, menggabungkan, dan menggabungkan kembali hasilnya. Namun, algoritme MapReduce tidak cocok untuk analisis interaktif atau program iteratif karena algoritme tersebut menulis data ke disk di antara setiap langkah komputasi. Ini mahal saat bekerja dengan kumpulan data besar.

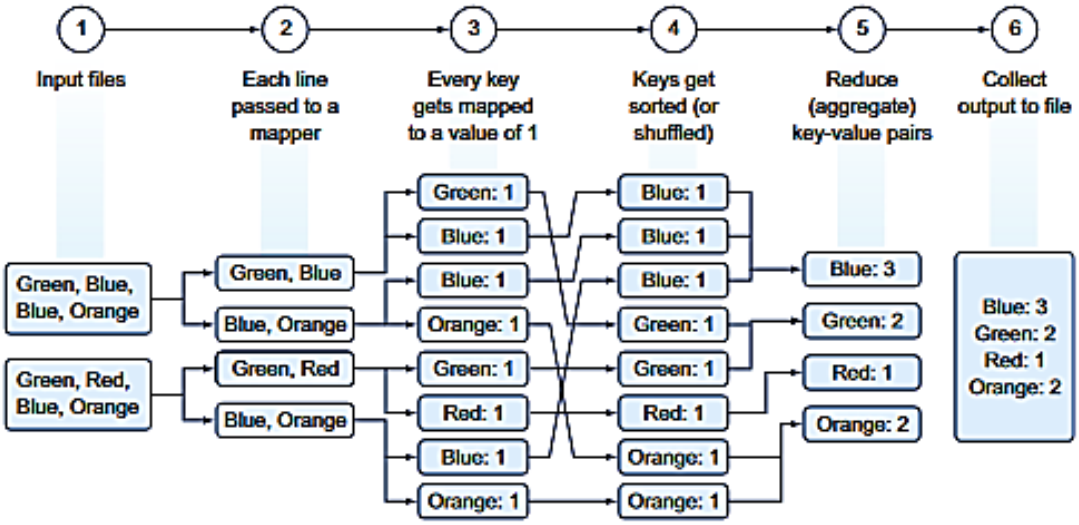


Gambar 5.3 Contoh aliran MapReduce yang disederhanakan untuk menghitung warna dalam teks masukan

Mari kita lihat bagaimana MapReduce bekerja pada contoh fiktif kecil. Anda adalah direktur sebuah perusahaan mainan. Setiap mainan memiliki dua warna, dan ketika klien memesan mainan dari halaman web, halaman web menempatkan file pesanan di Hadoop dengan warna mainan tersebut. Tugas Anda adalah mencari tahu berapa banyak unit warna yang perlu Anda siapkan. Anda akan menggunakan algoritme gaya MapReduce untuk menghitung warna. Pertama mari kita lihat versi yang disederhanakan pada gambar 5.3.
Seperti namanya, prosesnya kira-kira bermuara pada dua fase besar:

- Fase pemetaan—Dokumen dibagi menjadi pasangan kunci-nilai. Sampai kami mengurangi, kami dapat memiliki banyak duplikat.
- **Fase pengurangan—**Ini tidak berbeda dengan SQL "group by." Kejadian unik yang berbeda dikelompokkan bersama, dan bergantung pada fungsi pereduksi, hasil yang berbeda dapat dibuat. Di sini kami menginginkan hitungan per warna, jadi itulah yang mengembalikan fungsi pengurangan.

Pada kenyataannya ini sedikit lebih rumit dari ini.



Gambar 5.4 Contoh aliran MapReduce untuk menghitung warna pada teks masukan

Seluruh proses dijelaskan dalam enam langkah berikut dan digambarkan dalam gambar 5.4.

1. Membaca file masukan.
2. Melewati setiap baris ke pekerjaan mapper.
3. Pekerjaan mapper mem-parsing warna (kunci) dari file dan mengeluarkan file untuk setiap warna dengan berapa kali ditemukan (nilai). Atau lebih tepatnya dikatakan, itu memetakan kunci (warna) ke nilai (jumlah kejadian).
4. Kunci diacak dan diurutkan untuk memfasilitasi agregasi.
5. Fase pengurangan menjumlahkan jumlah kemunculan per warna dan menghasilkan satu file per kunci dengan jumlah total kemunculan untuk setiap warna.
6. Kunci dikumpulkan dalam file keluaran.

CATATAN Meskipun Hadoop membuat bekerja dengan data besar menjadi mudah, menyiapkan klaster kerja yang baik masih tidak sepele, tetapi manajer klaster seperti Apache Mesos meringankan beban tersebut. Pada kenyataannya, banyak perusahaan (menengah)

tidak memiliki kompetensi untuk memelihara instalasi Hadoop yang sehat. Inilah mengapa kami akan bekerja dengan Hortonworks Sandbox, ekosistem Hadoop yang telah terinstal dan terkonfigurasi. Instruksi instalasi dapat ditemukan di bagian 1.5: Contoh kerja pengantar Hadoop. Sekarang, dengan mengingat cara kerja Hadoop, mari kita lihat Spark.

5.1.2 Spark: mengganti MapReduce untuk kinerja yang lebih baik

Ilmuwan data sering melakukan analisis interaktif dan mengandalkan algoritme yang secara inheren berulang; perlu beberapa saat hingga suatu algoritme menyatu dengan solusi. Karena ini adalah titik lemah dari framework MapReduce, kami akan memperkenalkan Spark Framework untuk mengatasinya. Spark meningkatkan kinerja pada tugas-tugas tersebut dengan urutan besarnya.

Apa Itu Spark?

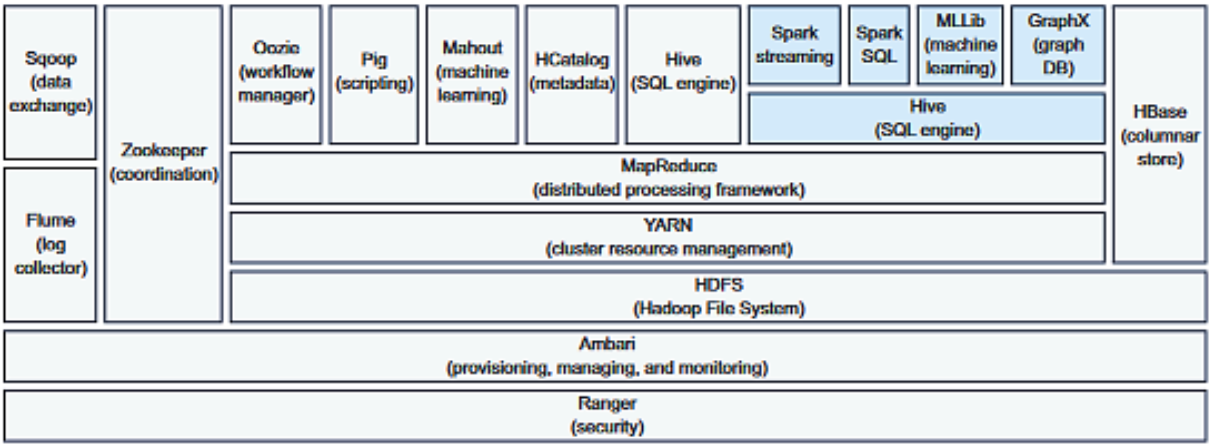
Spark adalah kerangka kerja komputasi cluster yang mirip dengan MapReduce. Spark, bagaimanapun, tidak menangani penyimpanan file pada sistem file (terdistribusi) itu sendiri, juga tidak menangani manajemen sumber daya. Untuk ini bergantung pada sistem seperti Hadoop File System, YARN, atau Apache Mesos. Hadoop dan Spark dengan demikian merupakan sistem yang saling melengkapi. Untuk pengujian dan pengembangan, Anda bahkan dapat menjalankan Spark di sistem lokal Anda.

Bagaimana Spark Memecahkan Masalah Mapreduce?

Meskipun kami sedikit menyederhanakan banyak hal demi kejelasan, Spark membuat semacam memori RAM bersama di antara komputer di klaster Anda. Hal ini memungkinkan pekerja yang berbeda untuk berbagi variabel (dan statusnya) dan dengan demikian menghilangkan kebutuhan untuk menulis hasil perantara ke disk. Lebih teknis dan lebih benar jika Anda menyukai hal itu: Spark menggunakan Resilient Distributed Datasets (RDD), yang merupakan abstraksi memori terdistribusi yang memungkinkan pemrogram melakukan perhitungan dalam memori pada kluster besar dengan cara yang toleran terhadap kesalahan.1 Karena ini merupakan -sistem memori, ini menghindari operasi disk yang mahal.

Komponen Berbeda Dari Ekosistem Spark

Spark core menyediakan lingkungan NoSQL yang cocok untuk analisis interaktif dan eksplorasi. Spark dapat dijalankan dalam mode batch dan interaktif dan mendukung Python.



Gambar 5.5 Framework Spark digunakan dalam kombinasi dengan framework Hadoop

Spark memiliki empat komponen besar lainnya, seperti tercantum di bawah ini dan digambarkan dalam gambar 5.5.

1. Spark streaming adalah alat untuk analisis waktu nyata.
2. Spark SQL menyediakan antarmuka SQL untuk bekerja dengan Spark.
3. MLLib adalah alat untuk pembelajaran mesin di dalam kerangka kerja Spark.
4. GraphX adalah basis data grafik untuk Spark. Kami akan masuk lebih dalam ke database grafik di bab 7.

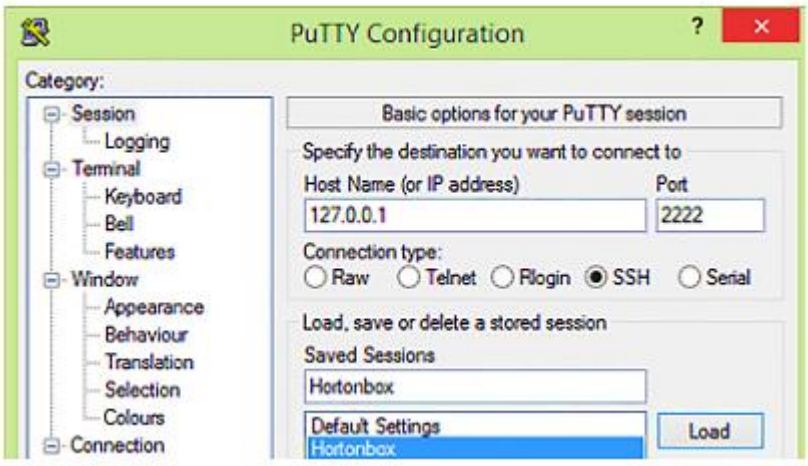
Sekarang mari selami data pinjaman menggunakan Hadoop, Hive, dan Spark.

5.2 STUDI KASUS: MENILAI RISIKO SAAT MEMINJAMKAN UANG

Diperkaya dengan pemahaman dasar tentang Hadoop dan Spark, kami sekarang siap untuk mengotori big data. Tujuan dari studi kasus ini adalah untuk mendapatkan pengalaman pertama dengan teknologi yang kami perkenalkan di awal bab ini, dan melihat bahwa sebagian besar Anda dapat (namun tidak harus) bekerja sama dengan teknologi lainnya. Catatan: Porsi data yang digunakan di sini tidak terlalu besar karena akan membutuhkan lebar pita yang serius untuk mengumpulkannya dan banyak node untuk mengikuti contoh.

Apa yang akan kita gunakan

- Horton Sandbox pada mesin virtual. Jika Anda belum mengunduh dan mengimpor ini ke perangkat lunak VM seperti VirtualBox, silakan kembali ke bagian 1.5 yang menjelaskan hal ini. Versi 2.3.2 dari Horton Sandbox digunakan saat menulis bab ini.
- Pustaka Python: Panda dan pywebhdsf. Mereka tidak perlu dipasang di lingkungan virtual lokal Anda kali ini; kami membutuhkannya langsung di Horton Sandbox. Oleh karena itu kita perlu menjalankan Horton Sandbox (di VirtualBox, misalnya) dan membuat beberapa persiapan.



Gambar 5.6 Menghubungkan ke Horton Sandbox menggunakan Putty

Di baris perintah Sandbox ada beberapa hal yang masih perlu Anda lakukan agar semua ini berfungsi, jadi sambungkan ke baris perintah. Anda dapat melakukan ini menggunakan program seperti Putty. Jika Anda tidak terbiasa dengan PuTTY, PuTTY menawarkan antarmuka baris perintah ke server dan dapat diunduh secara gratis di

http://www.chiark.greenend.org.uk/~sgtatham/putty/ download.html. Konfigurasi login Putty ditunjukkan pada gambar 5.6.

Pengguna dan kata sandi default adalah (pada saat penulisan) "root" dan "hadoop", masing-masing. Anda harus mengubah kata sandi ini pada login pertama.

Setelah terhubung, keluarkan perintah berikut:

- `yum -y install python-pip` — Ini menginstal pip, manajer paket Python.
- `pip install git+https://github.com/DavyCielen/pywebhdfs.git -upgrade` — Pada saat penulisan, ada masalah dengan perpustakaan pywebhdfs dan kami memperbaikinya di fork ini. Semoga Anda tidak memerlukan ini lagi saat membaca ini; masalah telah ditandai dan harus diselesaikan oleh pengelola paket ini.
- `pip install pandas` — Untuk menginstal Panda. Ini biasanya memakan waktu cukup lama karena ketergantungan.

File .ipynb tersedia untuk Anda buka di Jupyter atau (yang lebih lama) Ipython dan ikuti kode di bab ini. Instruksi penyiapan untuk Horton Sandbox diulangi di sana; pastikan untuk menjalankan kode langsung di Horton Sandbox. Sekarang, dengan urusan persiapan, mari kita lihat apa yang perlu kita lakukan. Dalam latihan ini, kita akan melalui beberapa langkah proses ilmu data:

Langkah 1: Tujuan penelitian. Ini terdiri dari dua bagian:

- Memberikan dasbor kepada manajer kami
- Menyiapkan data untuk orang lain untuk membuat dasbor mereka sendiri

Langkah 2: Pengambilan data

- Mengunduh data dari situs web klub pemberi pinjaman
- Menaruh data pada Sistem File Hadoop Horton Sandbox

Langkah 3: Persiapan data

- Mengubah data ini dengan Spark
- Menyimpan data yang telah disiapkan di Hive

Langkah 4 & 6: Eksplorasi dan pembuatan laporan

- Memvisualisasikan data dengan Qlik Sense

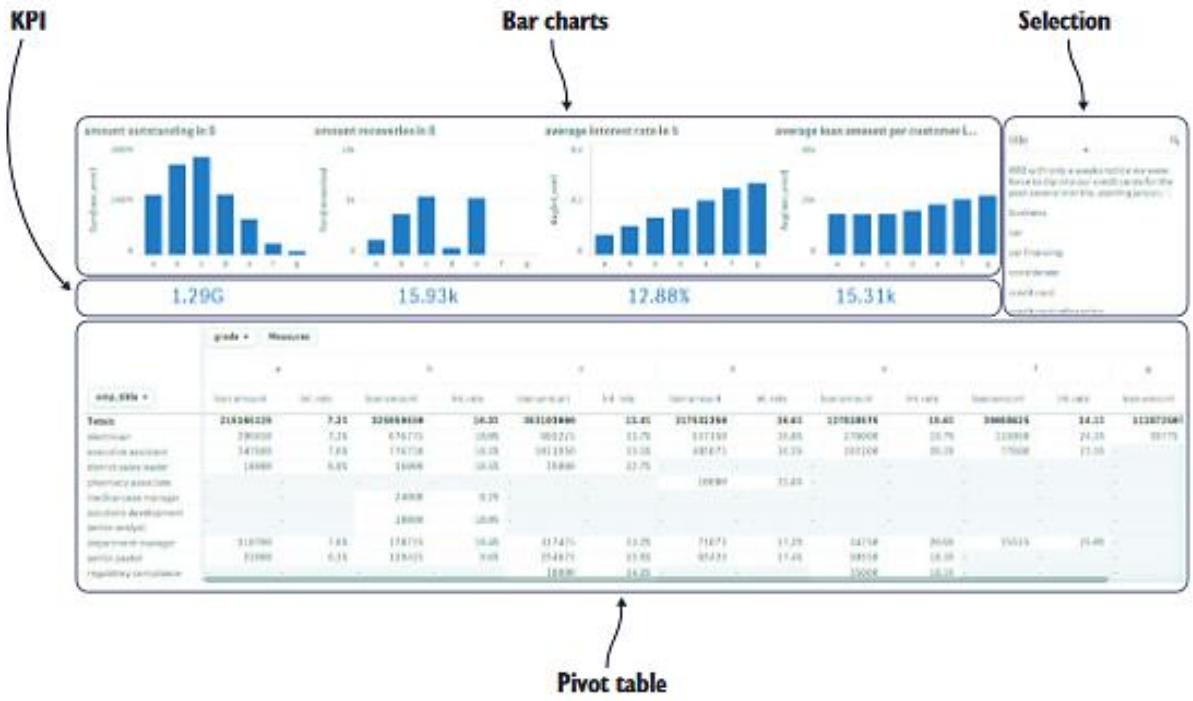
Kami tidak memiliki bangunan model dalam studi kasus ini, tetapi Anda akan memiliki infrastruktur untuk melakukannya sendiri jika Anda mau. Misalnya, Anda dapat menggunakan SPARK Machine learning untuk mencoba memprediksi kapan seseorang akan gagal membayar utangnya. Saatnya bertemu Klub Pemberi Pinjaman.

5.2.1 Langkah 1: Tujuan penelitian

Lending Club adalah organisasi yang menghubungkan orang yang membutuhkan pinjaman dengan orang yang memiliki uang untuk diinvestasikan. Bos Anda juga memiliki uang untuk diinvestasikan dan menginginkan informasi sebelum memberikan uang dalam jumlah besar. Untuk mencapai ini, Anda akan membuat laporan untuknya yang memberinya wawasan tentang peringkat rata-rata, risiko, dan pengembalian untuk meminjamkan uang kepada orang tertentu. Dengan melalui proses ini, Anda membuat data dapat diakses di alat dasbor, sehingga memungkinkan orang lain untuk menjelajahinya juga. Dalam arti tertentu, ini adalah tujuan kedua dari kasus ini: membuka data untuk BI swalayan. Kecerdasan Bisnis

swalayan sering diterapkan di organisasi berbasis data yang tidak memiliki analis untuk cadangan. Siapa pun di organisasi dapat melakukan pemotongan dan pemotongan sederhana sendiri sambil meninggalkan analitik yang lebih rumit untuk ilmuwan data.

Kami dapat melakukan studi kasus ini karena Lending Club menyediakan data anonim tentang pinjaman yang ada. Di akhir studi kasus ini, Anda akan membuat laporan seperti gambar 5.7.



Gambar 5.7 Hasil akhir dari latihan ini adalah dasbor penjelasan untuk membandingkan peluang pinjaman dengan peluang serupa.

Namun, hal pertama yang pertama; mari kita dapatkan data diri kita sendiri.

5.2.2 Langkah 2: Pengambilan data

Saatnya bekerja dengan Sistem File Hadoop (atau hdfs). Pertama kami akan mengirim perintah melalui baris perintah dan kemudian melalui bahasa skrip Python dengan bantuan paket pywebhdfs. Sistem file Hadoop mirip dengan sistem file normal, kecuali file dan folder disimpan di beberapa server dan Anda tidak tahu alamat fisik setiap file. Ini tidak asing jika Anda pernah bekerja dengan alat seperti Dropbox atau Google Drive. File yang Anda masukkan ke drive ini disimpan di suatu tempat di server tanpa Anda tahu persis di server mana. Seperti pada sistem file normal, Anda dapat membuat, mengganti nama, dan menghapus file dan folder.

Menggunakan Baris Perintah Untuk Berinteraksi Dengan Sistem File Hadoop

Mari kita ambil dulu daftar direktori dan file yang ada saat ini di folder root Hadoop menggunakan baris perintah. Ketik perintah hadoop fs –ls / di Putty untuk mencapai ini.
Pastikan Anda menghidupkan mesin virtual Anda dengan Hortonworks Sandbox sebelum mencoba koneksi. Di Putty Anda kemudian harus terhubung ke 127.0.0.1:2222, seperti yang

ditunjukkan sebelumnya pada gambar 5.6. Output dari perintah Hadoop ditunjukkan pada gambar 5.8. Anda juga dapat menambahkan argumen seperti hadoop fs –ls –R / untuk mendapatkan daftar rekursif dari semua file dan subdirektori.

```
[root@sandbox ~]# hadoop fs -ls /
Found 20 items
drwxrwxrwx   - admin  hadoop          0 2015-07-14 14:54 /LoanStats3c.cs
-rw-r--r--   1 root   hadoop  120834552 2015-07-14 14:47 /LoanStats3c.csv
drwxrwxrwx   - yarn   hadoop          0 2015-07-15 13:32 /app-logs
drwxr-xr-x   - hdfs   hdfs            0 2015-06-05 09:19 /apps
drwxr-xr-x   - admin  hadoop          0 2015-07-13 06:47 /book
drwxr-xr-x   - root   hadoop          0 2015-07-17 10:24 /chapter5
-rwxr-xr-x   1 hdfs   hadoop       4240 2015-07-14 19:32 /cout.json
```

Gambar 5.8 Output dari perintah Hadoop list: hadoop fs –ls /. Folder root Hadoop

Kami sekarang akan membuat direktori baru "bab 5" di hdfs untuk digunakan selama bab ini. Perintah berikut akan membuat direktori baru dan memberi semua orang akses ke folder:

```
sudo -u hdfs hadoop fs -mkdir /chapter5
sudo -u hdfs hadoop fs -chmod 777 /chapter5
```

Anda mungkin memperhatikan pola di sini. Perintah Hadoop sangat mirip dengan perintah sistem file lokal kami (gaya POSIX) tetapi mulai dengan Hadoop fs dan beri tanda hubung - sebelum setiap perintah. Tabel 5.1 memberikan ikhtisar tentang perintah sistem file populer di Hadoop dan mitra perintah sistem file lokalnya.

Tabel 5.1 Daftar perintah umum sistem file Hadoop

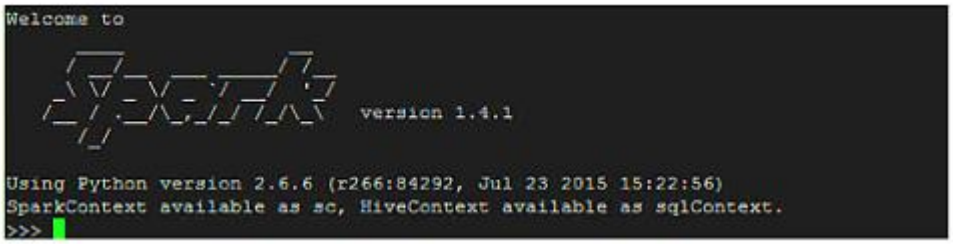
Sasaran	Perintah sistem file Hadoop	Perintah sistem file lokal
Dapatkan daftar file dan direktori dari direktori	hadoop fs –ls URI	ls URI
Buat direktori	hadoop fs –mkdir URI	mkdir URI
Hapus direktori	hadoop fs –rm –r URI	rm –r URI
Ubah izin file	hadoop fs –chmod MODE URI	chmod MODE URI
Pindahkan atau ganti nama file	hadoop fs –mv OLDURI NEWURI	mv OLDURI NEWURI

Ada dua perintah khusus yang sering Anda gunakan. Ini adalah

- Unggah file dari sistem file lokal ke sistem file terdistribusi (hadoop fs –putLOCALURI REMOTEURI).
- Unduh file dari sistem file terdistribusi ke sistem file lokal (hadoop –get REMOTEURI).

Mari kita perjelas ini dengan sebuah contoh. Misalkan Anda memiliki file .CSV di mesin virtual Linux tempat Anda terhubung ke kluster Linux Hadoop. Anda ingin menyalin file .CSV dari mesin virtual Linux Anda ke cluster hdfs. Gunakan perintah hadoop –put mycsv.csv /data.

Menggunakan Putty kita dapat memulai sesi Python di Horton Sandbox untuk mengambil data kita menggunakan skrip Python. Keluarkan perintah "pyspark" di baris perintah untuk memulai sesi. Jika semuanya baik-baik saja, Anda akan melihat layar selamat datang yang ditunjukkan pada gambar 5.9.



Gambar 5.9 Layar pembuka Spark untuk penggunaan interaktif dengan Python

Sekarang kami menggunakan kode Python untuk mengambil data untuk kami, seperti yang ditunjukkan pada daftar berikut.

Listing 5.1 Drawing in the Lending Club loan data

```
import requests
import zipfile
import StringIO
source = requests.get("https://resources.lendingclub.com/
    LoanStats3d.csv.zip", verify=False)
stringio = StringIO.StringIO(source.content)
unzipped = zipfile.ZipFile(stringio)
```

Downloads data from Lending Club. This is https so it should verify, but we won't bother (verify=False).

Creates virtual file.

Unzips data.

Kami mengunduh file "LoanStats3d.csv.zip" dari situs web Lending Club di https:// resources.lendingclub.com/LoanStats3d.csv.zip dan mengekstraknya. Kami menggunakan metode dari paket request, zipfile, dan stringio Python untuk mengunduh data, membuat file virtual, dan mengekstraknya. Ini hanya satu file; jika Anda menginginkan semua data mereka, Anda dapat membuat lingkaran, tetapi untuk tujuan demonstrasi ini akan berhasil. Seperti yang kami sebutkan sebelumnya, bagian penting dari studi kasus ini adalah persiapan data dengan teknologi big data. Namun, sebelum kita dapat melakukannya, kita perlu meletakkannya di sistem file Hadoop. PyWebHdfs adalah paket yang memungkinkan Anda berinteraksi dengan sistem file Hadoop dari Python. Itu menerjemahkan dan meneruskan perintah Anda untuk menghentikan panggilan untuk antarmuka webhdfs. Ini berguna karena Anda dapat menggunakan bahasa skrip favorit Anda untuk mengotomatiskan tugas, seperti yang diperlihatkan dalam daftar berikut.

Listing 5.2 Storing data on Hadoop

Stores it locally because we need to transfer it to Hadoop file system.

Does preliminary data cleaning using Pandas: removes top row and bottom 2 rows because they're useless. Opening original file will show you this.

```
import pandas as pd
from pywebhdfs.webhdfs import PyWebHdfsClient
subselection_csv = pd.read_csv(unzipped.open('LoanStats3d.csv'),
     skiprows=1,skipfooter=2,engine='python')
stored_csv = subselection_csv.to_csv('./stored_csv.csv')
hdfs = PyWebHdfsClient(user_name="hdfs",port=50070,host="sandbox")
hdfs.make_dir('chapter5')
with open('./stored_csv.csv') as file_data:
        hdfs.create_file('chapter5/LoanStats3d.csv',file_data,
     overwrite=True)
```

Creates folder "chapter5" on Hadoop file system.

Opens locally stored csv.

Connects to Hadoop Sandbox.

Creates .csv file on Hadoop file system.

Kami telah mengunduh dan membuka ritsleting file dalam daftar 5.1; sekarang di daftar 5.2 kami membuat sub-seleksi data menggunakan Panda dan menyimpannya secara lokal. Kemudian kami membuat direktori di Hadoop dan mentransfer file lokal ke Hadoop. Data yang diunduh dalam format .CSV dan karena agak kecil, kita dapat menggunakan pustaka Pandas untuk menghapus baris pertama dan dua baris terakhir dari file. Ini berisi komentar dan hanya akan membuat bekerja dengan file ini menjadi rumit di lingkungan Hadoop. Baris pertama kode kita mengimpor paket Pandas, sedangkan baris kedua mem-parsing file ke dalam memori dan menghapus dua baris data pertama dan terakhir. Baris kode ketiga menyimpan data ke sistem file lokal untuk digunakan nanti dan pemeriksaan mudah.
Sebelum melanjutkan, kita dapat memeriksa file kita menggunakan baris kode berikut:

```
print hdfs.get_file_dir_status('chapter5/LoanStats3d.csv')
```

Konsol PySpark akan memberi tahu kami bahwa file kami aman dan baik di sistem Hadoop, seperti yang ditunjukkan pada gambar 5.10.

```
>>> print hdfs.get_file_dir_status('chapter5/LoanStats3d.csv')#A
{u'FileStatus': {u'group': u'hdfs', u'permission': u'755', u'blockSize': 1342177
28, u'accessTime': 1449236321223, u'pathSuffix': u'', u'modificationTime': 14492
36321965, u'replication': 3, u'length': 120997124, u'childrenNum': 0, u'owner':
u'hdfs', u'storagePolicy': 0, u'type': u'FILE', u'fileId': 17520}}
```

Gambar 5.10 Mengambil status file di Hadoop melalui konsol PySpark

Dengan file yang sudah siap dan menunggu kita di Hadoop, kita dapat beralih ke persiapan data menggunakan Spark, karena tidak cukup bersih untuk langsung disimpan di Hive.

5.2.3 Langkah 3: Persiapan data

Sekarang setelah kami mengunduh data untuk dianalisis, kami akan menggunakan Spark untuk membersihkan data sebelum menyimpannya di Hive.

Persiapan Data Di Spark

Membersihkan data sering kali merupakan latihan interaktif, karena Anda menemukan masalah dan memperbaiki masalahnya, dan kemungkinan Anda akan melakukannya beberapa kali sebelum Anda memiliki data yang bersih dan segar. Contoh data kotor adalah string seperti "UsA", yang dikapitalisasi dengan tidak benar. Pada titik ini, kami tidak lagi bekerja di jobs.py tetapi menggunakan antarmuka baris perintah PySpark untuk berinteraksi langsung dengan Spark.

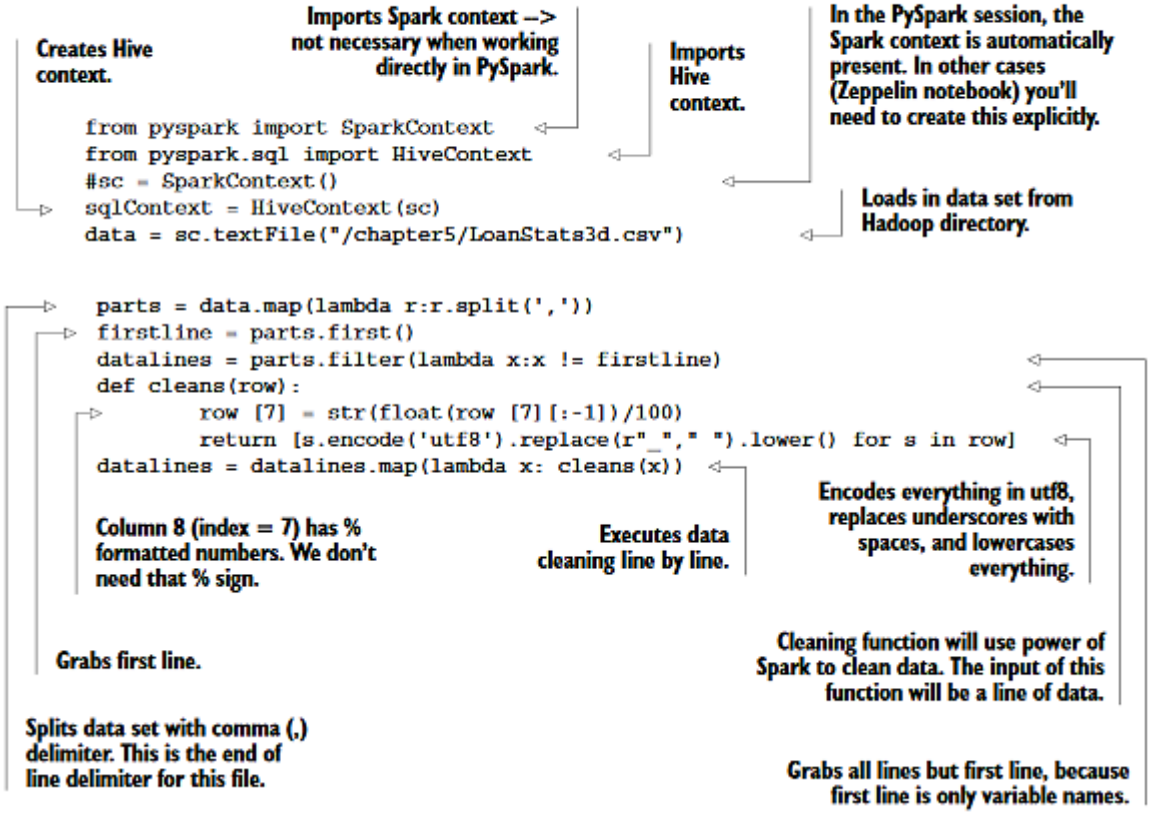
Spark sangat cocok untuk jenis analisis interaktif ini karena tidak perlu menyimpan data setelah setiap langkah dan memiliki model yang jauh lebih baik daripada Hadoop untuk berbagi data antar server (semacam memori terdistribusi).

Transformasi terdiri dari empat bagian:

1. Jalankan PySpark (seharusnya masih terbuka dari bagian 5.2.2) dan muat konteks Spark dan Hive.
2. Baca dan parsing file .CSV.
3. Pisahkan baris tajuk dari data.
4. Bersihkan data.

Oke, ke bisnis. Daftar berikut menunjukkan implementasi kode di konsol PySpark.

Listing 5.3 Connecting to Apache Spark



Mari selami lebih jauh detail untuk setiap langkah.

Langkah 1: Memulai Spark dalam mode interaktif dan memuat konteks

Impor konteks Spark tidak diperlukan di konsol PySpark karena konteks sudah tersedia sebagai variabel sc. Anda mungkin memperhatikan bahwa ini juga disebutkan saat membuka PySpark; lihat gambar 5.9 jika Anda melewatkannya. Kami kemudian memuat konteks Hive untuk memungkinkan kami bekerja secara interaktif dengan Hive. Jika Anda bekerja secara interaktif dengan Spark, konteks Spark dan Hive dimuat secara otomatis, tetapi jika Anda ingin menggunakannya dalam mode batch, Anda perlu memuatnya secara manual. Untuk mengirimkan kode secara berkelompok, Anda akan menggunakan perintah spark-submit filename.py pada baris perintah Horton Sandbox.

```
from pyspark import SparkContext
from pyspark.sql import HiveContext
sc = SparkContext()
sqlContext = HiveContext(sc)
```

Dengan pengaturan lingkungan, kami siap untuk mulai mem-parsing file .CSV.

Langkah 2: Membaca dan mem-parsing file .CSV

Selanjutnya kita membaca file dari sistem file Hadoop dan membaginya di setiap koma yang kita jumpai. Dalam kode kami, baris pertama membaca file .CSV dari sistem file Hadoop. Baris kedua membagi setiap baris saat bertemu dengan koma. Pengurai .CSV kami dirancang dengan naif karena kami belajar tentang Spark, tetapi Anda juga dapat menggunakan paket .CSV untuk membantu Anda mengurai baris dengan lebih tepat.

```
data = sc.textFile("/chapter5/LoanStats3d.csv")
parts = data.map(lambda r:r.split(','))
```

Perhatikan betapa miripnya ini dengan pendekatan pemrograman fungsional. Bagi mereka yang belum pernah menjumpainya, Anda dapat dengan naif membaca lambda r:r.split(',') sebagai "untuk setiap input r (baris dalam kasus ini), pisahkan input r ini saat menemukan koma." Seperti dalam kasus ini, “untuk setiap masukan” berarti “untuk setiap baris”, tetapi Anda juga dapat membacanya sebagai “pisahkan setiap baris dengan koma”. Sintaks mirip fungsional ini adalah salah satu karakteristik favorit saya dari Spark.

Langkah 3: Pisahkan baris tajuk dari data

Untuk memisahkan header dari data, kita membaca di baris pertama dan mempertahankan setiap baris yang tidak mirip dengan baris header:

```
firstline = parts.first()
datalines = parts.filter(lambda x:x != firstline)
```

Mengikuti praktik terbaik dalam data besar, kami tidak perlu melakukan langkah ini karena baris pertama sudah disimpan dalam file terpisah. Pada kenyataannya, file .CSV sering berisi baris header dan Anda harus melakukan operasi serupa sebelum dapat mulai membersihkan data.

Langkah 4: Bersihkan data

Pada langkah ini kami melakukan pembersihan dasar untuk meningkatkan kualitas data. Ini memungkinkan kami membuat laporan yang lebih baik. Setelah langkah kedua, data kita terdiri dari array. Kami akan memperlakukan setiap masukan untuk fungsi lambda sebagai

larik sekarang dan mengembalikan larik. Untuk memudahkan tugas ini, kami membuat fungsi pembantu yang membersihkan. Pembersihan kami terdiri dari memformat ulang input seperti "10,4%" menjadi 0,104 dan menyandikan setiap string sebagai utf-8, serta mengganti garis bawah dengan spasi dan menurunkan semua string. Baris kedua dari kode memanggil fungsi helper kita untuk setiap baris dari array.

```
def cleans(row):
        row [7] = str(float(row [7][:-1])/100)
        return [s.encode('utf8').replace(r"_"," ").lower() for s in row]
datalines = datalines.map(lambda x: cleans(x))
```

Data kami sekarang disiapkan untuk laporan, jadi kami perlu menyediakannya untuk alat pelaporan kami. Hive sangat cocok untuk ini, karena banyak alat pelaporan yang dapat terhubung dengannya. Mari kita lihat bagaimana melakukannya.

Simpan Data Di Hive

Untuk menyimpan data di Hive kita perlu menyelesaikan dua langkah:

1. Buat dan daftarkan metadata.
2. Jalankan pernyataan SQL untuk menyimpan data di Hive.

Di bagian ini, sekali lagi kita akan mengeksekusi potongan kode berikutnya di shell PySpark kesayangan kita, seperti yang ditunjukkan pada daftar berikut.

Listing 5.4 Storing data in Hive (full)

Creates metadata: the Spark SQL StructField function represents a field in a StructType. The StructField object is comprised of three fields: name (a string), dataType (a DataType), and "nullable" (a boolean). The field of name is the name of a StructField. The field of dataType specifies the data type of a StructField. The field of nullable specifies if values of a StructField can contain None values.

Imports SQL data types.

```
from pyspark.sql.types import *
fields = [StructField(field_name,StringType(),True) for field_name in
     firstline]
schema = StructType(fields)
schemaLoans = sqlContext.createDataFrame(datalines, schema)
schemaLoans.registerTempTable("loans")
```

Creates data frame from data (datalines) and data schema (schema).

Registers it as a table called loans.

StructType function creates the data schema. A StructType object requires a list of StructFields as input.

```
sqlContext.sql("drop table if exists LoansByTitle")
sql = '''create table LoansByTitle stored as parquet as select title,
     count(1) as number from loans group by title order by number desc'''
sqlContext.sql(sql)

sqlContext.sql('drop table if exists raw')
sql = '''create table raw stored as parquet as select title,
     emp_title,grade,home_ownership,int_rate,recoveries,
     collection_recovery_fee,loan_amnt,term from loans'''
```

Drops table (in case it already exists) and stores a subset of raw data in Hive.

Drops table (in case it already exists), summarizes, and stores it in Hive. LoansByTitle represents the sum of loans by job title.

Mari telusuri lebih dalam setiap langkah untuk klarifikasi lebih lanjut.

Langkah 1: Buat dan daftarkan metadata

Banyak orang lebih suka menggunakan SQL saat mereka bekerja dengan data. Ini juga dimungkinkan dengan Spark. Anda bahkan dapat membaca dan menyimpan data di Hive secara langsung seperti yang akan kami lakukan. Namun, sebelum Anda dapat melakukannya, Anda harus membuat metadata yang berisi nama kolom dan tipe kolom untuk setiap kolom. Baris kode pertama adalah impor. Baris kedua mem-parsing nama bidang dan jenis bidang dan menentukan apakah suatu bidang wajib. StructType mewakili baris sebagai larik bidang struct. Kemudian Anda menempatkannya di kerangka data yang terdaftar sebagai tabel (sementara) di Hive.

```
from pyspark.sql.types import *
fields = [StructField(field_name,StringType(),True) for field_name in firstline]
schema = StructType(fields)
schemaLoans = sqlContext.createDataFrame(datalines, schema)
schemaLoans.registerTempTable("loans")
```

Dengan metadata yang siap, kami sekarang dapat memasukkan data ke Hive.

Langkah 2: Jalankan kueri dan simpan tabel di Hive

Sekarang kami siap menggunakan dialek SQL pada data kami. Pertama kita akan membuat tabel ringkasan yang menghitung jumlah pinjaman per tujuan. Kemudian kami menyimpan subset dari data mentah yang dibersihkan di Hive untuk visualisasi di Qlik. Menjalankan perintah seperti SQL semudah meneruskan string yang berisi perintah SQL- ke fungsi sqlContext.sql. Perhatikan bahwa kami tidak menulis SQL murni karena kami berkomunikasi langsung dengan Hive. Hive memiliki dialek SQL sendiri yang disebut HiveQL. Dalam SQL kami, misalnya, kami segera memerintahkannya untuk menyimpan data sebagai file Parket. Parket adalah format file data besar yang populer.

```
sqlContext.sql("drop table if exists LoansByTitle")
sql = '''create table LoansByTitle stored as parquet as select title,
     count(1) as number from loans group by title order by number desc'''
sqlContext.sql(sql)

sqlContext.sql('drop table if exists raw')
sql = '''create table raw stored as parquet as select title,
     emp_title,grade,home_ownership,int_rate,recoveries,collection_recovery_f
     ee,loan_amnt,term from loans'''
sqlContext.sql(sql)
```

Dengan data yang disimpan di Hive, kami dapat menghubungkan alat visualisasi kami ke sana.

5.2.4 Langkah 4: Eksplorasi data & Langkah 6: Pembuatan laporan

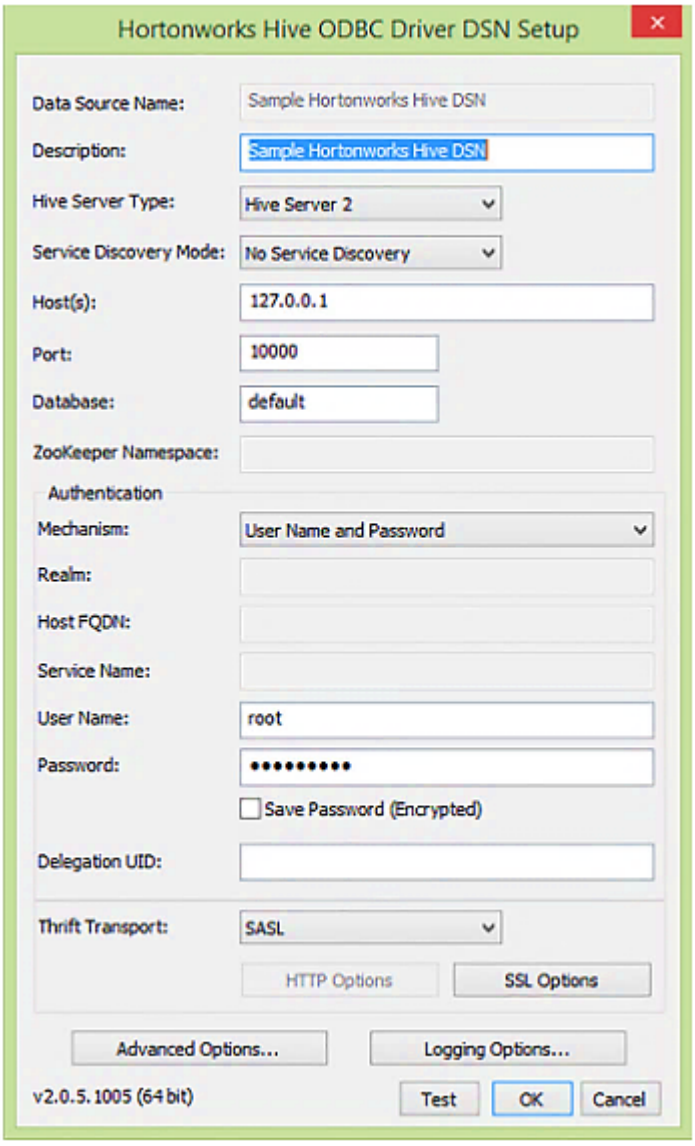
Kami akan membuat laporan interaktif dengan Qlik Sense untuk ditampilkan kepada manajer kami. Qlik Sense dapat diunduh dari http://www.qlik.com/try-or-buy/download-qlik-sense setelah berlangganan situs web mereka. Saat pengunduhan dimulai, Anda akan

diarahkan ke halaman yang berisi beberapa video informasi tentang cara memasang dan bekerja dengan Qlik Sense. Disarankan untuk menonton ini terlebih dahulu.

Kami menggunakan konektor Hive ODBC untuk membaca data dari Hive dan membuatnya tersedia untuk Qlik. Tersedia tutorial pemasangan konektor ODBC di Qlik. Untuk sistem operasi utama, ini dapat ditemukan di http://hortonworks.com/hdp/addons/.

CATATAN Di Windows, ini mungkin tidak berfungsi di luar kotak. Setelah Anda menginstal ODBC, pastikan untuk memeriksa manajer ODBC Windows Anda (CTRL+F dan cari ODBC). Di manajer, buka "System-DSN" dan pilih "Sample Hive Hortonworks DSN". Pastikan pengaturan Anda sudah benar (seperti yang ditunjukkan pada gambar 5.11) atau Qlik tidak akan terhubung ke Hortonworks Sandbox. Semoga Anda tidak lupa kata sandi Sandbox Anda; seperti yang Anda lihat pada gambar 5.11, Anda membutuhkannya lagi.

Sekarang buka Qlik Sense. Jika diinstal di Windows, Anda seharusnya mendapatkan opsi untuk menempatkan pintasan ke .exe di desktop Anda. Qlik bukan freeware; itu adalah produk komersial dengan versi umpan untuk satu pelanggan, tetapi itu sudah cukup untuk saat ini. Di bab terakhir kita akan membuat dasbor menggunakan pustaka JavaScript gratis.



Gambar 5.11 Konfigurasi ODBC Windows Hortonworks

Qlik dapat mengambil data langsung ke memori atau melakukan panggilan setiap kali ke Hive. Kami telah memilih metode pertama karena bekerja lebih cepat.
Bagian ini memiliki tiga langkah:

1. Muat data di dalam Qlik dengan koneksi ODBC.
2. Buat laporan.
3. Jelajahi data.

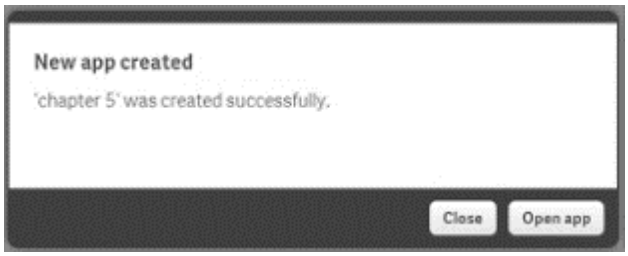
Langkah 1: Muat data di Qlik
Saat Anda memulai Qlik Sense, Anda akan melihat layar selamat datang dengan laporan yang ada (disebut aplikasi), seperti yang ditunjukkan pada gambar 5.12.

Gambar 5.12 Layar selamat datang Qlik Sense

Untuk memulai aplikasi baru, klik tombol Buat aplikasi baru di sebelah kanan layar, seperti yang ditunjukkan pada gambar 5.13. Ini membuka kotak dialog baru. Masukkan "bab 5" sebagai nama baru aplikasi kami.

Gambar 5.13 Kotak pesan Buat aplikasi baru

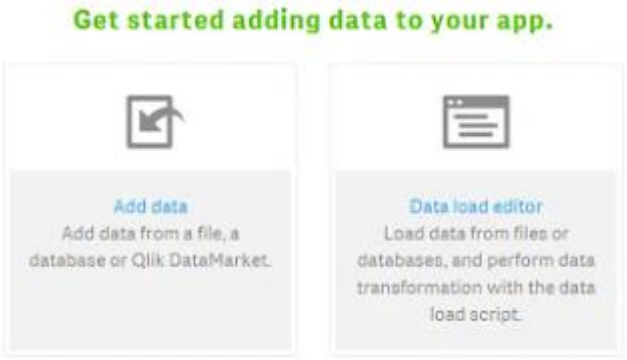
Muncul kotak konfirmasi (gambar 5.14) jika aplikasi berhasil dibuat.



Gambar 5.14 Kotak konfirmasi bahwa aplikasi berhasil dibuat.

Klik tombol Buka aplikasi dan layar baru akan meminta Anda untuk menambahkan data ke aplikasi (gambar 5.15).

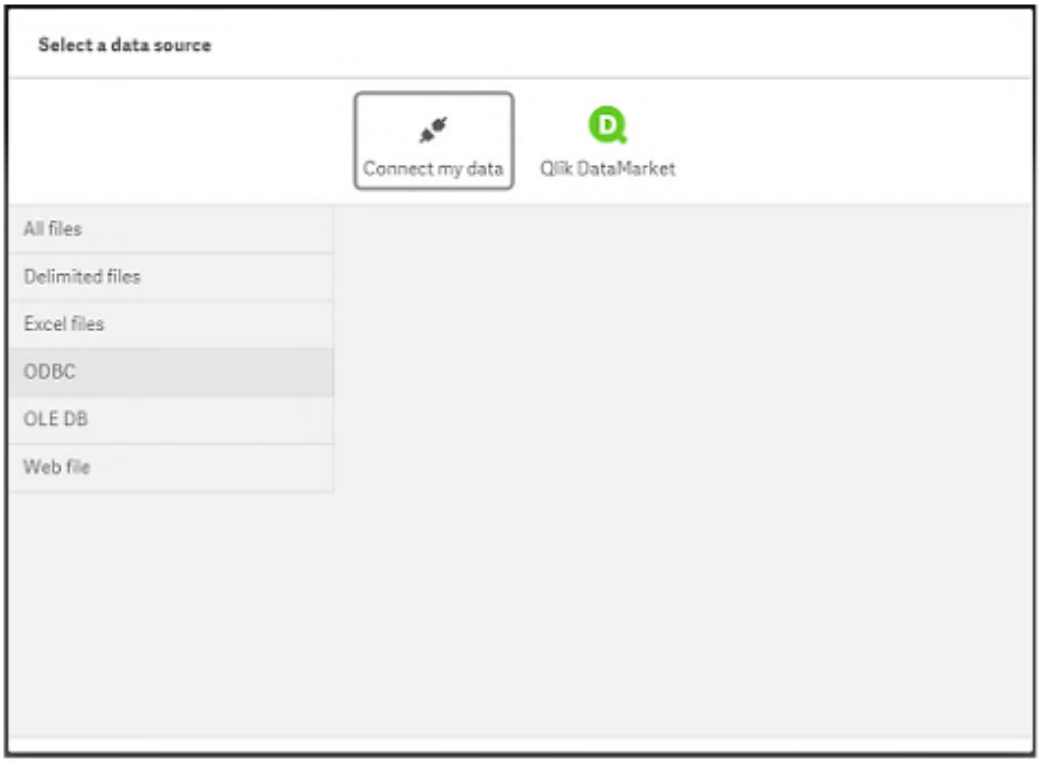
Klik tombol Tambah data dan pilih ODBC sebagai sumber data (gambar 5.16). Pada layar berikutnya (gambar 5.17) pilih User DSN, Hortonworks, dan tentukan root sebagai username dan hadoop sebagai password (atau yang baru Anda berikan saat login ke Sandbox untuk pertama kali).



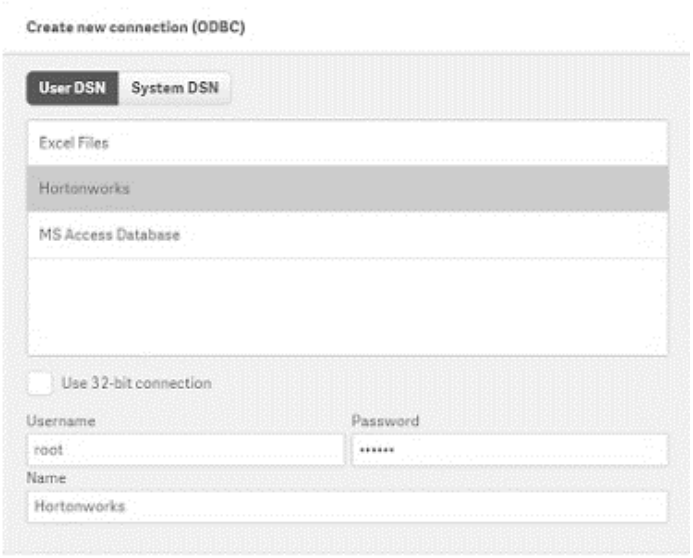
Gambar 5.15 Layar mulai menambahkan data muncul saat Anda membuka aplikasi baru.

CATATAN Opsi Hortonworks tidak muncul secara default. Anda perlu menginstal konektor HDP 2.3 ODBC agar opsi ini muncul (seperti yang dinyatakan sebelumnya). Jika Anda belum berhasil memasangnya saat ini, instruksi yang jelas untuk ini dapat ditemukan di https://blogs.perficient.com/multi-shoring/ blog/2015/09/29/how-to-connect -hortonworks-hive-from-qlikview-with-odbc-driver/.

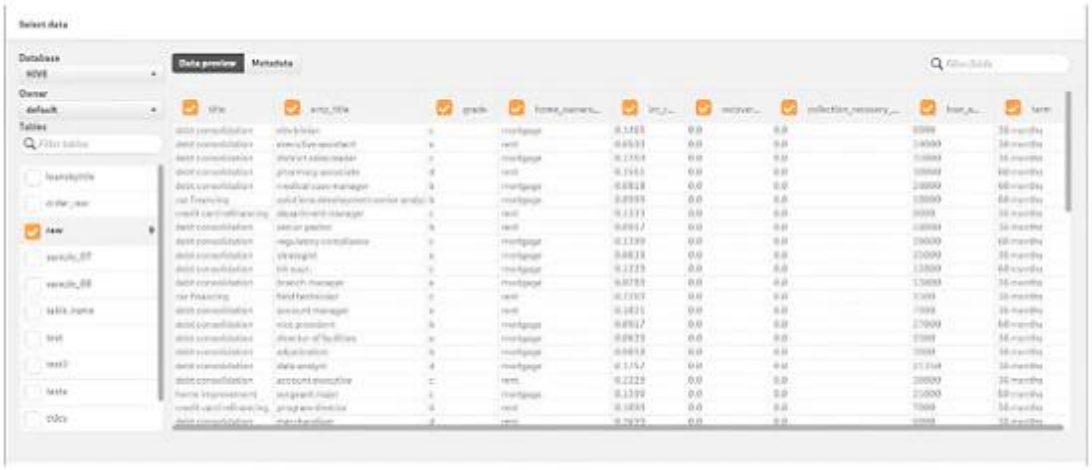
Klik panah yang menunjuk ke kanan untuk pergi ke layar berikutnya.



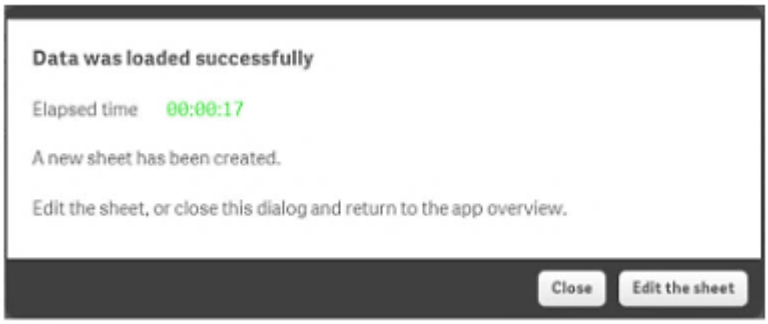
Gambar 5.16 Pilih ODBC sebagai sumber data di layar Pilih sumber data



Gambar 5.17 Pilih Hortonworks pada User DSN dan tentukan username dan password.



Gambar 5.18 Ikhtisar kolom data mentah antarmuka Hive



Gambar 5.19 Konfirmasi bahwa data dimuat di Qlik

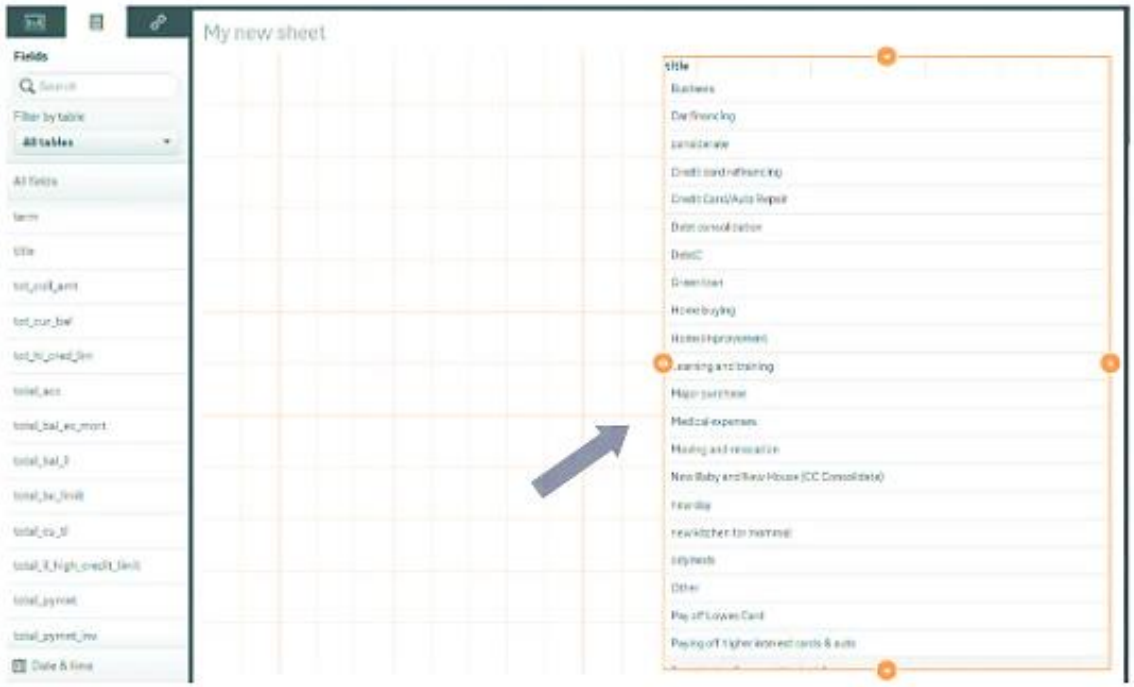
Pilih Hive data, dan default as user pada layar berikutnya (gambar 5.18). Pilih mentah sebagai Tabel untuk memilih dan memilih setiap kolom untuk diimpor; lalu klik tombol Muat dan Selesai untuk menyelesaikan langkah ini. Setelah langkah ini, diperlukan beberapa detik untuk memuat data di Qlik (gambar 5.19).

Langkah 2: Buat laporan

Pilih Edit lembar untuk mulai membuat laporan. Ini akan menambahkan editor laporan (gambar 5.20).

Gambar 5.20 Layar editor untuk laporan terbuka

Sublangkah 1: Menambahkan filter pilihan ke laporan Hal pertama yang akan kita tambahkan ke laporan adalah kotak pilihan yang menunjukkan mengapa setiap orang menginginkan pinjaman. Untuk melakukannya, jatuhkan ukuran judul dari panel aset kiri pada panel laporan dan berikan ukuran dan posisi yang nyaman (gambar 5.21). Klik pada tabel Bidang sehingga Anda dapat menarik dan melepas bidang.

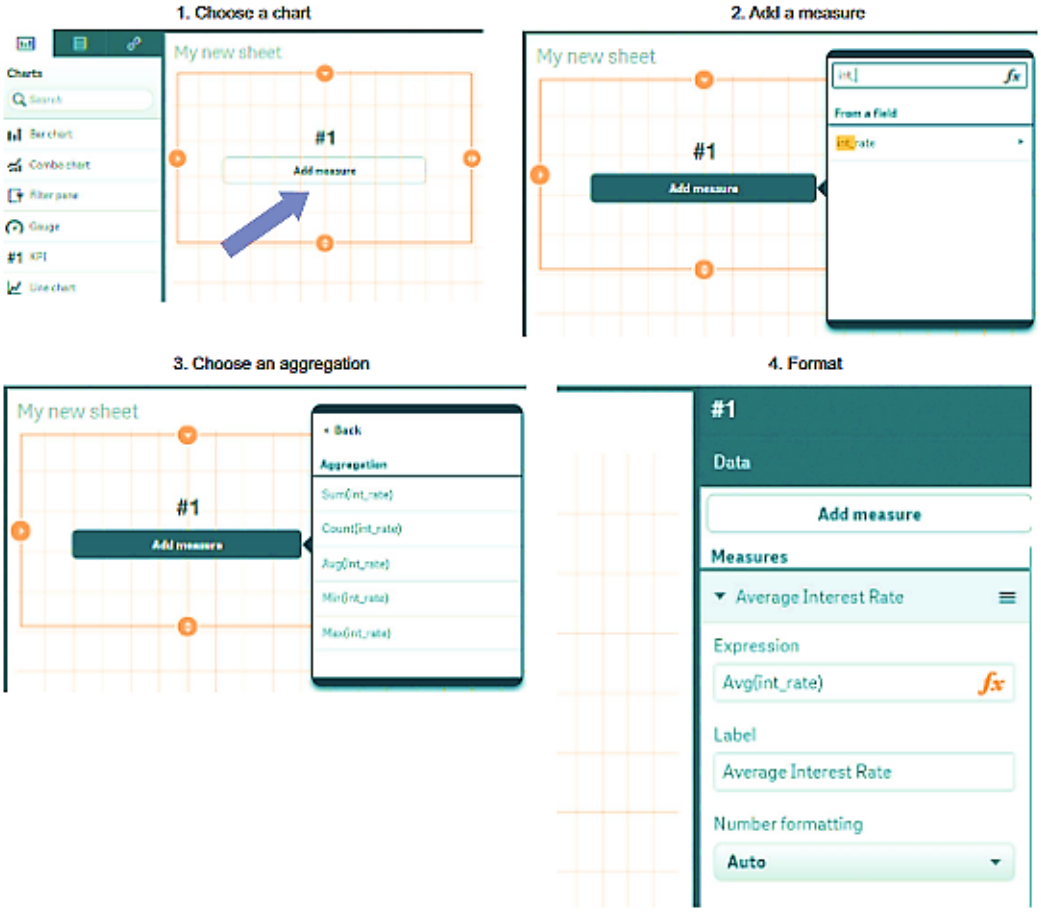


Gambar 5.21 Seret judul dari panel Bidang kiri ke panel laporan.

Sublangkah 2: Menambahkan KPI ke laporan Bagan KPI menunjukkan jumlah agregat untuk total populasi yang dipilih. Angka-angka seperti tingkat bunga rata-rata dan jumlah nasabah ditunjukkan pada bagan ini (gambar 5.22). Menambahkan KPI ke laporan membutuhkan empat langkah, seperti yang tercantum di bawah ini dan ditunjukkan pada gambar 5.23.

1. Pilih bagan—Pilih KPI sebagai bagan dan letakkan di layar laporan; mengubah ukuran dan posisi sesuai keinginan Anda.
2. Tambahkan takaran—Klik tombol tambah takaran di dalam bagan dan pilih int_rate.
3. Pilih metode agregasi—Avg(int_rate).
4. Format grafik—Pada panel kanan, isikan suku bunga rata-rata sebagai Label.

Gambar 5.22 Contoh grafik KPI

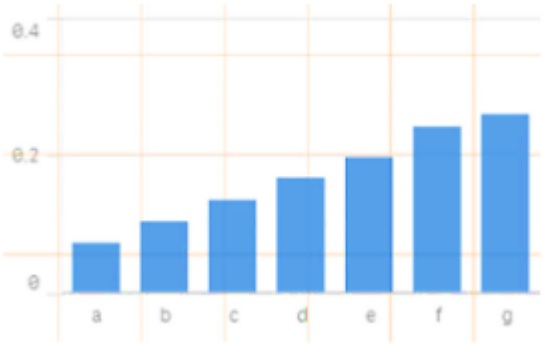


Gambar 5.23 Empat langkah untuk menambahkan grafik KPI ke laporan Qlik

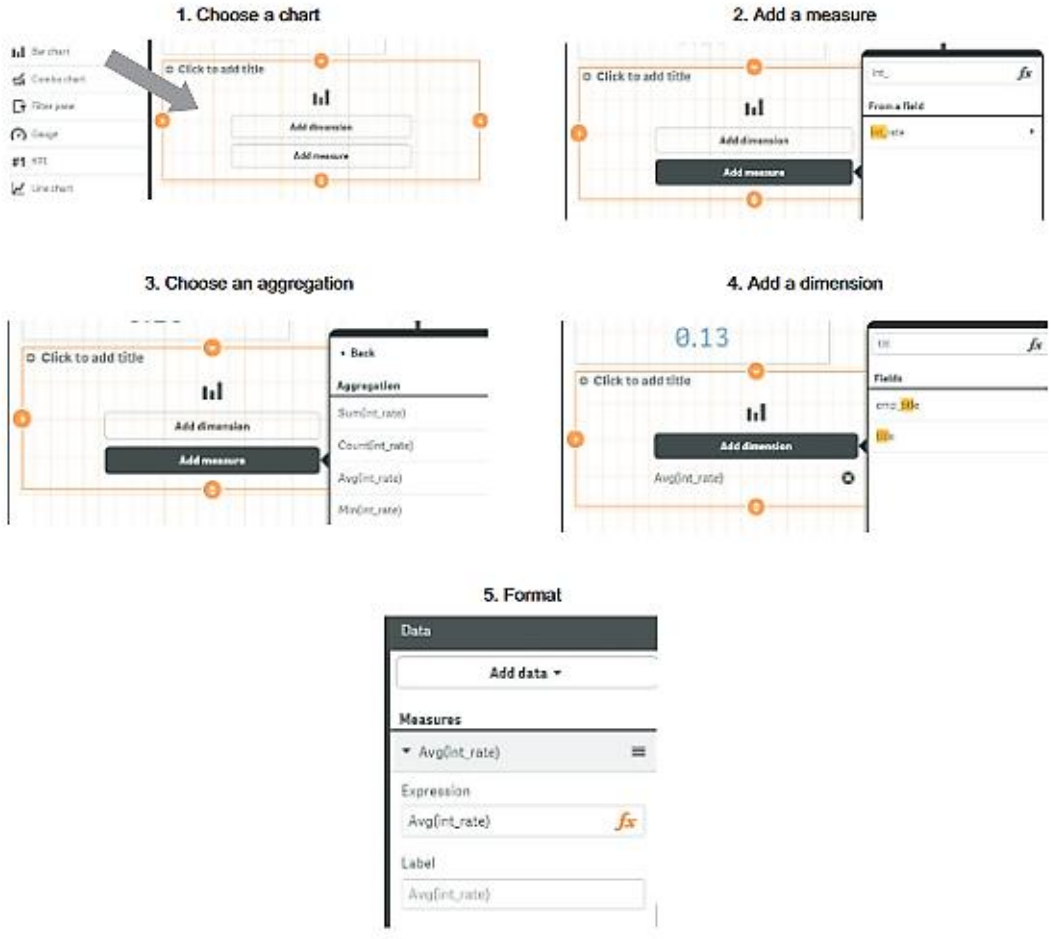
Secara total kami akan menambahkan empat bagan KPI ke laporan kami, jadi Anda harus mengulangi langkah-langkah ini untuk KPI berikut:

- Tingkat bunga rata-rata
- Jumlah total pinjaman
- Rata-rata jumlah pinjaman
- Pemulihan total

Sublangkah 3: Menambahkan diagram batang ke laporan kami Selanjutnya, kami akan menambahkan empat diagram batang ke laporan. Ini akan menunjukkan angka yang berbeda untuk setiap tingkat risiko. Satu diagram batang akan menjelaskan suku bunga rata-rata per kelompok risiko, dan diagram batang lainnya akan menunjukkan jumlah total pinjaman per kelompok risiko (gambar 5.24).



Gambar 5.24 Contoh diagram batang



Gambar 5.25 Menambahkan diagram batang membutuhkan lima langkah.

Menambahkan diagram batang ke laporan membutuhkan lima langkah, seperti yang tercantum di bawah ini dan ditunjukkan pada gambar 5.25.

1. **Pilih bagan—**Pilih bagan batang sebagai bagan dan letakkan di layar laporan; mengubah ukuran dan posisi sesuai keinginan Anda.
2. **Tambahkan takaran—**Klik tombol Tambah takaran di dalam bagan dan pilih int_rate.
3. **Pilih metode agregasi—**Avg(int_rate).
4. **Tambahkan dimensi—**Klik Tambahkan dimensi, dan pilih nilai sebagai dimensi.
5. **Format grafik**—Pada panel kanan, isikan suku bunga rata-rata sebagai Label.

Ulangi prosedur ini untuk kombinasi dimensi dan ukuran berikut:

- Suku bunga rata-rata per kelas
- Jumlah pinjaman rata-rata per kelas
- Total jumlah pinjaman per kelas
- Total pemulihan per kelas

Sublangkah 4: Menambahkan tabel silang ke laporan Misalkan Anda ingin mengetahui tingkat bunga rata-rata yang dibayarkan oleh direktur kelompok risiko C. Dalam hal ini Anda ingin mendapatkan ukuran (tingkat bunga) untuk kombinasi dua dimensi (jabatan dan tingkat risiko). Ini dapat dicapai dengan tabel pivot seperti pada gambar 5.26.

average interest rate per job title / risk grade

emp_title	grade: a	b	c	d
electrician	0.072455172	0.099825	0.13674545	0.16769333
executive assistant	0.069928571	0.1023193	0.13515811	0.16489091
district sales leader	0.0692	0.1049	0.1269	-
pharmacy associate	-	-	-	0.1561
medical case manager	-	0.0818	-	-
solutions development senior analyst	-	0.0999	-	-
department manager	0.075866667	0.10396667	0.132172	0.17185

Gambar 5.26 Contoh tabel pivot, menunjukkan tingkat bunga rata-rata yang dibayarkan per jabatan/kombinasi tingkat risiko

Menambahkan tabel pivot ke laporan membutuhkan enam langkah, seperti yang tercantum di bawah ini dan ditunjukkan pada gambar 5.27.

1. Pilih bagan—Pilih tabel pivot sebagai bagan dan letakkan di layar laporan; mengubah ukuran dan posisi sesuai keinginan Anda.
2. Tambahkan takaran—Klik tombol Tambah takaran di dalam bagan dan pilih int_rate.
3. Pilih metode agregasi—Avg(int_rate).
4. Tambahkan dimensi baris —Klik Tambahkan dimensi, dan pilih emp_title sebagai dimensi.
5. Tambahkan dimensi kolom—Klik Tambah data, pilih kolom, dan pilih nilai.
6. Format grafik—Pada panel kanan, isikan suku bunga rata-rata sebagai Label.

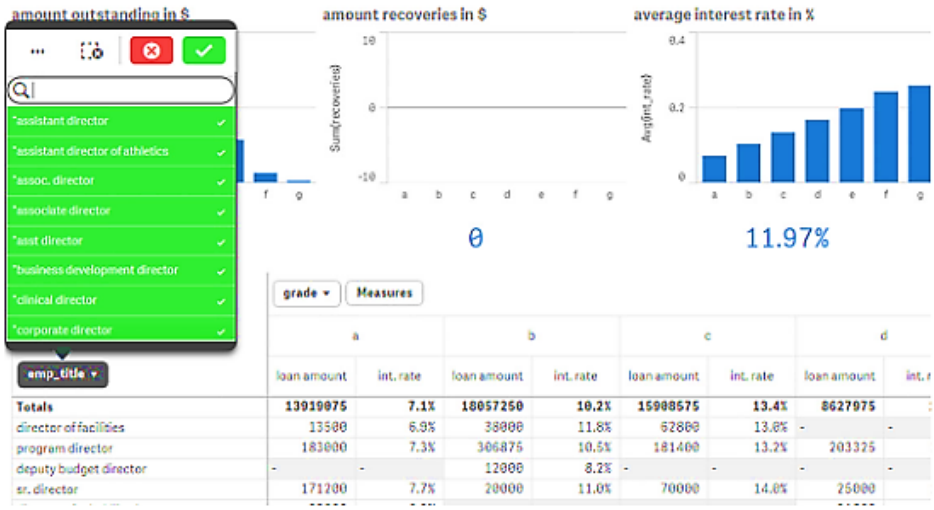
Gambar 5.27 Menambahkan tabel pivot memerlukan enam langkah.

Gambar 5.28 Hasil akhir dalam mode edit

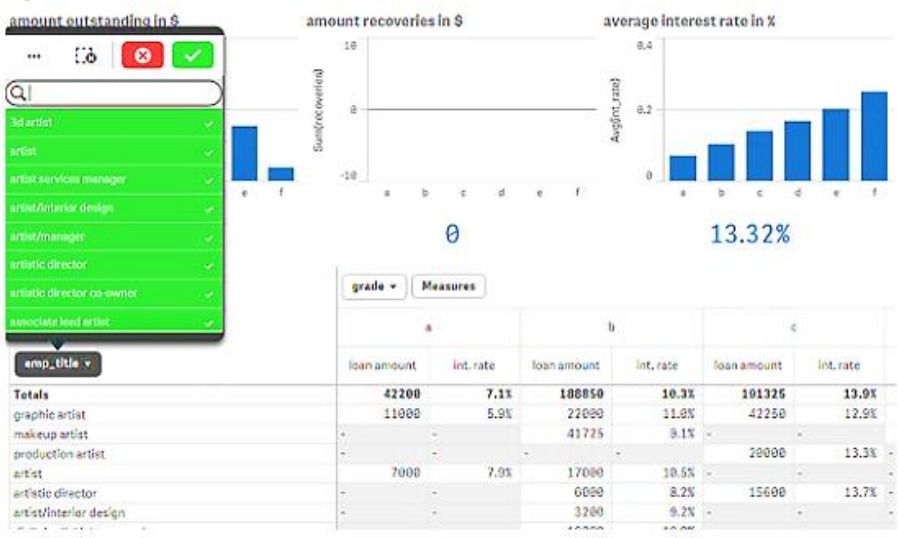
Setelah mengubah ukuran dan memposisikan ulang, Anda akan mendapatkan hasil yang serupa dengan gambar 5.28. Klik tombol Selesai di sebelah kiri dan Anda siap menjelajahi data.

Langkah 3: Jelajahi data

Hasilnya adalah grafik interaktif yang memperbarui sendiri berdasarkan pilihan yang Anda buat. Mengapa Anda tidak mencoba mencari informasi dari sutradara dan membandingkannya dengan artis? Untuk melakukannya, tekan emp_title di tabel pivot dan ketik director di kolom pencarian. Hasilnya seperti gambar 5.29. Dengan cara yang sama, kita dapat melihat para seniman, seperti yang ditunjukkan pada Gambar 5.30. Wawasan menarik lainnya datang dari membandingkan peringkat untuk tujuan pembelian rumah dengan tujuan konsolidasi utang.



Gambar 5.29 Ketika kita memilih direktur, kita dapat melihat bahwa mereka membayar rata-rata 11,97% untuk pinjaman.



Gambar 5.30 Saat kami memilih artis, kami melihat bahwa mereka membayar tingkat bunga rata-rata 13,32% untuk pinjaman.

Kami akhirnya melakukannya: Kami membuat laporan yang diinginkan manajer kami, dan dalam prosesnya kami membuka pintu bagi orang lain untuk membuat laporan mereka sendiri menggunakan data ini. Langkah selanjutnya yang menarik untuk Anda renungkan adalah menggunakan pengaturan ini untuk menemukan orang-orang yang cenderung gagal membayar hutang mereka. Untuk ini, Anda dapat menggunakan kemampuan pembelajaran Mesin Spark yang didorong oleh algoritme online seperti yang ditunjukkan di bab 4.

Dalam bab ini, kami mendapat pengantar langsung tentang kerangka kerja Hadoop dan Spark. Kami membahas banyak hal, tetapi jujur saja, Python membuat bekerja dengan teknologi data besar menjadi mudah. Di bab selanjutnya kita akan menggali lebih dalam dunia database NoSQL dan berhubungan dengan lebih banyak teknologi big data.

5.3 RINGKASAN

Dalam bab ini Anda mempelajarinya

- Hadoop adalah kerangka kerja yang memungkinkan Anda menyimpan file dan mendistribusikan perhitungan di banyak komputer.
- Hadoop menyembunyikan semua kerumitan bekerja dengan sekelompok komputer untuk Anda.
- Ekosistem aplikasi mengelilingi Hadoop dan Spark, mulai dari basis data hingga kontrol akses.
- Spark menambahkan struktur memori bersama ke Hadoop Framework yang lebih sesuai untuk pekerjaan ilmu data.
- Dalam bab studi kasus kami menggunakan PySpark (library Python) untuk berkomunikasi dengan Hive dan Spark dari Python. Kami menggunakan pustaka Python pywebhdfs untuk bekerja dengan pustaka Hadoop, tetapi Anda juga dapat melakukannya menggunakan baris perintah OS.
- Sangat mudah untuk menghubungkan alat BI seperti Qlik ke Hadoop.

BAB 6
DATABASE NOSQL

Dalam bab ini diharapkan mahsiswa mampu menguasai:

- Memahami database NoSQL dan mengapa database tersebut digunakan saat ini
- Mengidentifikasi perbedaan antara NoSQL dan database relasional
- Mendefinisikan prinsip ACID dan hubungannya dengan prinsip NoSQL BASE
- Mempelajari mengapa teorema CAP penting untuk pengaturan basis data multi-simpul
- Menerapkan proses ilmu data ke proyek dengan database Elasticsearch NoSQL

Bab ini dibagi menjadi dua bagian: awal teoretis dan akhir praktis.

- Di bagian pertama bab ini, kita akan melihat database NoSQL secara umum dan menjawab pertanyaan berikut: Mengapa mereka ada? Mengapa tidak sampai saat ini? Jenis apa yang ada dan mengapa Anda harus peduli?
- Pada bagian kedua kita akan mengatasi masalah kehidupan nyata—diagnosa dan pembuatan profil penyakit—menggunakan data yang tersedia secara gratis, Python, dan database NoSQL.

Tidak diragukan lagi Anda pernah mendengar tentang database NoSQL dan bagaimana mereka digunakan secara religius oleh banyak perusahaan teknologi tinggi. Tapi apa itu database NoSQL dan apa yang membuatnya sangat berbeda dari database relasional atau SQL yang biasa Anda gunakan? NoSQL adalah kependekan dari Not Only Structured Query Language, tetapi meskipun benar bahwa database NoSQL memungkinkan Anda untuk menanyakannya dengan SQL, Anda tidak harus fokus pada nama sebenarnya. Banyak perdebatan telah berkecamuk tentang nama dan apakah kelompok database baru ini bahkan harus memiliki nama kolektif sama sekali. Sebaliknya, mari kita lihat apa yang mereka wakili sebagai kebalikan dari sistem manajemen basis data relasional (RDBMS). Basis data tradisional berada di satu komputer atau server. Ini dulunya baik-baik saja selama data Anda tidak melebihi server Anda, tetapi itu tidak terjadi pada banyak perusahaan untuk waktu yang lama sekarang. Dengan pertumbuhan internet, perusahaan seperti Google dan Amazon merasa tertahan oleh database node tunggal ini dan mencari alternatif.

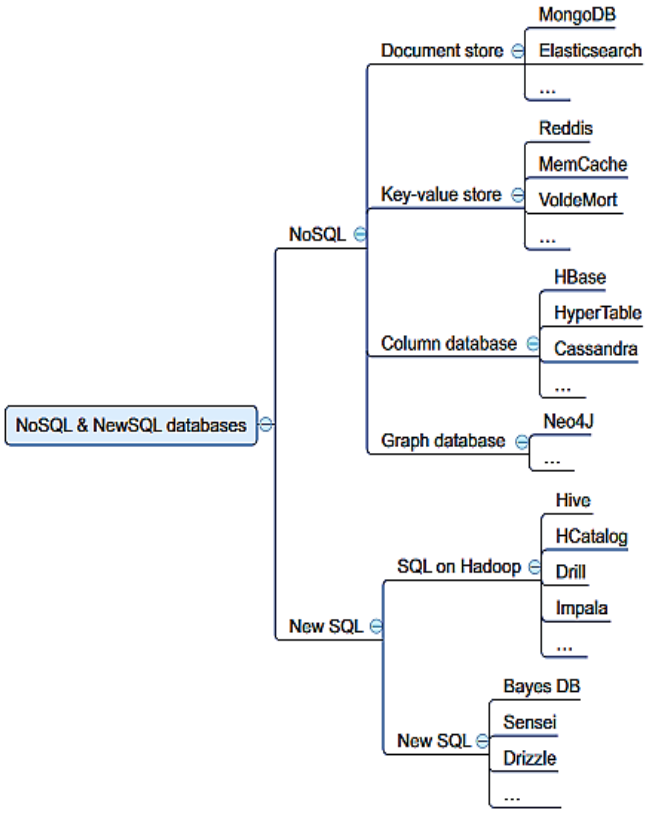
Banyak perusahaan menggunakan database NoSQL single-node seperti MongoDB karena mereka menginginkan skema yang fleksibel atau kemampuan untuk mengumpulkan data secara hierarkis. Berikut adalah beberapa contoh awal:

- Solusi NoSQL pertama Google adalah Google BigTable, yang menandai dimulainya basis data berbentuk kolom.
- Amazon hadir dengan Dynamo, toko bernilai kunci.
- Dua tipe database lagi muncul dalam pencarian partisi: penyimpanan dokumen dan database grafik.

Kami akan merinci masing-masing dari empat jenis nanti di bab ini. Harap dicatat bahwa, meskipun ukuran merupakan faktor penting, database ini tidak semata-mata berasal dari kebutuhan untuk menangani volume data yang lebih besar. Setiap V data besar memiliki

pengaruh (volume, variasi, kecepatan, dan terkadang kebenaran). Database grafik, misalnya, dapat menangani data jaringan. Penggemar basis data grafik bahkan mengklaim bahwa semuanya dapat dilihat sebagai jaringan. Misalnya, bagaimana Anda menyiapkan makan malam? Dengan bahan-bahan. Bahan-bahan ini disatukan untuk membentuk hidangan dan dapat digunakan bersama bahan lainnya untuk membentuk hidangan lainnya. Dilihat dari sudut pandang ini, bahan dan resep merupakan bagian dari suatu jaringan. Tetapi resep dan bahan juga dapat disimpan dalam basis data relasional atau penyimpanan dokumen Anda; itu semua bagaimana Anda melihat masalahnya. Di sinilah letak kekuatan NoSQL: kemampuan untuk melihat masalah dari sudut yang berbeda, membentuk struktur data ke kasus penggunaan. Sebagai seorang ilmuwan data, tugas Anda adalah menemukan jawaban terbaik untuk masalah apa pun. Meskipun terkadang hal ini masih lebih mudah dicapai dengan menggunakan RDBMS, sering kali database NoSQL tertentu menawarkan pendekatan yang lebih baik.

Apakah basis data relasional akan hilang di perusahaan dengan data besar karena kebutuhan akan partisi? Tidak, platform NewSQL (jangan bingung dengan NoSQL) adalah jawaban RDBMS untuk kebutuhan pengaturan cluster. Database NewSQL mengikuti model relasional tetapi mampu dibagi menjadi cluster terdistribusi seperti database NoSQL. Ini bukan akhir dari database relasional dan tentu saja bukan akhir dari SQL, karena platform seperti Hive menerjemahkan SQL menjadi pekerjaan MapReduce untuk Hadoop. Selain itu, tidak semua perusahaan membutuhkan big data; banyak yang melakukannya dengan baik dengan database kecil dan database relasional tradisional sangat cocok untuk itu.



Gambar 6.1 Database NoSQL dan NewSQL

Jika Anda melihat peta pikiran data besar yang ditunjukkan pada gambar 6.1, Anda akan melihat empat jenis database NoSQL. Keempat jenis ini adalah penyimpanan dokumen, penyimpanan nilai kunci, basis data grafik, dan basis data kolom. Peta pikiran juga menyertakan basis data relasional terpartisi NewSQL. Di masa depan, pemisahan besar antara NoSQL dan NewSQL ini akan menjadi usang karena setiap jenis database akan memiliki fokusnya sendiri, sambil menggabungkan elemen dari database NoSQL dan NewSQL. Garis perlahan-lahan kabur karena tipe RDBMS mendapatkan fitur NoSQL seperti pengindeksan berorientasi kolom yang terlihat di database kolom. Namun untuk saat ini, ini adalah cara yang baik untuk menunjukkan bahwa database relasional lama telah melewati penyiapan node tunggal, sementara tipe database lainnya muncul di bawah penyebut NoSQL. Mari kita lihat apa yang dihadirkan NoSQL.

6.1 PENGANTAR NOSQL

Seperti yang telah Anda baca, tujuan database NoSQL tidak hanya untuk menawarkan cara untuk mempartisi database dengan sukses melalui beberapa node, tetapi juga untuk menyajikan cara yang berbeda secara mendasar untuk memodelkan data yang tersedia agar sesuai dengan strukturnya untuk kasus penggunaannya. dan bukan bagaimana database relasional membutuhkannya untuk dimodelkan.

Untuk membantu Anda memahami NoSQL, kami akan mulai dengan melihat prinsip-prinsip inti ACID dari database relasional server tunggal dan menunjukkan bagaimana database NoSQL menulis ulang prinsip-prinsip tersebut menjadi prinsip-prinsip BASE sehingga akan bekerja jauh lebih baik dengan cara terdistribusi. Kita juga akan melihat teorema CAP, yang menjelaskan masalah utama dalam mendistribusikan basis data di beberapa node dan bagaimana pendekatan basis data ACID dan BASE.

6.1.1 ASAM: prinsip inti dari database relasional

Aspek utama dari database relasional tradisional dapat diringkas dengan konsep ACID:

- **Atomicity—**Prinsip “semua atau tidak sama sekali”. Jika catatan dimasukkan ke dalam database, itu dimasukkan sepenuhnya atau tidak sama sekali. Jika, misalnya, kegagalan daya terjadi di tengah tindakan penulisan basis data, Anda tidak akan mendapatkan setengah catatan; itu tidak akan ada sama sekali.
- **Konsistensi—**Prinsip penting ini menjaga integritas data. Tidak ada entri yang masuk ke database yang akan bertentangan dengan aturan yang telah ditentukan sebelumnya, seperti tidak adanya bidang wajib atau bidang menjadi numerik, bukan teks.
- **Isolasi—**Ketika sesuatu diubah dalam database, tidak ada yang dapat terjadi pada data yang sama persis pada saat yang sama. Sebaliknya, tindakan terjadi secara berurutan dengan perubahan lain. Isolasi adalah skala dari isolasi rendah ke isolasi tinggi. Pada skala ini, basis data tradisional berada di ujung "isolasi tinggi". Contoh isolasi rendah adalah Google Docs: Banyak orang dapat menulis ke dokumen pada waktu yang sama dan melihat perubahan satu sama lain terjadi secara instan. Dokumen Word tradisional, di ujung lain spektrum, memiliki isolasi yang tinggi; itu dikunci untuk diedit oleh pengguna pertama yang membukanya. Orang kedua yang membuka dokumen dapat melihat versi terakhir yang disimpan tetapi tidak dapat melihat perubahan yang belum disimpan atau mengedit

dokumen tanpa terlebih dahulu menyimpannya sebagai salinan. Jadi begitu seseorang membukanya, versi terbaru sepenuhnya diisolasi dari siapa pun kecuali editor yang mengunci dokumen tersebut.

- **Ketahanan—**Jika data telah masuk ke database, data tersebut akan bertahan secara permanen. Kerusakan fisik pada hard disk akan menghancurkan rekaman, tetapi pemadaman listrik dan kerusakan perangkat lunak seharusnya tidak terjadi.

ACID berlaku untuk semua database relasional dan database NoSQL tertentu, seperti database grafik Neo4j. Kami akan membahas database grafik lebih lanjut nanti di bab ini dan di bab 7. Untuk sebagian besar database NoSQL lainnya, prinsip lain berlaku: DASAR. Untuk memahami BASE dan mengapa ini berlaku untuk sebagian besar database NoSQL, kita perlu melihat Teorema CAP.

6.1.2 Teorema CAP: masalah dengan DB pada banyak node

Setelah database tersebar di server yang berbeda, sulit untuk mengikuti prinsip ACID karena konsistensi yang dijanjikan ACID; Teorema CAP menunjukkan mengapa ini menjadi masalah. Teorema CAP menyatakan bahwa database dapat berupa dua dari hal-hal berikut tetapi tidak pernah ketiganya:

- **Toleransi partisi—**Basis data dapat menangani partisi jaringan atau kegagalan jaringan.
- **Tersedia—**Selama node yang Anda sambungkan aktif dan berjalan dan Anda dapat menyambungkannya, node akan merespons, bahkan jika koneksi antara node database yang berbeda terputus.
- **Konsisten—**Node mana pun yang Anda sambungkan, Anda akan selalu melihat data yang sama persis.

Untuk database single-node, mudah untuk melihat bagaimana itu selalu tersedia dan konsisten:

- **Tersedia —**Selama node aktif, itu tersedia. Itu semua janji ketersediaan CAP.
- **Konsisten—**Tidak ada node kedua, jadi tidak ada yang tidak konsisten.

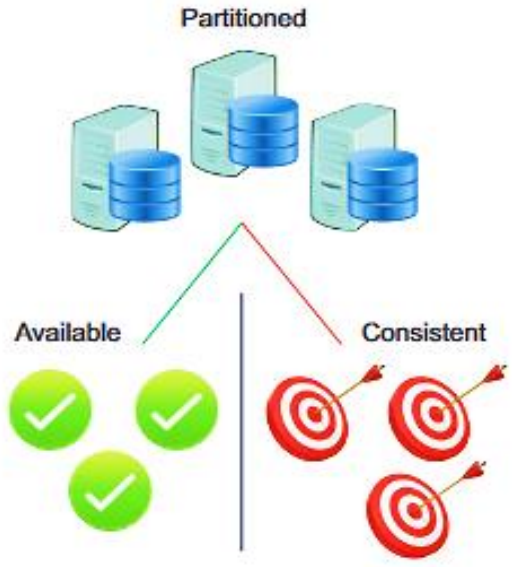
Segalanya menjadi menarik setelah database dipartisi. Kemudian Anda perlu membuat pilihan antara ketersediaan dan konsistensi, seperti yang ditunjukkan pada gambar 6.2.

Mari kita ambil contoh toko online dengan server di Eropa dan server di Amerika Serikat, dengan satu pusat distribusi. Seorang Jerman bernama Fritz dan seorang Amerika bernama Freddy sedang berbelanja pada waktu yang sama di toko online yang sama. Mereka melihat sebuah barang dan hanya satu yang masih ada: meja kopi perunggu berbentuk gurita. Terjadi bencana, dan komunikasi antara dua server lokal terputus untuk sementara. Jika Anda adalah pemilik toko, Anda memiliki dua opsi:

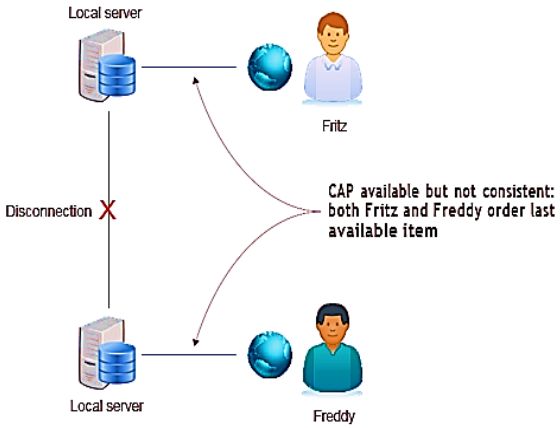
- **Ketersediaan—**Anda mengizinkan server untuk terus melayani pelanggan, dan Anda menyelesaikan semuanya setelahnya.
- **Konsistensi—**Anda menangguhkan semua penjualan sampai komunikasi terjalin kembali.

Dalam kasus pertama, Fritz dan Freddy sama-sama akan membeli meja kopi gurita, karena nomor stok terakhir yang diketahui untuk kedua node adalah “satu” dan kedua node diperbolehkan untuk menjualnya, seperti yang ditunjukkan pada gambar 6.3.

Jika meja kopi sulit didapat, Anda harus memberi tahu Fritz atau Freddy bahwa dia tidak akan menerima mejanya pada tanggal pengiriman yang dijanjikan atau, lebih buruk lagi, dia tidak akan pernah menerimanya. Sebagai pebisnis yang baik, Anda mungkin memberi kompensasi kepada salah satu dari mereka dengan kupon diskon untuk pembelian selanjutnya, dan semuanya mungkin baik-baik saja.



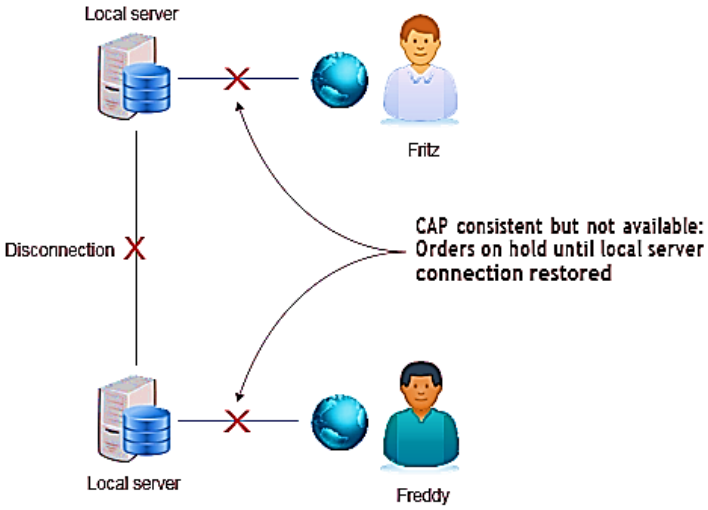
Gambar 6.2 Teorema CAP: saat mempartisi database, Anda harus memilih antara ketersediaan dan konsistensi.



Gambar 6.3 Teorema CAP: jika node terputus, Anda dapat memilih untuk tetap tersedia, tetapi data bisa menjadi tidak konsisten.

Opsi kedua (gambar 6.4) melibatkan penangguhan sementara permintaan yang masuk. Ini mungkin adil bagi Fritz dan Freddy jika setelah lima menit toko web buka kembali untuk bisnis, tetapi Anda mungkin kehilangan penjualan dan mungkin banyak lagi. Toko web cenderung memilih ketersediaan daripada konsistensi, tetapi itu bukanlah pilihan yang optimal dalam semua kasus. Ikuti festival populer seperti Tomorrowland. Festival cenderung memiliki kapasitas maksimum yang diperbolehkan untuk alasan keamanan. Jika Anda menjual lebih banyak tiket daripada yang diizinkan karena server Anda terus menjual selama kegagalan

komunikasi node, Anda dapat menjual dua kali lipat jumlah yang diizinkan pada saat komunikasi dibangun kembali. Dalam kasus seperti itu, mungkin lebih bijaksana untuk mencari konsistensi dan mematikan node untuk sementara. Festival seperti Tomorrowland terjual habis dalam beberapa jam pertama, jadi sedikit downtime tidak akan merugikan seperti harus menarik ribuan tiket masuk.



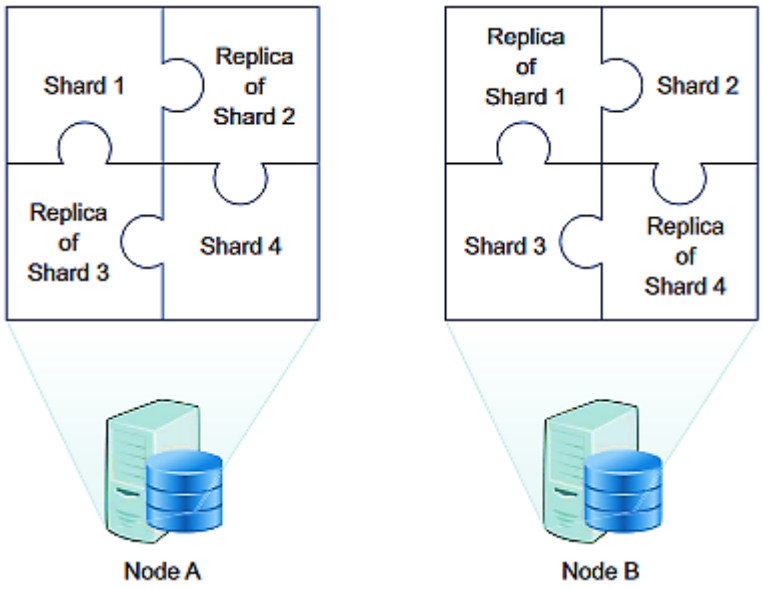
Gambar 6.4 Teorema CAP: jika node terputus, Anda dapat memilih untuk tetap konsisten dengan menghentikan akses ke database hingga koneksi dipulihkan

6.1.3 Prinsip DASAR database NoSQL

RDBMS mengikuti prinsip ASAM; Database NoSQL yang tidak mengikuti ACID, seperti penyimpanan dokumen dan penyimpanan nilai kunci, mengikuti BASE. BASE adalah sekumpulan janji basis data yang jauh lebih lembut:

- **Pada dasarnya tersedia**—Ketersediaan dijamin dalam pengertian CAP. Mengambil contoh toko web, jika sebuah node aktif dan berjalan, Anda dapat terus berbelanja. Bergantung pada bagaimana segala sesuatunya diatur, node dapat mengambil alih dari node lain. Elasticsearch, misalnya, adalah mesin pencari tipe dokumen NoSQL yang membagi dan mereplikasi datanya sedemikian rupa sehingga kegagalan node tidak selalu berarti kegagalan layanan, melalui proses sharding. Setiap shard dapat dilihat sebagai contoh server database individual, tetapi juga mampu berkomunikasi dengan shard lainnya untuk membagi beban kerja seefisien mungkin (gambar 6.5). Beberapa pecahan dapat hadir pada satu node. Jika setiap shard memiliki replika di node lain, kegagalan node dapat dengan mudah diatasi dengan membagi kembali pekerjaan di antara node yang tersisa.
- **Kondisi lunak**—Status sistem mungkin berubah seiring waktu. Ini sesuai dengan prinsip konsistensi akhir: sistem mungkin harus berubah untuk membuat data konsisten lagi. Di satu node, data mungkin mengatakan "A" dan di node lain mungkin mengatakan "B" karena diadaptasi. Nanti, pada resolusi konflik saat jaringan kembali online, kemungkinan “A” di node pertama diganti dengan “B.” Meskipun tidak ada yang melakukan apa pun untuk mengubah "A" menjadi "B" secara eksplisit, ini akan mengambil nilai ini karena menjadi konsisten dengan simpul lainnya.

- **Konsistensi akhir**—Basis data akan menjadi konsisten dari waktu ke waktu. Dalam contoh toko web, tabel dijual dua kali, yang mengakibatkan ketidakkonsistenan data. Setelah koneksi antara masing-masing node dibangun kembali, mereka akan berkomunikasi dan memutuskan bagaimana menyelesaikannya. Konflik ini dapat diselesaikan, misalnya, berdasarkan siapa cepat dia dapat, atau dengan memilih pelanggan yang akan menanggung biaya transportasi terendah. Database datang dengan perilaku default, tetapi mengingat bahwa ada keputusan bisnis yang sebenarnya untuk dibuat di sini, perilaku ini dapat ditimpa. Bahkan jika koneksi aktif dan berjalan, latensi dapat menyebabkan node menjadi tidak konsisten. Seringkali, produk disimpan dalam keranjang belanja online, tetapi memasukkan barang ke dalam keranjang tidak menguncinya untuk pengguna lain. Jika Fritz mengalahkan Freddy di tombol checkout, akan ada masalah saat Freddy pergi untuk check out. Ini dapat dengan mudah dijelaskan kepada pelanggan: dia sudah terlambat. Tetapi bagaimana jika keduanya menekan tombol checkout pada milidetik yang sama persis dan kedua penjualan terjadi?

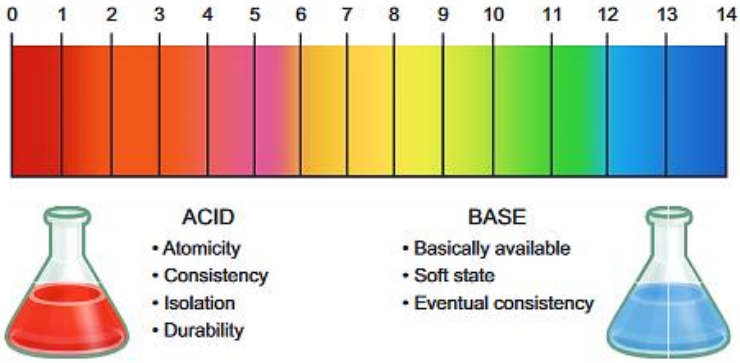


Gambar 6.5 Sharding: setiap shard dapat berfungsi sebagai database mandiri, tetapi mereka juga bekerja sama secara keseluruhan.

Contoh mewakili dua node, masing-masing berisi empat pecahan: dua pecahan utama dan dua replika. Kegagalan satu node didukung oleh yang lain.

ASAM versus BASIS

Prinsip DASAR agak dibuat agar sesuai dengan asam dan basa dari kimia: asam adalah cairan dengan nilai pH rendah. Basa adalah kebalikannya dan memiliki nilai pH yang tinggi. Kami tidak akan membahas detail kimia di sini, tetapi gambar 6.6 menunjukkan mnemonik bagi mereka yang akrab dengan padanan kimia asam dan basa.



Gambar 6.6 ACID versus BASE: database relasional tradisional versus sebagian besar database NoSQL. Nama-nama tersebut berasal dari konsep kimia skala pH. Nilai pH di bawah 7 bersifat asam; lebih tinggi dari 7 adalah basa. Pada skala ini, air permukaan rata-rata Anda berfluktuasi antara 6,5 dan 8,5.

6.1.4 Tipe database NoSQL

Seperti yang Anda lihat sebelumnya, ada empat jenis NoSQL besar: penyimpanan nilai kunci, penyimpanan dokumen, basis data berorientasi kolom, dan basis data grafik. Setiap jenis memecahkan masalah yang tidak dapat diselesaikan dengan database relasional. Implementasi aktual seringkali merupakan kombinasi dari ini. OrientDB, misalnya, adalah database multi-model, menggabungkan tipe NoSQL. OrientDB adalah database grafik di mana setiap node adalah dokumen.

Gambar 6.7 Database relasional berusaha menuju normalisasi (memastikan setiap potongan data disimpan hanya sekali). Setiap tabel memiliki pengidentifikasi unik (kunci primer) yang digunakan untuk memodelkan hubungan antara entitas (tabel), maka istilah relasional.

Sebelum masuk ke database NoSQL yang berbeda, mari kita lihat database relasional sehingga Anda memiliki sesuatu untuk dibandingkan. Dalam pemodelan data, banyak pendekatan yang mungkin dilakukan. Database relasional umumnya berusaha menuju normalisasi: memastikan setiap bagian data disimpan hanya sekali. Normalisasi menandai pengaturan struktural mereka. Jika, misalnya, Anda ingin menyimpan data tentang seseorang dan hobinya, Anda dapat melakukannya dengan dua tabel: satu tentang orang tersebut dan satu lagi tentang hobinya. Seperti yang Anda lihat pada gambar 6.7, tabel tambahan

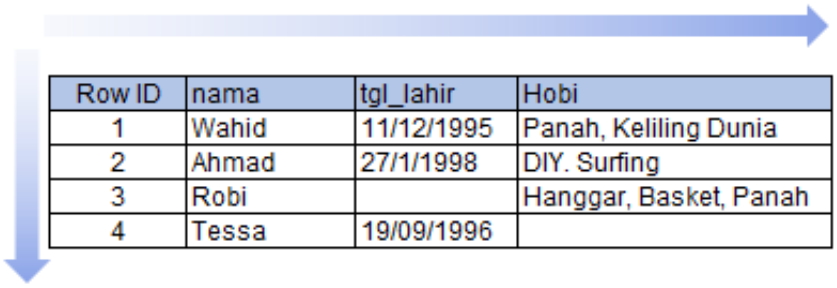
diperlukan untuk menghubungkan hobi dengan orang karena hubungan banyak-ke-banyak mereka: seseorang dapat memiliki banyak hobi dan hobi dapat membuat banyak orang mempraktikkannya. Database relasional skala penuh dapat terdiri dari banyak entitas dan tabel penghubung. Sekarang Anda memiliki sesuatu untuk dibandingkan dengan NoSQL, mari kita lihat berbagai jenisnya.

Database Berorientasi Kolom

Basis data relasional tradisional berorientasi pada baris, dengan setiap baris memiliki id baris dan setiap bidang dalam baris disimpan bersama dalam sebuah tabel. Katakanlah, misalnya, tidak ada data tambahan tentang hobi yang disimpan dan Anda hanya memiliki satu tabel untuk mendeskripsikan orang, seperti yang ditunjukkan pada gambar 6.8. Perhatikan bagaimana dalam skenario ini Anda mengalami sedikit denormalisasi karena hobi dapat diulang. Jika informasi hobi merupakan tambahan yang bagus tetapi tidak penting untuk kasus penggunaan Anda, menambahkannya sebagai daftar dalam kolom Hobi adalah pendekatan yang dapat diterima. Tetapi jika informasinya tidak cukup penting untuk tabel terpisah, apakah harus disimpan sama sekali?

Row ID	nama	tgl_lahir	Hobi
1	Wahid	11/12/1995	Panah, Keliling Dunia
2	Ahmad	27/1/1998	DIY. Surfing
3	Robi		Hanggar, Basket, Panah
4	Tessa	19/09/1996	

Gambar 6.8 Tata letak basis data berorientasi baris. Setiap entitas (orang) diwakili oleh satu baris, tersebar di beberapa kolom.



Gambar 6.9 Pencarian berorientasi baris: dari atas ke bawah dan untuk setiap entri, semua kolom dimasukkan ke dalam memori

Setiap kali Anda mencari sesuatu dalam database berorientasi baris, setiap baris dipindai, terlepas dari kolom mana yang Anda perlukan. Katakanlah Anda hanya menginginkan daftar ulang tahun di bulan September. Database akan memindai tabel dari atas ke bawah dan dari kiri ke kanan, seperti yang ditunjukkan pada gambar 6.9, dan akhirnya mengembalikan daftar ulang tahun.

Pengindeksan data pada kolom tertentu dapat secara signifikan meningkatkan kecepatan pencarian, tetapi pengindeksan setiap kolom membawa biaya tambahan dan database masih memindai semua kolom. Database kolom menyimpan setiap kolom secara

terpisah, memungkinkan pemindaian lebih cepat ketika hanya sejumlah kecil kolom yang terlibat; lihat gambar 6.10.

nama	Row ID
Wahid	1
Ahmad	2
Robi	3
Tessa	4

tgl_lahir	Row ID
11/12/1995	1
27/1/1998	2
19/09/1996	4

Hobi	Row ID
Panah	1,3
Keliling Dunia	1
DIY	2
Surfing	2
Hanggar	3
Basket	3

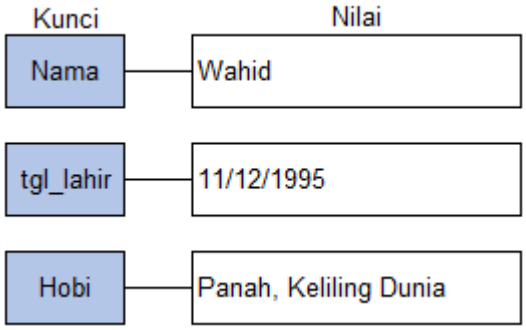
Gambar 6.10 Database berorientasi kolom menyimpan setiap kolom secara terpisah dengan nomor baris terkait. Setiap entitas (orang) dibagi menjadi beberapa tabel.

Tata letak ini terlihat sangat mirip dengan database berorientasi baris dengan indeks pada setiap kolom. Indeks basis data adalah struktur data yang memungkinkan pencarian cepat pada data dengan mengorbankan ruang penyimpanan dan penulisan tambahan (pembaruan indeks). Indeks memetakan nomor baris ke data, sedangkan basis data kolom memetakan data ke nomor baris; dengan cara itu menghitung menjadi lebih cepat, jadi mudah untuk melihat berapa banyak orang yang menyukai panahan, misalnya. Menyimpan kolom secara terpisah juga memungkinkan kompresi yang dioptimalkan karena hanya ada satu tipe data per tabel.

Kapan Anda harus menggunakan database berorientasi baris dan kapan Anda harus menggunakan database berorientasi kolom? Dalam database berorientasi kolom, mudah untuk menambahkan kolom lain karena tidak ada kolom yang terpengaruh olehnya. Tetapi menambahkan seluruh catatan membutuhkan penyesuaian semua tabel. Ini membuat basis data berorientasi baris lebih disukai daripada basis data berorientasi kolom untuk pemrosesan transaksi online (OLTP), karena ini menyiratkan penambahan atau perubahan catatan terus-menerus. Basis data berorientasi kolom bersinar saat melakukan analitik dan pelaporan: menjumlahkan nilai dan menghitung entri. Basis data berorientasi baris seringkali merupakan basis data operasional pilihan untuk transaksi aktual (seperti penjualan). Pekerjaan batch semalam memperbarui basis data berorientasi kolom, mendukung pencarian secepat kilat dan agregasi menggunakan algoritme MapReduce untuk laporan. Contoh toko keluarga kolom adalah Apache HBase, Cassandra Facebook, Hypertable, dan kakek dari toko kolom lebar, Google BigTable.

Toko Nilai Kunci

Penyimpanan nilai kunci adalah yang paling kompleks dari database NoSQL. Mereka adalah, seperti namanya, kumpulan pasangan kunci-nilai, seperti yang ditunjukkan pada gambar 6.11, dan kesederhanaan ini menjadikannya tipe database NoSQL yang paling dapat diskalakan, yang mampu menyimpan data dalam jumlah besar.



Gambar 6.11 Penyimpanan nilai kunci menyimpan semuanya sebagai kunci dan nilai.

Nilai dalam penyimpanan nilai kunci dapat berupa apa saja: string, angka, tetapi juga kumpulan pasangan nilai kunci baru yang dikemas dalam objek. Gambar 6.12 menunjukkan struktur nilai kunci yang sedikit lebih rumit. Contoh toko bernilai kunci adalah Redis, Voldemort, Riak, dan Amazon's Dynamo.

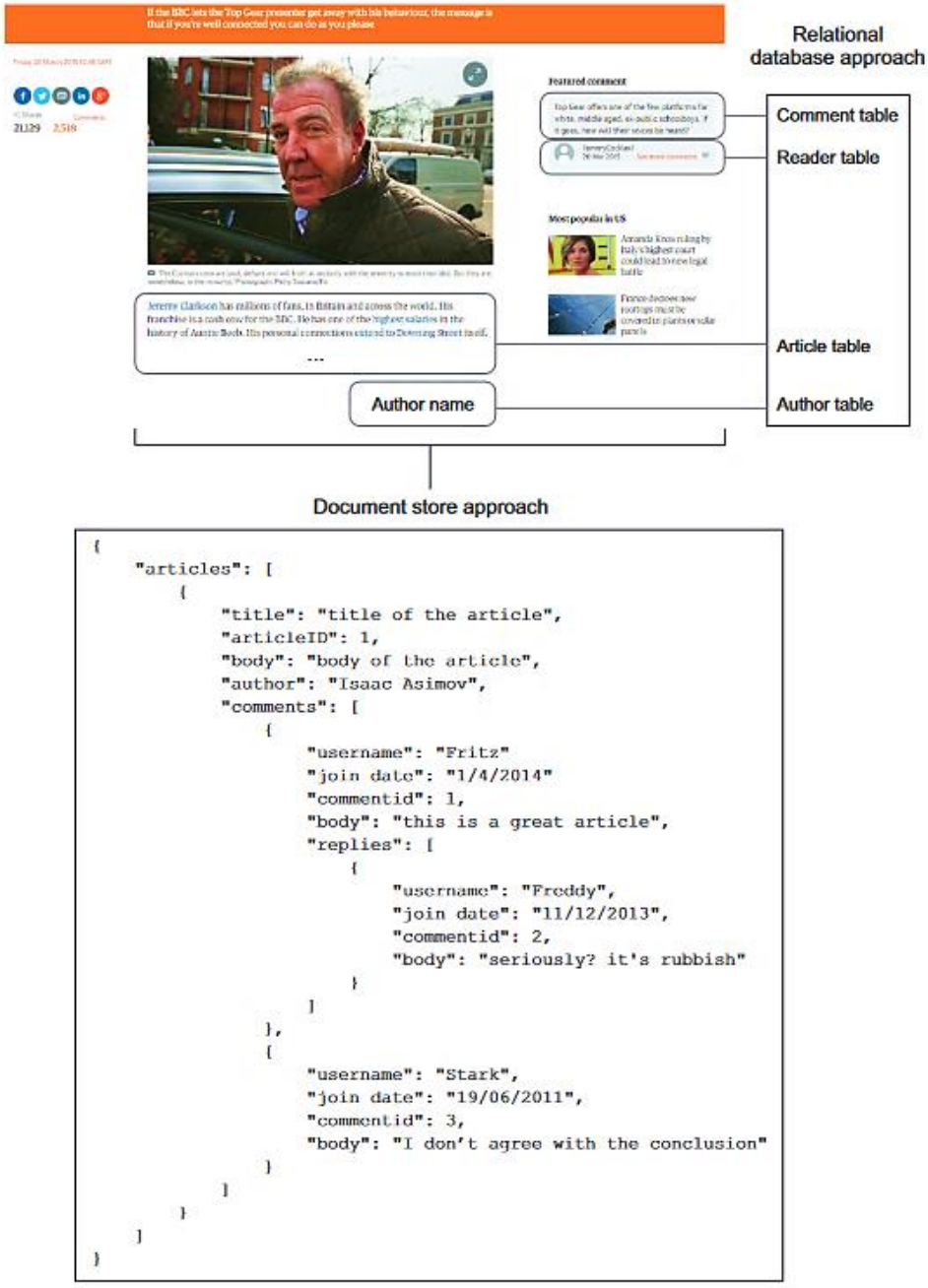
```
{"internal data":[{"entities":[
    {"customer":[
        {"id":1,"name":"Freddy"},
        {"id":2,"name":"Fritz"}
    ]},
    {"legal entities":[
        {"id":1,"company":"Maiton"}
    ]}]
},{"Products":[
    {"furniture":[
        {"id":1,"name":"Octopus Table","stock":1}
    ]
}]}]}
```

Gambar 6.12 Struktur bersarang nilai kunci

Toko Dokumen

Penyimpanan dokumen adalah satu tingkat kerumitan dari penyimpanan nilai kunci: penyimpanan dokumen mengasumsikan struktur dokumen tertentu yang dapat ditentukan dengan skema. Penyimpanan dokumen tampak paling alami di antara jenis database NoSQL karena dirancang untuk menyimpan dokumen sehari-hari sebagaimana adanya, dan memungkinkan kueri dan penghitungan yang rumit pada bentuk data yang sering kali sudah digabungkan ini. Cara hal-hal disimpan dalam database relasional masuk akal dari sudut pandang normalisasi: semuanya harus disimpan hanya sekali dan terhubung melalui kunci asing. Penyimpanan dokumen tidak terlalu peduli dengan normalisasi selama data berada dalam struktur yang masuk akal. Model data relasional tidak selalu cocok dengan kasus bisnis tertentu. Surat kabar atau majalah, misalnya, memuat artikel. Untuk menyimpan ini dalam basis data relasional, Anda perlu memotongnya terlebih dahulu: teks artikel masuk ke satu tabel, penulis dan semua informasi tentang penulis di tabel lain, dan komentar pada artikel saat dipublikasikan di situs web masuk ke tabel lain. Seperti yang ditunjukkan pada gambar 6.13, sebuah artikel surat kabar juga dapat disimpan sebagai satu kesatuan; ini menurunkan

beban kognitif bekerja dengan data bagi mereka yang terbiasa melihat artikel sepanjang waktu. Contoh penyimpanan dokumen adalah MongoDB dan CouchDB.



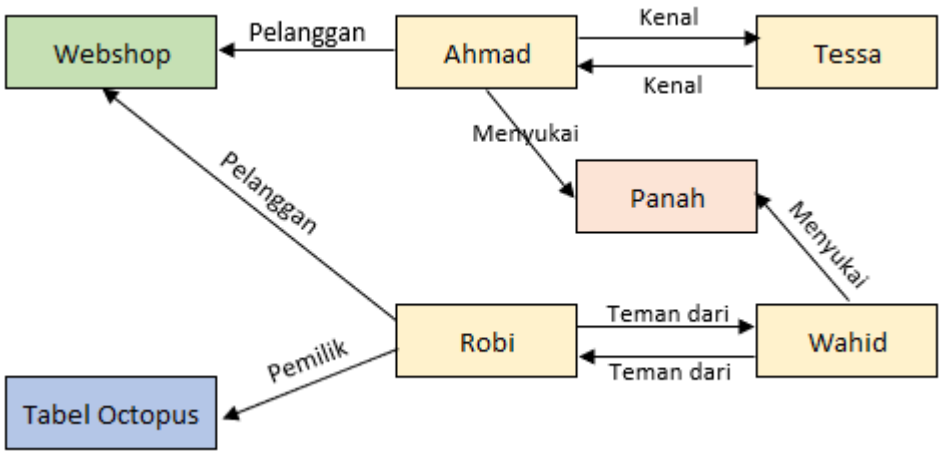
Gambar 6.13 Penyimpanan dokumen menyimpan dokumen secara keseluruhan, sedangkan RDMS memotong artikel dan menyimpannya dalam beberapa tabel. Contoh diambil dari situs web Guardian.

Database Grafik

Jenis database NoSQL besar terakhir adalah yang paling kompleks, diarahkan untuk menyimpan hubungan antar entitas secara efisien. Ketika data sangat saling terhubung, seperti untuk jejaring sosial, kutipan makalah ilmiah, atau kelompok aset modal, database grafik adalah jawabannya. Grafik atau data jaringan memiliki dua komponen utama:

- **Node—**Entitas itu sendiri. Di jejaring sosial ini bisa jadi orang.
- **Tepi—**Hubungan antara dua entitas. Hubungan ini diwakili oleh garis dan memiliki propertinya sendiri. Tepi dapat memiliki arah, misalnya jika tanda panah menunjukkan siapa bos siapa.

Grafik bisa menjadi sangat kompleks mengingat jenis relasi dan entitas yang cukup. Gambar 6.14 sudah menunjukkan bahwa kompleksitas dengan jumlah entitas yang terbatas. Database grafik seperti Neo4j juga mengklaim mendukung ACID, sedangkan penyimpanan dokumen dan penyimpanan nilai kunci mematuhi BASE.



Gambar 6.14 Contoh data grafik dengan empat tipe entitas (orang, hobi, perusahaan, dan furnitur) dan relasinya tanpa informasi edge atau node tambahan

257 systems in ranking, March 2015

Rank Mar 2015	Rank Feb 2015	Rank Mar 2014	DBMS	Database Model	Score Mar 2015	Score Feb 2015	Score Mar 2014
1.	1.	1.	Oracle	Relational DBMS	1469.09	+29.37	-22.71
2.	2.	2.	MySQL	Relational DBMS	1261.09	-11.36	-29.12
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1164.80	-12.68	-40.48
4.	4.	↑ 5.	MongoDB	Document store	275.01	+7.77	+75.03
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	264.44	+2.10	+29.38
6.	6.	6.	DB2	Relational DBMS	198.85	-3.57	+11.52
7.	7.	7.	Microsoft Access	Relational DBMS	141.69	+1.15	-4.79
8.	8.	↑ 10.	Cassandra	Wide column store	107.31	+0.23	+29.22
9.	9.	↓ 8.	SQLite	Relational DBMS	101.71	+2.14	+8.73
10.	10.	↑ 13.	Redis	Key-value store	97.05	-2.16	+43.59
11.	11.	↓ 9.	SAP Adaptive Server	Relational DBMS	85.37	-0.97	+3.81
12.	12.	12.	Solr	Search engine	81.88	+0.40	+20.74
13.	13.	↓ 11.	Teradata	Relational DBMS	72.78	+3.33	+10.15
14.	14.	↑ 16.	HBase	Wide column store	60.73	+3.59	+25.59
15.	↑ 16.	↑ 19.	Elasticsearch	Search engine	58.92	+6.09	+32.75

Gambar 6.15 Top 15 database berdasarkan popularitas menurut DB-Engines.com pada Maret 2015

Kemungkinannya tidak terbatas, dan karena dunia menjadi semakin saling terhubung, database grafik cenderung memenangkan medan dibandingkan jenis lainnya, termasuk database relasional yang masih dominan.

Gambar 6.15 menunjukkan bahwa dengan 9 entri, database relasional masih mendominasi 15 teratas pada saat buku ini ditulis, dan dengan hadirnya NewSQL kita belum bisa menghitungnya. Neo4j, database grafik paling populer, dapat ditemukan di posisi 23 pada saat penulisan, dengan Titan di posisi 53. Sekarang setelah Anda melihat masing-masing tipe database NoSQL, saatnya untuk mengotori tangan Anda dengan salah satunya.

6.2 STUDI KASUS: PENYAKIT APAKAH ITU?

Itu telah terjadi pada banyak dari kita: Anda tiba-tiba mengalami gejala medis dan hal pertama yang Anda lakukan adalah Google penyakit apa yang mungkin ditunjukkan oleh gejala tersebut; lalu Anda memutuskan apakah perlu menemui dokter. Mesin pencari web baik-baik saja untuk ini, tetapi database yang lebih berdedikasi akan lebih baik. Database seperti ini ada dan cukup canggih; mereka hampir bisa menjadi versi virtual dari Dr. House, seorang ahli diagnosa yang brilian di serial TV House M.D. Tapi mereka dibangun di atas data yang terlindungi dengan baik dan tidak semuanya dapat diakses oleh publik. Selain itu, meskipun perusahaan farmasi besar dan rumah sakit canggih memiliki akses ke dokter virtual ini, banyak dokter umum masih terpaku pada buku mereka saja. Asimetri informasi dan sumber daya ini tidak hanya menyedihkan dan berbahaya, tetapi juga tidak perlu ada sama sekali. Jika mesin pencari khusus penyakit yang sederhana digunakan oleh semua dokter umum di dunia, banyak kesalahan medis dapat dihindari.

Dalam studi kasus ini, Anda akan mempelajari cara membuat mesin telusur semacam itu di sini, meskipun hanya menggunakan sebagian kecil dari data medis yang dapat diakses secara bebas. Untuk mengatasi masalah tersebut, Anda akan menggunakan database NoSQL modern yang disebut Elasticsearch untuk menyimpan data, dan proses ilmu data untuk bekerja dengan data dan mengubahnya menjadi sumber daya yang cepat dan mudah dicari. Inilah cara Anda menerapkan prosesnya:

1. **Menetapkan tujuan penelitian**.
2. **Pengumpulan data**—Anda akan mendapatkan data dari Wikipedia. Ada lebih banyak sumber di luar sana, tetapi untuk tujuan demonstrasi, satu saja sudah cukup.
3. **Persiapan data**—Data Wikipedia mungkin tidak sempurna dalam formatnya saat ini. Anda akan menerapkan beberapa teknik untuk mengubahnya.
4. **Eksplorasi data**—Kasus penggunaan Anda istimewa karena langkah 4 proses sains data juga merupakan hasil akhir yang diinginkan: Anda ingin data Anda mudah dijelajahi.
5. **Pemodelan data**—Tidak ada pemodelan data nyata yang diterapkan dalam bab ini. Matriks istilah dokumen yang digunakan untuk pencarian seringkali menjadi titik awal untuk pemodelan topik tingkat lanjut. Kami tidak akan membahasnya di sini.
6. **Menyajikan hasil** —Untuk membuat data dapat dicari, Anda memerlukan antarmuka pengguna seperti situs web tempat orang dapat meminta dan mengambil informasi penyakit. Dalam bab ini Anda tidak akan melangkah lebih jauh untuk membangun antarmuka yang sebenarnya. Sasaran sekunder Anda: membuat profil kategori penyakit dengan kata kuncinya; Anda akan mencapai tahap proses ilmu data ini karena Anda akan menyajikannya sebagai cloud kata, seperti yang ada di gambar 6.16.

Gambar 6.16 Awan kata sampel pada kata kunci diabetes tidak berbobot

Untuk mengikuti kode, Anda memerlukan barang-barang ini:

- Sesi Python dengan perpustakaan elasticsearch-py dan Wikipedia terpasang (pip install elasticsearch dan pipinstallwikipedia)
- Contoh Elasticsearch yang disiapkan secara lokal; lihat lampiran A untuk petunjuk pemasangan
- Pustaka IPython

CATATAN Kode untuk bab ini tersedia untuk diunduh dari situs web Manning untuk buku ini di https://manning.com/books/introducing-data-science dan dalam format IPython.

Elasticsearch: mesin pencari open source/database NoSQL

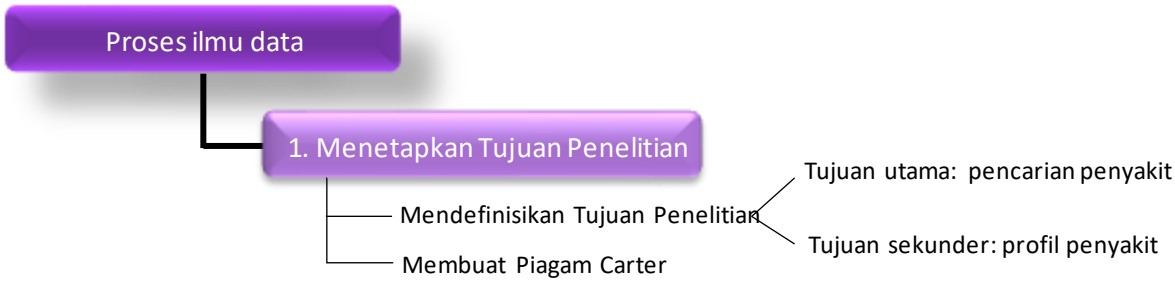
Untuk mengatasi masalah yang dihadapi, mendiagnosis penyakit, database NoSQL yang akan Anda gunakan adalah Elasticsearch. Seperti MongoDB, Elasticsearch adalah toko dokumen. Tapi tidak seperti MongoDB, Elasticsearch adalah mesin pencari. Sementara MongoDB hebat dalam melakukan kalkulasi kompleks dan pekerjaan MapReduce, tujuan utama Elasticsearch adalah pencarian teks lengkap. Elasticsearch akan melakukan perhitungan dasar pada data numerik terindeks seperti summing, counts, median, mean, standar deviasi, dan sebagainya, namun pada intinya tetap merupakan mesin pencari.

Elasticsearch dibangun di atas Apache Lucene, mesin pencari Apache yang dibuat pada tahun 1999. Lucene terkenal sulit ditangani dan lebih merupakan blok penyusun untuk aplikasi yang lebih ramah pengguna daripada solusi end-to-end itu sendiri. Tapi Lucene adalah mesin pencari yang sangat kuat, dan diikuti oleh Apache Solr pada tahun 2004, dibuka untuk penggunaan umum pada tahun 2006. Solr (sumber terbuka, platform pencarian perusahaan) dibangun di atas Apache Lucene dan saat ini masih yang paling serbaguna dan mesin pencari open source populer. Solr adalah platform hebat dan patut diselidiki jika Anda terlibat dalam proyek yang membutuhkan mesin pencari. Pada tahun 2010 Elasticsearch muncul, dengan cepat mendapatkan popularitas. Meskipun Solr masih sulit untuk disiapkan dan dikonfigurasikan, bahkan untuk proyek kecil, Elasticsearch sangat mudah. Solr masih memiliki keunggulan dalam jumlah kemungkinan plugin yang memperluas fungsionalitas intinya, tetapi

Elasticsearch dengan cepat mengejar ketinggalan dan saat ini kemampuannya memiliki kualitas yang sebanding.

6.2.1 Langkah 1: Menetapkan tujuan penelitian

Bisakah Anda mendiagnosa penyakit pada akhir bab ini, hanya menggunakan komputer rumah Anda sendiri dan perangkat lunak gratis serta data di luar sana? Mengetahui apa yang ingin Anda lakukan dan bagaimana melakukannya adalah langkah pertama dalam proses ilmu data, seperti yang ditunjukkan pada gambar 6.17.



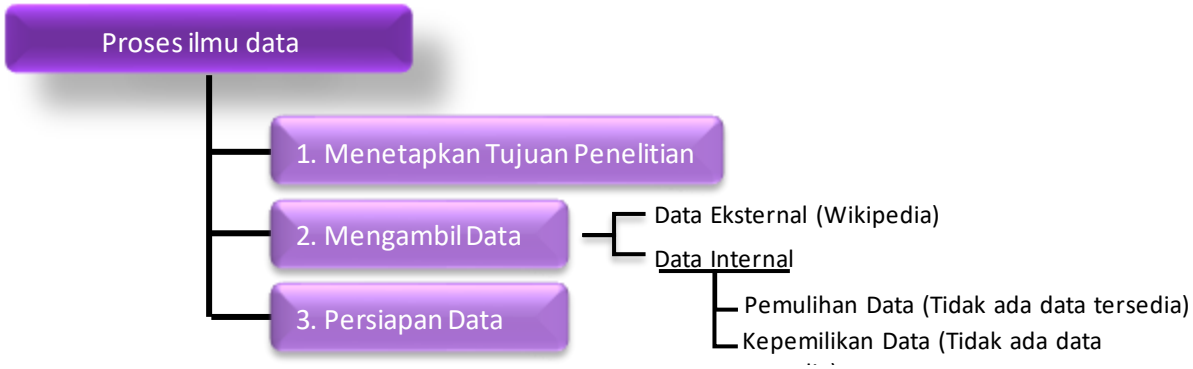
Gambar 6.17 Langkah 1 dalam proses ilmu data: menetapkan tujuan penelitian

- Tujuan utama Anda adalah membuat mesin pencari penyakit yang akan membantu dokter umum dalam mendiagnosis penyakit.
- Tujuan kedua Anda adalah membuat profil suatu penyakit: Kata kunci apa yang membedakannya dari penyakit lain?

Tujuan sekunder ini berguna untuk tujuan pendidikan atau sebagai masukan untuk penggunaan yang lebih lanjut seperti mendeteksi penyebaran epidemi dengan memanfaatkan media sosial. Dengan tujuan penelitian Anda dan rencana tindakan yang ditentukan, mari beralih ke langkah pengambilan data.

6.2.2 Langkah 2 dan 3: Pengambilan dan persiapan data

Pengambilan data dan persiapan data adalah dua langkah berbeda dalam proses ilmu data, dan meskipun ini tetap berlaku untuk studi kasus, kami akan mengeksplorasi keduanya di bagian yang sama.



Gambar 6.18 Proses sains data langkah 2: pengambilan data. Dalam hal ini tidak ada data internal; semua data akan diambil dari Wikipedia.

Dengan cara ini Anda dapat menghindari pengaturan penyimpanan intermedia lokal dan segera melakukan persiapan data saat data sedang diambil. Mari kita lihat di mana kita berada dalam proses ilmu data (lihat gambar 6.18).

Seperti yang ditunjukkan pada gambar 6.18, Anda memiliki dua kemungkinan sumber: data internal dan data eksternal.

- **Data internal**—Anda tidak memiliki informasi penyakit. Jika saat ini Anda bekerja di perusahaan farmasi atau rumah sakit, Anda mungkin lebih beruntung.
- **Data eksternal**—Yang dapat Anda gunakan untuk kasus ini hanyalah data eksternal. Anda memiliki beberapa kemungkinan, tetapi Anda akan menggunakan Wikipedia.
- Saat Anda menarik data dari Wikipedia, Anda harus menyimpannya di indeks pencarian Elastis lokal Anda, tetapi sebelum melakukannya, Anda harus menyiapkan datanya. Setelah data memasuki indeks Elasticsearch, data tidak dapat diubah; yang dapat Anda lakukan hanyalah menanyakannya. Lihat ikhtisar persiapan data pada gambar 6.19.

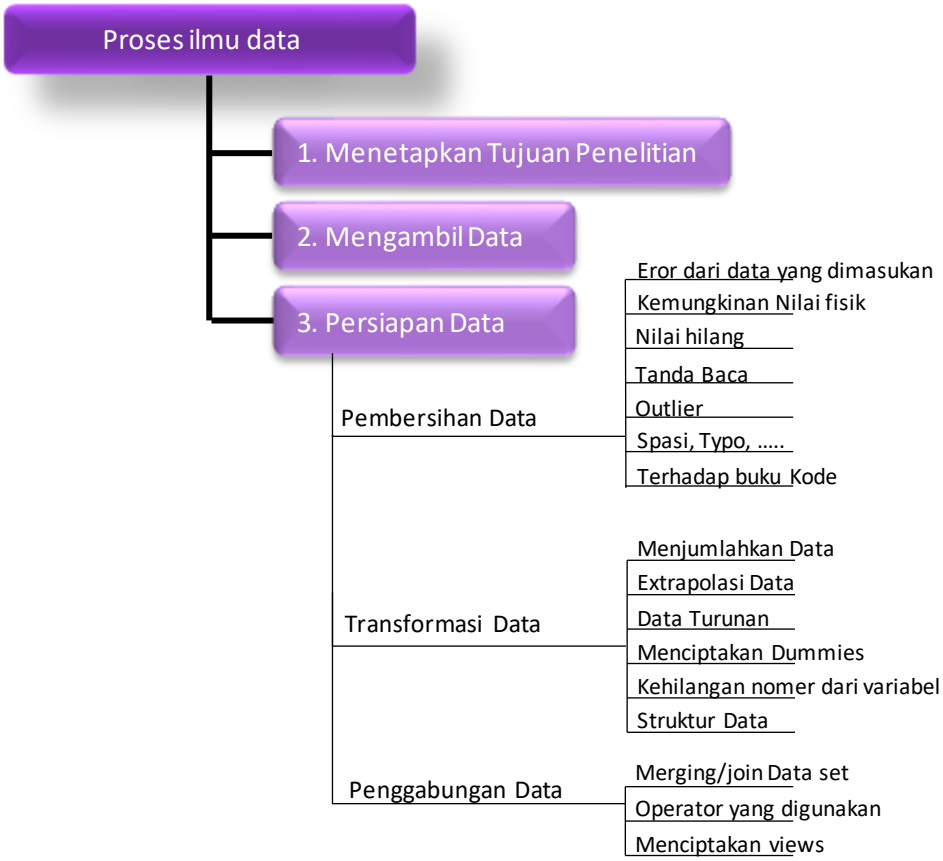
Seperti yang ditunjukkan pada gambar 6.19 ada tiga kategori persiapan data yang perlu dipertimbangkan:

- **Pembersihan data**—Data yang Anda ambil dari Wikipedia bisa jadi tidak lengkap atau salah. Kesalahan entri data dan kesalahan ejaan mungkin terjadi—bahkan informasi yang salah pun tidak dikecualikan. Untungnya, Anda tidak memerlukan daftar penyakit yang lengkap, dan Anda dapat menangani kesalahan pengejaan pada waktu pencarian; lebih lanjut tentang itu nanti. Berkat pustaka Python Wikipedia, data tekstual yang akan Anda terima sudah cukup bersih. Jika Anda mengikisnya secara manual, Anda perlu menambahkan pembersihan HTML, menghapus semua tag HTML. Kebenaran dari masalah ini adalah pencarian teks lengkap cenderung cukup kuat terhadap kesalahan umum seperti nilai yang salah. Bahkan jika Anda membuang tag HTML dengan sengaja, mereka tidak akan mempengaruhi hasil; tag HTML terlalu berbeda dari bahasa normal untuk mengganggu.
- **Transformasi data**—Anda tidak perlu banyak mengubah data pada saat ini; Anda ingin mencarinya apa adanya. Tapi Anda akan membuat perbedaan antara judul halaman, nama penyakit, dan isi halaman. Perbedaan ini hampir wajib untuk interpretasi hasil pencarian.
- **Menggabungkan data**—Semua data diambil dari satu sumber dalam kasus ini, sehingga Anda tidak perlu menggabungkan data. Perpanjangan yang mungkin untuk latihan ini adalah mendapatkan data penyakit dari sumber lain dan mencocokkan penyakitnya. Ini bukan tugas sepele karena tidak ada pengidentifikasi unik dan namanya seringkali sedikit berbeda.

Anda dapat melakukan pembersihan data hanya dalam dua tahap: saat menggunakan program Python yang menghubungkan Wikipedia ke Elasticsearch dan saat menjalankan sistem pengindeksan internal Elasticsearch:

- **Python**—Di sini Anda menentukan data apa yang akan Anda izinkan untuk disimpan oleh penyimpanan dokumen Anda, tetapi Anda tidak akan membersihkan data atau mengubah data pada tahap ini, karena Elasticsearch lebih baik melakukannya dengan sedikit usaha.

- **Elasticsearch**—Elasticsearch akan menangani manipulasi data (membuat indeks) secara tersembunyi. Anda masih dapat memengaruhi proses ini, dan Anda akan melakukannya secara lebih eksplisit nanti di bab ini.



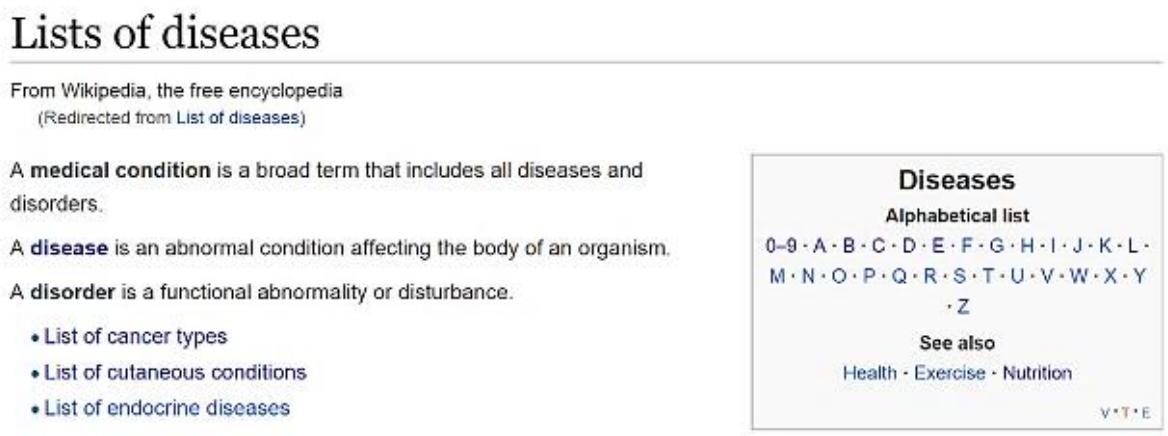
Gambar 6.19 Proses sains data langkah 3: persiapan data

Sekarang setelah Anda memiliki ikhtisar tentang langkah-langkah yang akan datang, mari mulai bekerja. Jika Anda mengikuti petunjuk dalam apendiks, Anda seharusnya sudah memiliki instans lokal Elastic-search dan berjalan. Pertama datang pengambilan data: Anda memerlukan informasi tentang berbagai penyakit. Anda memiliki beberapa cara untuk mendapatkan data semacam itu. Anda dapat meminta data perusahaan atau mendapatkan data dari Freebase atau sumber data terbuka dan gratis lainnya. Mendapatkan data Anda bisa menjadi tantangan, tetapi untuk contoh ini Anda akan menariknya dari Wikipedia. Ini agak ironis karena pencarian di situs Wikipedia sendiri ditangani oleh Elasticsearch. Wikipedia dulu memiliki sistemnya sendiri yang dibangun di atas Apache Lucene, tetapi menjadi tidak dapat dipertahankan, dan mulai Januari 2014 Wikipedia mulai menggunakan Elasticsearch sebagai gantinya.

Wikipedia memiliki halaman Daftar penyakit, seperti yang ditunjukkan pada gambar 6.20. Dari sini Anda dapat meminjam data dari daftar abjad.

Anda tahu data apa yang Anda inginkan; sekarang pergi ambil itu. Anda dapat mengunduh seluruh dump data Wikipedia. Jika mau, Anda dapat mengunduhnya ke http://meta.wikimedia.org/ wiki/Data_dump_torrents#enwiki.

Tentu saja, jika Anda mengindeks seluruh Wikipedia, indeks akan membutuhkan sekitar 40 GB penyimpanan. Jangan ragu untuk menggunakan solusi ini, tetapi demi mempertahankan penyimpanan dan lebar pita, kami akan membatasi diri dalam buku ini hanya untuk menarik data yang ingin kami gunakan. Pilihan lainnya adalah menggores halaman yang Anda butuhkan. Seperti Google, Anda dapat membuat program merayapi halaman dan mengambil seluruh HTML yang dirender. Ini akan membantu, tetapi Anda akan mendapatkan HTML yang sebenarnya, jadi Anda harus membersihkannya sebelum mengindeksnya. Juga, kecuali Anda adalah Google, situs web tidak terlalu menyukai perayap yang menggores halaman web mereka. Ini menciptakan jumlah lalu lintas yang tidak perlu, dan jika cukup banyak orang mengirim perayap, itu dapat membuat server HTTP bertekuk lutut, merusak kesenangan semua orang. Mengirim miliaran permintaan pada saat yang sama juga merupakan salah satu cara serangan denial of service (DoA) dilakukan. Jika Anda memang perlu mengikis situs web, buat skrip dalam jeda waktu antara setiap permintaan halaman. Dengan cara ini, pengikis Anda lebih mirip dengan perilaku pengunjung situs web biasa dan Anda tidak akan meledakkan server mereka.



Gambar 6.20 Halaman Daftar penyakit Wikipedia, titik awal untuk pengambilan data Anda

Untungnya, pencipta Wikipedia cukup pintar untuk mengetahui bahwa inilah yang akan terjadi dengan semua informasi ini terbuka untuk semua orang. Mereka telah menempatkan API di mana Anda dapat mengambil informasi Anda dengan aman. Anda dapat membaca selengkapnya di http://www.mediawiki.org/wiki/API:Main_page. Anda akan menggambar dari API. Dan Python tidak akan menjadi Python jika belum memiliki perpustakaan untuk melakukan pekerjaan itu. Sebenarnya ada beberapa, tapi yang termudah sudah cukup untuk kebutuhan Anda: Wikipedia. Aktifkan lingkungan virtual Python Anda dan instal semua perpustakaan yang Anda perlukan untuk sisa buku ini:

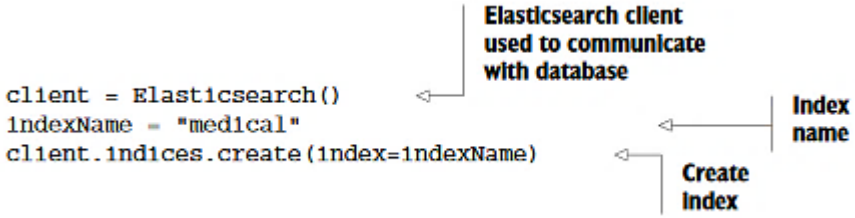
```
pip install wikipedia
pip install Elasticsearch
```

Anda akan menggunakan Wikipedia untuk masuk ke Wikipedia. Elasticsearch adalah pustaka utama Elasticsearch Python; dengan itu Anda dapat berkomunikasi dengan database Anda.

Buka juru bahasa Python favorit Anda dan impor perpustakaan yang diperlukan:

```
from elasticsearch import Elasticsearch
import wikipedia
```

Anda akan mengambil data dari API Wikipedia dan pada saat yang sama mengindeks pada instans Elasticsearch lokal Anda, jadi pertama-tama Anda harus menyiapkannya untuk penerimaan data.



Hal pertama yang Anda butuhkan adalah klien. Elasticsearch() dapat diinisialisasi dengan alamat tetapi defaultnya adalah localhost:9200. Elasticsearch()dan Elasticsearch ('localhost:9200') adalah hal yang sama: klien Anda terhubung ke node Elasticsearch lokal Anda. Kemudian Anda membuat indeks bernama "medical". Jika semua berjalan dengan baik, Anda akan melihat jawaban "acknowledged:true" seperti yang ditunjukkan pada gambar 6.21.

Elasticsearch mengklaim tanpa skema, artinya Anda dapat menggunakan Elasticsearch tanpa menentukan skema basis data dan tanpa memberi tahu Elasticsearch jenis data apa yang perlu diharapkan. Meskipun ini berlaku untuk kasus sederhana, Anda tidak dapat menghindari skema dalam jangka panjang, jadi mari kita buat, seperti yang ditunjukkan pada daftar berikut.

```
In [7]: client = Elasticsearch() #elasticsearch client used to communicate with the database
        indexName = "medical" #the index name
        #client.indices.delete(index=indexName) #delete an index
        client.indices.create(index=indexName) #create an index

Out[7]: {u'acknowledged': True}
```

Gambar 6.21 Membuat indeks Elasticsearch dengan Python-Elasticsearch

Listing 6.1 Adding a mapping to the document type

```
diseaseMapping = {
        'properties': {
            'name': {'type': 'string'},
            'title': {'type': 'string'},
            'fulltext': {'type': 'string'}
        }
    }
client.indices.put_mapping(index=indexName,
doc_type='diseases',body=diseaseMapping )
```

Defining a mapping and attributing it to the disease doc type.

The "diseases" doc type is updated with a mapping. Now we define the data it should expect.

Dengan cara ini Anda memberi tahu Elasticsearch bahwa indeks Anda akan memiliki jenis dokumen yang disebut "disease", dan Anda menyediakannya dengan jenis bidang untuk setiap bidang. Anda memiliki tiga bidang dalam dokumen penyakit: name, title, dan fulltext, semuanya bertipe string. Jika Anda tidak menyediakan pemetaan, Elasticsearch akan menebak jenisnya dengan melihat entri pertama yang diterimanya. Jika tidak mengenali bidang sebagai boolean, double, float, long, integer, atau date, itu akan menyetelnya menjadi string. Dalam hal ini, Anda tidak perlu menentukan pemetaan secara manual.

Sekarang mari beralih ke Wikipedia. Hal pertama yang ingin Anda lakukan adalah mengambil halaman Daftar penyakit, karena ini adalah titik masuk Anda untuk eksplorasi lebih lanjut:

```
dl = wikipedia.page("Lists_of_diseases")
```

Anda sekarang memiliki halaman pertama, tetapi Anda lebih tertarik pada halaman daftar karena mengandung link ke penyakit. Lihat tautannya:

```
dl.links
```

Halaman Daftar penyakit dilengkapi dengan lebih banyak tautan daripada yang akan Anda gunakan. Gambar 6.22 menunjukkan daftar abjad dimulai dari mata rantai keenam belas.

```
dl = wikipedia.page("Lists_of_diseases")
dl.links
```

```
In [9]: dl = wikipedia.page("Lists_of_diseases")
        dl.links

Out[9]: [u'Airborne disease',
         u'Contagious disease',
         u'Cryptogenic disease',
         u'Disease',
         u'Disseminated disease',
         u'Endocrine disease',
         u'Environmental disease',
         u'Eye disease',
         u'Lifestyle disease',
         u'List of abbreviations for diseases and disorders',
         u'List of autism-related topics',
         u'List of basic exercise topics',
         u'List of cancer types',
         u'List of communication disorders',
         u'List of cutaneous conditions',
         u'List of diseases (0\u20139)',
         u'List of diseases (A)',
         u'List of diseases (B)',
```

Gambar 6.22 Tautan pada halaman Wikipedia Daftar penyakit. Ini memiliki lebih banyak tautan daripada yang Anda perlukan.

Halaman ini memiliki banyak tautan, tetapi hanya daftar abjad yang menarik bagi Anda, jadi simpan hanya yang:

```
diseaseListArray = []
for link in dl.links[15:42]:
    try:
        diseaseListArray.append(wikipedia.page(link))
    except Exception,e:
        print str(e)
```

Anda mungkin memperhatikan bahwa subset tersebut di-hardcode, karena Anda tahu itu adalah entri ke-16 hingga ke-43 dalam array. Jika Wikipedia menambahkan bahkan satu tautan sebelum yang Anda minati, itu akan membuang hasilnya. Praktik yang lebih baik adalah menggunakan ekspresi reguler untuk tugas ini. Untuk tujuan eksplorasi, hardcoding nomor entri baik-baik saja, tetapi jika ekspresi reguler adalah sifat kedua bagi Anda atau Anda bermaksud mengubah kode ini menjadi pekerjaan batch, ekspresi reguler disarankan. Anda dapat menemukan informasi lebih lanjut tentang mereka di https://docs.python.org/2/howto/regex.html. Satu kemungkinan untuk versi regex adalah potongan kode berikut.

```
diseaseListArray = []
check = re.compile("List of diseases*")
for link in dl.links:
    if check.match(link):
        try:
            diseaseListArray.append(wikipedia.page(link))
        except Exception,e:
            print str(e)
```

```
In [16]: diseaseListArray

Out[16]: [<WikipediaPage 'List of diseases (0-9)'>,
          <WikipediaPage 'List of diseases (A)'>,
          <WikipediaPage 'List of diseases (B)'>,
          <WikipediaPage 'List of diseases (C)'>,
          <WikipediaPage 'List of diseases (D)'>,
          <WikipediaPage 'List of diseases (E)'>,
          <WikipediaPage 'List of diseases (F)'>,
          <WikipediaPage 'List of diseases (G)'>,
          <WikipediaPage 'List of diseases (H)'>,
```

Figure 6.23 First Wikipedia disease list, "list of diseases (0-9)"

Gambar 6.23 menunjukkan entri pertama dari apa yang Anda kejar: penyakit itu sendiri.

```
diseaseListArray[0].links
```

Saatnya mengindeks penyakit. Setelah diindeks, entri data dan persiapan data secara efektif berakhir, seperti yang ditunjukkan dalam daftar berikut.

Listing 6.2 Indexing diseases from Wikipedia

```
checkList = [["0","1","2","3","4","5","6","7","8","9"],
["A"],["B"],["C"],["D"],["E"],["F"],["G"],["H"],
["I"],["J"],["K"],["L"],["M"],["N"],["O"],["P"],
["Q"],["R"],["S"],["T"],["U"],["V"],["W"],["X"],["Y"],["Z"]]
docType = 'diseases'
for diseaselistNumber, diseaselist in enumerate(diseaseListArray):
    for disease in diseaselist.links:
        try:
            if disease[0] in checkList[diseaselistNumber]
and disease[0:3] !="List":
                currentPage = wikipedia.page(disease)
                client.index(index=indexName,
doc_type=docType,id = disease, body={"name": disease,
     "title":currentPage.title ,
"fulltext":currentPage.content})
        except Exception,e:
            print str(e)
```

Looping through disease lists.

The checklist is an array containing an array of allowed first characters. If a disease doesn't comply, skip it.

Document type you'll index.

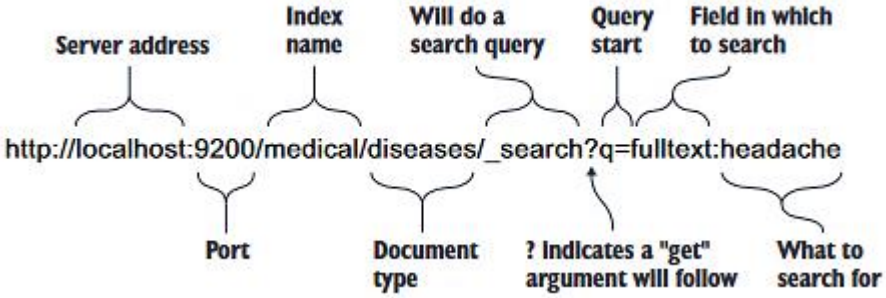
Looping through lists of links for every disease list.

First check if it's a disease, then index it.

Karena setiap halaman daftar akan memiliki tautan yang tidak Anda perlukan, periksa untuk melihat apakah suatu entri adalah penyakit. Anda menunjukkan untuk setiap daftar dari karakter apa penyakit itu dimulai, jadi Anda memeriksanya. Selain itu, Anda mengecualikan tautan yang dimulai dengan "daftar" karena ini akan muncul begitu Anda masuk ke daftar L penyakit. Pengecekannya agak naif, tetapi biaya untuk memiliki beberapa entri yang tidak diinginkan agak rendah karena algoritme pencarian akan mengecualikan hasil yang tidak relevan begitu Anda mulai melakukan kueri. Untuk setiap penyakit, Anda mengindeks nama penyakit dan teks lengkap halaman. Nama tersebut juga digunakan sebagai ID indeksnya; ini berguna untuk beberapa fitur Elasticsearch tingkat lanjut tetapi juga untuk pencarian cepat di browser. Misalnya, coba URL ini di browser Anda: http://localhost:9200/medical/diseases/11%20beta%20hydroxylase%20deficiency. Judul diindeks secara terpisah; dalam kebanyakan

kasus, nama tautan dan judul halaman akan identik dan terkadang judul berisi nama alternatif untuk penyakit tersebut.

Dengan setidaknya beberapa penyakit yang diindeks, Anda dapat menggunakan URI Elasticsearch untuk pencarian sederhana. Lihatlah pencarian seluruh tubuh untuk kata sakit kepala pada gambar 6.24. Anda sudah bisa melakukan ini saat mengindeks; Elasticsearch dapat memperbarui indeks dan mengembalikan kueri untuknya secara bersamaan.



Gambar 6.24 Kumpulan contoh URL Elasticsearch

Jika Anda tidak menanyakan indeks, Anda masih bisa mendapatkan beberapa hasil tanpa mengetahui apa pun tentang indeks tersebut. Menentukan http://localhost:9200/ medical/diseases/_search akan mengembalikan lima hasil pertama. Untuk tampilan data yang lebih terstruktur, Anda dapat meminta pemetaan jenis dokumen ini di http://localhost:9200/medical/penyakit/_mapping?pretty. Argumen pretty get menunjukkan JSON yang dikembalikan dalam format yang lebih mudah dibaca, seperti dapat dilihat pada gambar 6.25. Pemetaan tampaknya seperti yang Anda tentukan: semua bidang bertipe string.

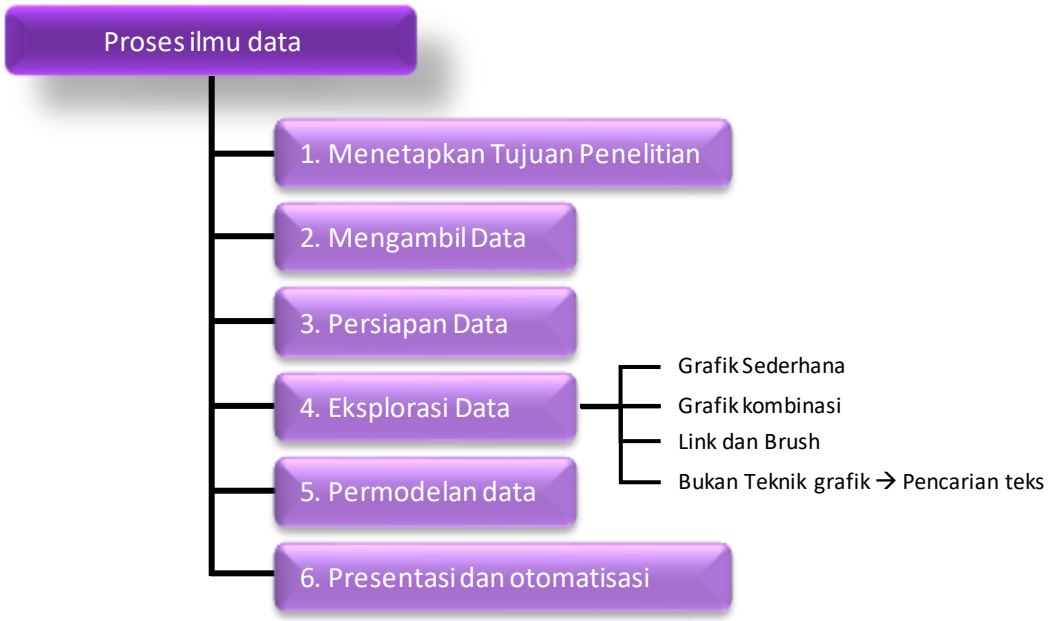
URL Elasticsearch memang berguna, namun tidak cukup untuk kebutuhan Anda. Anda masih memiliki penyakit untuk didiagnosis, dan untuk ini Anda akan mengirimkan permintaan POST ke Elasticsearch melalui pustaka Elasticsearch Python Anda. Dengan pengambilan data dan persiapan selesai, Anda dapat melanjutkan untuk mengeksplorasi data Anda.

Gambar 6.25 Pemetaan tipe dokumen penyakit melalui URL Elasticsearch

6.2.3 Langkah 4: Eksplorasi data

Itu bukan lupus. Itu tidak pernah lupus!
—Dr. Rumah Rumah M.D.

Eksplorasi data menandai studi kasus ini, karena tujuan utama proyek (diagnostik penyakit) adalah cara khusus untuk mengeksplorasi data dengan menanyakan gejala penyakit. Gambar 6.26 menunjukkan beberapa teknik eksplorasi data, tetapi dalam kasus ini non-grafis: menafsirkan hasil kueri penelusuran teks.



Gambar 6.26 Proses sains data langkah 4: eksplorasi data

Momen kebenaran ada di sini: dapatkah Anda menemukan penyakit tertentu dengan memberi makan gejalanya pada mesin pencari Anda? Mari pertama-tama pastikan Anda memiliki dasar-dasarnya dan berjalan. Impor perpustakaan Elasticsearch dan tentukan pengaturan pencarian global:

```
from elasticsearch import Elasticsearch
client = Elasticsearch()
indexName = "medical"
docType="diseases"
searchFrom = 0
searchSize= 3
```

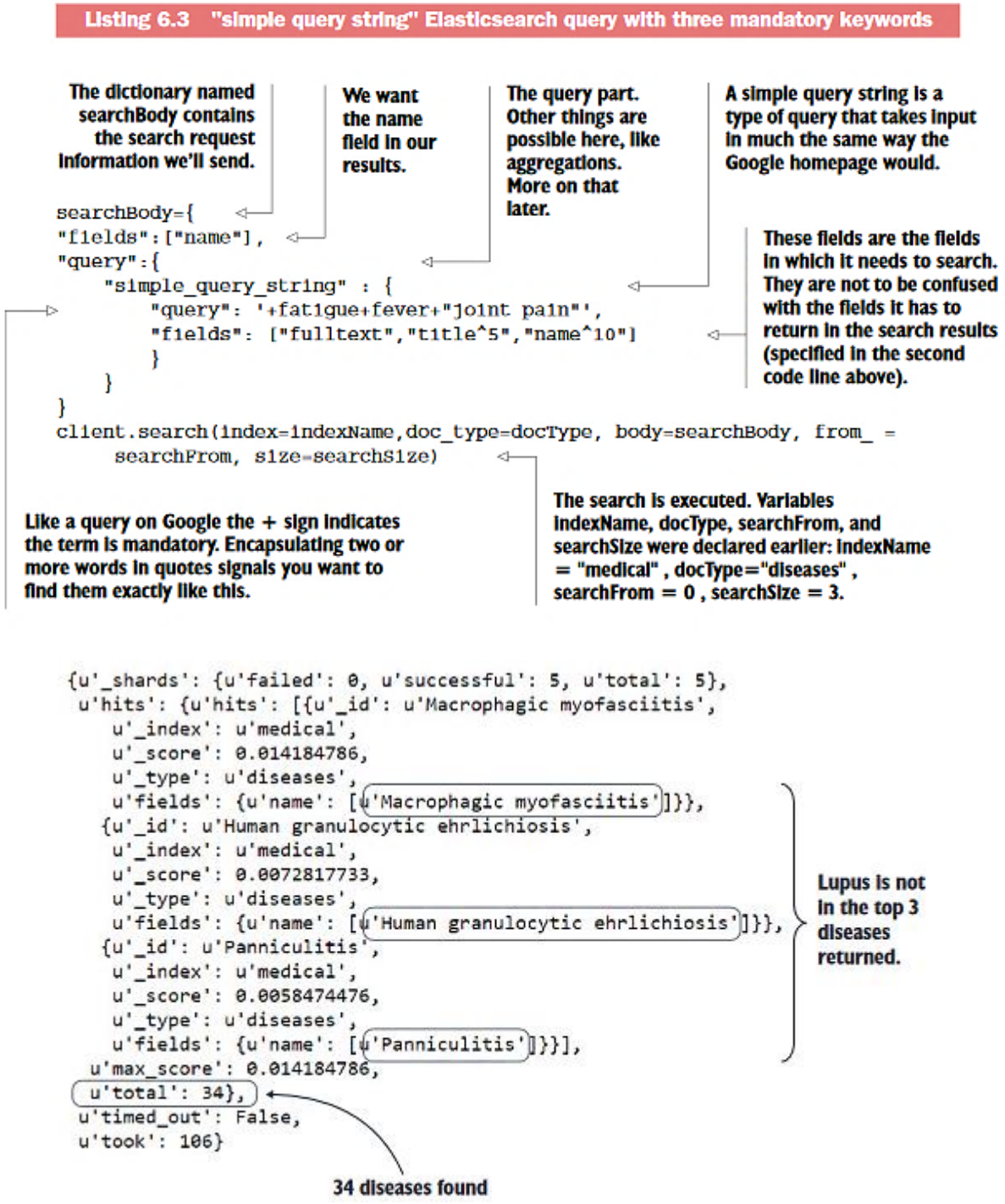
Anda hanya akan mengembalikan tiga hasil pertama; standarnya adalah lima. Elasticsearch memiliki bahasa kueri JSON yang rumit; setiap pencarian adalah permintaan POST ke server dan akan dijawab dengan jawaban JSON. Secara kasar, bahasa ini terdiri dari tiga bagian besar: kueri, filter, dan agregasi. Kueri mengambil kata kunci pencarian dan menempatkannya melalui satu atau lebih penganalisa sebelum kata-kata tersebut dicari di indeks. Kita akan membahas lebih dalam tentang penganalisis nanti di bab ini. Filter mengambil kata kunci seperti kueri tetapi tidak mencoba menganalisis apa yang Anda berikan; itu memfilter pada kondisi yang kami berikan. Dengan demikian, filter tidak terlalu rumit tetapi berkali-kali lebih

efisien karena filter juga disimpan sementara di dalam Elasticsearch jika Anda menggunakan filter yang sama dua kali. Agregasi dapat dibandingkan dengan grup SQL; ember kata-kata akan dibuat, dan untuk setiap ember statistik yang relevan dapat dihitung. Masing-masing dari ketiga kompartemen ini memiliki banyak opsi dan fitur, membuat penjabaran seluruh bahasa di sini menjadi tidak mungkin. Untungnya, tidak perlu membahas kerumitan yang dapat diwakili oleh kueri pencarian Elastis. Kami akan menggunakan "Bahasa kueri string kueri", cara untuk kueri data yang sangat mirip dengan bahasa kueri penelusuran Google. Jika, misalnya, Anda ingin agar istilah pencarian bersifat wajib, tambahkan tanda tambah (+); jika Anda ingin mengecualikan istilah pencarian, gunakan tanda minus (-). Meminta Elasticsearch tidak disarankan karena menurunkan kinerja; mesin pencari pertama-tama harus menerjemahkan string kueri ke dalam bahasa kueri JSON aslinya. Tapi untuk tujuan Anda itu akan bekerja dengan baik; selain itu, kinerja tidak akan menjadi faktor pada beberapa ribu rekaman yang Anda miliki di indeks. Sekarang saatnya untuk menanyakan data penyakit Anda.

Tujuan Utama Proyek: Mendiagnosis Penyakit Berdasarkan Gejalanya

Jika Anda pernah melihat serial televisi populer House M.D., kalimat “It’s never lupus” mungkin terdengar asing. Lupus adalah salah satu jenis penyakit autoimun, dimana sistem kekebalan tubuh menyerang bagian tubuh yang sehat. Mari kita lihat gejala apa yang dibutuhkan mesin pencari Anda untuk menentukan bahwa Anda sedang mencari lupus. Mulailah dengan tiga gejala: kelelahan, demam, dan nyeri sendi. Pasien imajiner Anda memiliki ketiganya (dan lebih banyak lagi), jadi jadikan semuanya wajib dengan menambahkan tanda tambah sebelum masing-masing:

Di searchBody, yang memiliki struktur JSON, Anda menentukan bidang yang ingin Anda lihat kembali, dalam hal ini nama penyakit sudah cukup. Anda menggunakan sintaks string kueri untuk mencari di semua bidang yang diindeks: teks lengkap, judul, dan nama. Dengan menambahkan ^ Anda dapat memberi bobot pada setiap bidang. Jika suatu gejala muncul di judul, itu lima kali lebih penting daripada di teks terbuka; jika itu terjadi pada nama itu sendiri, itu dianggap sepuluh kali lebih penting. Perhatikan bagaimana "nyeri sendi" dibungkus dengan tanda kutip. Jika Anda tidak memiliki tanda “ ”, persendian dan nyeri akan dianggap sebagai dua kata kunci yang terpisah, bukan satu frasa. Di Elasticsearch ini disebut pencocokan frasa. Mari kita lihat hasilnya pada gambar 6.27.



Gambar 6.27 Pencarian pertama Lupus dengan 34 hasil

Gambar 6.27 menunjukkan tiga hasil teratas dari 34 penyakit yang cocok. Hasilnya diurutkan berdasarkan skor pencocokannya, variabel _score. Skor yang cocok bukanlah hal yang sederhana untuk dijelaskan; ini mempertimbangkan seberapa cocok penyakit tersebut dengan kueri Anda dan berapa kali kata kunci ditemukan, bobot yang Anda berikan, dan seterusnya. Saat ini, lupus bahkan tidak muncul di tiga hasil teratas. Beruntung bagi Anda, lupus memiliki gejala lain yang berbeda: ruam. Ruam tidak selalu muncul di wajah seseorang, tetapi itu memang terjadi dan dari sinilah lupus mendapatkan namanya: ruam wajah membuat orang samar-samar menyerupai serigala. Pasien Anda mengalami ruam tetapi

bukan ruam khas di wajah, jadi tambahkan "ruam" pada gejalanya tanpa menyebutkan wajahnya.

```
"query": '+fatigue+fever+"joint pain"+rash',

{u'_shards': {u'failed': 0, u'successful': 5, u'total': 5},
 u'hits': {u'hits': [{u'_id': u'Human granulocytic ehrlichiosis',
    u'_index': u'medical',
    u'_score': 0.009902062,
    u'_type': u'diseases',
    u'fields': {u'name': [u'Human granulocytic ehrlichiosis']}},
   {u'_id': u'Lupus erythematosus',
    u'_index': u'medical',
    u'_score': 0.009000875,
    u'_type': u'diseases',
    u'fields': {u'name': [u'Lupus erythematosus']}},
   {u'_id': u'Panniculitis',
    u'_index': u'medical',
    u'_score': 0.007950994,
    u'_type': u'diseases',
    u'fields': {u'name': [u'Panniculitis']}}],
  u'max_score': 0.009902062,
  u'total': 6},
 u'timed_out': False,
 u'took': 15}
```

Gambar 6.28 Upaya pencarian Lupus kedua dengan enam hasil dan lupus di posisi tiga teratas

Hasil pencarian baru ditunjukkan pada gambar 6.28. Sekarang hasilnya dipersempit menjadi enam dan lupus masuk tiga besar. Pada titik ini, mesin pencari mengatakan Human Granulocytic Ehrlichiosis (HGE) lebih mungkin. HGE adalah penyakit yang disebarkan oleh kutu, seperti penyakit Lyme yang terkenal. Sekarang seorang dokter yang cakap pasti sudah mengetahui penyakit apa yang menjangkiti pasien Anda, karena dalam menentukan penyakit banyak faktor yang berperan, lebih dari yang dapat Anda masukkan ke dalam mesin pencari sederhana Anda. Misalnya, ruam hanya terjadi pada 10% pasien HGE dan 50% pasien lupus. Lupus muncul perlahan, sedangkan HGE dipicu oleh gigitan kutu. Basis data pembelajaran mesin tingkat lanjut yang diberi semua informasi ini dengan cara yang lebih terstruktur dapat membuat diagnosis dengan kepastian yang jauh lebih besar. Mengingat Anda perlu puas dengan halaman Wikipedia, Anda memerlukan gejala lain untuk mengonfirmasi bahwa itu lupus. Pasien mengalami nyeri dada, jadi tambahkan ini ke dalam daftar.''

```
"query": '+fatigue+fever+"joint pain"+rash+"chest pain"',
```

Hasilnya ditunjukkan pada gambar 6.29. Sepertinya itu lupus. Butuh beberapa saat untuk sampai pada kesimpulan ini, tetapi Anda sampai di sana. Tentu saja, cara Anda mempresentasikan Elasticsearch dengan gejalanya terbatas. Anda hanya menggunakan istilah tunggal (“kelelahan”) atau frasa literal (“nyeri sendi”). Ini berhasil untuk contoh ini, tetapi Elasticsearch lebih fleksibel dari ini. Ini dapat mengambil ekspresi reguler dan melakukan

pencarian kabur, tetapi itu di luar cakupan buku ini, meskipun beberapa contoh disertakan dalam kode yang dapat diunduh.

```
{u'_shards': {u'failed': 0, u'successful': 5, u'total': 5},
 u'hits': {u'hits': [{u'_id': u'Lupus erythematosus',
    u'_index': u'medical',
    u'_score': 0.010452312,
    u'_type': u'diseases',
    u'fields': {u'name': [u'Lupus erythematosus']}}],
  u'max_score': 0.010452312,
  u'total': 1},
 u'timed_out': False,
 u'took': 11}
```

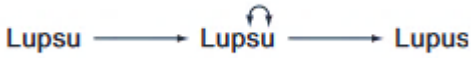
Gambar 6.29 Lupus pencarian ketiga: dengan gejala yang cukup untuk menentukan itu pasti lupus

Menangani Kesalahan Eja: Damerau-Levenshtein

Misalnya seseorang mengetik "lupsu", bukan "lupus". Kesalahan ejaan terjadi setiap saat dan di semua jenis dokumen buatan manusia. Untuk menangani data ini para ilmuwan sering menggunakan Damerau-Levenshtein. Jarak Damerau-Levenshtein antara dua string adalah jumlah operasi yang diperlukan untuk mengubah satu string menjadi string lainnya. Empat operasi diperbolehkan untuk menghitung jarak:

- **Penghapusan —**Menghapus karakter dari string.
- **Penyisipan—**Menambahkan karakter ke string.
- **Pergantian—**Menggantikan satu karakter dengan karakter lainnya. Tanpa substitusi dihitung sebagai satu operasi, mengubah satu karakter menjadi karakter lain akan membutuhkan dua operasi: satu penghapusan dan satu penyisipan.
- **Transposisi dua karakter yang berdekatan—**Tukar dua karakter yang berdekatan.

Operasi terakhir (transposisi) inilah yang membuat perbedaan antara jarak Levenshtein tradisional dan jarak Damerau-Levenshtein. Operasi terakhir inilah yang membuat kesalahan ejaan disleksia kita berada dalam batas yang dapat diterima. Damerau- Levenshtein memaafkan kesalahan transposisi ini, yang membuatnya bagus untuk mesin telusur, tetapi juga digunakan untuk hal lain seperti menghitung perbedaan antara string DNA. Gambar 6.30 menunjukkan bagaimana transformasi dari “lupsu” menjadi “lupus” dilakukan dengan satu transposisi.



Gambar 6.30 Transposisi karakter yang berdekatan adalah salah satu operasi dalam jarak Damerau-Levenshtein. Tiga lainnya adalah penyisipan, penghapusan, dan penggantian.

Hanya dengan ini, Anda telah mencapai tujuan pertama Anda: mendiagnosis penyakit. Tapi jangan lupa tentang tujuan proyek sekunder Anda: profil penyakit.

Tujuan Sekunder Proyek: Profil Penyakit

Yang Anda inginkan adalah daftar kata kunci yang sesuai dengan penyakit pilihan Anda. Untuk ini, Anda akan menggunakan agregasi istilah penting. Kalkulasi skor untuk menentukan kata mana yang signifikan sekali lagi merupakan kombinasi dari faktor-faktor, tetapi kira-kira bermuara pada perbandingan berapa kali sebuah istilah ditemukan dalam kumpulan hasil dibandingkan dengan semua dokumen lainnya. Dengan cara ini, Elasticsearch memprofilkan hasil Anda dengan menyediakan kata kunci yang membedakannya dari data lain. Mari kita lakukan pada diabetes, penyakit umum yang dapat terjadi dalam berbagai bentuk:

Listing 6.4 Significant terms Elasticsearch query for "diabetes"

The dictionary named searchBody contains the search request information we'll send.

We want the name field in our results.

The query part.

A filtered query has two possible components: a query and a filter. The query performs a search while the filter matches exact values only and is therefore way more efficient but restrictive.

The filter part of the filtered query. A query part isn't mandatory; a filter is sufficient.

We want to filter the name field and keep only if it contains the term diabetes.

```
searchBody={
"fields":["name"],
"query":{
    "filtered" : {
        "filter": {
            'term': {'name':'diabetes'}
        }
    }
},
"aggregations" : {
        "DiseaseKeywords" : {
            "significant_terms" : { "field" : "fulltext", "size":30 }
        }
    }
}
client.search(index=indexName,doc_type=docType,
body=searchBody, from_ = searchFrom, size=searchSize)
```

DiseaseKeywords is the name we give to our aggregation.

A significant term aggregation can be compared to keyword detection. The internal algorithm looks for words that are "more important" for the selected set of documents than they are in the overall population of documents.

An aggregation can generally be compared to a group by in SQL. It's mostly used to summarize values of a numeric variable over the distinct values within one or more variables.

Anda melihat kode baru di sini. Anda menyingkirkan pencarian string kueri dan menggunakan filter sebagai gantinya. Filter dikemas dalam bagian kueri karena permintaan pencarian dapat digabungkan dengan filter. Ini tidak terjadi dalam contoh ini, tetapi ketika ini terjadi, Elastic-search pertama-tama akan menerapkan filter yang jauh lebih efisien sebelum mencoba melakukan pencarian. Jika Anda ingin menelusuri subkumpulan data Anda, sebaiknya tambahkan filter untuk terlebih dahulu membuat subkumpulan ini. Untuk mendemonstrasikan ini, perhatikan dua potongan kode berikut. Mereka menghasilkan hasil yang sama tetapi mereka bukan hal yang persis sama.

String kueri sederhana yang mencari "diabetes" dalam nama penyakit:

```
"query":{
    "simple_query_string" : {
        "query": 'diabetes',
        "fields": ["name"]
        }
    }
```

Istilah filter menyaring semua penyakit dengan nama "diabetes":

```
"query":{
    "filtered" : {
        "filter": {
            'term': {'name':'diabetes'}
        }
    }
}
```

Meskipun tidak akan ditampilkan pada sejumlah kecil data yang Anda inginkan, filternya jauh lebih cepat daripada pencarian. Permintaan pencarian akan menghitung skor pencarian untuk masing-masing penyakit dan memberi peringkat yang sesuai, sedangkan filter hanya menyaring semua penyakit yang tidak sesuai. Filter sejauh ini tidak sekompleks pencarian yang sebenarnya: itu bisa "ya" atau "tidak" dan ini terbukti dalam output. Skornya adalah 1 untuk semuanya; tidak ada perbedaan yang dibuat dalam set hasil. Output terdiri dari dua bagian sekarang karena agregasi istilah yang signifikan. Sebelumnya Anda hanya mendapatkan hit; sekarang Anda memiliki hits dan agregasi. Pertama, lihat hit pada gambar 6.31.

```
u'hits': {u'hits': [{u'_id': u'Diabetes mellitus',
   u'_index': u'medical',
   u'_score': 1.0,
   u'_type': u'diseases',
   u'fields': {u'name': [u'Diabetes mellitus']}},
  {u'_id': u'Diabetes insipidus, nephrogenic type 3',
   u'_index': u'medical',
   u'_score': 1.0,
   u'_type': u'diseases',
   u'fields': {u'name': [u'Diabetes insipidus, nephrogenic type 3']}},
  {u'_id': u'Ectodermal dysplasia arthrogryposis diabetes mellitus',
   u'_index': u'medical',
   u'_score': 1.0,
   u'_type': u'diseases',
   u'fields': {u'name': [u'Ectodermal dysplasia arthrogryposis diabetes mellitus']}}],
 u'max_score': 1.0,
 u'total': 27},
u'timed_out': False,
u'took': 44}
```

Gambar 6.31 Keluaran dari kueri terfilter dengan filter "diabetes" pada nama penyakit

Ini seharusnya terlihat akrab sekarang dengan satu pengecualian penting: semua hasil memiliki skor 1. Selain lebih mudah dilakukan, filter di-cache oleh Elasticsearch untuk

sementara. Dengan cara ini, permintaan selanjutnya dengan filter yang sama bahkan lebih cepat, sehingga menghasilkan keunggulan performa yang sangat besar dibandingkan kueri penelusuran.

Kapan sebaiknya Anda menggunakan filter dan kapan kueri penelusuran? Aturannya sederhana: gunakan filter bila memungkinkan dan gunakan kueri penelusuran untuk penelusuran teks lengkap saat peringkat di antara hasil diperlukan untuk mendapatkan hasil yang paling menarik di atas. Sekarang perhatikan suku-suku penting pada gambar 6.32.


```
{u'_shards': {u'failed': 0, u'successful': 5, u'total': 5},
 u'aggregations': {u'DiseaseKeywords': {u'buckets': [{u'bg_count': 18,
     u'doc_count': 9,
     u'key': u'siphon',
     u'score': 62.84567901234568},
    {u'bg_count': 18,
     u'doc_count': 9,
     u'key': u'diabainein',
     u'score': 62.84567901234568},
    {u'bg_count': 18,
     u'doc_count': 9,
     u'key': u'bainein',
     u'score': 62.84567901234568},
    {u'bg_count': 20,
     u'doc_count': 9,
     u'key': u'passer',
     u'score': 56.52777777777778},
    {u'bg_count': 14,
     u'doc_count': 7,
     u'key': u'ndi',
     u'score': 48.87997256515774},
```


Gambar 6.32 Agregasi istilah signifikan diabetes, lima kata kunci pertama

Jika Anda melihat lima kata kunci pertama pada gambar 6.32, Anda akan melihat bahwa empat kata kunci teratas terkait dengan asal mula diabetes. Paragraf Wikipedia berikut menawarkan bantuan:

Kata diabetes (/ˌdaɪ.əˈbiːtiːz/ atau /ˌdaɪ.əˈbiːtɨs/) berasal dari bahasa Latin diabe–te–s, yang selanjutnya berasal dari bahasa Yunani Kuno διαβήτης (diabe–te–s) yang secara harfiah berarti berarti "pelintas; menyedot" [69]. Tabib Yunani kuno Aretaeus dari Cappadocia (fl. Abad ke-1 M) menggunakan kata itu, dengan arti yang dimaksud "keluarnya urin yang berlebihan," sebagai nama penyakit [70, 71, 72]. Pada akhirnya, kata tersebut berasal dari bahasa Yunani διαβαίνειν (diabainein), artinya "melalui", [69] yang terdiri dari διά - (dia-), artinya "melalui" dan βαίνειν (bainein), artinya "pergi" [70]. Kata "diabetes" pertama kali tercatat dalam bahasa Inggris, dalam bentuk diabete, dalam sebuah teks kedokteran yang ditulis sekitar tahun 1425.

—Halaman Wikipedia Diabetes_mellitus

Ini memberi tahu Anda dari mana kata diabetes berasal: "orang yang lewat; menyedot" dalam bahasa Yunani. Itu juga menyebutkan diabainein dan bainein. Anda mungkin telah mengetahui bahwa kata kunci yang paling relevan untuk suatu penyakit adalah definisi dan asal sebenarnya. Untungnya kami meminta 30 kata kunci, jadi mari kita pilih beberapa kata

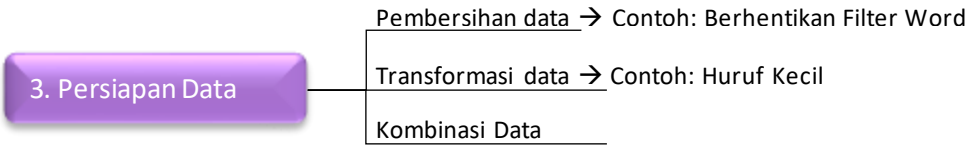
kunci yang lebih menarik seperti ndi. ndi adalah versi huruf kecil dari NDI, atau “Nephrogenic Diabetes Insipidus,” bentuk diabetes yang didapat paling umum. Kata kunci huruf kecil dikembalikan karena begitulah cara mereka disimpan dalam indeks saat kami memasukkannya melalui penganalisis standar saat mengindeks. Kami tidak menentukan apa pun saat mengindeks, jadi penganalisis standar digunakan secara default. Kata kunci menarik lainnya di 30 teratas adalah avp, gen yang berhubungan dengan diabetes; haus, gejala diabetes; dan Amiloride, obat untuk diabetes. Kata kunci ini tampaknya memang menggambarkan diabetes, tetapi kami kehilangan kata kunci multi-istilah; kami hanya menyimpan istilah individu dalam indeks karena ini adalah perilaku default. Kata-kata tertentu tidak akan pernah muncul dengan sendirinya karena tidak sering digunakan tetapi masih signifikan bila digunakan bersama istilah lain. Saat ini kami kehilangan hubungan antara istilah-istilah tertentu. Ambil avp, misalnya; jika avp selalu ditulis dalam bentuk lengkapnya "Diabetes Insipidus Nefrogenik", itu tidak akan diambil. Menyimpan n-gram (kombinasi n jumlah kata) menghabiskan ruang penyimpanan, dan menggunakannya untuk kueri atau agregasi membebani server pencarian. Memutuskan di mana harus berhenti adalah latihan keseimbangan dan bergantung pada data dan kasus penggunaan Anda.

Secara umum, bigram (kombinasi dari dua istilah) berguna karena bigram yang bermakna ada dalam bahasa alami, meskipun 10 gram tidak terlalu banyak. Konsep kunci Bigram akan berguna untuk pembuatan profil penyakit, tetapi untuk membuat agregasi istilah penting bigram tersebut, Anda perlu menyimpannya sebagai bigram dalam indeks Anda. Seperti yang sering terjadi dalam ilmu data, Anda harus mundur beberapa langkah untuk membuat beberapa perubahan. Mari kembali ke tahap persiapan data.

6.2.4 Langkah 3 ditinjau kembali: Persiapan data untuk pembuatan profil penyakit

Seharusnya tidak mengejutkan bahwa Anda kembali ke persiapan data, seperti yang ditunjukkan pada gambar 6.33. Bagaimanapun, proses ilmu data adalah proses yang berulang. Saat Anda mengindeks data Anda, Anda hampir tidak melakukan pembersihan data atau transformasi data. Anda dapat menambahkan pembersihan data sekarang, misalnya dengan menghentikan pemfilteran kata. Stop word adalah kata-kata yang sangat umum sehingga sering dibuang karena dapat mencemari hasil. Kami tidak akan menghentikan pemfilteran kata (atau pembersihan data lainnya) di sini, tetapi jangan ragu untuk mencobanya sendiri.

Untuk mengindeks bigram, Anda perlu membuat filter token dan penganalisis teks Anda sendiri. Filter token mampu melakukan transformasi pada token. Filter token khusus Anda perlu menggabungkan token untuk membuat n-gram, juga disebut herpes zoster. Tokenizer pencarian elastis default disebut tokenizer standar, dan itu akan mencari batas kata, seperti ruang antar kata, untuk memotong teks menjadi token atau istilah yang berbeda. Lihat pengaturan baru untuk indeks penyakit Anda, seperti yang ditampilkan dalam daftar berikut.



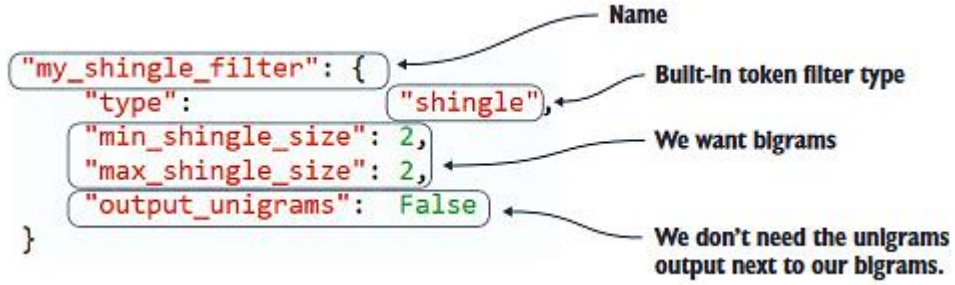
Gambar 6.33 Proses ilmu data langkah 3: persiapan data. Pembersihan data untuk teks dapat menghentikan pemfilteran kata; transformasi data dapat berupa huruf kecil dari karakter.

Listing 6.5 Updating Elasticsearch index settings

```
settings={
    "analysis": {
            "filter": {
                "my_shingle_filter": {
                    "type":             "shingle",
                    "min_shingle_size": 2,
                    "max_shingle_size": 2,
                    "output_unigrams":  False
                }
            },
            "analyzer": {
                "my_shingle_analyzer": {
                    "type":             "custom",
                    "tokenizer":        "standard",
                    "filter": [
                        "lowercase",
                        "my_shingle_filter"
                    ]
                }
            }
        }
    }
client.indices.close(index=indexName)
client.indices.put_settings(index=indexName , body = settings)
client.indices.open(index=indexName)
```

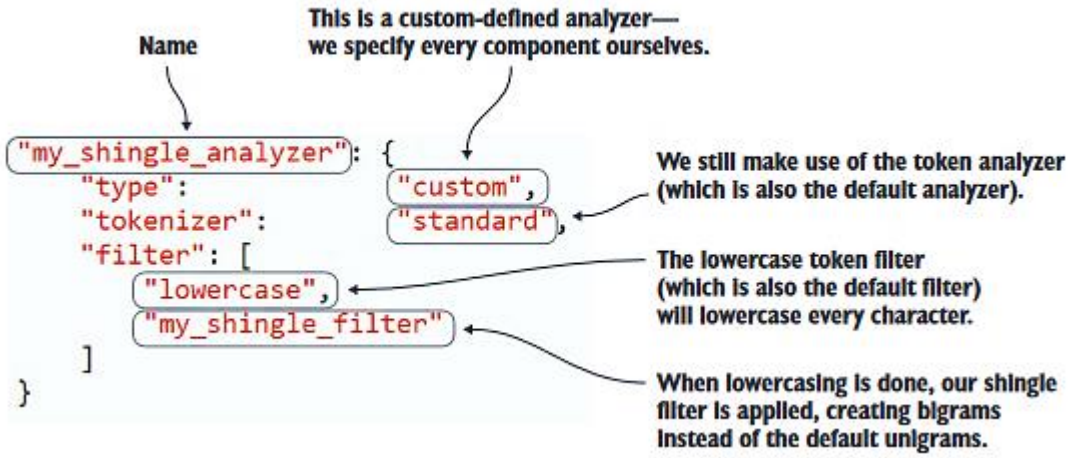
Before you can change certain settings, the index needs to be closed. After changing the settings, you can reopen the index.

Anda membuat dua elemen baru: filter token yang disebut "filter sirap saya" dan penganalisa baru yang disebut "my_shingle_analyzer." Karena n-gram sangat umum, Elasticsearch hadir dengan tipe filter token sirap bawaan. Yang perlu Anda sampaikan adalah bahwa Anda menginginkan bigram “min_shingle_size” : 2, “max_shingle_size” : 2, seperti yang ditunjukkan pada gambar 6.34. Anda bisa menggunakan trigram dan lebih tinggi, tetapi untuk tujuan demonstrasi ini sudah cukup.



Gambar 6.34 Filter token sirap untuk menghasilkan bigram

Penganalisis yang ditunjukkan pada gambar 6.35 adalah kombinasi dari semua operasi yang diperlukan untuk beralih dari teks input ke indeks. Ini menggabungkan filter sirap, tetapi lebih dari ini. Tokenizer membagi teks menjadi token atau istilah; Anda kemudian dapat menggunakan filter huruf kecil sehingga tidak ada perbedaan saat menelusuri "Diabetes" versus "diabetes". Terakhir, Anda menerapkan filter sirap Anda, membuat bigram Anda.



Gambar 6.35 Penganalisis khusus dengan tokenisasi standar dan filter token sirap untuk menghasilkan bigram

Perhatikan bahwa Anda harus menutup indeks sebelum memperbarui pengaturan. Anda kemudian dapat membuka kembali indeks dengan aman karena mengetahui bahwa pengaturan Anda telah diperbarui. Tidak semua perubahan pengaturan mengharuskan indeks ditutup, tetapi yang ini perlu. Anda dapat menemukan ikhtisar pengaturan apa yang memerlukan penutupan indeks di http://www.elastic.co/guide/en/elasticsearch/reference/current/indices-update-settings.html. Indeks sekarang siap menggunakan penganalisis baru Anda. Untuk ini, Anda akan membuat jenis dokumen baru, penyakit2, dengan pemetaan baru, seperti yang ditunjukkan pada daftar berikut.

Listing 6.6 Create more advanced Elasticsearch doctype mapping

```
docType = 'diseases2'
diseaseMapping = {
        'properties': {
            'name': {'type': 'string'},
            'title': {'type': 'string'},
            'fulltext': {
                "type": "string",
                "fields": {
                    "shingles": {
                        "type":     "string",
                        "analyzer": "my_shingle_analyzer"
                    }
```

The new disease mapping differs from the old one by the addition of the fulltext.shingles field that contains your bigrams.

```
                }
            }
        }
    }
client.indices.put_mapping(index=indexName,
doc_type=docType,body=diseaseMapping )
```

Dalam fulltext Anda sekarang memiliki parameter tambahan, fields. Di sini Anda dapat menentukan semua isotop fulltext yang berbeda. Anda hanya memiliki satu; itu menggunakan nama shingles dan akan menganalisis fulltext dengan my_shingle_analyzer baru Anda. Anda masih memiliki akses ke fulltext asli Anda, dan Anda tidak menentukan penganalisis untuk ini, jadi yang standar akan digunakan seperti sebelumnya. Anda dapat mengakses yang baru dengan memberikan nama properti diikuti dengan nama bidangnya: fulltext.shingles. Yang perlu Anda lakukan sekarang adalah melalui langkah-langkah sebelumnya dan mengindeks data menggunakan API Wikipedia, seperti yang ditunjukkan pada daftar berikut.

Listing 6.7 Reindexing Wikipedia disease explanations with new doctype mapping

```
dl = wikipedia.page("Lists_of_diseases")
diseaseListArray = []
for link in dl.links[15:42]:
    try:
        diseaseListArray.append(wikipedia.page(link))
    except Exception,e:
        print str(e)

checkList = [["0","1","2","3","4","5","6","7","8","9"],
["A"],["B"],["C"],["D"],["E"],["F"],["G"],
["H"],["I"],["J"],["K"],["L"],["M"],["N"],
["O"],["P"],["Q"],["R"],["S"],["T"],["U"],
["V"],["W"],["X"],["Y"],["Z"]]
```

The checklist is an array containing allowed "first characters." If a disease doesn't comply, you skip it.

Loop through disease lists.

```
for diseaselistNumber, diseaselist in enumerate(diseaseListArray):
    for disease in diseaselist.links: #loop through lists of links for every
     disease list
        try:
            if disease[0] in checkList[diseaselistNumber]
and disease[0:3] !="List":
                currentPage = wikipedia.page(disease)
                client.index(index=indexName,
doc_type=docType,id = disease, body={"name": disease,
"title":currentPage.title ,
"fulltext":currentPage.content})
        except Exception,e:
            print str(e)
```

First check if it's a disease, then index it.

Tidak ada yang baru di sini, hanya saja kali ini Anda akan mengindeks diseases2 doc_type, bukan diseases . Setelah ini selesai, Anda dapat melanjutkan lagi ke langkah 4, eksplorasi data, dan memeriksa hasilnya.

6.2.5 Langkah 4 ditinjau kembali: Eksplorasi data untuk pembuatan profil penyakit

Anda sekali lagi sampai pada eksplorasi data. Anda dapat menyesuaikan kueri agregasi dan menggunakan bidang baru Anda untuk memberi Anda konsep kunci bigram yang terkait dengan diabetes:

Listing 6.8 Significant terms aggregation on "diabetes" with bigrams

```
searchBody={
"fields":["name"],
"query":{
    "filtered" : {
        "filter": {
            'term': {'name':'diabetes'}
        }
    }
},
"aggregations" : {
        "DiseaseKeywords" : {
            "significant_terms" : { "field" : "fulltext", "size" : 30 }
        },
        "DiseaseBigrams": {
            "significant_terms" : { "field" : "fulltext.shingles",
"size" : 30 }
        }
    }
}
client.search(index=indexName,doc_type=docType,
body=searchBody, from_ = 0, size=3)
```

Agregat baru Anda, yang disebut DiseaseBigrams, menggunakan bidang fulltext.shingles untuk memberikan beberapa wawasan baru tentang diabetes. Istilah kunci baru ini muncul:

- **Keputihan yang berlebihan**—Pasien diabetes harus sering buang air kecil.
- **Menyebabkan poliuria**—Hal ini menunjukkan hal yang sama: diabetes menyebabkan pasien sering buang air kecil.
- **Tes deprivasi**—Ini sebenarnya adalah trigram, "tes deprivasi air", tetapi ini mengenali tes deprivasi karena Anda hanya memiliki bigram. Ini adalah tes untuk menentukan apakah seorang pasien menderita diabetes.
- **Rasa haus yang berlebihan**—Anda sudah menemukan "haus" dengan pencarian kata kunci unigram Anda, tetapi secara teknis pada saat itu bisa berarti "tidak haus".

Ada bigram menarik lainnya, unigram, dan mungkin juga trigram. Secara keseluruhan, mereka dapat digunakan untuk menganalisis teks atau kumpulan teks sebelum membacanya. Perhatikan bahwa Anda mencapai hasil yang diinginkan tanpa sampai ke tahap pemodelan. Kadang-kadang setidaknya ada informasi berharga dalam jumlah yang sama untuk ditemukan dalam eksplorasi data seperti dalam pemodelan data. Sekarang setelah Anda sepenuhnya mencapai tujuan sekunder Anda, Anda dapat melanjutkan ke langkah 6 proses ilmu data: presentasi dan otomatisasi.

6.2.6 Langkah 6: Presentasi dan otomatisasi

Tujuan utama Anda, diagnosis penyakit, berubah menjadi alat diagnostik swalayan dengan mengizinkan dokter untuk menanyakannya melalui, misalnya, aplikasi web. Anda tidak akan membangun situs web dalam kasus ini, tetapi jika Anda berencana melakukannya, harap baca sidebar "Elasticsearch aplikasi web."

Elasticsearch untuk aplikasi web

Seperti database lainnya, merupakan praktik yang buruk untuk memaparkan REST API Elasticsearch Anda secara langsung ke bagian depan aplikasi web. Jika situs web dapat langsung membuat permintaan POST ke database Anda, siapa pun dapat dengan mudah menghapus data Anda: selalu ada kebutuhan akan lapisan perantara. Lapisan tengah ini bisa jadi Python jika itu cocok untuk Anda. Dua solusi Python populer adalah Django atau kerangka kerja Django REST dalam kombinasi dengan ujung depan independen. Django umumnya digunakan untuk membangun aplikasi bolak-balik (aplikasi web di mana server membangun ujung depan secara dinamis, mengingat data dari database dan sistem templating). Kerangka kerja Django REST adalah plugin untuk Django, mengubah Django menjadi layanan REST, membuatnya menjadi bagian dari aplikasi satu halaman. Aplikasi satu halaman adalah aplikasi web yang menggunakan satu halaman web sebagai jangkar tetapi mampu mengubah konten secara dinamis dengan mengambil file statis dari server HTTP dan data dari RESTful API. Kedua pendekatan (pulang pergi dan satu halaman) baik-baik saja, selama server Elasticsearch sendiri tidak terbuka untuk umum, karena tidak memiliki langkah keamanan bawaan. Keamanan dapat ditambahkan ke Elasticsearch secara langsung menggunakan "Shield", layanan berbayar Elasticsearch.

Tujuan kedua, pembuatan profil penyakit, juga dapat dibawa ke tingkat antarmuka pengguna; memungkinkan untuk membiarkan hasil pencarian menghasilkan cloud kata yang meringkas hasil pencarian secara visual. Kami tidak akan membahasnya sejauh itu dalam buku ini, tetapi jika Anda tertarik untuk menyiapkan sesuatu seperti ini dengan Python, gunakan perpustakaan word_cloud (pemasangan pip word_cloud). Atau jika Anda lebih suka JavaScript, D3.js adalah cara yang baik. Anda dapat menemukan contoh implementasi di http://www.jasondavies.com/wordcloud/#%2F%2Fwww.jasondavies.com%2Fwordcloud%2Fabout%2F.

Gambar 6.36 Awan kata unigram pada kata kunci diabetes tidak berbobot dari Elasticsearch

Menambahkan kata kunci Anda di situs web berbasis D3.js ini akan menghasilkan cloud kata unigram seperti yang ditunjukkan pada gambar 6.36 yang dapat dimasukkan ke dalam presentasi hasil proyek Anda. Istilah-istilah tersebut tidak diboboti oleh skor mereka dalam

kasus ini, tetapi sudah memberikan representasi yang bagus dari temuan tersebut. Banyak peningkatan yang dimungkinkan untuk aplikasi Anda, terutama di bidang persiapan data. Tapi menyelami semua kemungkinan di sini akan membawa kita terlalu jauh; jadi kita telah sampai pada akhir bab ini. Di bagian berikutnya kita akan melihat data streaming.

6.3 RINGKASAN

Dalam bab ini, Anda mempelajari hal berikut:

- NoSQL adalah singkatan dari "Not Only Structured Query Language" dan telah muncul dari kebutuhan untuk menangani jumlah dan jenis data yang meningkat secara eksponensial, serta meningkatnya kebutuhan akan skema yang lebih beragam dan fleksibel seperti jaringan dan struktur hierarkis.
- Penanganan semua data ini memerlukan partisi basis data karena tidak ada satu mesin pun yang mampu melakukan semua pekerjaan. Saat mempartisi, Teorema CAP berlaku: Anda dapat memiliki ketersediaan atau konsistensi tetapi tidak pernah keduanya sekaligus.
- Database relasional dan database grafik berpegang pada prinsip ACID: atomitas, konsistensi, isolasi, dan daya tahan. Basis data NoSQL umumnya mengikuti prinsip BASE: ketersediaan dasar, status lunak, dan konsistensi akhir.
- Empat jenis database NoSQL terbesar

  – Penyimpanan nilai kunci—Pada dasarnya sekumpulan pasangan nilai kunci yang disimpan dalam database. Basis data ini bisa sangat besar dan sangat serbaguna tetapi kompleksitas datanya rendah. Contoh yang terkenal adalah Redis.

  – Basis data kolom lebar—Basis data ini sedikit lebih kompleks daripada penyimpanan nilai kunci karena mereka menggunakan kolom tetapi dengan cara yang lebih efisien daripada RDBMS biasa. Kolom pada dasarnya dipisahkan, memungkinkan Anda mengambil data dalam satu kolom dengan cepat. Database terkenal adalah Cassandra.

  – Penyimpanan dokumen—Basis data ini sedikit lebih rumit dan menyimpan data sebagai dokumen. Saat ini yang paling populer adalah MongoDB, tetapi dalam studi kasus kami, kami menggunakan Elasticsearch, yang merupakan penyimpanan dokumen dan mesin pencari.

  – Database grafik—Database ini dapat menyimpan struktur data yang paling kompleks, karena memperlakukan entitas dan relasi antar entitas dengan perhatian yang sama. Kompleksitas ini menimbulkan biaya dalam kecepatan pencarian. Yang populer adalah Neo4j, tetapi GraphX (basis data grafik yang terkait dengan Apache Spark) adalah pemenangnya.
- Elasticsearch adalah penyimpanan dokumen dan mesin pencari teks lengkap yang dibangun di atas Apache Lucene, mesin pencari sumber terbuka. Ini dapat digunakan untuk membuat token, melakukan kueri agregasi, melakukan kueri dimensional (segi), kueri pencarian profil, dan banyak lagi.

BAB 7
DATABASE GRAFIK

Dalam bab ini mahasiswa diharapkan mampu:

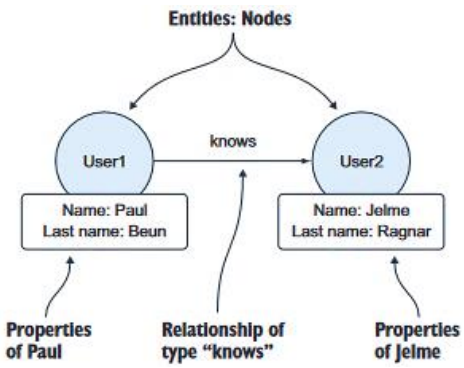
- Memperkenalkan data terhubung dan hubungannya dengan grafik dan database grafik
- Mempelajari bagaimana database grafik berbeda dari database relasional
- Menemukan database grafik Neo4j
- Menerapkan proses ilmu data ke proyek mesin pemberi rekomendasi dengan basis data grafik Neo4j

Di satu sisi kita menghasilkan data dalam skala besar, mendorong Google, Amazon, dan Facebook untuk menemukan cara cerdas untuk menangani hal ini, di sisi lain kita dihadapkan pada data yang semakin saling terhubung dari sebelumnya. Grafik dan jaringan meresap dalam kehidupan kita. Dengan menghadirkan beberapa contoh yang memotivasi, kami berharap dapat mengajari pembaca cara mengenali masalah graf ketika muncul dengan sendirinya. Dalam bab ini kita akan melihat bagaimana memanfaatkan koneksi tersebut untuk semua yang layak menggunakan database grafik, dan mendemonstrasikan bagaimana menggunakan Neo4j, database grafik yang populer.

7.1 DATABASE DATA DAN GRAFIK YANG TERHUBUNG

Mari kita mulai dengan membiasakan diri dengan konsep data terhubung dan representasinya sebagai data grafik.

- **Data terhubung—**Seperti namanya, data terhubung dicirikan oleh fakta bahwa data yang ada memiliki hubungan yang membuatnya terhubung.
- **Grafik—**Sering disebut dalam kalimat yang sama dengan data terhubung. Grafik sangat cocok untuk mewakili konektivitas data dengan cara yang bermakna.
- **Database grafik—**Diperkenalkan di bab 6. Alasan topik ini perlu mendapat perhatian khusus adalah karena, selain fakta bahwa ukuran data bertambah, juga menjadi lebih saling berhubungan. Tidak banyak upaya yang diperlukan untuk menghasilkan contoh data terhubung yang terkenal.

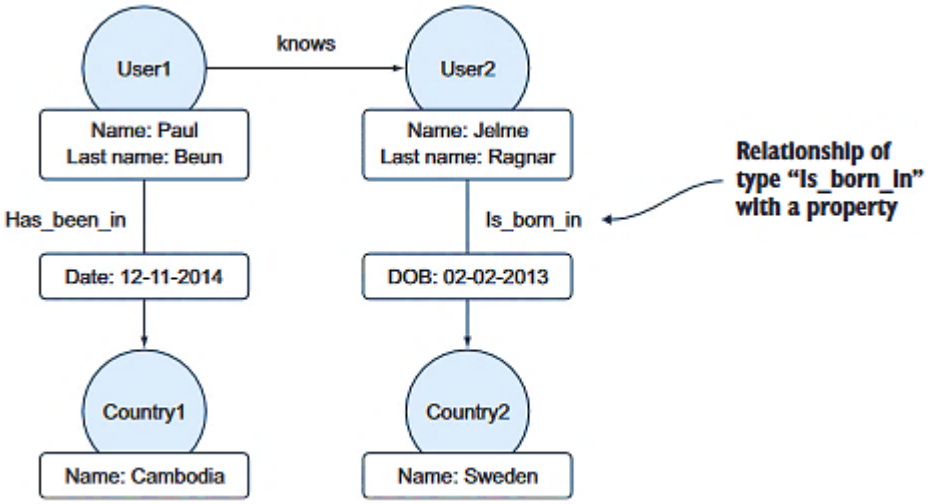


Gambar 7.1 Contoh data terhubung sederhana: dua entitas atau node (User1, User2), masing-masing dengan properti (nama depan, nama belakang).

Contoh menonjol dari data yang mengambil bentuk jaringan adalah data media sosial. Media sosial memungkinkan kita untuk berbagi dan bertukar data dalam jaringan, sehingga menghasilkan sejumlah besar data yang terhubung. Hal ini dapat kita ilustrasikan dengan contoh sederhana. Anggaplah kita memiliki dua orang di data kita, Pengguna1 dan Pengguna2. Selanjutnya kita mengetahui nama depan dan nama belakang User1 (nama depan: Paul dan nama belakang: Beun) dan User2 (nama depan: Jelme dan nama belakang: Ragnar). Cara alami untuk merepresentasikannya adalah dengan menggambarnya di papan tulis, seperti yang ditunjukkan pada gambar 7.1.

Terminologi gambar 7.1 dijelaskan di bawah ini:

- **Entitas—**Kami memiliki dua entitas yang mewakili orang (Pengguna1 dan Pengguna2). Entitas ini memiliki properti "nama" dan "nama belakang".
- **Properti—**Properti ditentukan oleh key-value pair. Dari grafik ini kita juga dapat menyimpulkan bahwa User1 dengan properti “nama” Paul mengetahui User2 dengan properti “nama” Jelme.
- **Hubungan—**Ini adalah hubungan antara Paul dan Jelme. Perhatikan bahwa hubungan itu memiliki arah: Paullah yang "mengenal" Jelme dan bukan sebaliknya. User1 dan User2 keduanya mewakili orang dan karenanya dapat dikelompokkan.
- **Label—**Dalam database grafik, seseorang dapat mengelompokkan node dengan menggunakan label. Pengguna1 dan Pengguna2 dalam hal ini dapat diberi label sebagai "Pengguna".



Gambar 7.2 Contoh data terhubung yang lebih rumit di mana dua entitas lagi telah disertakan (Negara1 dan Negara2) dan dua hubungan baru ("Has_been_in" dan "Is_born_in")

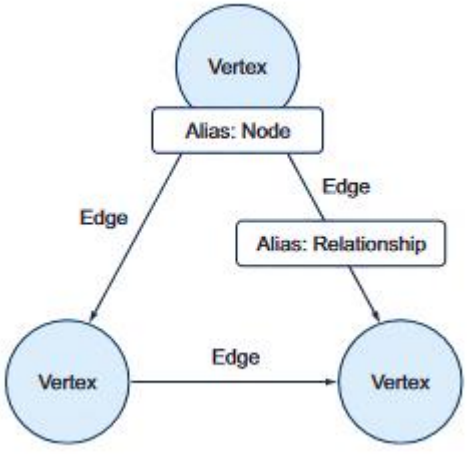
Data yang terhubung seringkali berisi lebih banyak entitas dan koneksi. Pada gambar 7.2 kita dapat melihat grafik yang lebih luas. Dua entitas lagi disertakan: Negara1 dengan nama Kamboja dan Negara2 dengan nama Swedia. Ada dua hubungan lagi: "Has_been_in" dan "Is_born_in". Di grafik sebelumnya, hanya entitas yang menyertakan properti, kini relasinya juga berisi properti. Grafik semacam itu dikenal sebagai grafik properti. Relasi yang

menghubungkan node User1 dan Country1 bertipe “Has_been_in” dan memiliki properti “Date” yang mewakili nilai data. Demikian pula, User2 terhubung ke Negara2 tetapi melalui tipe hubungan yang berbeda, yaitu tipe “Is_born_in”. Perhatikan bahwa jenis hubungan memberi kita konteks hubungan antar node. Node dapat memiliki banyak hubungan.

Representasi data kami semacam ini memberi kami cara intuitif untuk menyimpan data yang terhubung. Untuk menjelajahi data kami, kami perlu melintasi grafik mengikuti jalur yang telah ditentukan sebelumnya untuk menemukan pola yang kami cari. Bagaimana jika seseorang ingin tahu di mana Paul berada? Diterjemahkan ke dalam terminologi basis data grafik, kami ingin menemukan pola "Paul telah masuk". Untuk menjawab ini, kami akan mulai dari simpul dengan nama "Paul" dan melintasi ke Kamboja melalui hubungan "Has_been_in". Oleh karena itu traversal grafik, yang sesuai dengan kueri basis data, adalah sebagai berikut:

1. Node awal—Dalam hal ini node dengan nama properti "Paul"
2. Jalur traversal—Dalam hal ini jalur dimulai dari simpul Paul dan menuju ke Kamboja
3. Node akhir—Node negara dengan properti nama “Kamboja”

Untuk lebih memahami bagaimana database grafik berurusan dengan data yang terhubung, sebaiknya memperluas sedikit lebih banyak tentang grafik secara umum. Grafik dipelajari secara ekstensif dalam domain ilmu komputer dan matematika dalam bidang yang disebut teori grafik. Teori graf adalah studi tentang graf, di mana graf mewakili struktur matematis yang digunakan untuk memodelkan hubungan berpasangan antar objek, seperti yang ditunjukkan pada gambar 7.3. Apa yang membuat mereka begitu menarik adalah bahwa mereka memiliki struktur yang cocok untuk memvisualisasikan data yang terhubung. Grafik ditentukan oleh simpul (juga dikenal sebagai simpul di dunia basis data grafik) dan tepi (juga dikenal sebagai hubungan). Konsep-konsep ini membentuk dasar-dasar dasar yang menjadi dasar struktur data grafik.



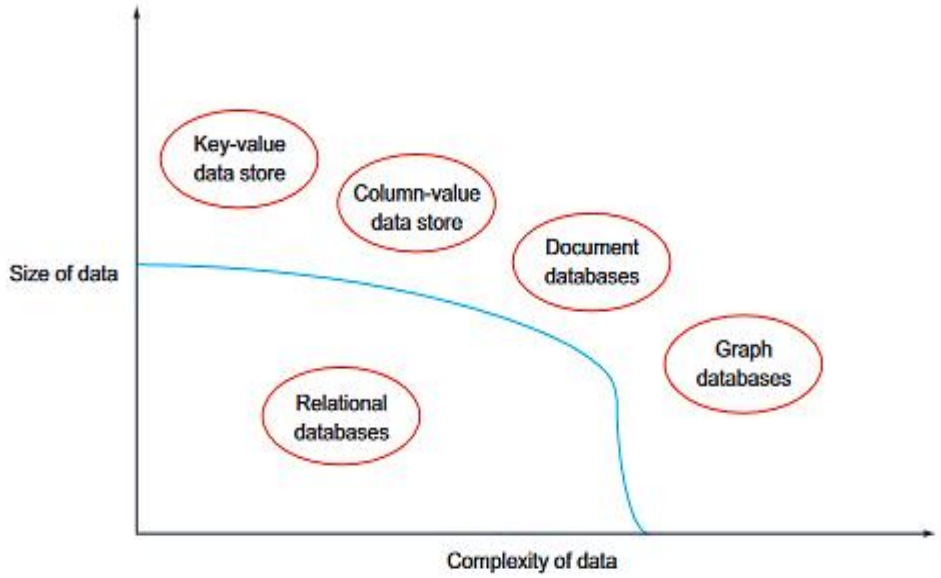
Gambar 7.3 Pada intinya sebuah graf terdiri dari node (juga dikenal sebagai simpul) dan sisi (yang menghubungkan simpul), seperti yang diketahui dari definisi matematis dari sebuah graf. Kumpulan objek ini mewakili grafik.

Dibandingkan dengan struktur data lainnya, fitur khusus dari data yang terhubung adalah sifatnya yang non-linier: setiap entitas dapat dihubungkan ke entitas lain melalui berbagai jenis relasi dan entitas perantara serta jalur. Dalam grafik, Anda dapat membuat

pembagian antara grafik berarah dan tidak berarah. Sisi-sisi graf berarah memiliki—bagaimana bisa sebaliknya—sebuah arah. Meskipun orang dapat berargumen bahwa setiap masalah entah bagaimana dapat direpresentasikan sebagai masalah grafik, penting untuk memahami kapan ideal untuk melakukannya dan kapan tidak.

7.1.1 Mengapa dan kapan saya harus menggunakan database grafik?

Pencarian untuk menentukan basis data grafik mana yang harus digunakan bisa menjadi proses yang terlibat untuk dilakukan. Salah satu aspek penting dalam proses pengambilan keputusan ini adalah menemukan representasi yang tepat untuk data Anda. Sejak awal 1970-an jenis database yang paling umum yang harus diandalkan adalah database relasional. Kemudian, yang lain muncul, seperti database hierarkis (misalnya, IMS), dan kerabat terdekat database grafik: database jaringan (misalnya, IDMS). Namun selama beberapa dekade terakhir, bentang alam menjadi jauh lebih beragam, memberikan lebih banyak pilihan kepada pengguna akhir tergantung pada kebutuhan khusus mereka. Mempertimbangkan perkembangan terbaru dari data yang tersedia, dua karakteristik sangat cocok untuk disorot di sini. Yang pertama adalah ukuran data dan yang lainnya kompleksitas data, seperti yang ditunjukkan pada gambar 7.4.



Gambar 7.4 Gambar ini mengilustrasikan posisi basis data grafik pada ruang dua dimensi di mana satu dimensi mewakili ukuran data yang sedang ditangani, dan dimensi lainnya mewakili kompleksitas dalam hal bagaimana menghubungkan data tersebut.

Ketika database relasional tidak dapat lagi mengatasi kompleksitas kumpulan data karena keterhubungannya, tetapi bukan ukurannya, database grafik mungkin menjadi pilihan terbaik Anda. Seperti yang ditunjukkan oleh gambar 7.4, kita harus bergantung pada database grafik ketika datanya kompleks tetapi masih kecil. Meskipun "kecil" adalah hal yang relatif di sini, kita masih berbicara tentang ratusan juta node. Menangani kerumitan adalah aset utama database grafik dan "alasan" utama Anda menggunakannya. Untuk menjelaskan kompleksitas

apa yang dimaksud di sini, pertama-tama pikirkan tentang cara kerja basis data relasional tradisional.

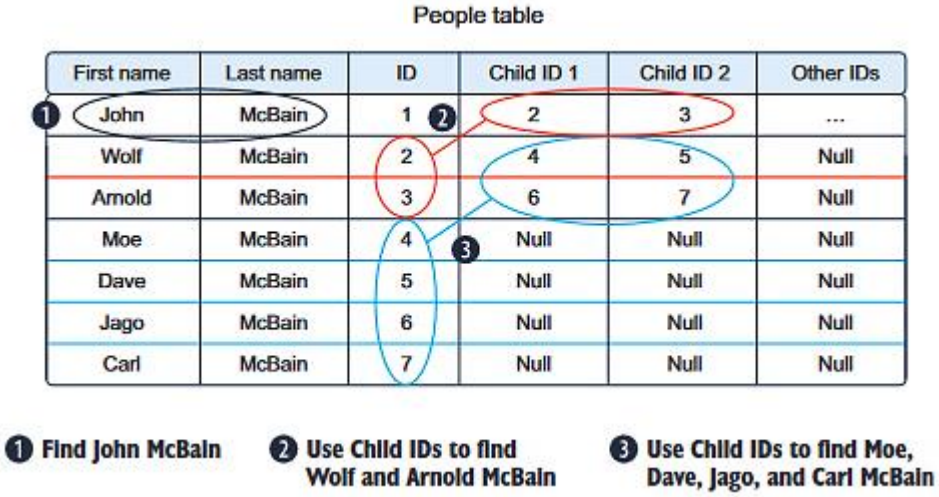
Berlawanan dengan apa yang ditunjukkan oleh nama database relasional, tidak banyak yang relasional tentangnya kecuali bahwa kunci asing dan kunci utama adalah yang menghubungkan tabel. Sebaliknya, hubungan dalam database grafik adalah warga negara kelas satu. Melalui aspek ini, mereka cocok untuk memodelkan dan menanyakan data yang terhubung. Database relasional lebih suka berusaha untuk meminimalkan redundansi data. Proses ini dikenal sebagai normalisasi basis data, di mana tabel didekomposisi menjadi tabel yang lebih kecil (lebih sedikit redundan) sambil mempertahankan semua informasi tetap utuh. Dalam database yang dinormalisasi seseorang perlu melakukan perubahan atribut hanya dalam satu tabel. Tujuan dari proses ini adalah untuk mengisolasi perubahan data dalam satu tabel. Sistem manajemen basis data relasional (RDBMS) adalah pilihan yang baik sebagai basis data untuk data yang cocok dengan format tabular. Hubungan dalam data dapat diekspresikan dengan menggabungkan tabel. Kecocokan mereka mulai diturunkan ketika gabungan menjadi lebih rumit, terutama ketika mereka menjadi gabungan banyak-ke-banyak. Waktu kueri juga akan meningkat ketika ukuran data Anda mulai meningkat, dan pemeliharaan database akan menjadi tantangan yang lebih besar. Faktor-faktor ini akan menghambat kinerja database Anda. Grafik database, di sisi lain, secara inheren menyimpan data sebagai node dan hubungan. Meskipun basis data grafik diklasifikasikan sebagai jenis basis data NoSQL, ada kecenderungan untuk menyajikannya sebagai kategori tersendiri. Seseorang mencari pembenaran untuk ini dengan mencatat bahwa jenis database NoSQL lainnya berorientasi pada agregasi, sedangkan database grafik tidak.

Database relasional mungkin, misalnya, memiliki tabel yang mewakili "orang" dan propertinya. Setiap orang terkait dengan orang lain melalui kekerabatan (dan persahabatan, dan seterusnya); setiap baris mungkin mewakili satu orang, tetapi menghubungkannya ke baris lain di tabel orang akan menjadi pekerjaan yang sangat sulit. Apakah Anda menambahkan variabel yang menyimpan pengidentifikasi unik anak pertama dan tambahan untuk menyimpan ID anak kedua? Di mana Anda berhenti? Anak kesepuluh?

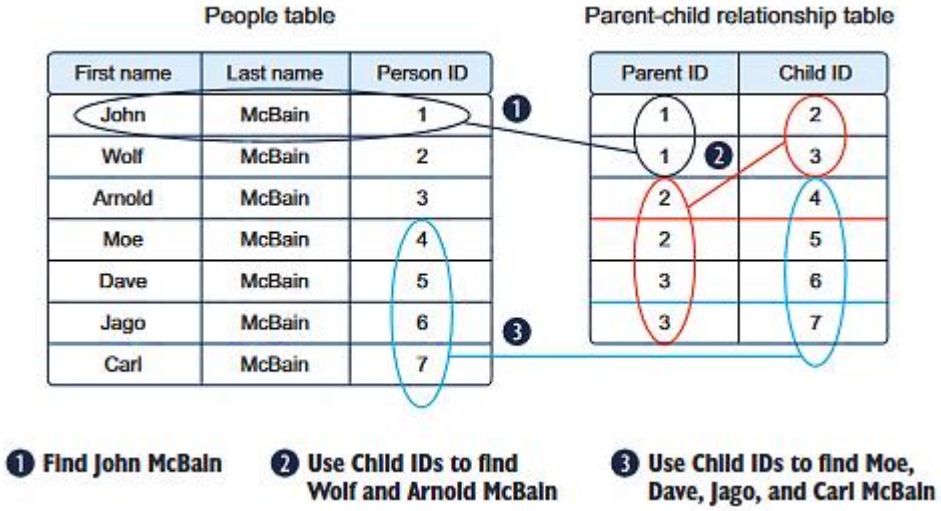
Alternatifnya adalah menggunakan tabel perantara untuk hubungan anak-orang tua, tetapi Anda memerlukan tabel terpisah untuk jenis hubungan lain seperti persahabatan. Dalam kasus terakhir ini Anda tidak mendapatkan proliferasi kolom tetapi proliferasi tabel: satu tabel hubungan untuk setiap jenis hubungan. Bahkan jika Anda entah bagaimana berhasil memodelkan data sedemikian rupa sehingga semua hubungan keluarga ada, Anda memerlukan pertanyaan yang sulit untuk mendapatkan jawaban atas pertanyaan sederhana seperti "Saya ingin cucu John McBain." Pertama, Anda perlu menemukan anak-anak John McBain. Setelah Anda menemukan anak-anaknya, Anda perlu menemukan anak mereka. Pada saat Anda menemukan semua cucu, Anda telah mencapai meja "orang" tiga kali:

1. Temukan McBain dan jemput anak-anaknya.
2. Cari anak-anak dengan ID yang Anda dapatkan dan dapatkan ID anak-anak mereka.
3. Temukan cucu McBain.

Gambar 7.5 menunjukkan pencarian rekursif dalam database relasi yang diperlukan untuk mendapatkan dari John McBain ke cucunya jika semuanya ada dalam satu tabel. Gambar 7.6 adalah cara lain untuk memodelkan data: hubungan orangtua-anak adalah tabel terpisah.Pencarian rekursif seperti ini tidak efisien, untuk sedikitnya.



Gambar 7.5 Pencarian rekursif versi 1: semua data dalam satu tabel



Gambar 7.6 Pencarian rekursif versi 2: menggunakan tabel hubungan induk-anak

Database grafik bersinar ketika jenis kerumitan ini muncul. Mari kita lihat yang paling populer di antara mereka.

7.2 MEMPERKENALKAN NEO4J: DATABASE GRAFIK

Data yang terhubung umumnya disimpan dalam database grafik. Basis data ini secara khusus dirancang untuk mengatasi struktur data yang terhubung. Lanskap database grafik yang tersedia agak beragam akhir-akhir ini. Tiga yang paling terkenal dalam urutan penurunan popularitas adalah Neo4j, OrientDb, dan Titan.

Neo4j adalah database grafik yang menyimpan data dalam grafik yang berisi node dan hubungan (keduanya diperbolehkan mengandung properti). Jenis basis data grafik ini dikenal sebagai grafik properti dan cocok untuk menyimpan data yang terhubung. Ini memiliki skema fleksibel yang akan memberi kita kebebasan untuk mengubah struktur data kita jika diperlukan, memberi kita kemampuan untuk menambahkan data baru dan hubungan baru jika diperlukan. Ini adalah proyek sumber terbuka, teknologi matang, mudah dipasang, mudah digunakan, dan didokumentasikan dengan baik. Neo4j juga memiliki antarmuka berbasis browser yang memfasilitasi pembuatan grafik untuk tujuan visualisasi. Untuk mengikuti, ini akan menjadi saat yang tepat untuk menginstal Neo4j.

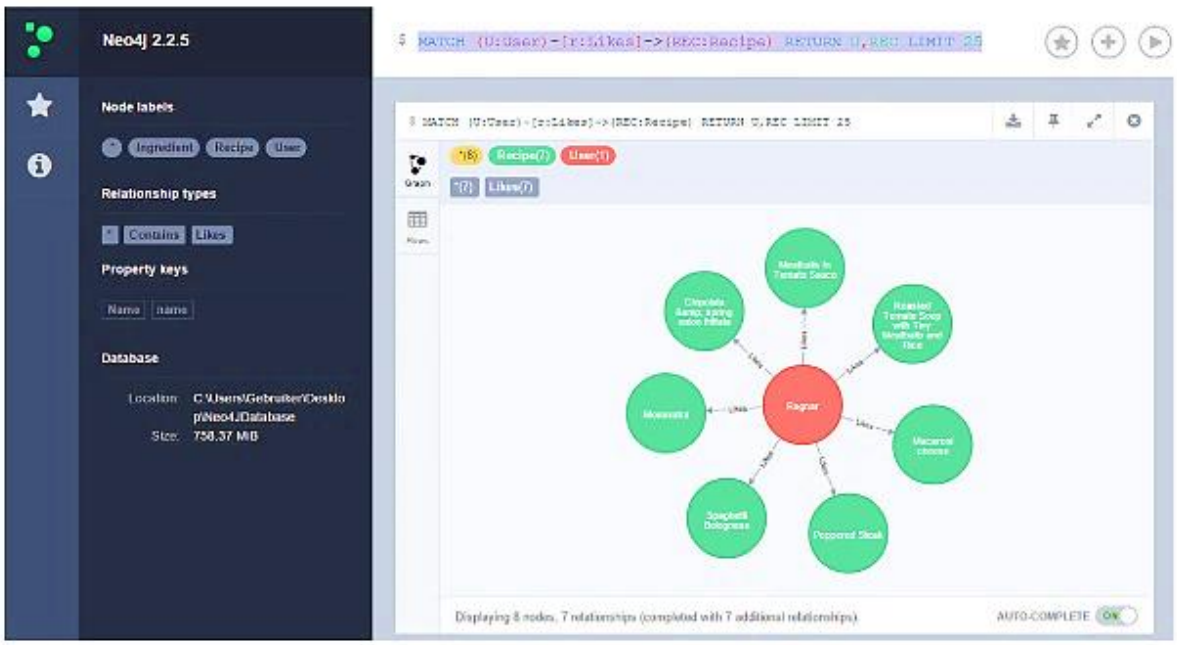
Sekarang mari kita perkenalkan empat struktur dasar di Neo4j:

- **Node—**Mewakili entitas seperti dokumen, pengguna, resep, dan sebagainya. Properti tertentu dapat ditugaskan ke node.
- **Hubungan—**Ada di antara node yang berbeda. Mereka dapat diakses baik berdiri sendiri atau melalui node yang mereka lampirkan. Relasi juga dapat berisi properti, oleh karena itu disebut model grafik properti. Setiap relasi memiliki nama dan arah, yang bersama-sama menyediakan konteks semantik untuk node yang dihubungkan oleh relasi tersebut.
- **Properti—**Node dan relasi dapat memiliki properti. Properti ditentukan oleh pasangan kunci-nilai.
- **Label—**Dapat digunakan untuk mengelompokkan node serupa untuk memfasilitasi traversal yang lebih cepat melalui grafik.

Sebelum melakukan analisis, kebiasaan yang baik adalah merancang database Anda dengan hati-hati agar sesuai dengan kueri yang ingin Anda jalankan saat melakukan analisis. Database grafik memiliki karakteristik yang menyenangkan karena ramah papan tulis. Jika seseorang mencoba menggambar pengaturan masalah di papan tulis, gambar ini akan sangat mirip dengan desain database untuk masalah yang ditentukan. Oleh karena itu, gambar papan tulis seperti itu akan menjadi titik awal yang baik untuk mendesain database kita.

Sekarang bagaimana cara mengambil datanya? Untuk menjelajahi data kami, kami perlu melintasi grafik mengikuti jalur yang telah ditentukan untuk menemukan pola yang kami telusuri. Peramban Neo4j adalah lingkungan yang ideal untuk membuat dan bermain-main dengan data Anda yang terhubung hingga Anda mendapatkan jenis representasi yang tepat untuk kueri yang optimal, seperti yang ditunjukkan pada gambar 7.7. Skema fleksibel dari basis data grafik sangat cocok untuk kita di sini. Di browser ini Anda dapat mengambil data Anda dalam baris atau sebagai grafik. Neo4j memiliki bahasa kueri sendiri untuk memudahkan pembuatan dan kemampuan kueri grafik.

Cypher adalah bahasa yang sangat ekspresif yang cukup mirip dengan SQL untuk meningkatkan proses pembelajaran bahasa. Di bagian berikut, kami akan membuat data kami sendiri menggunakan Cypher dan memasukkannya ke Neo4j. Kemudian kita bisa bermain-main dengan data.

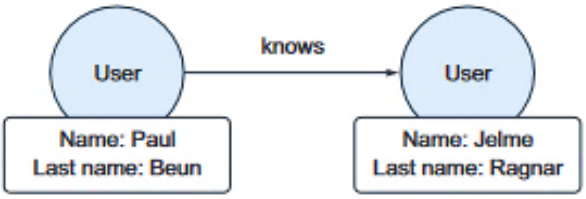


Gambar 7.7 Antarmuka Neo4j 2.2.5 dengan kueri yang diselesaikan dari studi kasus bab

7.2.1 Cypher: bahasa kueri grafik

Mari perkenalkan Cypher dan sintaks dasarnya untuk operasi grafik. Ide dari bagian ini adalah untuk menyajikan cukup tentang Cypher agar kita mulai menggunakan browser Neo4j. Di akhir bagian ini Anda harus dapat membuat data terhubung Anda sendiri menggunakan Cypher di browser Neo4j dan menjalankan kueri dasar untuk mengambil hasil kueri. Untuk pengenalan yang lebih luas tentang Cypher, Anda dapat mengunjungi http://neo4j.com/docs/stable/cypher-query-lang.html. Kita akan mulai dengan menggambar grafik sosial sederhana yang disertai dengan kueri dasar untuk mengambil pola yang telah ditentukan sebelumnya sebagai contoh. Pada langkah selanjutnya kita akan menggambar grafik yang lebih kompleks yang memungkinkan kita menggunakan kueri yang lebih rumit di Cypher. Ini akan membantu kita untuk mengenal Cypher dan mengarahkan kita ke jalan untuk mewujudkan kasus penggunaan kita. Selain itu, kami akan menunjukkan cara membuat data terkoneksi simulasi kami sendiri menggunakan Cypher.

Gambar 7.8 menunjukkan grafik sosial sederhana dari dua node, yang dihubungkan oleh hubungan tipe “knows”. Node memiliki properti "nama" dan "nama belakang". Sekarang, jika kita ingin mengetahui pola berikut, “Siapa yang Paulus kenal?” kami akan menanyakan ini menggunakan Cypher. Untuk menemukan pola di Cypher, kita akan mulai dengan klausa Pertandingan. Dalam query ini kita akan mulai mencari di node User dengan nama properti “Paul”. Perhatikan bagaimana simpul diapit dalam tanda kurung, seperti yang ditunjukkan dalam cuplikan kode di bawah ini, dan hubungannya diapit oleh tanda kurung siku. Relasi diberi nama dengan awalan titik dua (:), dan arahnya dijelaskan menggunakan panah. Placeholder p2 akan berisi semua simpul Pengguna yang memiliki hubungan tipe "knows" sebagai hubungan masuk. Dengan klausa kembali kita dapat mengambil hasil kueri.



Gambar 7.8 Contoh grafik sosial sederhana dengan dua pengguna dan satu hubungan

```
Match(p1:User { name: 'Paul' } )-[:knows]->(p2:User)
Return p2.name
```

Perhatikan hubungan yang erat tentang bagaimana kita merumuskan pertanyaan kita secara lisan dan cara database grafik menerjemahkannya menjadi traversal. Di Neo4j, ekspresi yang mengesankan ini dimungkinkan oleh bahasa kueri grafiknya, Cypher. Untuk membuat contoh lebih menarik, mari kita asumsikan bahwa data kita diwakili oleh grafik pada gambar 7.9.

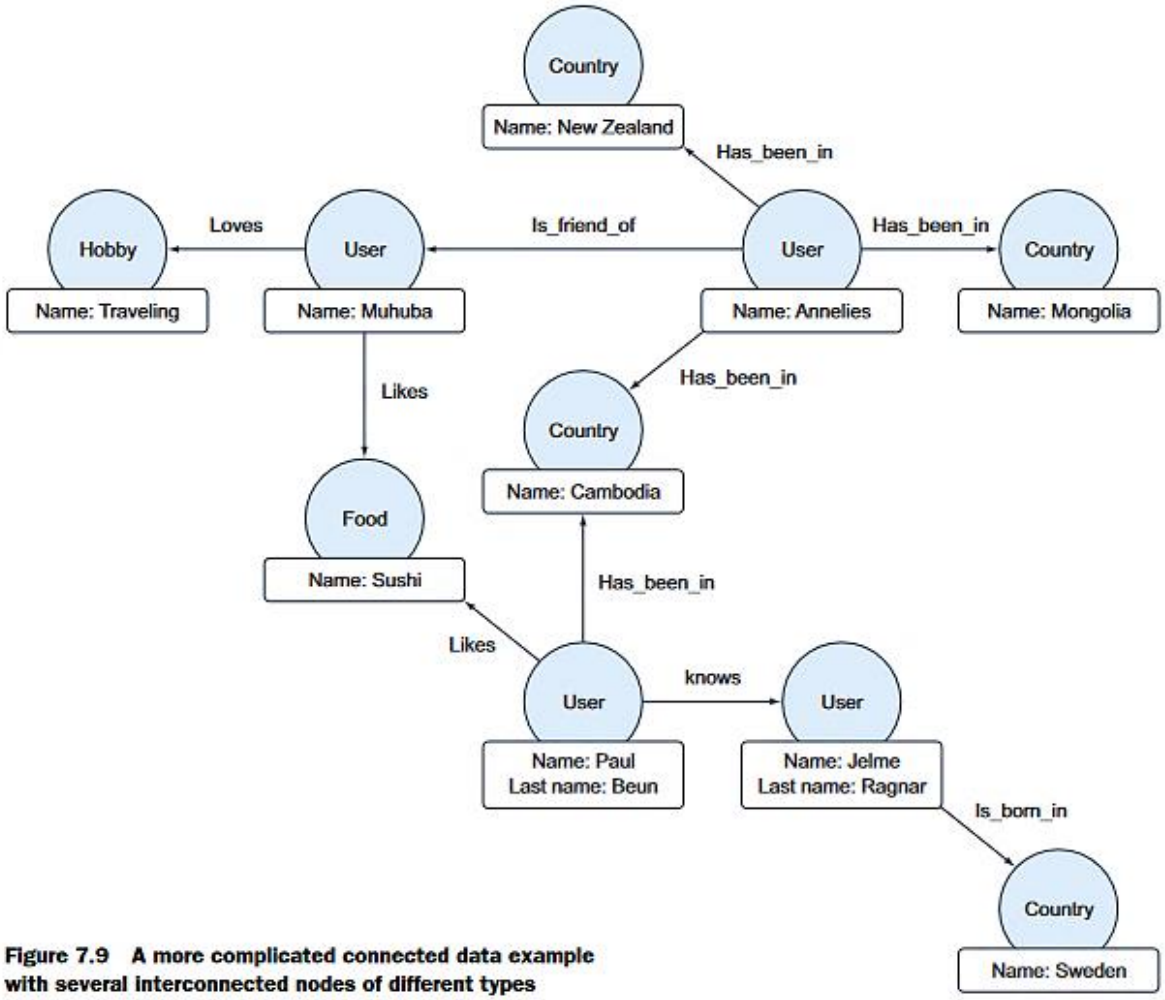


Figure 7.9 A more complicated connected data example with several interconnected nodes of different types

Gambar 7.9 Contoh data terkoneksi yang lebih rumit dengan beberapa node yang saling terhubung dari tipe yang berbeda

Kita dapat memasukkan data yang terhubung pada gambar 7.9 ke dalam Neo4j dengan menggunakan Cypher. Kita dapat menulis perintah Cypher secara langsung di antarmuka

berbasis browser Neo4j, atau secara alternatif melalui driver Python (lihat http://neo4j.com/developer/python/ untuk gambaran umum). Ini adalah cara yang baik untuk merasakan langsung dengan data yang terhubung dan database grafik.

Untuk menulis pernyataan create yang sesuai di Cypher, pertama-tama kita harus memiliki pemahaman yang baik tentang data mana yang ingin kita simpan sebagai node dan mana yang sebagai relasi, seperti apa propertinya, dan apakah label akan berguna. Keputusan pertama adalah memutuskan data mana yang harus dianggap sebagai node dan mana yang sebagai hubungan untuk menyediakan konteks semantik untuk node ini. Pada gambar 7.9 kami telah memilih untuk mewakili pengguna dan negara tempat mereka berada sebagai node. Data yang menyediakan informasi tentang node tertentu, misalnya nama yang diasosiasikan dengan node, dapat direpresentasikan sebagai properti. Semua data yang memberikan konteks tentang dua atau lebih node akan dianggap sebagai sebuah hubungan. Node yang berbagi fitur umum, misalnya Kamboja dan Swedia keduanya merupakan negara, juga akan dikelompokkan melalui label. Pada gambar 7.9 hal ini sudah dilakukan.

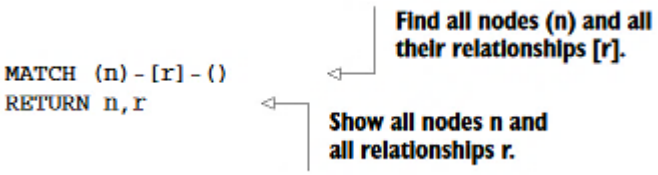
Dalam daftar berikut kami mendemonstrasikan bagaimana berbagai objek dapat dikodekan di Cypher melalui satu pernyataan pembuatan besar. Ketahuilah bahwa Cypher peka huruf besar-kecil.

Listing 7.1 Cypher data creation statement

```
CREATE (user1:User {name :'Annelies'}),
 (user2:User {name :'Paul' , LastName: 'Beun'}),
 (user3:User {name :'Muhuba'}),
 (user4:User {name : 'Jelme' , LastName: 'Ragnar'}),
 (country1:Country { name:'Mongolia'}),
 (country2:Country { name:'Cambodia'}),
 (country3:Country { name:'New Zealand'}),
 (country4:Country { name:'Sweden'}),
 (food1:Food { name:'Sushi' }),
 (hobby1:Hobby { name:'Travelling'}),
 (user1)-[:Has_been_in]->(country1),
 (user1)-[: Has_been_in]->(country2),
 (user1)-[: Has_been_in]->(country3),
 (user2)-[: Has_been_in]->(country2),
 (user1)-[: Is_mother_of]->(user4),
 (user2)-[: knows]->(user4),
 (user1)-[: Is_friend_of]->(user3),
 (user2)-[: Likes]->( food1),
 (user3)-[: Likes]->( food1),
 (user4)-[: Is_born_in]->(country4)
```

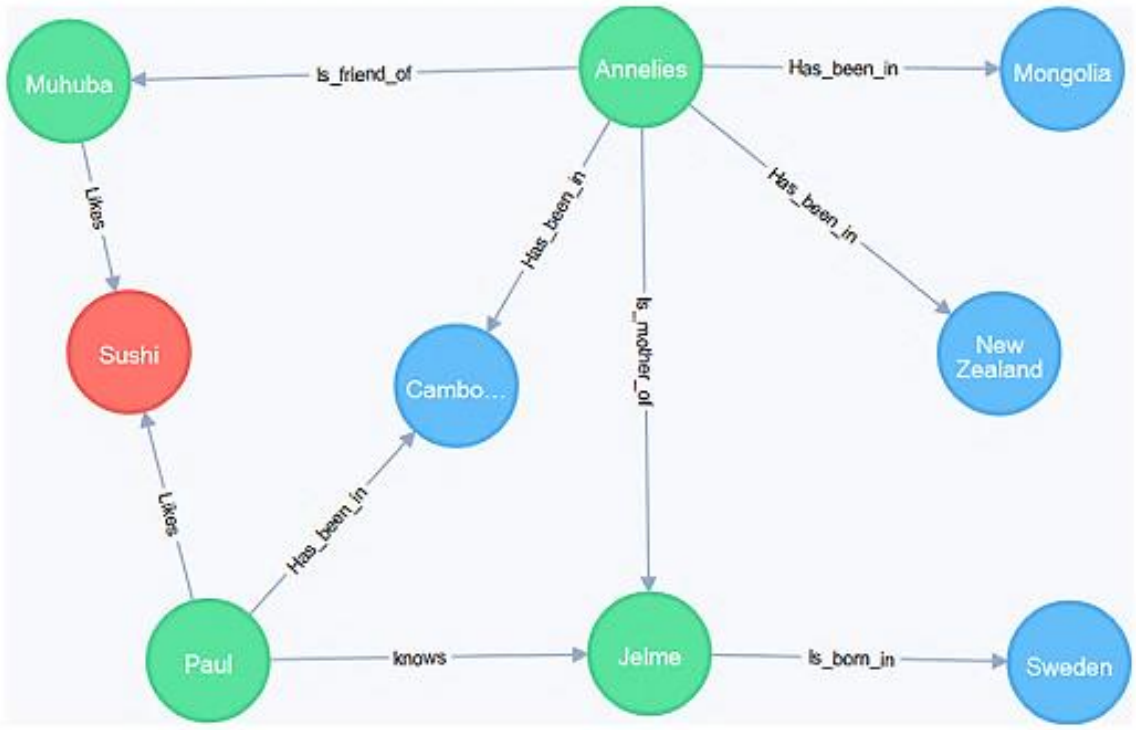
Menjalankan pernyataan create ini sekaligus memiliki keuntungan bahwa keberhasilan eksekusi ini akan memastikan kita bahwa database grafik telah berhasil dibuat. Jika ada kesalahan, grafik tidak akan dibuat. Dalam skenario nyata, seseorang juga harus menentukan indeks dan batasan untuk memastikan pencarian cepat dan tidak mencari seluruh database. Kami belum melakukan ini di sini karena kumpulan data simulasi kami kecil. Namun, ini dapat dengan mudah dilakukan dengan menggunakan Cypher. Lihat dokumentasi Cypher untuk mengetahui lebih lanjut tentang indeks dan kendala (http://neo4j.com/docs/stable/cypherdoc-labels-constraints-and-indexes.html). Sekarang

setelah kami membuat data kami, kami dapat menanyakannya. Kueri berikut akan mengembalikan semua node dan relasi dalam database:



Gambar 7.10 memperlihatkan database yang telah kita buat. Kita dapat membandingkan grafik ini dengan grafik yang kita bayangkan di papan tulis kita. Di papan tulis kami, kami mengelompokkan simpul orang dalam label "Pengguna" dan simpul negara dalam label "Negara". Meskipun node dalam gambar ini tidak diwakili oleh labelnya, label tersebut ada di database kami. Selain itu, kami juga melewatkan simpul (Hobby) dan hubungan bertipe “Loves”. Ini dapat dengan mudah ditambahkan melalui pernyataan gabungan yang akan membuat simpul dan hubungan jika belum ada:

```
Merge (user3)-[: Loves]->( hobby1)
```



Gambar 7.10 Grafik yang digambar pada gambar 7.9 sekarang telah dibuat di antarmuka web Neo4j.

Node tidak diwakili oleh label mereka tetapi dengan nama mereka. Kami dapat menyimpulkan dari grafik bahwa kami kehilangan label Hobby dengan nama Bepergian. Alasannya adalah karena kita lupa menyertakan node ini dan hubungannya yang sesuai dalam pernyataan buat.

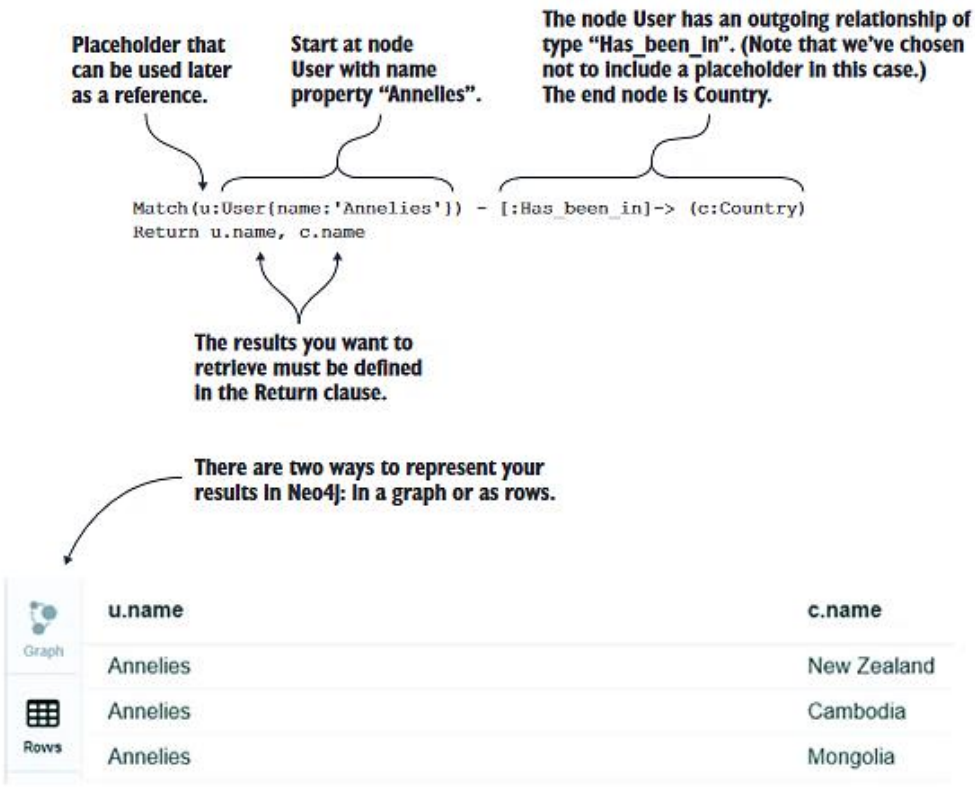
Kita bisa mengajukan banyak pertanyaan di sini. Misalnya:

Pertanyaan 1: Negara mana saja yang pernah dikunjungi Annelies? Kode Cypher untuk membuat jawaban (ditunjukkan pada gambar 7.11) adalah

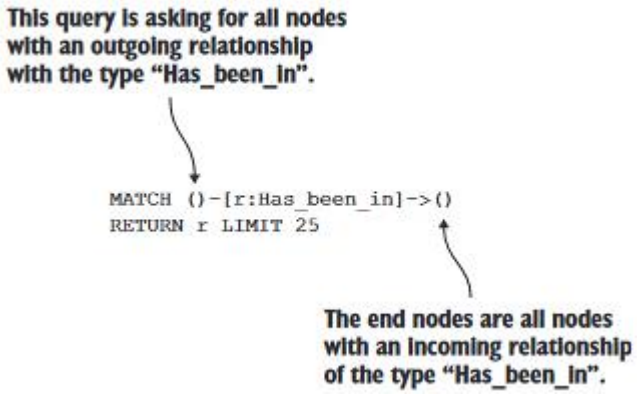
```
Match(u:User{name:'Annelies'}) - [:Has_been_in]-> (c:Country)
Return u.name, c.name
```

Pertanyaan 2: Siapa yang pernah ke mana? Kode Cypher (dijelaskan pada gambar 7.12) adalah

```
Match ()-[r: Has_been_in]->()
Return r LIMIT 25
```

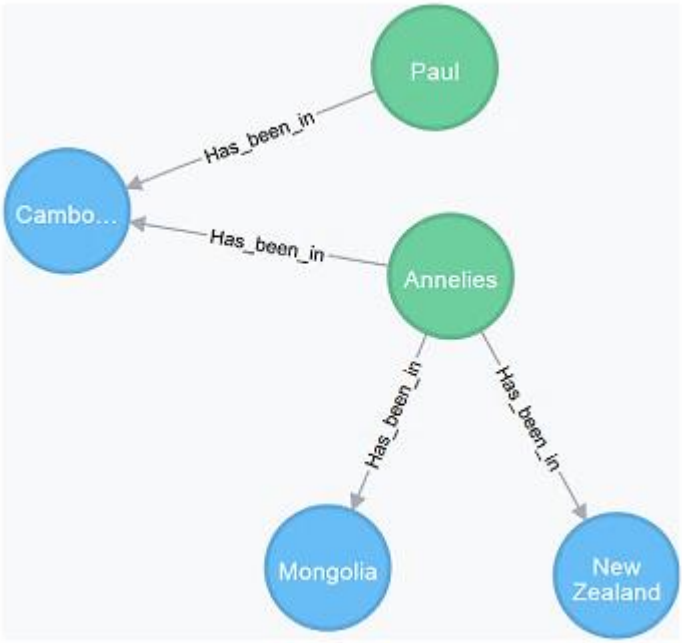


Gambar 7.11 Hasil pertanyaan 1: Negara mana saja yang pernah dikunjungi Annelies? Kita bisa melihat tiga negara yang pernah didatangi Annelies, menggunakan deretan presentasi Neo4j. Penjelajahan hanya membutuhkan waktu 97 milidetik.



Gambar 7.12 Siapa yang pernah ke mana? Penumpukan kueri dijelaskan.

Saat kami menjalankan kueri ini, kami mendapatkan jawaban yang ditunjukkan pada gambar 7.13.



Gambar 7.13 Hasil pertanyaan 2: Siapa yang pernah ke mana? Hasil traversal kami sekarang ditampilkan dalam representasi grafik Neo4j. Sekarang kita bisa melihat bahwa Paul, selain Annelies, juga pernah ke Kamboja.

Dalam pertanyaan 2 kami telah memilih untuk tidak menentukan simpul awal. Oleh karena itu, Cypher akan pergi ke semua node yang ada di database untuk menemukan node yang memiliki hubungan keluar dengan tipe "Has_been_in". Seseorang harus menghindari untuk tidak menentukan simpul awal karena, tergantung pada ukuran database Anda, kueri seperti itu bisa memakan waktu lama untuk menyatu. Bermain-main dengan data untuk mendapatkan database grafik yang tepat juga berarti banyak penghapusan data. Cypher memiliki pernyataan hapus yang cocok untuk menghapus sejumlah kecil data. Kueri berikut menunjukkan cara menghapus semua node dan hubungan dalam database:

```
MATCH(n)
Optional MATCH (n)-[r]-()
Delete n,r
```

Sekarang setelah kita mengenal data yang terhubung dan memiliki pengetahuan dasar tentang bagaimana data tersebut dikelola dalam database grafik, kita dapat melangkah lebih jauh dan melihat aplikasi nyata dan langsung dari data yang terhubung. Grafik sosial, misalnya, dapat digunakan untuk menemukan kumpulan node yang terhubung erat di dalam komunitas grafik. Orang-orang dalam sebuah cluster yang tidak saling mengenal kemudian dapat diperkenalkan satu sama lain. Konsep mencari node yang terhubung erat, node yang memiliki banyak kesamaan fitur, adalah konsep yang banyak digunakan. Di bagian selanjutnya kita akan

menggunakan ide ini, di mana tujuannya adalah untuk menemukan klaster di dalam jaringan bahan.

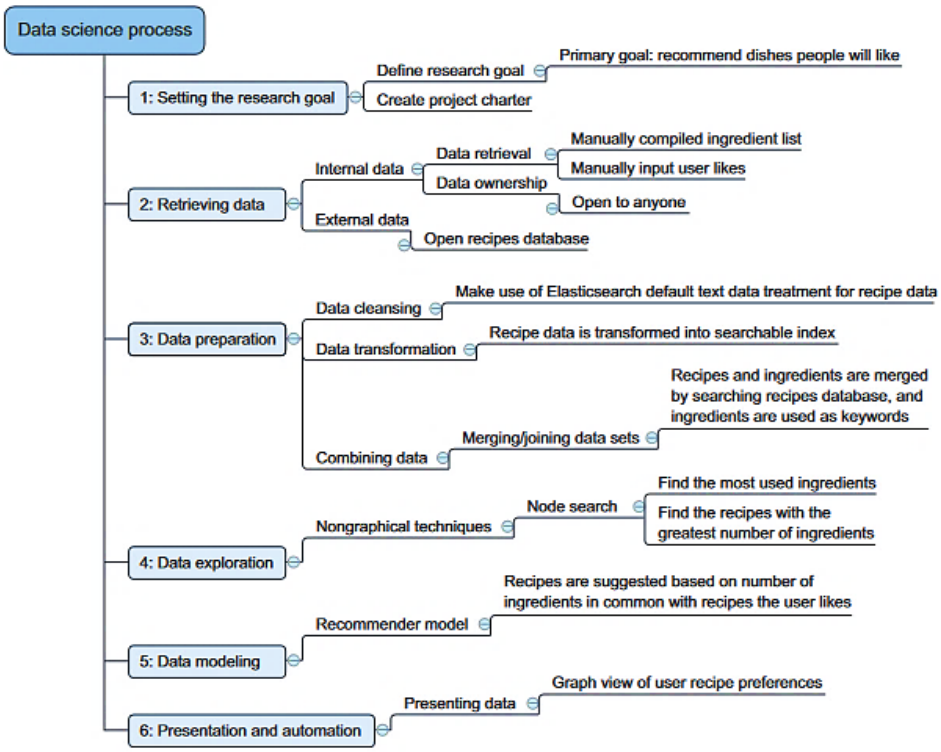
7.3 CONTOH DATA TERHUBUNG MESIN REKOMENDASI RESEP

Salah satu kasus penggunaan yang paling populer untuk database grafik adalah pengembangan mesin pemberi rekomendasi. Mesin pemberi rekomendasi diadopsi secara luas melalui janji mereka untuk membuat konten yang relevan. Hidup di era dengan data yang melimpah dapat membuat banyak konsumen kewalahan. Perusahaan melihat kebutuhan yang jelas untuk inventif dalam cara menarik pelanggan melalui konten yang dipersonalisasi, sehingga menggunakan kekuatan mesin pemberi rekomendasi.

Dalam studi kasus kami, kami akan merekomendasikan resep berdasarkan preferensi hidangan pengguna dan jaringan bahan. Selama persiapan data, kami akan menggunakan Elasticsearch untuk mempercepat proses dan memungkinkan lebih banyak fokus pada basis data grafik yang sebenarnya. Tujuan utamanya di sini adalah untuk mengganti daftar bahan dari data unduhan "kotor" dengan bahan dari daftar "bersih" kami sendiri.

7.3.1 Langkah 1: Menetapkan tujuan penelitian

Mari kita lihat apa yang akan terjadi jika kita mengikuti proses ilmu data (gambar 7.14). Tujuan utama kami adalah menyiapkan mesin pemberi rekomendasi yang akan membantu pengguna situs memasak menemukan resep yang tepat. Pengguna dapat menyukai beberapa resep dan kami akan mendasarkan rekomendasi hidangan kami pada bahan yang tumpang tindih dalam jaringan resep. Ini adalah pendekatan yang sederhana dan intuitif, namun sudah memberikan hasil yang cukup akurat. Mari kita lihat tiga elemen data yang kita butuhkan.



Gambar 7.14 Ilmu data diterapkan pada model rekomendasi data terhubung

7.3.2 Langkah 2: Pengambilan data

Untuk latihan ini, kami memerlukan tiga jenis data:

- Resep dan bahannya masing-masing
- Daftar bahan berbeda yang ingin kami modelkan
- Setidaknya satu pengguna dan preferensinya untuk hidangan tertentu

Seperti biasa, kami dapat membaginya menjadi data yang tersedia atau dibuat secara internal dan data yang diperoleh secara eksternal.

- **Data internal—**Kami tidak memiliki preferensi atau bahan apa pun dari pengguna, tetapi ini adalah bagian terkecil dari data kami dan mudah dibuat. Beberapa preferensi input manual sudah cukup untuk membuat rekomendasi. Pengguna mendapatkan hasil yang lebih menarik dan akurat, semakin banyak umpan balik yang dia berikan. Kami akan memasukkan preferensi pengguna nanti dalam studi kasus. Daftar bahan dapat disusun secara manual dan akan tetap relevan untuk tahun-tahun mendatang, jadi silakan gunakan daftar dalam materi yang dapat diunduh untuk tujuan apa pun, komersial atau lainnya.
- **Data eksternal—**Resep adalah hal yang berbeda. Ada ribuan bahan, tapi ini bisa digabungkan menjadi jutaan hidangan. Namun, kami beruntung karena daftar yang cukup besar tersedia secara gratis di https://github.com/fictivekin/openrecipes. Terima kasih banyak kepada Fictive Kin untuk kumpulan data berharga ini dengan lebih dari seratus ribu resep. Tentu ada duplikat di sini, tetapi mereka tidak akan merusak kasus penggunaan kita seburuk itu.

Kami sekarang memiliki dua file data yang kami miliki: daftar 800+ bahan (ingredients.txt) dan lebih dari seratus ribu resep di file recipe.json. Contoh daftar bahan dapat dilihat pada daftar berikut.

Listing 7.2 Ingredients list text file sample

```
Ditalini
Egg Noodles
Farfalle
Fettuccine
Fusilli
Lasagna
Linguine
Macaroni
Orzo
```

File JSON "openrecipes" berisi lebih dari seratus ribu resep dengan banyak properti seperti tanggal publikasi, lokasi sumber, waktu persiapan, deskripsi, dan sebagainya. Kami hanya tertarik pada nama dan daftar bahan. Resep sampel ditampilkan dalam daftar berikut.

Listing 7.3 A sample JSON recipe

```
{ "_id" : { "$oid" : "5160756b96cc62079cc2db15" },
    "name" : "Drop Biscuits and Sausage Gravy",
    "ingredients" : "Biscuits\n3 cups All-purpose Flour\n2 Tablespoons Baking
     Powder\n1/2 teaspoon Salt\n1-1/2 stick (3/4 Cup) Cold Butter, Cut Into
     Pieces\n1-1/4 cup Buttermilk\n SAUSAGE GRAVY\n1 pound Breakfast Sausage,
     Hot Or Mild\n1/3 cup All-purpose Flour\n4 cups Whole Milk\n1/2 teaspoon
     Seasoned Salt\n2 teaspoons Black Pepper, More To Taste",
    "url" : "http://thepioneerwoman.com/cooking/2013/03/drop-biscuits-and-
     sausage-gravy/",
    "image" : "http://static.thepioneerwoman.com/cooking/files/2013/03/
     bisgrav.jpg",
    "ts" : { "$date" : 1365276011104 },
    "cookTime" : "PT30M",
    "source" : "thepioneerwoman",
    "recipeYield" : "12",
    "datePublished" : "2013-03-11",
    "prepTime" : "PT10M",
    "description" : "Late Saturday afternoon, after Marlboro Man had returned
     home with the soccer-playing girls, and I had returned home with the..."
}
```

Karena kita berurusan dengan data teks di sini, masalahnya ada dua: pertama, menyiapkan data tekstual seperti yang dijelaskan dalam bab penambangan teks. Kemudian, setelah data dibersihkan secara menyeluruh, data tersebut dapat digunakan untuk menghasilkan rekomendasi resep berdasarkan jaringan bahan. Bab ini tidak berfokus pada persiapan data teks karena ini dijelaskan di tempat lain, jadi kami akan membiarkan diri kami menikmati jalan pintas selama persiapan data yang akan datang.

7.3.3 Langkah 3: Persiapan data

Kami sekarang memiliki dua file data yang dapat kami gunakan, dan kami perlu menggabungkannya menjadi satu basis data grafik. Data resep "kotor" menimbulkan masalah yang dapat kami atasi dengan menggunakan daftar bahan bersih kami dan penggunaan mesin pencari dan database NoSQL Elasticsearch. Kita sudah mengandalkan Elasticsearch di bab sebelumnya dan sekarang ini akan membersihkan data resep untuk kita secara implisit saat membuat indeks. Kami kemudian dapat mencari data ini untuk menautkan setiap bahan ke setiap resep di mana itu terjadi. Kita dapat membersihkan data teks menggunakan Python murni, seperti yang kita lakukan di bab penambangan teks, tetapi ini menunjukkan bahwa ada baiknya untuk mengetahui poin kuat dari setiap database NoSQL; jangan menyematkan diri Anda pada satu teknologi, tetapi gunakan bersama-sama untuk kepentingan proyek.

Listing 7.4 Importing recipe data into Elasticsearch

```
from elasticsearch import Elasticsearch
import json

client = Elasticsearch ()
indexName = "gastronomical"
docType = 'recipes'

client.indices.create(index=indexName)

file_name = 'C:/Users/Gebruiker/Downloads/recipes.json'

recipeMapping = {
        'properties': {
            'name': {'type': 'string'},
            'ingredients': {'type': 'string'}
        }
    }

client.indices.put_mapping(index=indexName,doc_type=docType,body=recipeMapping )

with open(file_name, encoding="utf8") as data_file:
    recipeData = json.load(data_file)

for recipe in recipeData:
    print recipe.keys()
    print recipe['_id'].keys()
    client.index(index=indexName,
        doc_type=docType,id = recipe['_id']['$oid'],
        body={"name": recipe['name'], "ingredients":recipe['ingredients']})
```

Import modules.

Elasticsearch client used to communicate with database.

Create index.

Location of JSON recipe file: change this to match your own setup!

Mapping for Elasticsearch "recipe" doctype.

Load JSON recipe file into memory. Another way to do this would be:

```
recipeData = []
with open(file_name) as f:
  for line in f:
    recipeData.append(json.loads(line))
```

Index recipes. Only name and ingredients are important for our use case. In case a timeout problem occurs it's possible to increase the timeout delay by specifying, for example, timeout=30 as an argument.

Listing 7.5 Using the Elasticsearch index to fill the graph database

```
from elasticsearch import Elasticsearch
from py2neo import Graph, authenticate, Node, Relationship

client = Elasticsearch ()
indexName = "gastronomical"
docType = 'recipes'

authenticate("localhost:7474", "user", "password")
graph_db = Graph("http://localhost:7474/db/data/")

filename = 'C:/Users/Gebruiker/Downloads/ingredients.txt'
ingredients =[]
with open(filename) as f:
    for line in f:
        ingredients.append(line.strip())

print ingredients

ingredientnumber = 0
grandtotal = 0
for ingredient in ingredients:

    try:
        IngredientNode = graph_db.merge_one("Ingredient","Name",ingredient)
    except:
        continue

    ingredientnumber +=1
    searchbody = {
        "size" : 99999999,
        "query": {
            "match_phrase":
                {
                    "ingredients":{

                        "query":ingredient,
                    }
                }
    }
}
result = client.search(index=indexName,doc_type=docType,body=searchbody)

print ingredient
print ingredientnumber
print "total: " +  str(result['hits']['total'])

grandtotal = grandtotal + result['hits']['total']
print "grand total: " +  str(grandtotal)

for recipe in result['hits']['hits']:

    try:
        RecipeNode =
 graph_db.merge_one("Recipe","Name",recipe['_source']['name'])
        NodesRelationship = Relationship(RecipeNode, "Contains",
 IngredientNode)
        graph_db.create_unique(NodesRelationship)
        print "added: " + recipe['_source']['name'] + " contains " +
 ingredient

    except:
        continue

print "************************************"
```

Import modules

Elasticsearch client used to communicate with database

Authenticate with your own username and password

Graph database entity

Ingredients text file gets loaded into memory

Strip because of the /n you get otherwise from reading the .txt

Loop through ingredients and fetch Elasticsearch result

Create node in graph database for current ingredient

Phrase matching used, as some ingredients consist of multiple words

Loop through recipes found for this particular ingredient

Create relationship between this recipe and ingredient

Create node for each recipe that is not already in graph database

Mari kita mulai dengan memasukkan data resep kita ke dalam Elasticsearch. Jika Anda tidak mengerti apa yang terjadi, silakan periksa kembali studi kasus bab 6 dan itu akan menjadi jelas. Pastikan untuk mengaktifkan instans Elasticsearch lokal Anda dan aktifkan lingkungan Python dengan modul Elasticsearch terpasang sebelum menjalankan cuplikan kode di daftar berikut. Direkomendasikan untuk tidak menjalankan kode ini "sebagaimana adanya" di Ipython (atau Jupyter) karena kode ini mencetak setiap kunci resep ke layar dan browser Anda hanya dapat menangani begitu banyak keluaran. Matikan pernyataan cetak atau jalankan di IDE Python lain. Kode dalam potongan ini dapat ditemukan di "Persiapan Data Bagian 1.py".

Jika semuanya berjalan dengan baik, kami sekarang memiliki indeks Elasticsearch dengan nama "gastronomical" yang dihuni oleh ribuan resep. Perhatikan bahwa kami mengizinkan duplikat dari resep yang sama dengan tidak menetapkan nama resep sebagai kunci dokumen. Jika, misalnya, sebuah resep disebut “lasagna” maka ini bisa berupa lasagna salmon, lasagna daging sapi, lasagna ayam, atau jenis lainnya. Tidak ada satu resep pun yang dipilih sebagai prototipe lasagna; semuanya diunggah ke Elasticsearch dengan nama yang sama: "lasagna". Ini adalah pilihan, jadi jangan ragu untuk memutuskan sebaliknya. Ini akan memiliki dampak yang signifikan, seperti yang akan kita lihat nanti. Pintu sekarang terbuka untuk pengunggahan sistematis ke basis data grafik lokal kami. Pastikan instance database grafik lokal Anda diaktifkan saat menerapkan kode berikut. Nama pengguna kami untuk database ini adalah Neo4j default dan kata sandinya adalah Neo4ja; pastikan untuk menyesuaikan ini untuk pengaturan lokal Anda. Untuk ini, kami juga memerlukan pustaka Python khusus Neo4j yang disebut py2neo. Jika Anda belum melakukannya, sekarang saatnya untuk menginstalnya ke lingkungan virtual Anda menggunakan pip install py2neo atau conda install py2neo saat menggunakan Anaconda. Sekali lagi, maklum kode ini akan merusak browser Anda saat dijalankan langsung di Ipython atau Jupiter. Kode dalam daftar ini dapat ditemukan di "Persiapan Data Bagian 2.py".

Hebat, sekarang kami bangga menjadi pemilik database grafik yang penuh dengan resep! Saatnya untuk eksplorasi data yang terhubung.

7.3.4 Langkah 4: Eksplorasi data

Tidak ada yang menghentikan Anda untuk menjalankan kode Cypher di lingkungan ini, tetapi Cypher juga dapat dijalankan melalui pustaka py2neo. Satu pertanyaan menarik yang dapat kami ajukan adalah bahan mana yang paling banyak muncul di semua resep? Apa yang kemungkinan besar akan masuk ke sistem pencernaan kita jika kita memilih dan memakan hidangan secara acak dari database ini?

```
from py2neo 1mport Graph, authent1cate, Node, Relat1onsh1p
authent1cate("localhost:7474", "user", "password")
graph_db = Graph("http://localhost:7474/db/data/")graph_db.cypher.execute("
   MATCH (REC:Rec1pe)-[r:Conta1ns]->(ING:Ingred1ent) WITH ING, count(r) AS num
   RETURN ING.Name as Name, num ORDER BY num DESC LIMIT 10;")
```

Permintaan dibuat di Cypher dan mengatakan: untuk semua resep dan bahan-bahannya, hitung jumlah relasi per bahan dan kembalikan sepuluh bahan dengan relasi terbanyak dan jumlah masing-masing. Hasilnya ditunjukkan pada Gambar 7.15.

Sebagian besar daftar 10 teratas pada gambar 7.15 seharusnya tidak mengejutkan. Dengan garam yang menempati urutan teratas dalam daftar kita, kita tidak perlu terkejut menemukan penyakit pembuluh darah sebagai pembunuh nomor satu di sebagian besar negara barat. Pertanyaan menarik lainnya yang muncul di benak sekarang adalah dari sudut pandang yang berbeda: resep mana yang membutuhkan bahan paling banyak?

```
from py2neo import Graph, Node, Relationship
graph_db = Graph("http://neo4j:neo4ja@localhost:7474/db/data/")
graph_db.cypher.execute("
    MATCH (REC:Recipe)-[r:Contains]->(ING:Ingredient) WITH REC, count(r) AS num
    RETURN REC.Name as Name, num ORDER BY num DESC LIMIT 10;")
```

	Name	num
1	Salt	53885
2	Oil	42585
3	Sugar	38519
4	Pepper	38118
5	Butter	35610
6	Garlic	29879
7	Flour	28175
8	Olive Oil	25979
9	Onion	24888
10	Cloves	22832

Gambar 7.15 10 bahan teratas yang muncul di sebagian besar resep

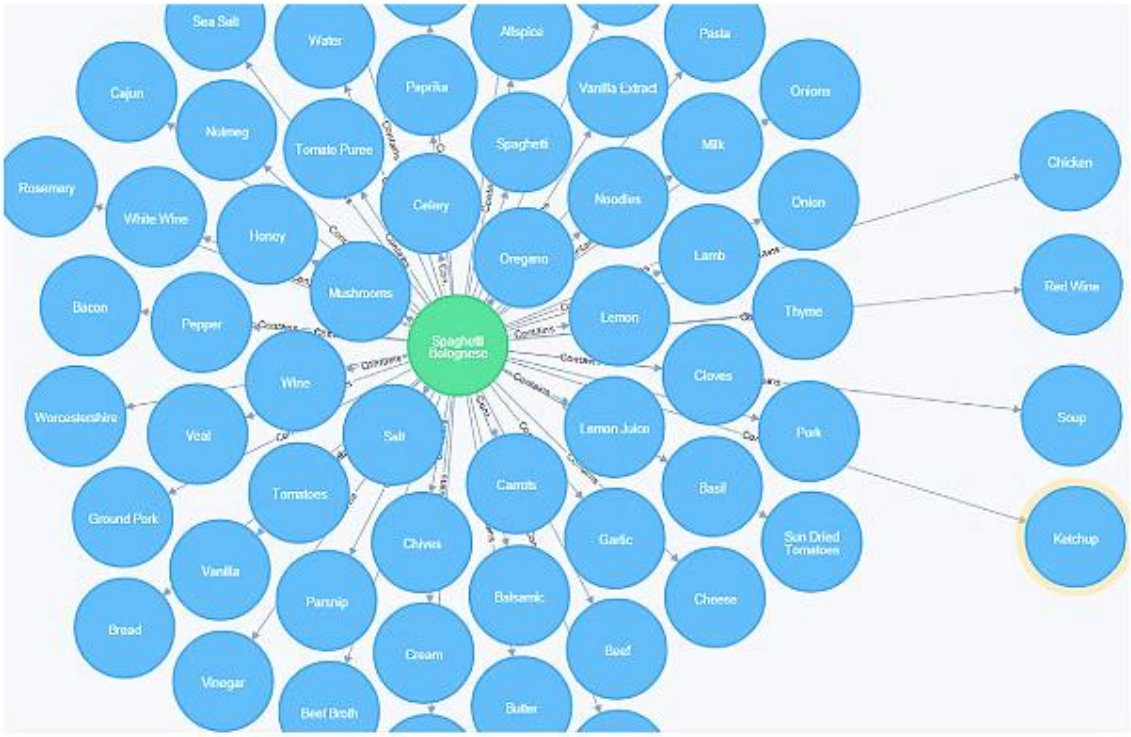
Permintaannya hampir sama seperti sebelumnya, tetapi alih-alih mengembalikan bahannya, kami meminta resepnya. Hasilnya adalah gambar 7.16.

	Name	num
1	Spaghetti Bolognese	59
2	Chicken Tortilla Soup	56
3	Kedgeree	55
4	Butternut Squash Soup	54
5	Hearty Beef Stew	53
6	Chicken Tikka Masala	52
7	Fish Tacos	52
8	Cooking For Others: 25 Years of Jor, 1 of BGSK	51
9	hibernation fare	50
10	Gazpacho	50

Gambar 7.16 10 hidangan teratas yang dapat dibuat dengan bahan paling beragam

Sekarang ini mungkin pemandangan yang mengejutkan. Spaghetti Bolognese hampir tidak terdengar seperti jenis hidangan yang membutuhkan 59 bahan. Mari kita lihat lebih dekat bahan-bahan yang tercantum untuk Spaghetti Bolognese.

```
from py2neo import Graph, Node, Relationship
graph_db = Graph("http://neo4j:neo4ja@localhost:7474/db/data/")
graph_db.cypher.execute("MATCH (REC1:Recipe{Name:'Spaghetti Bolognese'})-
    [r:Contains]->(ING:Ingredient) RETURN REC1.Name, ING.Name;")
```



Gambar 7.17 Kemungkinan bahan Spaghetti Bolognese

Kueri Cypher hanya mencantumkan bahan-bahan yang ditautkan ke Spaghetti Bolognese. Gambar 7.17 menunjukkan hasil di antarmuka web Neo4j. Mari ingatkan diri kita sendiri tentang pernyataan yang kita buat saat mengindeks data di Elasticsearch. Pencarian cepat Elasticsearch pada Spaghetti Bolognese menunjukkan kepada kita bahwa hal itu terjadi berkali-kali, dan semua kejadian ini digunakan untuk menghubungkan bahan-bahan dengan Spaghetti Bolognese sebagai resep. Kita tidak harus melihat Spaghetti Bolognese sebagai resep tunggal tetapi lebih sebagai kumpulan cara orang membuat “Spaghetti Bolognese” mereka sendiri. Ini membuat cara yang menarik untuk melihat data ini. Orang dapat membuat versi hidangan mereka sendiri dengan saus tomat, anggur merah, dan ayam atau bahkan menambahkan sup. Dengan sajian “Spaghetti Bolognese” yang begitu terbuka untuk ditafsirkan, tak heran banyak orang yang menyukainya. Kisah Spaghetti Bolognese adalah gangguan yang menarik tetapi bukan tujuan kami datang. Saatnya untuk merekomendasikan hidangan ke rakus kami "Ragnar".

7.3.5 Langkah 5: Pemodelan data

Dengan pengetahuan kami tentang data yang sedikit diperkaya, kami mencapai tujuan latihan ini: rekomendasi. Untuk ini kami memperkenalkan pengguna yang kami sebut "Ragnar", yang menyukai beberapa hidangan. Informasi baru ini perlu diserap oleh basis data grafik kami sebelum kami dapat mengharapkannya untuk menyarankan hidangan baru. Oleh karena itu, mari kita buat node pengguna Ragnar dengan beberapa preferensi resep.

Listing 7.6 Creating a user node who likes certain recipes in the Neo4j graph database

```
from py2neo import Graph, Node, Relationship

graph_db = Graph("http://neo4j:neo4ja@localhost:7474/db/data/")

UserRef = graph_db.merge_one("User","Name","Ragnar")

RecipeRef = graph_db.find_one("Recipe",property_key="Name",
     property_value="Spaghetti Bolognese")
NodesRelationship = Relationship(UserRef, "Likes", RecipeRef)
graph_db.create_unique(NodesRelationship) #Commit his like to database

graph_db.create_unique(Relationship(UserRef, "Likes",
     graph_db.find_one("Recipe",property_key="Name",
     property_value="Roasted Tomato Soup with Tiny Meatballs
     and Rice")))
graph_db.create_unique(Relationship(UserRef, "Likes",
     graph_db.find_one("Recipe",property_key="Name",
     property_value="Moussaka")))
graph_db.create_unique(Relationship(UserRef, "Likes",
     graph_db.find_one("Recipe",property_key="Name",
     property_value="Chipolata &amp;amp; spring onion frittata")))
graph_db.create_unique(Relationship(UserRef, "Likes",
     graph_db.find_one("Recipe",property_key="Name",
     property_value="Meatballs In Tomato Sauce")))
graph_db.create_unique(Relationship(UserRef, "Likes",
     graph_db.find_one("Recipe",property_key="Name",
     property_value="Macaroni cheese")))
graph_db.create_unique(Relationship(UserRef, "Likes",
     graph_db.find_one("Recipe",property_key="Name",
     property_value="Peppered Steak")))
```

Import modules

Make graph database connection object

Create new user called "Ragnar"

Ragnar likes Spaghetti Bolognese

Find recipe by the name of Spaghetti Bolognese

Create a like relationship between Ragnar and the spaghetti

Repeat the same process as in the lines above but for several other dishes

Dalam daftar 7.6 penikmat makanan kami Ragnar ditambahkan ke database bersama dengan preferensinya untuk beberapa hidangan. Jika kita memilih Ragnar di antarmuka Neo4j, kita mendapatkan gambar 7.18. Kueri Cypher untuk ini adalah

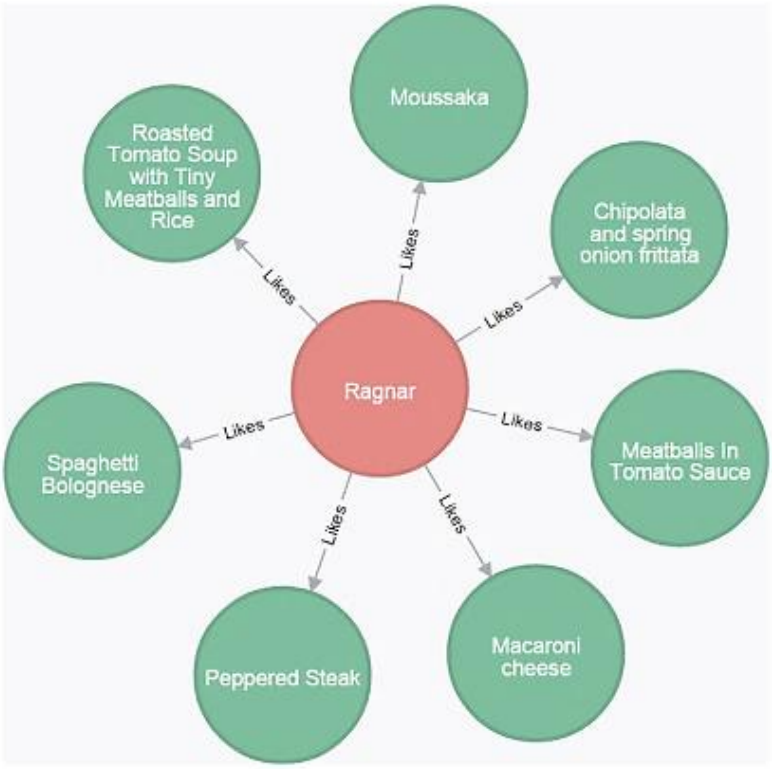
```
MATCH (U:User)-[r:Likes]->(REC:Recipe) RETURN U,REC LIMIT 25
```

Tidak ada kejutan di gambar 7.18: banyak orang menyukai Spaghetti Bolognese, begitu pula ahli gastronomi Skandinavia kami, Ragnar.

Untuk mesin rekomendasi sederhana yang ingin kami buat, yang perlu kami lakukan hanyalah meminta database grafik untuk memberi kami hidangan terdekat dalam hal bahan. Sekali lagi, ini adalah pendekatan dasar untuk sistem pemberi rekomendasi karena tidak memperhitungkan faktor-faktor seperti

- Tidak suka bahan atau hidangan.
- Jumlah suka atau tidak suka. Skor dari 10 bukannya biner suka atau tidak suka bisa membuat perbedaan.
- Jumlah bahan yang ada di piring.
- Ambang batas untuk bahan tertentu menjadi jelas dalam rasanya. Bahan-bahan tertentu, seperti cabai, akan memberikan dampak yang lebih besar untuk dosis yang lebih kecil daripada bahan lainnya.

- Alergi makanan. Meskipun hal ini secara implisit dimodelkan dalam suka atau tidak suka hidangan dengan bahan tertentu, alergi makanan bisa menjadi sangat penting sehingga satu kesalahan saja bisa berakibat fatal. Menghindari alergen harus menimpa seluruh sistem rekomendasi.
- Lebih banyak hal untuk Anda renungkan.



Gambar 7.18 Pengguna Ragnar menyukai beberapa hidangan

Ini mungkin sedikit mengejutkan, tetapi satu perintah Cypher sudah cukup.

```
from py2neo import Graph, Node, Relationship
graph_db = Graph("http://neo4j:neo4ja@localhost:7474/db/data/")
graph_db.cypher.execute("
   MATCH (USR1:User{Name:'Ragnar'})-[l1:Likes]->(REC1:Recipe),
         (REC1)-[c1:Contains]->(ING1:Ingredient)
      WITH  ING1,REC1 MATCH (REC2:Recipe)-[c2:Contains]->(ING1:Ingredient)
      WHERE REC1 <> REC2
   RETURN REC2.Name,count(ING1) AS IngCount ORDER BY IngCount DESC LIMIT 20;")
```

Pertama, semua resep yang disukai Ragnar dikumpulkan. Kemudian bahan-bahannya digunakan untuk mengambil semua hidangan lain yang membaginya. Bahan-bahan tersebut kemudian dihitung untuk setiap hidangan yang terhubung dan diberi peringkat dari banyak bahan umum hingga sedikit. Hanya 20 hidangan teratas yang disimpan; ini menghasilkan tabel gambar 7.19.

	REC2.Name	IngCount
1	Spaghetti and Meatballs	104
2	Hearty Beef Stew	91
3	Cassoulet	89
4	Lasagne	88
5	Spaghetti & Meatballs	86
6	Good old lasagne	84
7	Beef Wellington	84
8	Braised Short Ribs	83
9	Lasagna	83
10	Italian Wedding Soup	82
11	French Onion Soup	82
12	Coq au vin	82
13	Shepherd's pie	81
14	Great British pork: from head to toe	81
15	Three Meat Cannelloni Bake	81
16	Cioppino	81
17	hibernation fare	80
18	Spaghetti and Meatballs Recipe with Oven Roasted Tomato Sauce	80
19	Braised Lamb Shanks	80
20	Lamb and Eggplant Casserole (Moussaka)	80

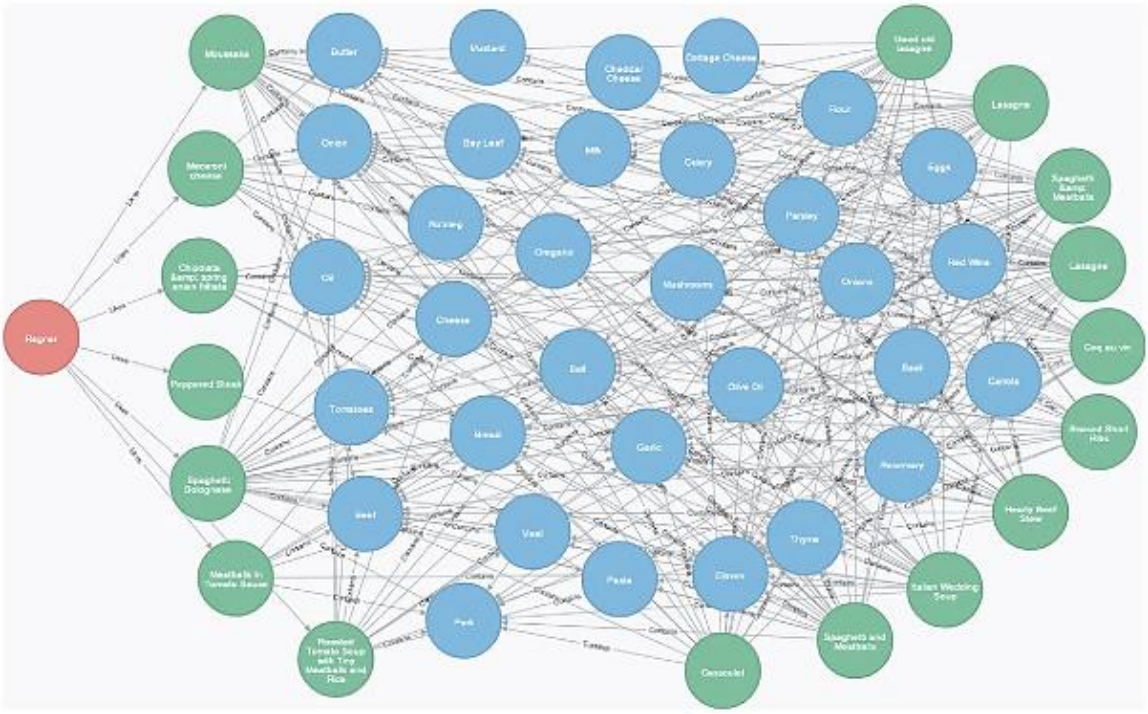
Gambar 7.19 Hasil rekomendasi resep; 20 hidangan teratas yang mungkin disukai pengguna

Dari gambar 7.19 kita dapat menyimpulkan sudah waktunya bagi Ragnar untuk mencoba Spaghetti and Meatballs, hidangan yang dibuat sangat terkenal oleh animasi Disney Lady and the Tramp. Ini memang terdengar seperti rekomendasi yang bagus untuk seseorang yang sangat menyukai hidangan yang mengandung pasta dan bakso, tetapi seperti yang dapat kita lihat dari jumlah bahannya, lebih banyak bahan yang mendukung saran ini. Untuk memberi kita sedikit petunjuk tentang apa yang ada di baliknya, kita dapat menunjukkan hidangan yang disukai, rekomendasi teratas, dan beberapa bahan yang tumpang tindih dalam satu gambar grafik ringkasan.

7.3.6 Langkah 6: Presentasi

Antarmuka web Neo4j memungkinkan kita untuk menjalankan model dan mengambil grafik yang tampak bagus yang meringkas bagian dari logika di balik rekomendasi. Ini menunjukkan bagaimana hidangan yang direkomendasikan terkait dengan hidangan yang disukai melalui bahan-bahannya. Ini ditunjukkan pada gambar 7.20 dan merupakan hasil akhir untuk studi kasus kami.

Dengan gambar grafik yang indah ini kita dapat menyimpulkan bab kita dalam pengetahuan bahwa Ragnar memiliki beberapa hidangan lezat untuk dinantikan. Jangan lupa untuk mencoba sendiri sistem rekomendasi dengan memasukkan preferensi Anda sendiri.



Gambar 7.20 Keterkaitan hidangan pilihan pengguna dan 10 hidangan teratas yang direkomendasikan melalui sub-pilihan bahan yang tumpang tindih

7.4 RINGKASAN

Dalam bab ini Anda belajar

- Database grafik sangat berguna saat menghadapi data di mana hubungan antar entitas sama pentingnya dengan entitas itu sendiri. Dibandingkan dengan database NoSQL lainnya, mereka dapat menangani kompleksitas terbesar tetapi data paling sedikit.
- Struktur data grafik terdiri dari dua komponen utama:
  – Node—Ini adalah entitas itu sendiri. Dalam studi kasus kami, ini adalah resep dan bahan.
  – Tepi—Hubungan antar entitas. Hubungan, seperti node, bisa dari semua jenis jenis (misalnya "berisi", "suka", "telah berkunjung") dan dapat memiliki properti spesifiknya sendiri seperti nama, bobot, atau ukuran lainnya.
- Kami melihat Neo4j, database grafik paling populer saat ini. Untuk instruksi tentang cara menginstalnya, Anda dapat melihat lampiran B. Kami melihat cara menambahkan data ke Neo4j, menanyakannya menggunakan Cypher, dan cara mengakses antarmuka webnya.
- Cypher adalah bahasa kueri khusus basis data Neo4j, dan kami melihat beberapa contoh. Kami juga menggunakannya dalam studi kasus sebagai bagian dari sistem rekomendasi hidangan kami.
- Dalam studi kasus bab ini, kami menggunakan Elasticsearch untuk membersihkan timbunan data resep yang sangat besar. Kami kemudian mengonversi data ini ke database Neo4j dengan resep dan bahan. Tujuan dari studi kasus ini adalah untuk merekomendasikan hidangan kepada orang-orang berdasarkan minat yang ditunjukkan sebelumnya pada hidangan lain. Untuk ini kami memanfaatkan keterhubungan resep melalui bahan-bahannya. Pustaka py2neo memungkinkan kami untuk berkomunikasi dengan server Neo4j dari Python.

- Ternyata database grafik tidak hanya berguna untuk mengimplementasikan sistem rekomendasi tetapi juga untuk eksplorasi data. Salah satu hal yang kami temukan adalah keragaman (dari segi bahan) resep Spaghetti Bolognese di luar sana.
- Kami menggunakan antarmuka web Neo4j untuk membuat representasi visual tentang cara kami mendapatkan dari preferensi hidangan hingga rekomendasi hidangan melalui node bahan.

BAB 8
PENAMBANGAN TEKS DAN ANALISIS TEKS

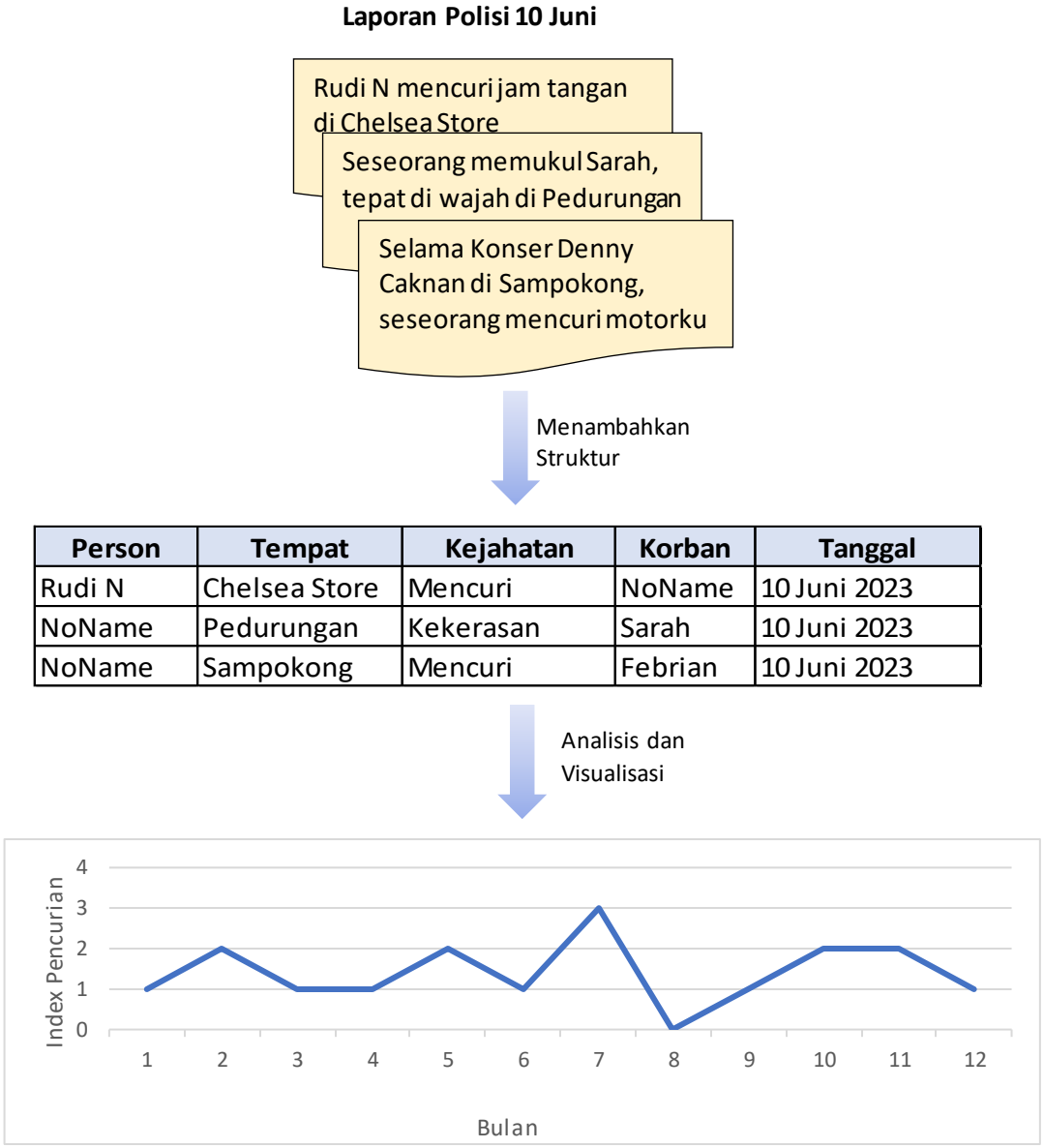
Dalam bab ini diharapkan mahasiswa akan memahami:

- Memahami pentingnya penambangan teks
- Memperkenalkan konsep paling penting dalam penambangan teks
- Bekerja melalui proyek penambangan teks

Sebagian besar informasi yang direkam manusia di dunia adalah dalam bentuk teks tertulis. Kita semua belajar membaca dan menulis sejak bayi sehingga kita dapat mengekspresikan diri melalui tulisan dan mempelajari apa yang diketahui, dipikirkan, dan dirasakan orang lain. Kami menggunakan keterampilan ini setiap saat saat membaca atau menulis email, blog, pesan teks, atau buku ini, jadi tidak heran jika bahasa tertulis muncul secara alami bagi sebagian besar dari kita. Bisnis yakin bahwa banyak nilai dapat ditemukan dalam teks yang dibuat orang, dan memang demikian karena berisi informasi tentang apa yang disukai, tidak disukai, apa yang mereka ketahui atau ingin ketahui, dambakan dan inginkan, kesehatan atau suasana hati mereka saat ini, dan masih banyak lagi. Banyak dari hal-hal ini dapat relevan bagi perusahaan atau peneliti, tetapi tidak ada satu orang pun yang dapat membaca dan menafsirkan sendiri tsunami bahan tertulis ini. Sekali lagi, kita perlu beralih ke komputer untuk melakukan pekerjaan untuk kita.

Sayangnya, bagaimanapun, bahasa alami tidak datang sebagai "alami" untuk komputer seperti halnya manusia. Memperoleh makna dan menyaring yang tidak penting dari yang penting masih merupakan sesuatu yang lebih baik dilakukan manusia daripada mesin mana pun. Untungnya, ilmuwan data dapat menerapkan teknik penambangan teks dan analitik teks khusus untuk menemukan informasi yang relevan dalam tumpukan teks yang seharusnya membutuhkan waktu berabad-abad untuk membacanya sendiri.

Penambangan teks atau analitik teks adalah disiplin yang menggabungkan ilmu bahasa dan ilmu komputer dengan teknik pembelajaran statistik dan mesin. Penambangan teks digunakan untuk menganalisis teks dan mengubahnya menjadi bentuk yang lebih terstruktur. Kemudian ia mengambil bentuk terstruktur ini dan mencoba memperoleh wawasan darinya. Saat menganalisis kejahatan dari laporan polisi, misalnya, penambangan teks membantu Anda mengenali orang, tempat, dan jenis kejahatan dari laporan tersebut. Kemudian struktur baru ini digunakan untuk mendapatkan wawasan tentang evolusi kejahatan. Lihat gambar 8.1.



Gambar 8.1 Dalam analitik teks, (biasanya) tantangan pertama adalah menyusun teks masukan; kemudian dapat dianalisis secara menyeluruh.

Meskipun bahasa tidak terbatas pada bahasa alami, fokus bab ini adalah Pemrosesan Bahasa Alami (NLP). Contoh bahasa non-alami adalah log mesin, matematika, dan kode Morse. Secara teknis bahkan bahasa Esperanto, Klingon, dan Naga tidak termasuk dalam bidang bahasa alami karena mereka diciptakan dengan sengaja alih-alih berkembang dari waktu ke waktu; mereka tidak menjadi "alami" bagi kita. Bahasa terakhir ini tetap cocok untuk komunikasi alami (ucapan, tulisan); mereka memiliki tata bahasa dan kosa kata seperti semua bahasa alami, dan teknik penambangan teks yang sama dapat diterapkan pada mereka.

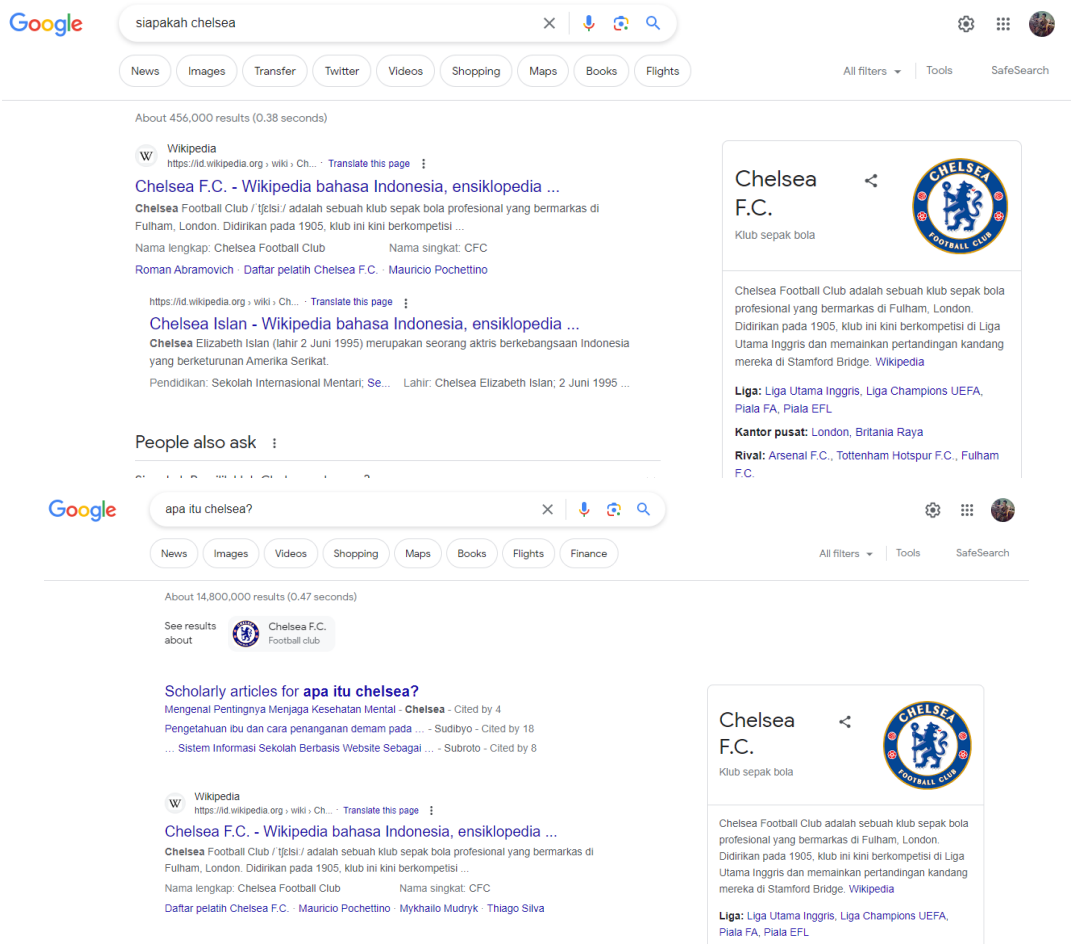
8.1 PENAMBANGAN TEKS DI DUNIA NYATA

Dalam kehidupan Anda sehari-hari, Anda telah menemukan aplikasi penambangan teks dan bahasa alami. Korektor pelengkapan otomatis dan ejaan terus menganalisis teks yang Anda ketik sebelum mengirim email atau pesan teks. Ketika Facebook melengkapi status Anda

secara otomatis dengan nama seorang teman, itu dilakukan dengan bantuan teknik yang disebut pengenalan entitas bernama, meskipun ini hanya salah satu komponen dari repertoar mereka. Tujuannya tidak hanya untuk mendeteksi bahwa Anda sedang mengetik kata benda, tetapi juga untuk menebak bahwa Anda merujuk pada seseorang dan mengenali siapa itu. Contoh lain dari pengenalan entitas bernama ditunjukkan pada gambar 8.2. Google tahu Chelsea adalah klub sepak bola tetapi merespons secara berbeda ketika ditanya tentang seseorang.

Google menggunakan banyak jenis penambangan teks saat menampilkan hasil kueri kepada Anda. Apa yang muncul di benak Anda ketika seseorang mengatakan “Chelsea”? Chelsea bisa jadi banyak hal: seseorang; klub sepak bola; sebuah lingkungan di Manhattan, New York atau London; pasar makanan; pertunjukan bunga; dan seterusnya. Google mengetahui hal ini dan memberikan jawaban berbeda untuk pertanyaan "Siapa Chelsea?" versus "Apa itu Chelsea?" Untuk memberikan jawaban yang paling relevan, Google harus melakukan (antara lain) semua hal berikut:

- Praproses semua dokumen yang dikumpulkannya untuk entitas bernama
- Lakukan identifikasi bahasa
- Mendeteksi jenis entitas yang Anda maksud
- Mencocokkan kueri dengan hasil
- Mendeteksi jenis konten yang akan dikembalikan (PDF, peka dewasa)

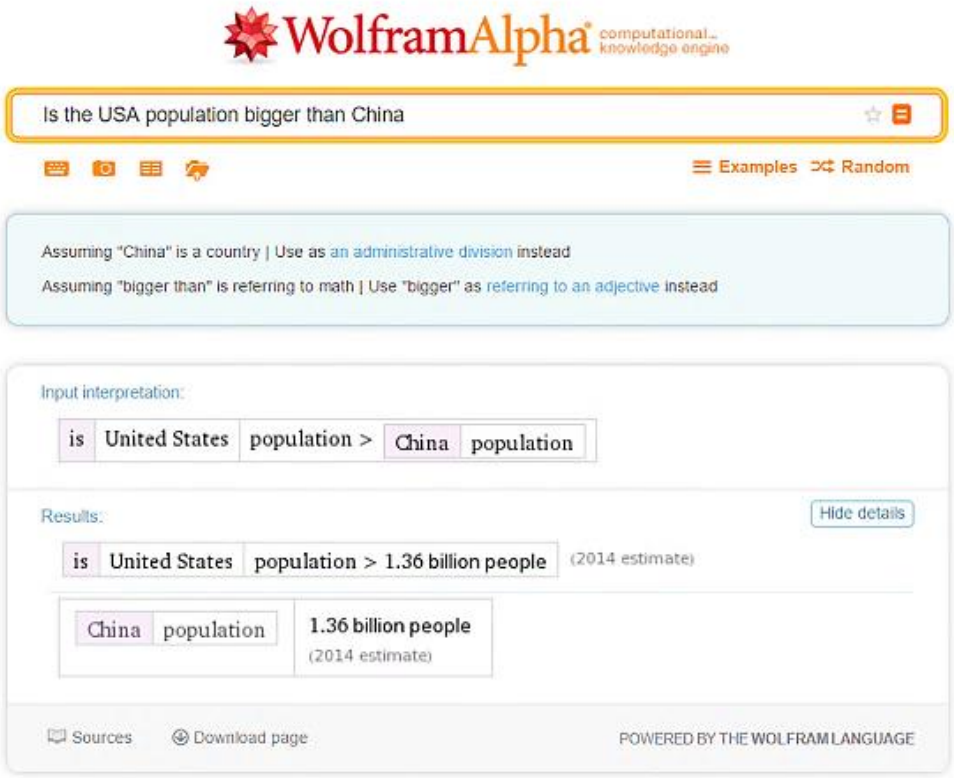


Gambar 8.2 Berbagai jawaban untuk pertanyaan “Siapakah Chelsea?” dan "Apa itu Chelsea?" menyiratkan bahwa Google menggunakan teknik penambangan teks untuk menjawab pertanyaan ini.

Contoh ini menunjukkan bahwa penambangan teks tidak hanya tentang makna langsung dari teks itu sendiri tetapi juga melibatkan atribut meta seperti bahasa dan jenis dokumen. Google menggunakan penambangan teks lebih dari sekadar menjawab pertanyaan. Selain melindungi pengguna Gmail dari spam, ia juga membagi email ke dalam berbagai kategori seperti sosial, pembaruan, dan forum, seperti yang ditunjukkan pada gambar 8.3. Mungkin untuk melangkah lebih jauh daripada menjawab pertanyaan sederhana saat Anda menggabungkan teks dengan logika dan matematika lainnya.

Gambar 8.3 Email dapat secara otomatis dibagi menurut kategori berdasarkan konten dan asal.

Hal ini memungkinkan pembuatan mesin penalaran otomatis yang digerakkan oleh kueri bahasa alami. Gambar 8.4 menunjukkan bagaimana “Wolfram Alpha,” mesin pengetahuan komputasional, menggunakan penambangan teks dan penalaran otomatis untuk menjawab pertanyaan “Apakah populasi AS lebih besar dari China?”



Gambar 8.4 Mesin Wolfram Alpha menggunakan penambangan teks dan penalaran logis untuk menjawab pertanyaan.

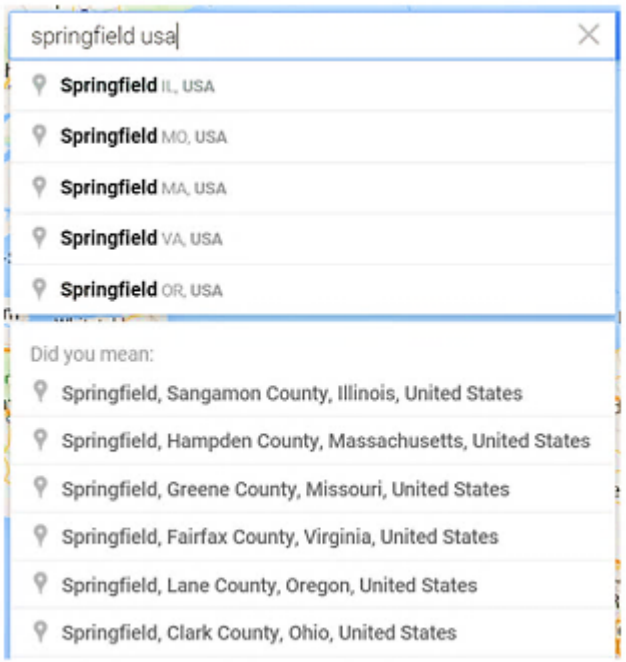
Jika ini tidak cukup mengesankan, IBM Watson mengejutkan banyak orang pada tahun 2011 ketika mesin tersebut dipasang melawan dua pemain manusia dalam permainan Jeopardy. Jeopardy adalah acara kuis Amerika di mana orang menerima jawaban atas pertanyaan dan poin dinilai untuk menebak pertanyaan yang benar untuk jawaban itu. Lihat gambar 8.5. Aman untuk mengatakan putaran ini mengarah ke kecerdasan buatan. IBM Watson adalah mesin kognitif yang dapat menafsirkan bahasa alami dan menjawab pertanyaan berdasarkan basis pengetahuan yang luas.

Gambar 8.5 IBM Watson memenangkan Jeopardy melawan pemain manusia.

Penambangan teks memiliki banyak aplikasi, termasuk, namun tidak terbatas pada, berikut ini:

- Identifikasi entitas
- Deteksi plagiarisme
- Identifikasi topik
- Pengelompokan teks
- Terjemahan
- Peringkasan teks otomatis
- Deteksi penipuan
- Pemfilteran spam
- Analisis sentimen

Penambangan teks bermanfaat, tetapi apakah itu sulit? Maaf mengecewakan: Ya, benar. Saat melihat contoh Wolfram Alpha dan IBM Watson, Anda mungkin mendapat kesan bahwa penambangan teks itu mudah. Sayangnya tidak. Pada kenyataannya penambangan teks adalah tugas yang rumit dan bahkan banyak hal yang tampaknya sederhana tidak dapat dilakukan dengan memuaskan. Misalnya, ambil tugas menebak alamat yang benar. Gambar 8.6 menunjukkan betapa sulitnya mengembalikan hasil yang tepat dengan kepastian dan bagaimana Google Maps meminta Anda untuk informasi lebih lanjut saat mencari "Springfield." Dalam hal ini manusia tidak akan melakukan yang lebih baik tanpa konteks tambahan, tetapi ambiguitas ini adalah salah satu dari banyak masalah yang Anda hadapi dalam aplikasi penambangan teks.



Gambar 8.6 Google Maps meminta lebih banyak konteks karena ambiguitas kueri "Springfield."

Masalah lainnya adalah kesalahan ejaan dan bentuk ejaan kata yang berbeda (benar). Ambil tiga referensi berikut ke New York: "NY", "Neww York", dan "New York". Bagi manusia, mudah untuk melihat bahwa mereka semua mengacu pada kota New York. Karena cara otak kita menginterpretasikan teks, memahami teks dengan kesalahan ejaan menjadi hal yang wajar bagi kita; orang bahkan mungkin tidak menyadarinya. Tetapi untuk komputer, ini adalah string yang tidak terkait kecuali jika kami menggunakan algoritme untuk memberi tahu bahwa string tersebut merujuk ke entitas yang sama. Masalah terkait adalah sinonim dan penggunaan kata ganti. Coba tetapkan orang yang tepat untuk kata ganti "dia" di kalimat berikutnya: "John memberikan bunga kepada orang tua Marleen saat pertama kali bertemu orang tuanya. Dia sangat senang dengan sikap ini." Cukup mudah, bukan? Bukan untuk komputer.

Kita dapat memecahkan banyak masalah serupa dengan mudah, tetapi sering kali terbukti sulit untuk sebuah mesin. Kita dapat melatih algoritme yang bekerja dengan baik pada masalah tertentu dalam lingkup yang terdefinisi dengan baik, tetapi algoritme yang lebih umum yang bekerja di semua kasus adalah monster yang sama sekali berbeda. Misalnya, kami dapat mengajari komputer untuk mengenali dan mengambil nomor akun AS dari teks, tetapi ini tidak dapat digeneralisasikan dengan baik ke nomor akun dari negara lain.

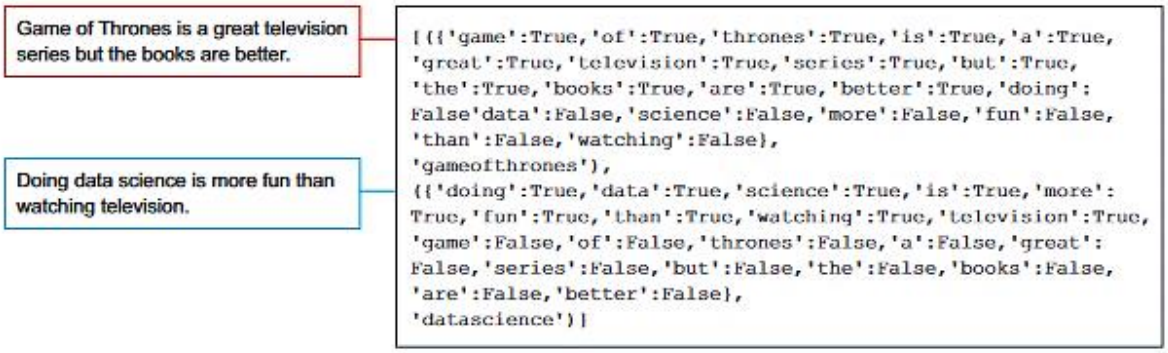
Algoritma bahasa juga peka terhadap konteks bahasa yang digunakan, meskipun bahasa itu sendiri tetap sama. Model bahasa Inggris tidak akan berfungsi untuk bahasa Arab dan sebaliknya, tetapi meskipun kita tetap menggunakan bahasa Inggris—algoritme yang dilatih untuk data Twitter kemungkinan tidak akan bekerja dengan baik pada teks hukum. Mari kita ingat ini ketika kita beralih ke bab studi kasus: tidak ada solusi yang sempurna dan cocok untuk semua dalam penambangan teks.

8.2 TEKNIK PENAMBANGAN TEKS

Selama studi kasus kami yang akan datang, kami akan mengatasi masalah klasifikasi teks: secara otomatis mengklasifikasikan teks yang tidak dikategorikan ke dalam kategori tertentu. Untuk mendapatkan dari data tekstual mentah ke tujuan akhir kami, kami memerlukan beberapa teknik penambangan data yang memerlukan informasi latar belakang agar kami dapat menggunakannya secara efektif. Konsep penting pertama dalam penambangan teks adalah "tas kata-kata".

8.2.1 Kantong kata-kata

Untuk membangun model klasifikasi kami, kami akan menggunakan pendekatan kumpulan kata. Kantong kata-kata adalah cara paling sederhana untuk menyusun data tekstual: setiap dokumen diubah menjadi vektor kata. Jika ada kata tertentu dalam vektor itu diberi label "Benar"; yang lainnya diberi label "Salah". Gambar 8.7 menunjukkan contoh sederhana dari hal ini, seandainya hanya ada dua dokumen: satu tentang acara televisi Game of Thrones dan satu lagi tentang ilmu data. Dua vektor kata bersama-sama membentuk matriks jangka dokumen. Matriks dokumen-term memegang kolom untuk setiap istilah dan baris untuk setiap dokumen. Nilai-nilainya adalah milik Anda untuk diputuskan. Dalam bab ini kita akan menggunakan binary: term is present? Benar atau salah.



Gambar 8.7 Sebuah teks diubah menjadi sekumpulan kata dengan memberi label pada setiap kata (istilah) dengan "Benar" jika ada dalam dokumen dan "Salah" jika tidak ada.

Contoh dari gambar 8.7 memberi Anda gambaran tentang data terstruktur yang kita perlukan untuk memulai analisis teks, tetapi ini sangat disederhanakan: tidak ada satu kata pun yang disaring dan tidak ada stemming (kita akan membahasnya nanti) yang diterapkan. Korpus besar dapat memiliki ribuan kata unik. Jika semua harus diberi label seperti ini tanpa pemfilteran apa pun, mudah untuk melihat bahwa kami mungkin akan mendapatkan volume data yang besar. Kumpulan kata kode biner seperti yang ditunjukkan pada gambar 8.7 hanyalah salah satu cara untuk menyusun data; ada teknik lain.

Frekuensi Term—Frekuensi Dokumen Terbalik (TF-IDF)

Formula yang terkenal untuk mengisi matriks term dokumen adalah TF-IDF atau Term Frequency dikali Inverse Document Frequency. Kantung kata biner menetapkan Benar atau Salah (istilah ada atau tidak), sementara frekuensi sederhana menghitung berapa kali istilah itu muncul. TF-IDF sedikit lebih rumit dan memperhitungkan berapa kali istilah muncul dalam

dokumen (TF). TF dapat berupa hitungan istilah sederhana, hitungan biner (Benar atau Salah), atau hitungan istilah skala logaritmik. Itu tergantung pada apa yang terbaik untuk Anda. Jika TF adalah frekuensi jangka, rumus TF adalah sebagai berikut:

$TF = f_{td}$, TF adalah frekuensi; (f) dari istilah (t) dalam dokumen (d).

Tetapi TF-IDF juga memperhitungkan semua dokumen lain karena Inverse Document Frequency. IDF memberikan gambaran tentang seberapa umum kata tersebut di seluruh korpus: semakin tinggi frekuensi dokumen semakin umum, dan semakin banyak kata umum kurang informatif. Misalnya kata “a” atau “the” sepertinya tidak memberikan informasi spesifik pada sebuah teks. Rumus IDF dengan penskalaan logaritmik adalah bentuk IDF yang paling umum digunakan:

$$IDF = \log(\frac{N}{|\{d \in D: t \in d\}|})$$

dengan N adalah jumlah total dokumen dalam korpus, dan $|\{d \in D: t \in d\}|$ menjadi jumlah dokumen (d) di mana istilah (t) muncul.

Skor TF-IDF menyatakan hal ini tentang sebuah istilah: seberapa pentingkah kata ini untuk membedakan dokumen ini dari yang lain dalam korpus? Rumus TF-IDF demikian

$$\frac{1F}{IDF} = f_{td}/\log(\frac{N}{|\{d \in D: t \in d\}|})$$

Kami tidak akan menggunakan TF-IDF, tetapi saat mengatur langkah selanjutnya dalam penambangan teks, ini harus menjadi salah satu hal pertama yang akan Anda temui. TF-IDF juga digunakan oleh Elasticsearch di belakang layar di bab 6. Ini adalah cara yang baik jika Anda ingin menggunakan TF-IDF untuk analitik teks; tinggalkan penambangan teks ke perangkat lunak khusus seperti SOLR atau Pencarian elastis dan ambil matriks dokumen/istilah untuk analitik teks dari sana.

Sebelum masuk ke kumpulan kata-kata yang sebenarnya, banyak langkah manipulasi data lainnya terjadi:

- **Tokenisasi—**Teks dipotong-potong disebut "token" atau "istilah". Token ini adalah unit informasi paling dasar yang akan Anda gunakan untuk model Anda. Istilahnya sering berupa kata-kata tetapi ini bukan keharusan. Seluruh kalimat dapat digunakan untuk analisis. Kami akan menggunakan unigram: istilah yang terdiri dari satu kata. Namun, seringkali berguna untuk menyertakan bigram (dua kata per token) atau trigram (tiga kata per token) untuk menangkap makna tambahan dan meningkatkan kinerja model Anda. Ini memang membutuhkan biaya, karena Anda membuat vektor istilah yang lebih besar dengan memasukkan bigram dan/atau trigram ke dalam persamaan.

- **Hentikan pemfilteran kata—**Setiap bahasa dilengkapi dengan kata-kata yang nilainya kecil dalam analitik teks karena sering digunakan. NLTK hadir dengan daftar pendek kata berhenti berbahasa Inggris yang dapat kami filter. Jika teks dipatok menjadi kata-kata, seringkali masuk akal untuk menghilangkan vektor kata dari kata-kata penghenti informasi rendah ini.
- **Huruf kecil—**Kata-kata dengan huruf kapital muncul di awal kalimat, yang lain karena merupakan kata benda atau kata sifat yang tepat. Kami tidak mendapatkan nilai tambah dengan membuat perbedaan itu dalam matriks istilah kami, jadi semua istilah akan diatur ke huruf kecil.

Teknik penyusunan data lainnya adalah stemming. Yang ini membutuhkan lebih banyak elaborasi.

8.2.2 Stemming dan lemmatisasi

Stemming adalah proses mengembalikan kata-kata ke bentuk akarnya; dengan cara ini Anda akan mendapatkan lebih sedikit variasi dalam data. Ini masuk akal jika kata-kata memiliki arti yang mirip tetapi ditulis berbeda karena, misalnya, satu dalam bentuk jamak. Stemming mencoba menyatukan dengan memotong bagian-bagian kata. Misalnya "pesawat" dan "pesawat" keduanya menjadi "pesawat".

Teknik lain, yang disebut lemmatisasi, memiliki tujuan yang sama tetapi melakukannya dengan cara yang lebih sensitif secara tata bahasa. Misalnya, sementara stemming dan lemmatisasi akan mengurangi "mobil" menjadi "mobil", lemmatisasi juga dapat mengembalikan kata kerja terkonjugasi ke bentuk tak terkonjugasinya seperti "are" menjadi "be". Yang mana yang Anda gunakan bergantung pada kasus Anda, dan lemmatisasi sangat diuntungkan dari Penandaan POS (Penandaan Bagian Ucapan). POS Tagging adalah proses menghubungkan label gramatikal ke setiap bagian kalimat. Anda mungkin melakukannya secara manual di sekolah sebagai latihan bahasa. Ambil kalimat “Game of Thrones adalah serial televisi.” Jika kami menerapkan Penandaan POS di atasnya, kami dapatkan:

({game":NN"},{"of":"IN"}.{"thrones":"NNS"},{"is": "VBZ"},{"a": "DTT"},{"television": "NN"},{"series": "NN"})

NN adalah kata benda, IN adalah preposisi, NNS adalah kata benda dalam bentuk jamak, VBZ adalah kata kerja orang ketiga tunggal, dan DT adalah penentu. Tabel 8.1 memiliki daftar lengkapnya.

Tabel 8.1 Daftar semua tag POS

Tag	Arti	Tag	Arti
CC	Coordinating conjunction	**CD**	Cardinal number
DT	Determiner	**EX**	Existential
FW	Foreign word	**IN**	Preposition or subordinating conjunction
JJ	Adjective	**JJR**	Adjective, comparative
JJS	Adjective, superlative	**LS**	List item marker
MD	Modal	**NN**	Noun, singular or mass

NNS	Noun, plural	**NNP**	Proper noun, singular
NNPS	Proper noun, plural	**PDT**	Predeterminer
POS	Possessive ending	**PRP**	Personal pronoun
PRP$	Possessive pronoun	**RB**	Adverb
RBR	Adverb, comparative	**RBS**	Adverb, superlative
RP	Particle	**SYM**	Symbol
UH	Interjection	**VB**	Verb, base form
VBD	Verb, past tense	**VBG**	Verb, gerund or present participle
VBN	Verb, past participle	**VBP**	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present	**WDT**	Wh-determiner
WP	Wh-pronoun	**WP$**	Possessive wh-pronoun
WRB	Wh-adverb		

Penandaan POS adalah kasus penggunaan tokenisasi kalimat daripada tokenisasi kata. Setelah Penandaan POS selesai, Anda masih dapat melanjutkan ke tokenisasi kata, tetapi Penanda POS membutuhkan seluruh kalimat. Menggabungkan Penandaan POS dan lemmatisasi cenderung memberikan data yang lebih bersih daripada hanya menggunakan stemmer. Demi kesederhanaan, kami akan tetap berpegang pada studi kasus, tetapi pertimbangkan ini sebagai kesempatan untuk menguraikan latihan.

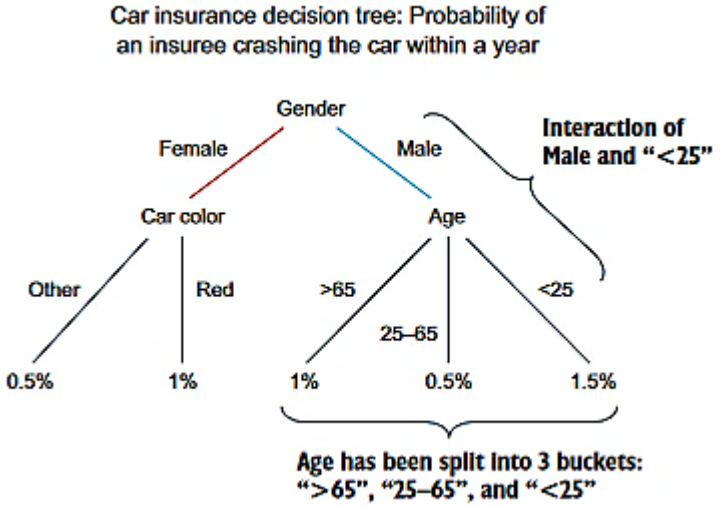
Kami sekarang mengetahui hal terpenting yang akan kami gunakan untuk melakukan pembersihan dan manipulasi data (penambangan teks). Untuk analitik teks kita, mari tambahkan pengklasifikasi pohon keputusan ke repertoar kita.

8.2.3 Pengklasifikasi pohon keputusan

Bagian analisis data dari studi kasus kami juga akan dibuat sederhana. Kami akan menguji pengklasifikasi Naïve Bayes dan pengklasifikasi pohon keputusan. Seperti yang terlihat di bab 3, pengklasifikasi Naïve Bayes disebut demikian karena menganggap setiap variabel masukan tidak bergantung satu sama lain, yang naif, terutama dalam penambangan teks. Ambil contoh sederhana dari "ilmu data", "analisis data", atau "permainan singgasana". Jika kita memotong data menjadi unigram, kita mendapatkan variabel terpisah berikut (jika kita mengabaikan stemming dan semacamnya): “data”, “sains”, “analisis”, “permainan”, “dari”, dan “tahta”. Jelas tautan akan hilang. Hal ini pada gilirannya dapat diatasi dengan membuat bigram (ilmu data, analisis data) dan trigram (permainan singgasana).

Pengklasifikasi pohon keputusan, bagaimanapun, tidak menganggap variabel independen satu sama lain dan secara aktif membuat variabel dan keranjang interaksi. Variabel interaksi adalah variabel yang menggabungkan variabel lain. Misalnya "data" dan "sains" mungkin merupakan prediktor yang baik dengan hak mereka sendiri, tetapi mungkin keduanya terjadi bersamaan dalam teks yang sama mungkin memiliki nilainya sendiri. Ember agak sebaliknya. Alih-alih menggabungkan dua variabel, sebuah variabel dipecah menjadi beberapa variabel baru. Ini masuk akal untuk variabel numerik. Gambar 8.8 menunjukkan

seperti apa bentuk pohon keputusan dan di mana Anda dapat menemukan interaksi dan pengelompokan.

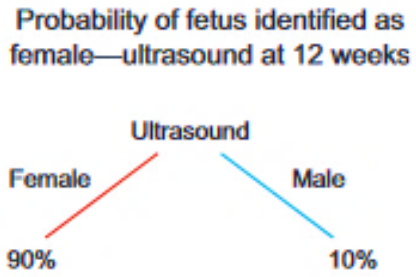


Gambar 8.8 Model pohon keputusan fiktif. Pohon keputusan secara otomatis membuat keranjang dan mengandaikan interaksi antara variabel input.

Sedangkan Naïve Bayes mengandaikan independensi semua variabel input, pohon keputusan dibangun di atas asumsi saling ketergantungan. Tapi bagaimana cara membangun struktur ini? Sebuah pohon keputusan memiliki beberapa kemungkinan kriteria yang dapat digunakan untuk membagi menjadi cabang-cabang dan memutuskan variabel mana yang lebih penting (lebih dekat ke akar pohon) daripada yang lain. Yang akan kita gunakan dalam pengklasifikasi pohon keputusan NLTK adalah "pengumpulan informasi". Untuk memahami perolehan informasi, pertama-tama kita perlu melihat entropi. Entropi adalah ukuran ketidakpastian atau kekacauan. Contoh sederhananya adalah jenis kelamin bayi. Saat wanita hamil, jenis kelamin janin bisa laki-laki atau perempuan, tapi kita tidak tahu yang mana. Jika Anda menebak, Anda memiliki peluang 50% untuk menebak dengan benar (memberi atau menerima, karena distribusi gender tidak 100% seragam). Namun, selama masa kehamilan Anda berkesempatan untuk melakukan USG untuk mengetahui jenis kelamin janin. Ultrasonografi tidak pernah 100% konklusif, tetapi semakin jauh perkembangan janin, semakin akurat jadinya. Perolehan akurasi ini, atau perolehan informasi, ada karena ketidakpastian atau penurunan entropi. Katakanlah USG pada usia kehamilan 12 minggu memiliki akurasi 90% dalam menentukan jenis kelamin bayi. Ketidakpastian 10% masih ada, tetapi USG memang mengurangi ketidakpastian dari 50% menjadi 10%. Itu pembeda yang cukup bagus. Pohon keputusan mengikuti prinsip yang sama, seperti yang ditunjukkan pada gambar 8.9.

Jika tes gender lain memiliki kekuatan prediktif yang lebih besar, itu bisa menjadi akar dari pohon dengan tes ultrasound berada di cabang, dan ini dapat berlanjut hingga kita kehabisan variabel atau pengamatan. Kita bisa kehabisan observasi, karena pada setiap cabang split kita juga membagi input data. Ini adalah kelemahan besar dari pohon keputusan, karena pada tingkat daun kekokohan pohon rusak jika terlalu sedikit pengamatan yang tersisa;

pohon keputusan mulai menyesuaikan data. Overfitting memungkinkan model untuk salah mengira keacakan untuk korelasi nyata. Untuk mengatasi hal ini, sebuah pohon keputusan dipangkas: cabang-cabangnya yang tidak berarti dikeluarkan dari model akhir.
Sekarang setelah kita melihat teknik baru yang paling penting, mari selami studi kasus.



Gambar 8.9 Pohon keputusan dengan satu variabel: kesimpulan dokter dari pemeriksaan USG saat hamil. Berapa peluang janin tersebut berjenis kelamin perempuan?

8.3 STUDI KASUS: MENGKLASIFIKASIKAN KIRIMAN REDDIT

Sementara text mining memiliki banyak aplikasi, dalam studi kasus bab ini kita fokus pada klasifikasi dokumen. Seperti yang ditunjukkan sebelumnya di bab ini, inilah yang Google lakukan saat mengatur email Anda dalam kategori atau mencoba membedakan spam dari email biasa. Ini juga banyak digunakan oleh pusat kontak yang memproses pertanyaan atau keluhan pelanggan yang masuk: keluhan tertulis terlebih dahulu melewati filter deteksi topik sehingga dapat ditugaskan ke orang yang tepat untuk ditangani. Klasifikasi dokumen juga merupakan salah satu fitur wajib dari sistem pemantauan media sosial. Kicauan yang dipantau, posting forum atau Facebook, artikel surat kabar, dan banyak sumber daya internet lainnya diberi label topik. Dengan cara ini mereka dapat digunakan kembali dalam laporan. Analisis sentimen adalah jenis klasifikasi teks tertentu: apakah penulis posting negatif, positif, atau netral pada sesuatu? “Sesuatu” itu dapat dikenali dengan pengenalan entitas.

Dalam studi kasus ini, kami akan menggunakan postingan dari Reddit, sebuah situs web yang juga dikenal sebagai “halaman depan internet” yang diproklamirkan sendiri, dan berupaya melatih model yang mampu membedakan apakah seseorang berbicara tentang “ilmu data” atau "permainan singgasana".

Hasil akhirnya bisa berupa presentasi model kita atau aplikasi interaktif lengkap. Di bab 9 kita akan fokus pada pembuatan aplikasi untuk pengguna akhir, jadi untuk saat ini kita akan tetap menyajikan model klasifikasi kita. Untuk mencapai tujuan kami, kami akan membutuhkan semua bantuan dan alat yang bisa kami dapatkan, dan itu terjadi Python sekali lagi siap untuk menyediakannya.

8.3.1 Mengenal Perangkat Bahasa Alami

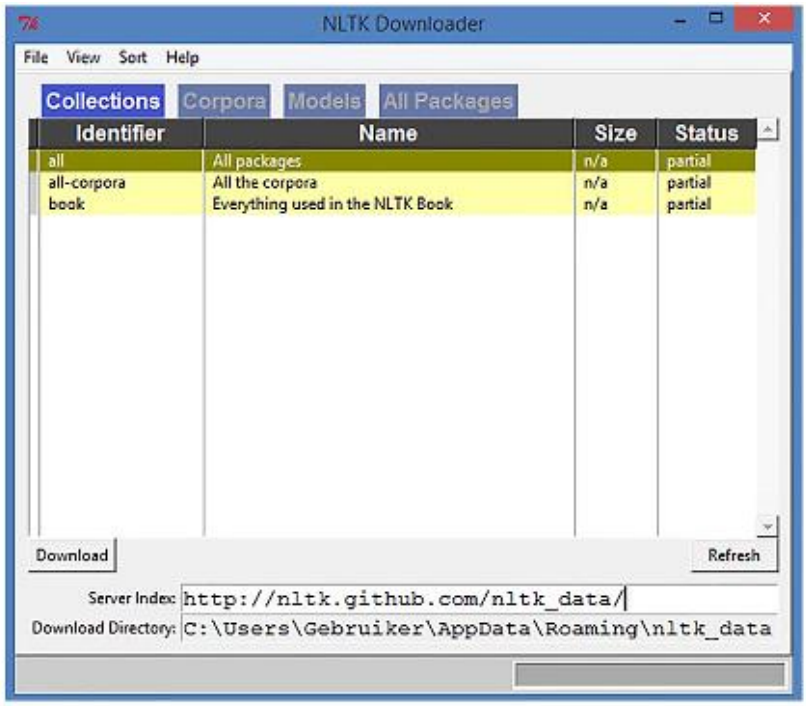
Python mungkin bukan bahasa yang paling efisien eksekusi di dunia, tetapi ia memiliki paket matang untuk penambangan teks dan pemrosesan bahasa: Natural Language Toolkit (NLTK). NLTK adalah kumpulan algoritme, fungsi, dan karya beranotasi yang akan memandu Anda mengambil langkah pertama dalam penambangan teks dan pemrosesan bahasa alami. NLTK juga didokumentasikan dengan sangat baik di nltk.org. NLTK, bagaimanapun, tidak

sering digunakan untuk pekerjaan tingkat produksi, seperti perpustakaan lain seperti scikit-learn.

Menginstal NLTK dan kumpulannya

Instal NLTK dengan penginstal paket favorit Anda. Jika Anda menggunakan Anaconda, itu sudah terinstal dengan pengaturan Anaconda default. Kalau tidak, Anda bisa menggunakan "pip" atau "easy_install". Ketika ini selesai, Anda masih perlu menginstal model dan corpora yang disertakan agar berfungsi penuh. Untuk ini, jalankan kode Python berikut:

- impor nltk
- nltk.download()



Gambar 8.10 Pilih Semua Paket untuk menyelesaikan instalasi NLTK sepenuhnya.

Bergantung pada instalasi Anda, ini akan memberi Anda pop-up atau lebih banyak opsi baris perintah. Gambar 8.10 menunjukkan kotak pop-up yang Anda dapatkan saat mengeluarkan perintah nltk.download(). Anda dapat mengunduh semua corpora jika Anda mau, tetapi untuk bab ini kami hanya akan menggunakan "punkt" dan "stopwords". Unduhan ini akan disebutkan secara eksplisit dalam kode yang disertakan dengan buku ini.

Dua file notebook IPython tersedia untuk bab ini:

- **Pengumpulan data**—Berisi bagian pengumpulan data dari studi kasus bab ini.
- **Persiapan dan analisis data**—Data yang disimpan dimasukkan ke dalam persiapan data dan kemudian dianalisis.

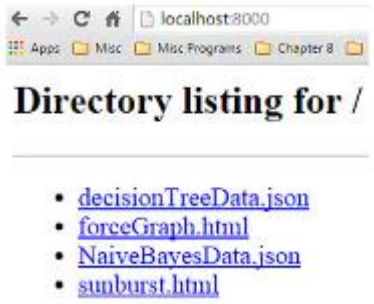
Semua kode dalam studi kasus yang akan datang dapat ditemukan di dua file ini dalam urutan yang sama dan juga dapat dijalankan seperti itu. Selain itu, dua grafik interaktif tersedia untuk diunduh:

- **forceGraph.html—**Mewakili 20 fitur teratas dari model Naïve Bayes kami

- **Sunburst.html —** Mewakili empat cabang teratas dari model pohon keputusan kami

Untuk membuka dua halaman HTML ini, diperlukan server HTTP, yang bisa Anda dapatkan menggunakan Python dan jendela perintah:

- Buka jendela perintah (Linux, Windows, apa pun yang Anda suka).
- Pindah ke folder yang berisi file HTML dan file data JSON: decisionTreeData.json untuk diagram sunburst dan NaiveBayesData.json untuk grafik gaya. File HTML harus tetap berada di lokasi yang sama dengan file datanya atau Anda harus mengubah JavaScript di file HTML.
- Buat server HTTP Python dengan perintah berikut: python –m Simple- HTTPServer 8000
- Buka browser dan buka localhost:8000; di sini Anda dapat memilih file HTML, seperti yang ditunjukkan pada gambar 8.11.



Gambar 8.11 Server HTTP Python menyajikan keluaran bab ini

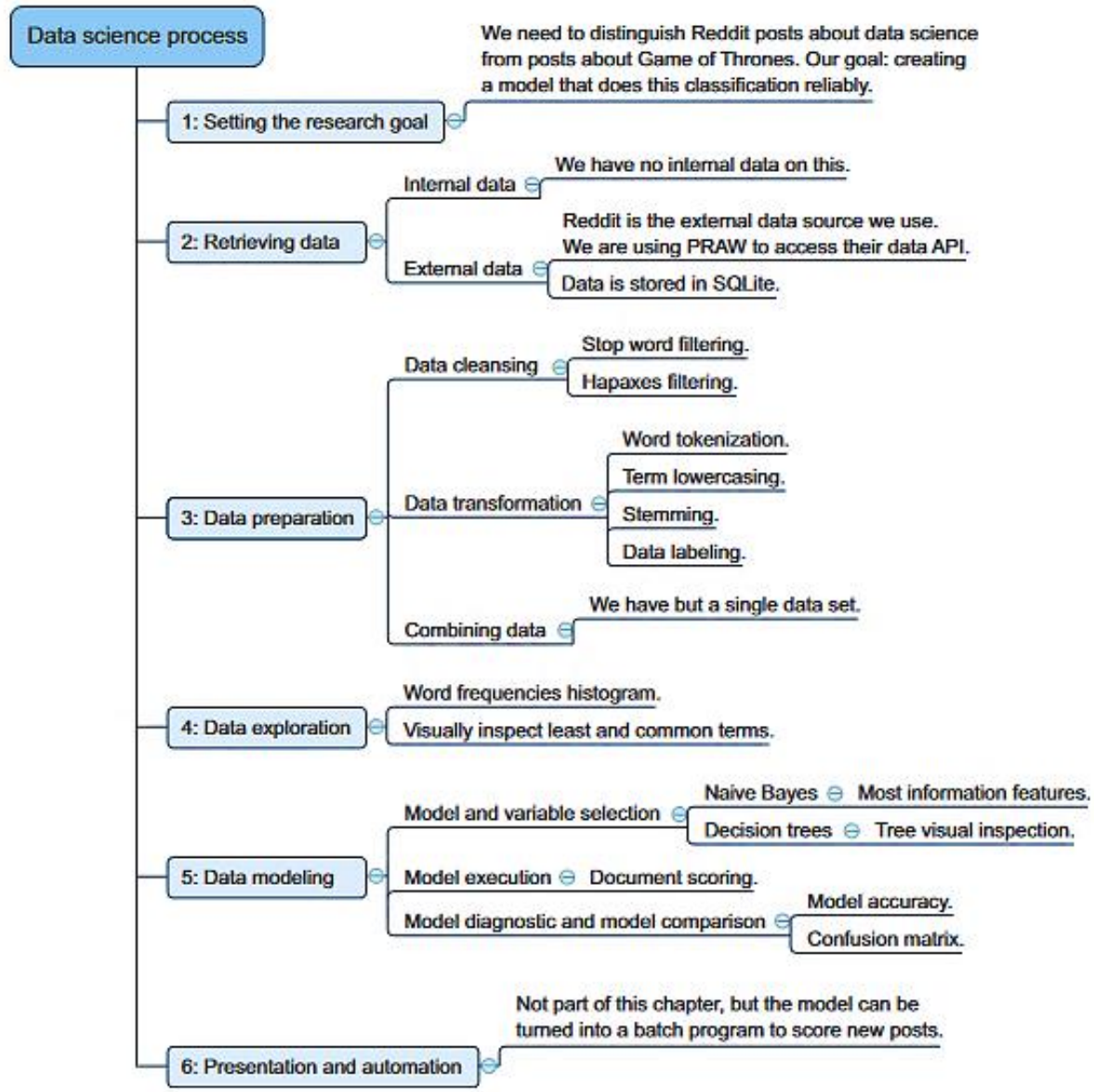
Paket Python yang akan kita gunakan di bab ini:

- **NLTK—** Untuk penambangan teks
- **PRAW—** Memungkinkan mengunduh postingan dari Reddit
- **SQLite3—** Memungkinkan kita untuk menyimpan data dalam format SQLite
- **Matplotlib—** Pustaka plot untuk memvisualisasikan data

Pastikan untuk menginstal semua pustaka dan kumpulan yang diperlukan sebelum melanjutkan. Namun, sebelum kita terjun ke tindakan, mari kita lihat langkah-langkah yang akan kita ambil untuk mencapai tujuan membuat model klasifikasi topik.

8.3.2 Ikhtisar proses ilmu data dan langkah 1: Tujuan penelitian

Untuk menyelesaikan latihan penambangan teks ini, sekali lagi kami akan menggunakan proses ilmu data. Gambar 8.12 menunjukkan proses ilmu data yang diterapkan pada kasus klasifikasi Reddit kami. Tidak semua elemen yang digambarkan pada gambar 8.12 mungkin masuk akal pada saat ini, dan sisa bab ini didedikasikan untuk menyelesaikannya dalam praktik saat kami bekerja menuju tujuan penelitian kami: membuat model klasifikasi yang mampu membedakan tulisan tentang "ilmu data" dari posting tentang "Game of Thrones." Tanpa basa-basi lagi, ayo ambil data kita.



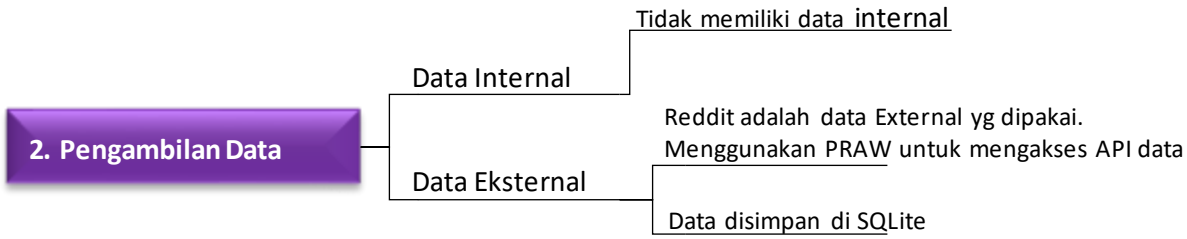
Gambar 8.12 Ikhtisar proses sains data yang diterapkan pada studi kasus klasifikasi topik Reddit

8.3.3 Langkah 2: Pengambilan data

Kami akan menggunakan data Reddit untuk kasus ini, dan bagi mereka yang tidak terbiasa dengan Reddit, luangkan waktu untuk memahami konsepnya di www.reddit.com. Reddit menyebut dirinya "halaman depan internet" karena pengguna dapat memposting hal-hal yang menurut mereka menarik dan/atau ditemukan di suatu tempat di internet, dan hanya hal-hal yang dianggap menarik oleh banyak orang yang ditampilkan sebagai "populer" di berandanya. Bisa dibilang Reddit memberikan gambaran mengenai hal-hal yang sedang trending di internet. Setiap pengguna dapat memposting dalam kategori yang telah ditentukan sebelumnya yang disebut "subreddit". Saat sebuah postingan dibuat, pengguna lain dapat mengomentarinya dan dapat memberikan suara positif jika mereka menyukai konten tersebut atau memberikan suara negatif jika mereka tidak menyukainya. Karena sebuah kiriman selalu menjadi bagian dari subreddit, kami memiliki metadata ini saat kami

terhubung ke API Reddit untuk mendapatkan data kami. Kami secara efektif mengambil data berlabel karena kami menganggap bahwa postingan di subreddit "gameofthrones" ada hubungannya dengan "gameofthrones".

Untuk mendapatkan data kami, kami menggunakan pustaka Reddit Python API resmi yang disebut PRAW. Setelah kami mendapatkan data yang kami butuhkan, kami akan menyimpannya dalam file mirip database ringan yang disebut SQLite. SQLite sangat ideal untuk menyimpan data dalam jumlah kecil karena tidak memerlukan penyiapan apa pun untuk digunakan dan akan merespons kueri SQL seperti yang dilakukan database relasional biasa. Media penyimpanan data lainnya bisa digunakan; jika Anda lebih suka database Oracle atau Postgres, Python memiliki pustaka yang sangat bagus untuk berinteraksi dengannya tanpa perlu menulis SQL. SQLAlchemy juga akan berfungsi untuk file SQLite. Gambar 8.13 menunjukkan langkah pengambilan data dalam proses ilmu data.



Gambar 8.13 Langkah pengambilan data proses sains data untuk kasus klasifikasi topik Reddit

Buka juru bahasa Python favorit Anda; saatnya beraksi, seperti yang ditunjukkan pada daftar 8.1. Pertama, kami perlu mengumpulkan data kami dari situs web Reddit. Jika Anda belum melakukannya, gunakan pip install praw atau conda install praw (Anaconda) sebelum menjalankan skrip berikut.

CATATAN Kode untuk langkah 2 juga dapat ditemukan di file IPython "Pengumpulan data Bab 8." Ini tersedia di bagian unduhan buku ini.

Pertama-tama mari impor pustaka yang diperlukan. Sekarang kita memiliki akses ke kemampuan SQLite dan PRAW, kita perlu menyiapkan database lokal kecil kita untuk data yang akan diterima. Dengan menentukan koneksi ke file SQLite, kami secara otomatis membuatnya jika belum ada. Kami kemudian menentukan kursor data yang mampu mengeksekusi pernyataan SQL apa pun, jadi kami menggunakannya untuk menentukan struktur database kami. Basis data akan berisi dua tabel: tabel topik berisi topik Reddit, yang mirip dengan seseorang yang memulai posting baru di forum, dan tabel kedua berisi komentar dan ditautkan ke tabel topik melalui kolom "topicID". Kedua tabel tersebut memiliki hubungan satu (tabel topik) ke banyak (tabel komentar). Untuk studi kasus, kami akan membatasi diri untuk menggunakan tabel topik, tetapi pengumpulan data akan menggabungkan keduanya karena ini memungkinkan Anda untuk bereksperimen dengan data tambahan ini jika Anda menginginkannya. Untuk mengasah keterampilan penambangan teks, Anda dapat melakukan

analisis sentimen pada komentar topik dan mencari tahu topik apa yang menerima komentar negatif atau positif. Anda kemudian dapat mengkorelasikannya dengan fitur model yang akan kami hasilkan di akhir bab ini.

Listing 8.1 Setting up SQLLite database and Reddit API client

```
import praw
import sqlite3

conn = sqlite3.connect('reddit.db')
c = conn.cursor()

c.execute('''DROP TABLE IF EXISTS topics''')
c.execute('''DROP TABLE IF EXISTS comments''')
c.execute('''CREATE TABLE topics
             (topicTitle text, topicText text, topicID text,
topicCategory text)''')
c.execute('''CREATE TABLE comments
             (commentText text, commentID text ,
topicTitle text, topicText text, topicID text ,
 topicCategory text)''')

user_agent = "Introducing Data Science Book"
r = praw.Reddit(user_agent=user_agent)

subreddits = ['datascience','gameofthrones']

limit = 1000
```

Import PRAW and SQLite3 libraries.

Set up connection to SQLite database.

Execute SQL statements to create topics and comments table.

Create PRAW user agent so we can use Reddit API.

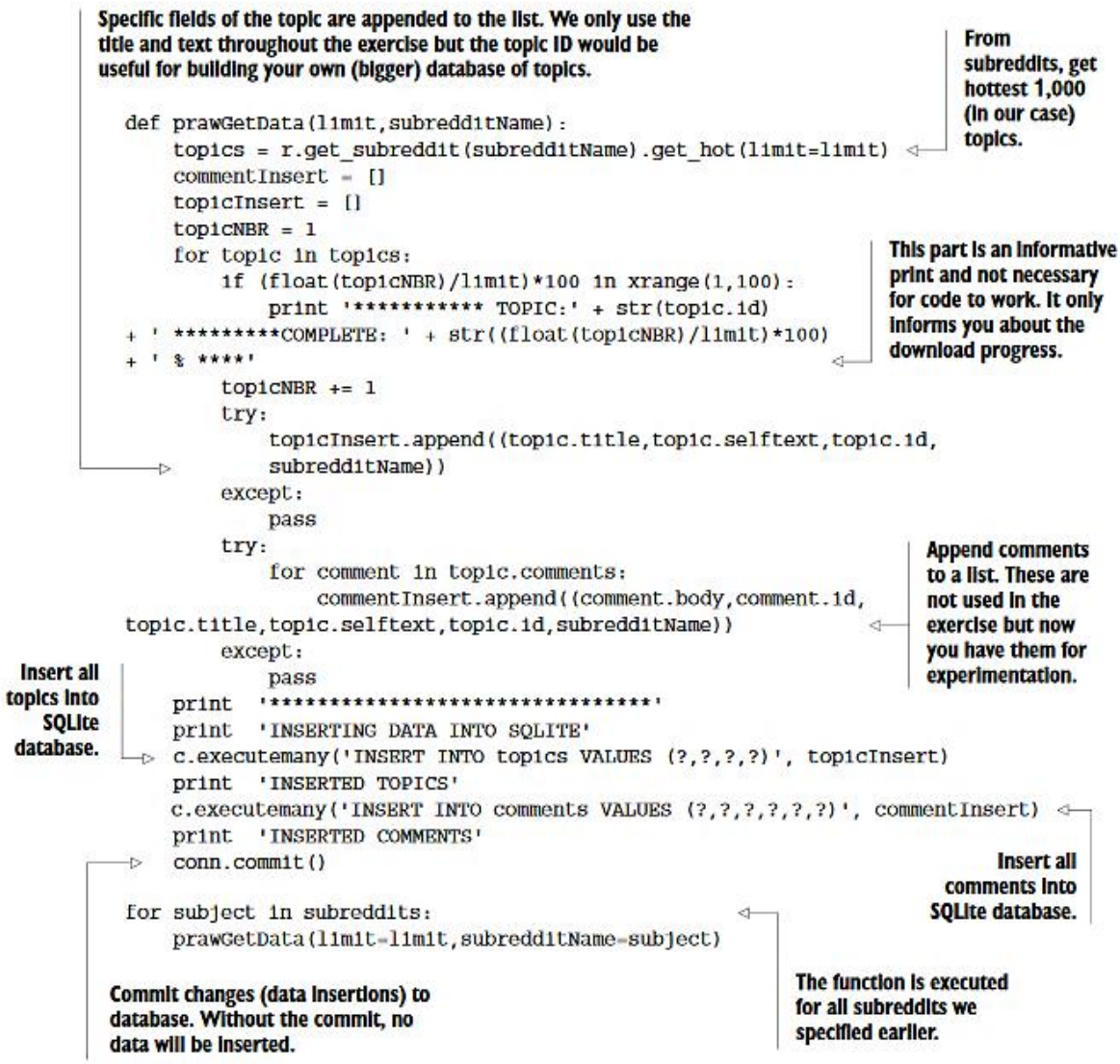
Our list of subreddits we'll draw into our SQLite database.

Maximum number of posts we'll fetch from Reddit per category. Maximum Reddit allows at any single time is also 1,000.

Kita perlu membuat klien PRAW untuk mendapatkan akses ke data. Setiap subreddit dapat dikenali dari namanya, dan kami tertarik pada "ilmu data" dan "gameofthrones". Batas tersebut mewakili jumlah maksimum topik (postingan, bukan komentar) yang akan kami ambil dari Reddit. Seribu juga merupakan jumlah maksimum yang diizinkan API untuk kami ambil pada setiap permintaan yang diberikan, meskipun kami dapat meminta lebih banyak nanti ketika orang telah memposting hal baru. Bahkan kita bisa menjalankan permintaan API secara berkala dan mengumpulkan data dari waktu ke waktu. Meskipun pada waktu tertentu Anda dibatasi untuk seribu posting, tidak ada yang menghentikan Anda untuk mengembangkan database Anda sendiri selama berbulan-bulan. Perlu dicatat bahwa skrip berikut mungkin membutuhkan waktu sekitar satu jam untuk diselesaikan. Jika Anda tidak ingin menunggu, silakan lanjutkan dan gunakan file SQLite yang dapat diunduh. Selain itu, jika Anda menjalankannya sekarang, kemungkinan besar Anda tidak akan mendapatkan keluaran yang sama persis seperti saat pertama kali dijalankan untuk membuat keluaran yang ditampilkan di bab ini.

Mari kita lihat fungsi pengambilan data kami, seperti yang ditunjukkan pada daftar berikut.

Listing 8.2 Reddit data retrieval and storage in SQLite



Fungsi prawGetData() mengambil topik "terhangat" di subredditnya, menambahkannya ke larik, lalu mendapatkan semua komentar terkaitnya. Ini berlangsung sampai seribu topik tercapai atau tidak ada lagi topik untuk diambil dan semuanya disimpan dalam database SQLite. Pernyataan cetak ada untuk memberi tahu Anda tentang kemajuannya dalam mengumpulkan seribu topik. Yang perlu kita lakukan hanyalah menjalankan fungsi untuk setiap subreddit. Jika Anda ingin analisis ini menyertakan lebih dari dua subreddit, ini adalah masalah menambahkan kategori tambahan ke larik subreddit. Dengan data yang dikumpulkan, kami siap untuk beralih ke persiapan data.

8.3.4 Langkah 3: Persiapan data

Seperti biasa, persiapan data adalah langkah paling penting untuk mendapatkan hasil yang benar. Untuk penambangan teks, ini bahkan lebih benar karena kita bahkan tidak memulai dengan data terstruktur.

Kode yang akan datang tersedia online sebagai file IPython "Persiapan dan analisis data Bab 8." Mari kita mulai dengan mengimpor pustaka yang diperlukan dan menyiapkan database SQLite, seperti yang ditunjukkan pada daftar berikut.

Listing 8.3 Text mining, libraries, corpora dependencies, and SQLite database connection

```
import sqlite3
import nltk
import matplotlib.pyplot as plt
from collections import OrderedDict
import random

nltk.download('punkt')
nltk.download('stopwords')

conn = sqlite3.connect('reddit.db')
c = conn.cursor()
```

Import all required libraries

Download corpora we make use of

Make a connection to SQLite database that contains our Reddit data

Jika Anda belum mengunduh korpus NLTK lengkap, sekarang kami akan mengunduh bagian yang akan kami gunakan. Jangan khawatir jika Anda sudah mengunduhnya, skrip akan mendeteksi jika corpora Anda mutakhir. Data kita masih tersimpan di file Reddit SQLite jadi mari buat koneksi ke sana. Bahkan sebelum menjelajahi data kami, kami mengetahui setidaknya dua hal yang harus kami lakukan untuk membersihkan data: hentikan pemfilteran kata dan huruf kecil. Fungsi filter kata umum akan membantu kita menyaring bagian yang tidak bersih. Mari kita buat satu di daftar berikut.

Listing 8.4 Word filtering and lowercasing functions

```
def wordFilter(excluded,wordrow):
    filtered = [word for word in wordrow if word not in excluded]
    return filtered
stopwords = nltk.corpus.stopwords.words('english')
def lowerCaseArray(wordrow):
    lowercased = [word.lower() for word in wordrow]
    return lowercased
```

wordFilter() function will remove a term from an array of terms

Stop word variable contains English stop words per default present in NLTK

lowerCaseArray() function transforms any term to its lowercased version

Kata-kata berhenti bahasa Inggris akan menjadi yang pertama meninggalkan data kami. Kode berikut akan memberi kita kata-kata berhenti ini:

```
stopwords = nltk.corpus.stopwords.words('english')
print stopwords
```

Gambar 8.14 menunjukkan daftar stop word bahasa Inggris di NLTK.

```
stopwords = nltk.corpus.stopwords.words('english')
print stopwords
```

```
[u'i', u'me', u'my', u'myself', u'we', u'our', u'ours', u'ourselves', u'you',
u'your', u'yours', u'yourself', u'yourselves', u'he', u'him', u'his', u'himsel
f', u'she', u'her', u'hers', u'herself', u'it', u'its', u'itself', u'they', u'th
em', u'their', u'theirs', u'themselves', u'what', u'which', u'who', u'whom', u't
his', u'that', u'these', u'those', u'am', u'is', u'are', u'was', u'were', u'be',
u'been', u'being', u'have', u'has', u'had', u'having', u'do', u'does', u'did',
u'doing', u'a', u'an', u'the', u'and', u'but', u'if', u'or', u'because', u'as',
u'until', u'while', u'of', u'at', u'by', u'for', u'with', u'about', u'against',
u'between', u'into', u'through', u'during', u'before', u'after', u'above', u'bel
ow', u'to', u'from', u'up', u'down', u'in', u'out', u'on', u'off', u'over', u'un
der', u'again', u'further', u'then', u'once', u'here', u'there', u'when', u'wher
e', u'why', u'how', u'all', u'any', u'both', u'each', u'few', u'more', u'most',
u'other', u'some', u'such', u'no', u'nor', u'not', u'only', u'own', u'same', u's
o', u'than', u'too', u'very', u's', u't', u'can', u'will', u'just', u'don', u'sh
ould', u'now']
```

Gambar 8.14 Daftar stop word bahasa Inggris di NLTK

Dengan semua komponen yang diperlukan, mari kita lihat fungsi pemrosesan data pertama kami dalam daftar berikut.

Listing 8.5 First data preparation function and execution

We'll use data['all_words'] for data exploration.

```
def data_processing(sql):
    c.execute(sql)
    data = {'wordMatrix':[],'all_words':[]}
    row = c.fetchone()
    while row is not None:
        wordrow = nltk.tokenize.word_tokenize(row[0]+" "+row[1])
        wordrow_lowercased = lowerCaseArray(wordrow)
        wordrow_nostopwords = wordFilter(stopwords,wordrow_lowercased)
        data['all_words'].extend(wordrow_nostopwords)
        data['wordMatrix'].append(wordrow_nostopwords)
        row = c.fetchone()
    return data

subreddits = ['datascience','gameofthrones']
data = {}
for subject in subreddits:
    data[subject] = data_processing(sql='''SELECT
     topicTitle,topicText,topicCategory FROM topics
WHERE topicCategory = '''+"'"+subject+"'")
```

Create pointer to AWLite data.

Fetch data row by row.

row[0] is title, row[1] is topic text; we turn them into a single text blob.

Get new document from SQLite database.

Our subreddits as defined earlier.

Call data processing function for every subreddit.

data['wordMatrix'] is a matrix comprised of word vectors; 1 vector per document.

Fungsi data_processing() kita menerima pernyataan SQL dan mengembalikan matriks jangka dokumen. Ini dilakukan dengan mengulangi data satu entri (topik Reddit) pada satu waktu dan menggabungkan judul topik dan teks isi topik menjadi satu vektor kata dengan menggunakan tokenisasi kata. Tokenizer adalah skrip penanganan teks yang memotong teks menjadi beberapa bagian. Anda memiliki banyak cara berbeda untuk menandai teks: Anda

dapat membaginya menjadi kalimat atau kata, Anda dapat membaginya dengan spasi dan tanda baca, atau Anda dapat mempertimbangkan karakter lain, dan seterusnya. Di sini kami memilih tokenizer kata NLTK standar. Tokenizer kata ini sederhana; yang dilakukannya hanyalah membagi teks menjadi beberapa istilah jika ada spasi di antara kata-kata tersebut. Kami kemudian huruf kecil vektor dan memfilter kata-kata berhenti. Perhatikan bagaimana urutan itu penting di sini; kata berhenti di awal kalimat tidak akan difilter jika kita terlebih dahulu memfilter kata berhenti sebelum huruf kecil. Misalnya dalam "Saya suka Game of Thrones", "Saya" tidak akan ditulis dengan huruf kecil dan karenanya tidak akan disaring. Kami kemudian membuat matriks kata (term-document matrix) dan daftar yang berisi semua kata. Perhatikan bagaimana kami memperluas daftar tanpa memfilter ganda; dengan cara ini kita dapat membuat histogram pada kemunculan kata selama eksplorasi data. Mari jalankan fungsi untuk dua kategori topik kita. Gambar 8.15 menunjukkan vektor kata pertama dari kategori “datascience”.

```
print data['datascience']['wordMatrix'][0]
```

```
print data['datascience']['wordMatrix'][0]

[u'data', u'science', u'freelancing', u"'m", u'currently', u'master
s', u'program', u'studying', u'business', u'analytics', u"'m", u'try
ing', u'get', u'data', u'freelancing', u'.', u"'m", u'still', u'lear
ning', u'skill', u'set', u'typically', u'see', u'right', u"'m", u'fa
irly', u'proficient', u'sql', u'know', u'bit', u'r.', u'freelancer
s', u'find', u'jobs', u'?']
```

Gambar 8.15 Vektor kata pertama dari kategori “datascience” setelah percobaan pengolahan data pertama

Ini pasti terlihat tercemar: tanda baca disimpan sebagai istilah terpisah dan beberapa kata bahkan belum dipisah. Eksplorasi data lebih lanjut harus mengklarifikasi beberapa hal untuk kita.

8.3.5 Langkah 4: Eksplorasi data

Kami sekarang memiliki semua persyaratan kami terpisah, tetapi ukuran data yang tipis menghalangi kami untuk memahami dengan baik apakah itu cukup bersih untuk penggunaan sebenarnya. Dengan melihat satu vektor, kita sudah menemukan beberapa masalah: beberapa kata belum dipisahkan dengan benar dan vektor berisi banyak istilah karakter tunggal. Istilah karakter tunggal mungkin merupakan pembeda topik yang baik dalam kasus tertentu. Misalnya, teks ekonomi akan memuat lebih banyak tanda $, £, dan ¤ daripada teks medis. Tetapi dalam kebanyakan kasus istilah satu karakter ini tidak berguna. Pertama, mari kita lihat distribusi frekuensi istilah kita.

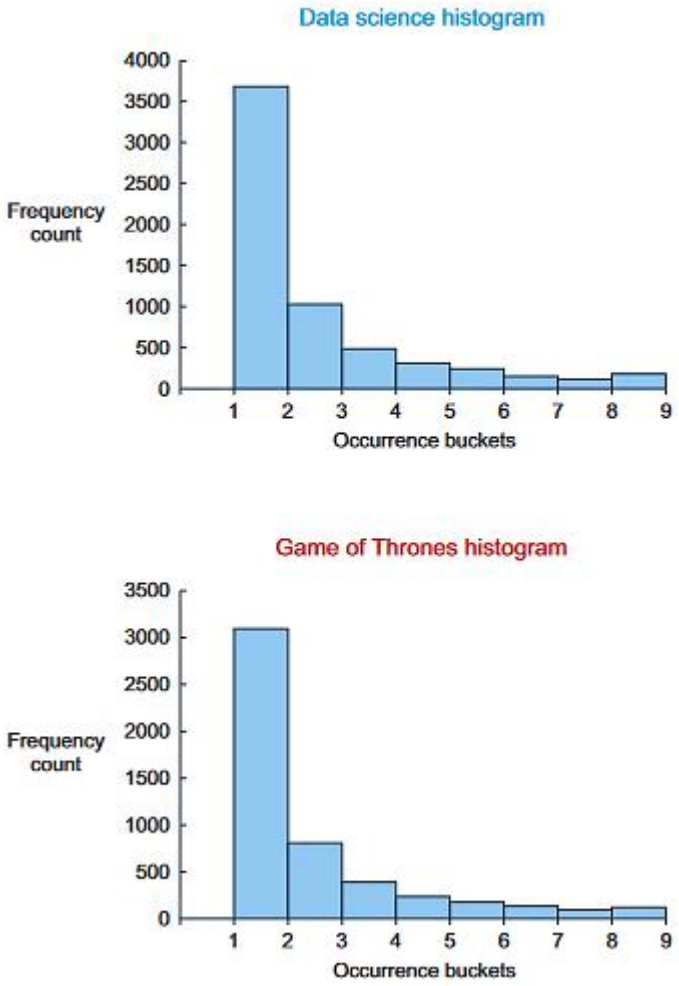
```
wordfreqs_cat1 = nltk.FreqDist(data['datascience']['all_words'])
plt.hist(wordfreqs_cat1.values(), bins = range(10))
plt.show()
wordfreqs_cat2 = nltk.FreqDist(data['gameofthrones']['all_words'])
plt.hist(wordfreqs_cat2.values(), bins = range(20))
plt.show()
```

Dengan menggambar histogram dari distribusi frekuensi (gambar 8.16) kami segera menyadari bahwa sebagian besar istilah kami hanya muncul dalam satu dokumen. Istilah kemunculan tunggal seperti ini disebut hapax, dan dari segi model istilah tersebut tidak berguna karena satu kemunculan fitur tidak pernah cukup untuk membuat model yang andal. Ini adalah kabar baik bagi kita; memotong hapax ini akan secara signifikan mengecilkan data kami tanpa merusak model akhir kami. Mari kita lihat beberapa istilah kemunculan tunggal ini.

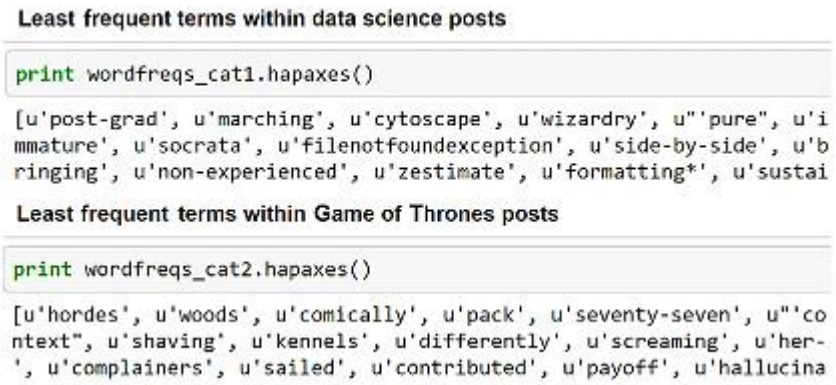
```
print wordfreqs_cat1.hapaxes()
print wordfreqs_cat2.hapaxes()
```

Istilah yang kami lihat di gambar 8.17 masuk akal, dan jika kami memiliki lebih banyak data, istilah tersebut kemungkinan akan lebih sering muncul.

```
print wordfreqs_cat1.hapaxes()
print wordfreqs_cat2.hapaxes()
```



Gambar 8.16 Histogram frekuensi istilah ini menunjukkan matriks istilah "ilmu data" dan "permainan singgasana" memiliki lebih dari 3.000 istilah yang muncul sekali.



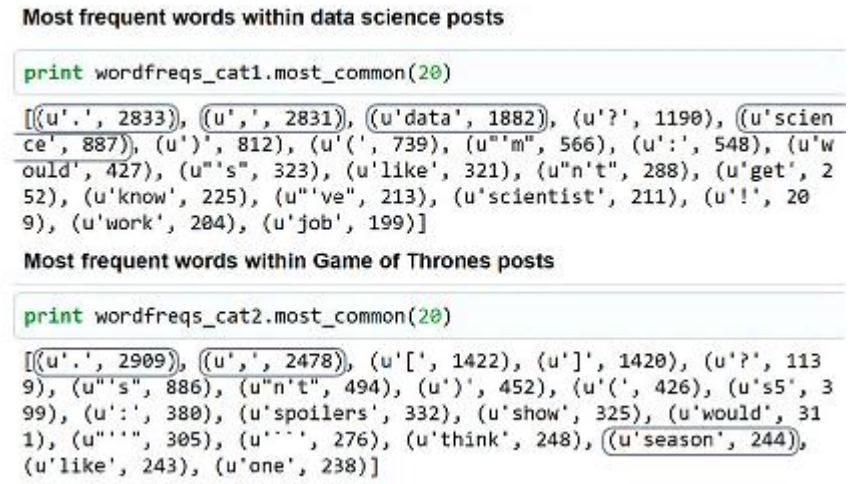
Gambar 8.17 Istilah kemunculan tunggal "ilmu data" dan "permainan singgasana" (hapaxes)

Banyak dari istilah ini adalah ejaan yang salah dari yang berguna, seperti: Jaimie adalah Jaime (Lannister), Milisandre adalah Melisandre, dan seterusnya. Tesaurus khusus Game of Thrones yang layak dapat membantu kami menemukan dan mengganti kesalahan ejaan ini dengan algoritme pencarian fuzzy. Ini membuktikan pembersihan data dalam penambangan teks dapat berlangsung tanpa batas waktu jika Anda menginginkannya; menjaga upaya dan hasil seimbang sangat penting di sini.

Sekarang mari kita lihat kata-kata yang paling sering.

```
print wordfreqs_cat1.most_common(20)
print wordfreqs_cat2.most_common(20)
```

Gambar 8.18 menunjukkan hasil meminta 20 kata paling umum teratas untuk setiap kategori.



Gambar 8.18 Top 20 kata paling sering untuk postingan "data science" dan "game of thrones".

Sekarang ini terlihat menggembirakan: beberapa kata umum memang tampak spesifik untuk topiknya. Kata-kata seperti "data", "sains", dan "musim" cenderung menjadi pembeda yang baik. Hal penting lainnya yang perlu diperhatikan adalah banyaknya istilah karakter

tunggal seperti “.” Dan ","; kami akan menyingkirkan ini. Dengan pengetahuan tambahan ini, mari kita merevisi skrip persiapan data kita.

8.3.6 Langkah 3 ditinjau kembali: Persiapan data diadaptasi

Eksplorasi data singkat ini telah menarik perhatian kita pada beberapa perubahan nyata yang dapat kita lakukan untuk menyempurnakan teks kita. Satu lagi yang penting adalah membendung istilah.

Daftar berikut menunjukkan algoritma stemming sederhana yang disebut “snowball stemming.” Stemmer bola salju ini bisa bahasa tertentu, jadi kami akan menggunakan bahasa Inggris; namun, ini mendukung banyak bahasa.

Listing 8.6 The Reddit data processing revised after data exploration

```
stemmer = nltk.SnowballStemmer("english")
def wordStemmer(wordrow):
    stemmed = [stemmer.stem(word) for word in wordrow]
    return stemmed

manual_stopwords = [',','.',')',',','(','m',"'m","n't",'e.g',"'ve",'s','#','/
    ','``',"'s","'",'!','r',']','=','[','s','&','%','*','...','1','2','3','
    4','5','6','7','8','9','10','--',"'",';','-',':']

def data_processing(sql,manual_stopwords):
    #create pointer to the sqlite data
    c.execute(sql)
    data = {'wordMatrix':[],'all_words':[]}
    interWordMatrix = []
    interWordList = []

    row = c.fetchone()
    while row is not None:
        tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+|[^\w\s]+')

        wordrow = tokenizer.tokenize(row[0]+" "+row[1])
        wordrow_lowercased = lowerCaseArray(wordrow)
        wordrow_nostopwords = wordFilter(stopwords,wordrow_lowercased)

        wordrow_nostopwords =
            wordFilter(manual_stopwords,wordrow_nostopwords)
        wordrow_stemmed = wordStemmer(wordrow_nostopwords)

        interWordList.extend(wordrow_stemmed)
        interWordMatrix.append(wordrow_stemmed)

        row = c.fetchone()

    wordfreqs = nltk.FreqDist(interWordList) s
    hapaxes = wordfreqs.hapaxes()
    for wordvector in interWordMatrix:
        wordvector_nohapexes = wordFilter(hapaxes,wordvector)
        data['wordMatrix'].append(wordvector_nohapexes)
        data['all_words'].extend(wordvector_nohapexes)

    return data

for subject in subreddits:
    data[subject] = data_processing(sql='''SELECT
     topicTitle,topicText,topicCategory FROM topics
WHERE topicCategory = '''+"'"+subject+"'",
manual_stopwords=manual_stopwords)
```

Stop words array defines terms to remove/ignore.

Now we define our revised data preparation.

Fetch data (reddit posts) one by one from SQLite database.

row[0] and row[1] contain the title and text of the post, respectively. We combine them into a single text blob.

Remove manually added stop words from text blob.

Temporary word list used to remove hapaxes later on.

Temporary word matrix; will become final word matrix after hapaxes removal.

Get new topic.

Make frequency distribution of all terms.

Get list of hapaxes.

Loop through temporary word matrix.

Remove hapaxes in each word vector.

Append correct word vector to final word matrix.

Extend list of all terms with corrected word vector.

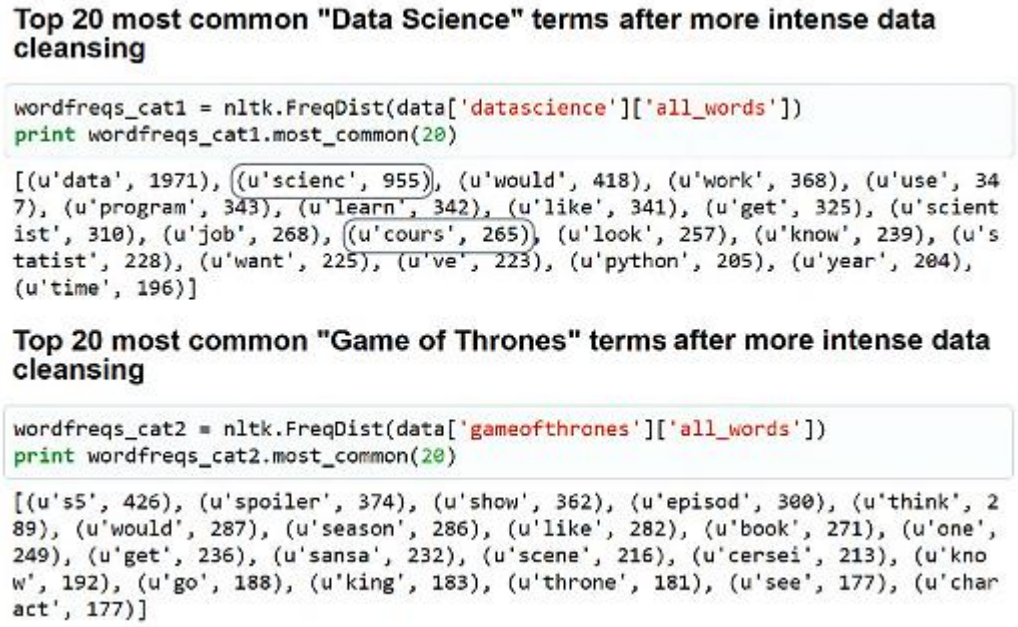
Run new data processing function for both subreddits.

Perhatikan perubahan sejak fungsi data_processing() terakhir. Tokenizer kami sekarang menjadi tokenizer ekspresi reguler. Ekspresi reguler bukan bagian dari buku ini dan sering dianggap menantang untuk dikuasai, tetapi yang dilakukan sederhana ini hanyalah memotong teks menjadi kata-kata. Untuk kata-kata, kombinasi alfanumerik diperbolehkan

(\w), jadi tidak ada lagi karakter khusus atau tanda baca. Kami juga menerapkan kata stemmer dan menghapus daftar kata berhenti tambahan. Dan, semua hapax dihilangkan di bagian akhir karena semuanya harus di-stem terlebih dahulu. Mari jalankan persiapan data kita lagi. Jika kami melakukan analisis eksplorasi yang sama seperti sebelumnya, kami akan melihatnya lebih masuk akal, dan kami tidak memiliki hapax lagi.

```
print wordfreqs_cat1.hapaxes()
print wordfreqs_cat2.hapaxes()
```

Mari kita ambil lagi 20 kata teratas dari setiap kategori (lihat gambar 8.19).



Gambar 8.19 Top 20 kata yang paling sering muncul di postingan Reddit “data science” dan “game of thrones” setelah persiapan data

Kita dapat melihat pada Gambar 8.19 bagaimana kualitas data meningkat secara luar biasa. Perhatikan juga bagaimana kata-kata tertentu dipersingkat karena stemming yang kita terapkan. Misalnya, "sains" dan "sains" telah menjadi "sains;" "kursus" dan "kursus" telah menjadi "kursus", dan seterusnya. Istilah yang dihasilkan bukan kata-kata yang sebenarnya tetapi masih dapat ditafsirkan. Jika Anda bersikeras pada istilah Anda tetap kata-kata yang sebenarnya, lemmatization akan menjadi cara untuk pergi.

Dengan proses pembersihan data “selesai” (komentar: latihan pembersihan penambangan teks hampir tidak pernah dapat diselesaikan sepenuhnya), yang tersisa hanyalah beberapa transformasi data untuk mendapatkan data dalam format kantong kata-kata. Pertama, mari kita beri label pada semua data kita dan juga buat sampel 100 pengamatan per kategori, seperti yang ditunjukkan pada daftar berikut.

Listing 8.7 Final data transformation and data splitting before modeling

Holdout sample is comprised of unlabeled data from the two subreddits: 100 observations from each data set. The labels are kept in a separate data set.

Holdout sample will be used to determine the model's flaws by constructing a confusion matrix.

```
holdoutLength  = 100

labeled_data1 = [(word,'datascience') for word in
     data['datascience']['wordMatrix'][holdoutLength:]]
labeled_data2 = [(word,'gameofthrones') for word in
     data['gameofthrones']['wordMatrix'][holdoutLength:]]
labeled_data = []
labeled_data.extend(labeled_data1)
labeled_data.extend(labeled_data2)
```

We create a single data set with every word vector tagged as being either 'datascience' or 'gameofthrones.' We keep part of the data aside for holdout sample.

```
holdout_data = data['datascience']['wordMatrix'][:holdoutLength]
holdout_data.extend(data['gameofthrones']['wordMatrix'][:holdoutLength])
holdout_data_labels = ([('datascience')
for _ in xrange(holdoutLength)] + [('gameofthrones') for _ in
     xrange(holdoutLength)])

data['datascience']['all_words_dedup'] =
list(OrderedDict.fromkeys(
data['datascience']['all_words']))
data['gameofthrones']['all_words_dedup'] =
list(OrderedDict.fromkeys(
data['gameofthrones']['all_words']))
all_words = []
all_words.extend(data['datascience']['all_words_dedup'])
all_words.extend(data['gameofthrones']['all_words_dedup'])
all_words_dedup = list(OrderedDict.fromkeys(all_words))
```

A list of all unique terms is created to build the bag of words data we need for training or scoring a model.

Data for model training and testing is first shuffled.

```
prepared_data = [({word: (word in x[0]) for word
in all_words_dedup}, x[1]) for x in labeled_data]
prepared_holdout_data = [({word: (word in x[0])
for word in all_words_dedup})
for x in holdout_data]

random.shuffle(prepared_data)
train_size = int(len(prepared_data) * 0.75)
train = prepared_data[:train_size]
test = prepared_data[train_size:]
```

Data is turned into a binary bag of words format.

Size of training data will be 75% of total and remaining 25% will be used for testing model performance.

Sampel holdout akan digunakan untuk pengujian akhir model kami dan pembuatan matriks kebingungan. Matriks konfusi adalah cara untuk memeriksa seberapa baik kinerja model pada data yang sebelumnya tidak terlihat. Matriks menunjukkan berapa banyak observasi yang diklasifikasikan dengan benar dan salah.

Sebelum membuat atau melatih dan menguji data, kita perlu melakukan satu langkah terakhir: menuangkan data ke dalam format kumpulan kata-kata di mana setiap istilah diberi label "Benar" atau "Salah" bergantung pada kehadirannya di pos tertentu. Kami juga perlu melakukan ini untuk sampel ketidakhadiran yang tidak berlabel.

Data yang telah kita siapkan sekarang berisi setiap istilah untuk setiap vektor, seperti yang ditunjukkan pada gambar 8.20.

```
print prepared_data[0]
```

```
print prepared_data[0]
({u'sunspear': False, u'profici': False, u'pardon': False, u'selye
s': False, u'four': False, u'davo': False, u'sleev': False, u'slee
                              ⋮
u'daeron': False, u'portion': False, u'emerg': False, u'fifti': Fals
e, u'decemb': False, u'defend': False, u'sincer': False}, 'datascien
ce')
```

Gambar 8.20 Kumpulan kata-kata biner yang siap untuk dimodelkan adalah data yang sangat jarang.

Kami membuat matriks yang besar namun jarang, memungkinkan kami untuk menerapkan teknik dari bab 5 jika terlalu besar untuk ditangani di mesin kami. Namun, dengan tabel sekecil itu, tidak perlu untuk itu sekarang dan kami dapat melanjutkan untuk mengocok dan membagi data menjadi set pelatihan dan pengujian.

Sementara bagian terbesar dari data Anda harus selalu digunakan untuk pelatihan model, ada rasio pemisahan yang optimal. Di sini kami memilih pembagian 3-1, tetapi jangan ragu untuk memainkannya. Semakin banyak pengamatan yang Anda miliki, semakin banyak kebebasan yang Anda miliki di sini. Jika Anda memiliki sedikit pengamatan, Anda perlu mengalokasikan lebih banyak untuk melatih model. Kami sekarang siap untuk beralih ke bagian yang paling bermanfaat: analisis data.

8.3.7 Langkah 5: Analisis data

Untuk analisis kami, kami akan menyesuaikan dua algoritme klasifikasi ke data kami: Naïve Bayes dan pohon keputusan. Naïve Bayes telah dijelaskan di bab 3 dan pohon keputusan di awal bab ini. Pertama mari kita uji kinerja pengklasifikasi Naïve Bayes kita. NLTK hadir dengan pengklasifikasi, tetapi jangan ragu untuk menggunakan algoritme dari paket lain seperti SciPy.

```
classifier = nltk.NaiveBayesClassifier.train(train)
```

Dengan pengklasifikasi yang dilatih, kita dapat menggunakan data uji untuk mengukur akurasi secara keseluruhan.

```
nltk.classify.accuracy(classifier, test)
```

```
nltk.classify.accuracy(classifier, test)
0.9681528662420382
```

Gambar 8.21 Ketepatan klasifikasi adalah ukuran yang menunjukkan persentase pengamatan yang diklasifikasikan dengan benar pada data uji.

Akurasi pada data uji diperkirakan lebih besar dari 90%, seperti yang terlihat pada gambar 8.21. Akurasi klasifikasi adalah jumlah pengamatan yang diklasifikasikan dengan benar sebagai persentase dari jumlah total pengamatan. Harap diperhatikan, bahwa ini bisa berbeda dalam kasus Anda jika Anda menggunakan data yang berbeda.

```
nltk.classify.accuracy(classifier, test)
```

Itu angka yang bagus. Sekarang kita bisa bersandar dan rileks, bukan? Tidak terlalu. Mari kita uji lagi pada 200 sampel ketidakhadiran pengamatan dan kali ini buat matriks kebingungan.

```
classified_data = classifier.classify_many(prepared_holdout_data)
cm = nltk.ConfusionMatrix(holdout_data_labels, classified_data)
print cm
```

Matriks konfusi pada gambar 8.22 menunjukkan kepada kita bahwa 97% mungkin berlebihan karena kita memiliki 28 (23 + 5) kasus kesalahan klasifikasi. Sekali lagi, ini bisa berbeda dengan data Anda jika Anda mengisi sendiri file SQLite.

```
                |     g |
                |     a |
                |  d  m |
                |  a  e |
                |  t  o |
                |  a  f |
                |  s  t |
                |  c  h |
                |  i  r |
                |  e  o |
                |  n  n |
                |  c  e |
                |  e  s |
----------------+-------+
  datascience   |<77>23 |
gameofthrones   |  5<95>|
----------------+-------+
(row = reference; col = test)
```

Gambar 8.22 Matriks kebingungan model Naïve Bayes menunjukkan 28 (23 + 5) pengamatan dari 200 salah klasifikasi

Dua puluh delapan kesalahan klasifikasi berarti kami memiliki akurasi 86% pada sampel ketidakhadiran. Ini perlu dibandingkan dengan menetapkan pos baru secara acak ke grup "datascience" atau "gameofthrones". Jika kami menetapkannya secara acak, kami dapat mengharapkan akurasi 50%, dan model kami tampaknya berperforma lebih baik dari itu. Mari kita lihat apa yang digunakannya untuk menentukan kategori dengan menggali fitur model yang paling informatif.

```
print(classifier.show_most_informative_features(20))
```

Gambar 8.23 menunjukkan 20 istilah teratas yang mampu membedakan kedua kategori tersebut.

```
Most Informative Features
                   data = True              datasc : gameof =    365.1 : 1.0
                  scene = True              gameof : datasc =     63.8 : 1.0
                 season = True              gameof : datasc =     62.4 : 1.0
                   king = True              gameof : datasc =     47.6 : 1.0
                     tv = True              gameof : datasc =     45.1 : 1.0
                   kill = True              gameof : datasc =     31.5 : 1.0
                 compani = True             datasc : gameof =     28.5 : 1.0
                analysi = True              datasc : gameof =     27.1 : 1.0
                process = True              datasc : gameof =     25.5 : 1.0
                  appli = True              datasc : gameof =     25.5 : 1.0
               research = True              datasc : gameof =     23.2 : 1.0
                 episod = True              gameof : datasc =     22.2 : 1.0
                 market = True              datasc : gameof =     21.7 : 1.0
                  watch = True              gameof : datasc =     21.6 : 1.0
                    man = True              gameof : datasc =     21.0 : 1.0
                  north = True              gameof : datasc =     20.8 : 1.0
                     hi = True              datasc : gameof =     20.4 : 1.0
                  level = True              datasc : gameof =     19.1 : 1.0
                  learn = True              datasc : gameof =     16.9 : 1.0
                    job = True              datasc : gameof =     16.6 : 1.0
```

Gambar 8.23 Istilah terpenting dalam model klasifikasi Naïve Bayes

Istilah "data" diberikan bobot yang berat dan tampaknya menjadi indikator terpenting apakah suatu topik termasuk dalam kategori ilmu data. Istilah seperti "adegan", "musim", "raja", "tv", dan "bunuh" adalah indikasi yang baik bahwa topiknya adalah Game of Thrones, bukan ilmu data. Semua hal ini sangat masuk akal, sehingga model lulus baik akurasi maupun kewarasan. Naïve Bayes bekerja dengan baik, jadi mari kita lihat pohon keputusan dalam daftar berikut.

Listing 8.8 Decision tree model training and evaluation

Create confusion matrix based on classification results and actual labels

Train decision tree classifier

Test classifier accuracy

Attempt to classify holdout data (scoring)

Show confusion matrix

```
classifier2 = nltk.DecisionTreeClassifier.train(train)
nltk.classify.accuracy(classifier2, test)
classified_data2 = classifier2.classify_many(prepared_holdout_data)
cm = nltk.ConfusionMatrix(holdout_data_labels, classified_data2)
print cm
```

```
nltk.classify.accuracy(classifier2, test)
0.9333333333333333
```

Gambar 8.24 Akurasi model pohon keputusan

Seperti terlihat pada gambar 8.24, akurasi yang dijanjikan adalah 93%. Kita sekarang tahu lebih baik daripada hanya mengandalkan tes tunggal ini, jadi sekali lagi kita beralih ke matriks kebingungan pada kumpulan data kedua, seperti yang ditunjukkan pada gambar 8.25.

Gambar 8.25 menunjukkan cerita yang berbeda. Pada 200 pengamatan sampel ketidakhadiran ini, model pohon keputusan cenderung mengklasifikasikan dengan baik saat postingan tentang Game of Thrones tetapi gagal total saat dihadapkan pada postingan ilmu data. Tampaknya model tersebut lebih menyukai Game of Thrones, dan dapatkah Anda menyalahkannya? Mari kita lihat model sebenarnya, meskipun dalam hal ini kita akan menggunakan Naïve Bayes sebagai model terakhir kita.

```
print(classifier2.pseudocode(depth=4))
```

```
                |     g |
                |     a |
                |  d  m |
                |  a  e |
                |  t  o |
                |  a  f |
                |  s  t |
                |  c  h |
                |  i  r |
                |  e  o |
                |  n  n |
                |  c  e |
                |  e  s |
--------------+-------+
  datascience |<26>74 |
gameofthrones |  2<98>|
--------------+-------+
(row = reference; col = test)
```

Gambar 8.25 Confusion matrix pada model pohon keputusan

Seperti namanya, pohon keputusan memiliki model seperti pohon, seperti yang ditunjukkan pada Gambar 8.26.

```
if data == False:
  if learn == False:
    if python == False:
      if tool == False: return 'gameofthrones'
      if tool == True: return 'datascience'
    if python == True: return 'datascience'
  if learn == True:
    if go == False:
      if wrong == False: return 'datascience'
      if wrong == True: return 'gameofthrones'
    if go == True:
      if upload == False: return 'gameofthrones'
      if upload == True: return 'datascience'
if data == True: return 'datascience'
```

Gambar 8.26 Representasi struktur pohon model pohon keputusan

Naïve Bayes mempertimbangkan semua istilah dan memiliki bobot yang dikaitkan, tetapi model pohon keputusan melewatinya secara berurutan, mengikuti jalur dari akar ke

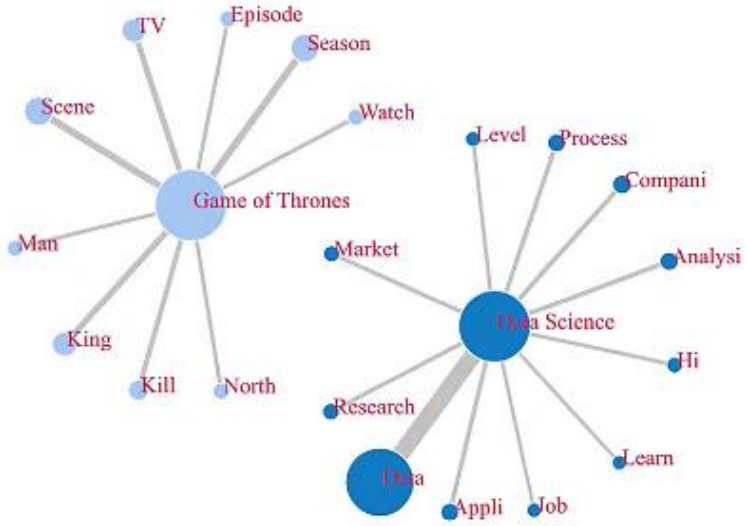
cabang dan daun terluar. Gambar 8.26 hanya menunjukkan empat lapisan teratas, dimulai dengan istilah "data". Jika "data" ada di postingan, itu selalu ilmu data. Jika "data" tidak dapat ditemukan, ia akan memeriksa istilah "pelajari", dan seterusnya. Kemungkinan alasan mengapa pohon keputusan ini tidak bekerja dengan baik adalah kurangnya pemangkasan. Ketika sebuah pohon keputusan dibangun, ia memiliki banyak daun, seringkali terlalu banyak. Sebuah pohon kemudian dipangkas ke tingkat tertentu untuk meminimalkan overfitting. Keuntungan besar dari pohon keputusan adalah efek interaksi implisit antara kata-kata yang diperhitungkan saat membangun cabang. Ketika beberapa istilah bersama membuat klasifikasi yang lebih kuat daripada istilah tunggal, pohon keputusan akan mengungguli Naïve Bayes. Kami tidak akan membahas detailnya di sini, tetapi pertimbangkan ini sebagai salah satu langkah selanjutnya yang dapat Anda ambil untuk menyempurnakan model.

Kami sekarang memiliki dua model klasifikasi yang memberi kami wawasan tentang perbedaan dua konten subreddit. Langkah terakhir adalah membagikan informasi yang baru ditemukan ini kepada orang lain.

8.3.8 Langkah 6: Presentasi dan otomatisasi

Sebagai langkah terakhir, kita perlu menggunakan apa yang telah kita pelajari dan mengubahnya menjadi aplikasi yang berguna atau mempresentasikan hasil kita kepada orang lain. Bab terakhir dari buku ini membahas membangun sebuah aplikasi interaktif, karena ini adalah sebuah proyek itu sendiri. Untuk saat ini kami akan puas dengan cara yang bagus untuk menyampaikan temuan kami. Grafik yang bagus atau, lebih baik lagi, grafik interaktif, dapat menarik perhatian; itu adalah lapisan gula pada kue presentasi. Meskipun mudah dan menggoda untuk merepresentasikan angka-angka seperti itu atau grafik batang paling banyak, alangkah baiknya untuk melangkah lebih jauh.

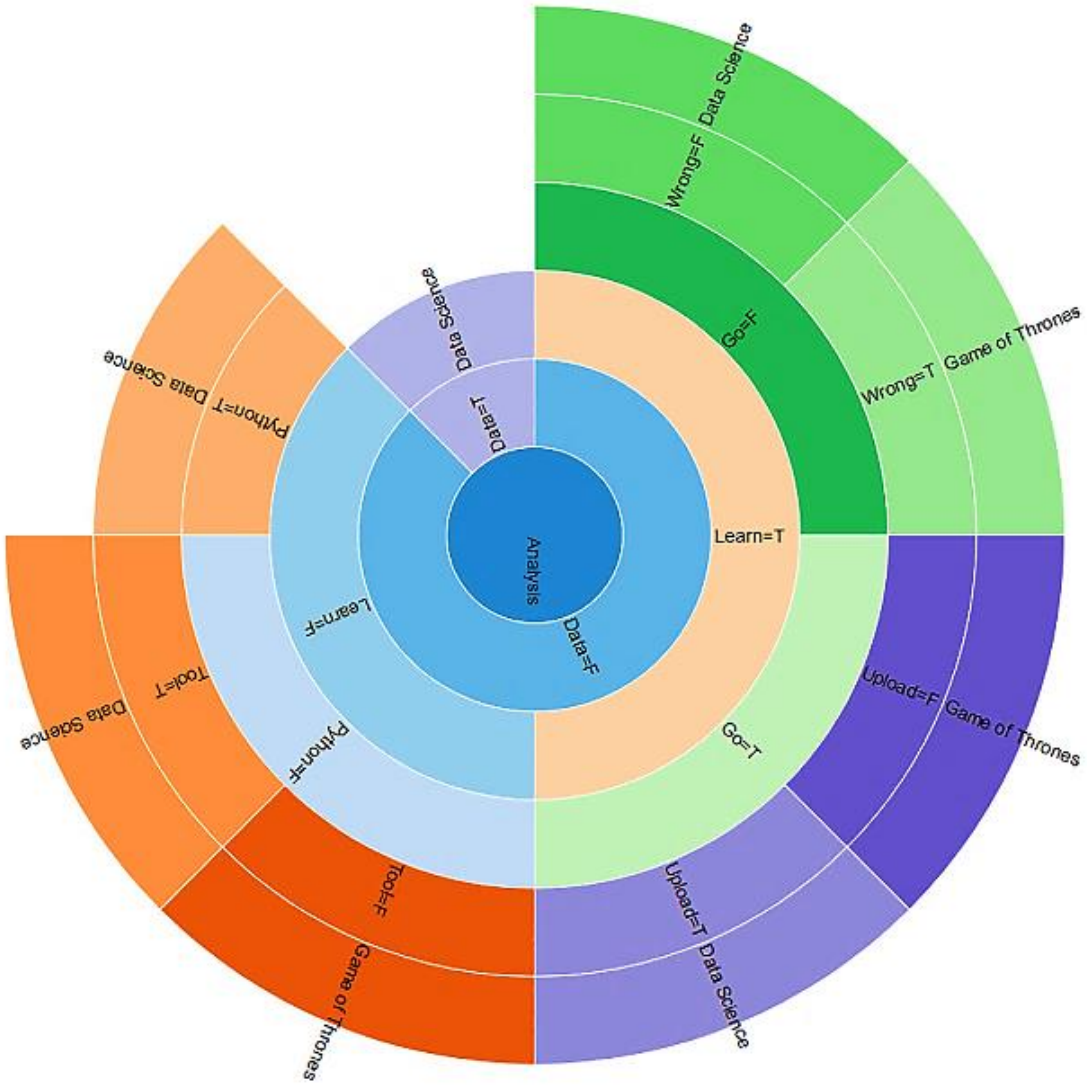
Misalnya, untuk merepresentasikan model Naïve Bayes, kita dapat menggunakan grafik gaya (gambar 8.27), di mana gelembung dan ukuran tautan menunjukkan seberapa kuat keterkaitan sebuah kata dengan subreddit “game of thrones” atau “ilmu data”. Perhatikan bagaimana kata-kata pada gelembung sering terpotong; ingat ini karena stemming yang kami terapkan.



Gambar 8.27 Grafik gaya interaktif dengan 20 suku signifikan Naïve Bayes teratas dan bobotnya

Meskipun gambar 8.27 itu sendiri statis, Anda dapat membuka file HTML "forceGraph.html" untuk menikmati efek grafik gaya d3.js seperti yang dijelaskan sebelumnya di bab ini. d3.js berada di luar cakupan buku ini, tetapi Anda tidak memerlukan pengetahuan mendalam tentang d3.js untuk menggunakannya. Serangkaian contoh ekstensif dapat digunakan dengan sedikit penyesuaian pada kode yang disediakan di https://github.com/mbostock/d3/wiki/Gallery. Yang Anda butuhkan hanyalah akal sehat dan sedikit pengetahuan tentang JavaScript. Kode untuk contoh grafik gaya dapat ditemukan di http://bl.ocks.org/mbostock/4062045.

Kami juga dapat mewakili pohon keputusan kami dengan cara yang agak orisinal. Kita bisa menggunakan versi mewah dari diagram pohon yang sebenarnya, tetapi diagram sunburst berikut ini lebih orisinal dan sama-sama menyenangkan untuk digunakan. Gambar 8.28 menunjukkan lapisan atas diagram sunburst. Dimungkinkan untuk memperbesar dengan mengeklik segmen lingkaran. Anda dapat memperkecil kembali dengan mengklik lingkaran tengah. Kode untuk contoh ini dapat ditemukan di http://bl.ocks.org/metmajer/5480307.



Gambar 8.28 Diagram Sunburst dibuat dari empat cabang teratas model pohon keputusan

Menampilkan hasil Anda dengan cara yang orisinal dapat menjadi kunci keberhasilan proyek. Orang tidak pernah menghargai upaya yang Anda lakukan untuk mencapai hasil Anda jika Anda tidak dapat mengomunikasikannya dan itu berarti bagi mereka. Visualisasi data asli di sana-sini pasti membantu dalam hal ini.

8.4 RINGKASAN

- Penambangan teks banyak digunakan untuk hal-hal seperti identifikasi entitas, deteksi plagiarisme, identifikasi topik, terjemahan, deteksi penipuan, pemfilteran spam, dan banyak lagi.
- Python memiliki perangkat yang matang untuk penambangan teks yang disebut NLTK, atau perangkat bahasa alami. NLTK bagus untuk bermain-main dan mempelajari tali; untuk aplikasi kehidupan nyata, bagaimanapun, Scikit-learn biasanya dianggap lebih "siap produksi". Scikit-learn banyak digunakan di bab-bab sebelumnya.
- Penyusunan data tekstual lebih intensif dari pada penyiapan data numerik dan melibatkan teknik ekstra, seperti

  – Stemming—Memotong akhir kata dengan cara yang cerdas sehingga dapat dicocokkan dengan beberapa versi konjugasi atau jamak dari kata tersebut.

  – Lemmatisasi — Seperti stemming, ini dimaksudkan untuk menghilangkan ganda, tetapi tidak seperti stemming, ini melihat arti kata.

  – Hentikan pemfilteran kata—Kata-kata tertentu muncul terlalu sering sehingga tidak berguna dan memfilternya dapat meningkatkan model secara signifikan. Kata henti sering kali spesifik untuk korpus.

  – Tokenisasi—Memotong teks menjadi beberapa bagian. Token dapat berupa kata tunggal, kombinasi kata (n-gram), atau bahkan seluruh kalimat.

  – Penandaan POS—Penandaan sebagian ucapan. Terkadang berguna untuk mengetahui apa fungsi kata tertentu dalam sebuah kalimat untuk memahaminya dengan lebih baik.
- Dalam studi kasus kami, kami berusaha membedakan postingan Reddit di "Game of Thrones" versus postingan di "ilmu data". Dalam upaya ini kami mencoba Naïve Bayes dan pengklasifikasi pohon keputusan. Naïve Bayes menganggap semua fitur tidak bergantung satu sama lain; pengklasifikasi pohon keputusan mengasumsikan ketergantungan, memungkinkan untuk model yang berbeda.
- Dalam contoh kita, Naïve Bayes menghasilkan model yang lebih baik, tetapi sangat sering pengklasifikasi pohon keputusan melakukan pekerjaan yang lebih baik, biasanya ketika lebih banyak data tersedia.
- Kami menentukan perbedaan kinerja menggunakan matriks kebingungan yang kami hitung setelah menerapkan kedua model pada data baru (namun berlabel).
- Saat menyajikan temuan kepada orang lain, menyertakan visualisasi data menarik yang mampu menyampaikan hasil Anda dengan cara yang mudah diingat dapat membantu.

BAB 9
VISUALISASI DATA KE PENGGUNA AKHIR

Dalam bab inii diharapkan mahasiswa mampu menguasai:

- Mempertimbangkan pilihan untuk visualisasi data bagi pengguna akhir Anda
- Menyiapkan aplikasi Crossfilter MapReduce dasar
- Membuat dasbor dengan dc.js
- Bekerja dengan alat pengembangan dasbor

BAB BERFOKUS APLIKASI Anda akan segera menyadari bahwa bab ini tentu saja berbeda dari bab 3 hingga 8 karena fokusnya di sini terletak pada langkah 6 dari proses ilmu data. Lebih khusus lagi, yang ingin kami lakukan di sini adalah membuat aplikasi ilmu data kecil. Oleh karena itu, kami tidak akan mengikuti langkah-langkah proses ilmu data di sini. Data yang digunakan dalam studi kasus hanya sebagian nyata tetapi berfungsi sebagai data yang mengalir baik dari tahap persiapan data maupun pemodelan data. Nikmati perjalanannya.

Seringkali, ilmuwan data harus menyampaikan wawasan baru mereka kepada pengguna akhir. Hasilnya dapat dikomunikasikan dalam beberapa cara:

- **Presentasi satu kali—**Pertanyaan penelitian adalah kesepakatan sekali pakai karena keputusan bisnis yang diturunkan darinya akan mengikat organisasi ke jalur tertentu selama bertahun-tahun yang akan datang. Ambil contoh, keputusan investasi perusahaan: Apakah kita mendistribusikan barang kita dari dua pusat distribusi atau hanya satu? Di mana mereka harus ditempatkan untuk efisiensi yang optimal? Saat keputusan dibuat, latihan tidak boleh diulang sampai Anda pensiun. Dalam hal ini, hasilnya disampaikan sebagai laporan dengan presentasi sebagai lapisan gula pada kue.
- **Area pandang baru pada data Anda—**Contoh paling nyata di sini adalah segmentasi pelanggan. Tentu, segmen-segmen itu sendiri akan dikomunikasikan melalui laporan dan presentasi, tetapi pada intinya mereka membentuk alat, bukan hasil akhir itu sendiri. Ketika segmentasi pelanggan yang jelas dan relevan ditemukan, itu dapat diumpankan kembali ke database sebagai dimensi baru pada data asalnya. Dari situ, masyarakat bisa membuat laporan sendiri, seperti berapa banyak produk yang terjual ke masing-masing segmen pelanggan.
- **Dasbor waktu nyata—**Terkadang tugas Anda sebagai ilmuwan data tidak berakhir saat Anda menemukan informasi baru yang Anda cari. Anda dapat mengirim informasi Anda kembali ke database dan menyelesaikannya. Tetapi ketika orang lain mulai membuat laporan tentang nugget emas yang baru ditemukan ini, mereka mungkin salah mengartikannya dan membuat laporan yang tidak masuk akal. Sebagai ilmuwan data yang menemukan informasi baru ini, Anda harus memberi contoh: membuat laporan pertama yang dapat disegarkan sehingga orang lain, terutama reporter dan TI, dapat memahaminya dan mengikuti jejak Anda. Pembuatan dashboard pertama juga merupakan cara untuk mempersingkat waktu pengiriman wawasan Anda kepada pengguna akhir yang ingin menggunakannya sehari-hari. Dengan cara ini, setidaknya mereka sudah memiliki

sesuatu untuk dikerjakan sampai departemen pelaporan menemukan waktu untuk membuat laporan permanen pada perangkat lunak pelaporan perusahaan.

Anda mungkin telah memperhatikan bahwa beberapa faktor penting berperan:

- *Keputusan seperti apa yang Anda dukung?* Apakah strategis atau operasional? Keputusan strategis seringkali hanya mengharuskan Anda untuk menganalisis dan melaporkan satu kali, sedangkan keputusan operasional mengharuskan laporan diperbarui secara berkala.
- *Seberapa besar organisasi Anda?* Dalam yang lebih kecil, Anda akan bertanggung jawab atas seluruh siklus: mulai dari pengumpulan data hingga pelaporan. Dalam kasus yang lebih besar, tim reporter mungkin tersedia untuk membuat dasbor untuk Anda. Tetapi bahkan dalam situasi terakhir ini, memberikan dashboard prototipe dapat bermanfaat karena memberikan contoh dan seringkali mempersingkat waktu pengiriman.

Meskipun keseluruhan buku didedikasikan untuk menghasilkan wawasan, dalam bab terakhir ini kita akan berfokus pada penyampaian dasbor operasional. Membuat presentasi untuk mempromosikan temuan Anda atau menyajikan wawasan strategis berada di luar cakupan buku ini.

9.1 OPSI VISUALISASI DATA

Anda memiliki beberapa opsi untuk mengirimkan dasbor ke pengguna akhir Anda. Di sini kita akan fokus pada satu opsi, dan di akhir bab ini Anda akan dapat membuat dasbor sendiri. Kasus bab ini adalah apotek rumah sakit dengan stok beberapa ribu obat. Pemerintah mengeluarkan norma baru untuk semua apotek: semua obat harus diperiksa sensitivitasnya terhadap cahaya dan disimpan dalam wadah khusus yang baru. Satu hal yang tidak disediakan pemerintah ke apotek adalah daftar sebenarnya dari obat-obatan peka cahaya. Hal ini tidak menjadi masalah bagi Anda sebagai data scientist karena setiap obat memiliki leaflet informasi pasien yang berisi informasi tersebut. Anda menyaring informasi dengan penggunaan penambangan teks yang cerdas dan menetapkan tag "peka cahaya" atau "tidak peka cahaya" untuk setiap obat. Informasi ini kemudian diunggah ke database pusat. Selain itu, apotek perlu mengetahui berapa banyak wadah yang dibutuhkan. Untuk ini, mereka memberi Anda akses ke data stok apotek. Saat Anda menggambar sampel hanya dengan variabel yang Anda butuhkan, kumpulan data terlihat seperti gambar 9.1 saat dibuka di Excel.

	A	B	C	D	E	F
1	MedName	LightSen	Date	StockOut	StockIn	Stock
2	Acupan 30 mg	No	1/01/2015	-8	150	142
3	Acupan 30 mg	No	2/01/2015	-6	5	141
4	Acupan 30 mg	No	3/01/2015	-2	0	139
5	Acupan 30 mg	No	4/01/2015	0	5	144
6	Acupan 30 mg	No	5/01/2015	-8	0	136
7	Acupan 30 mg	No	6/01/2015	-1	0	135
8	Acupan 30 mg	No	7/01/2015	-1	15	149
9	Acupan 30 mg	No	8/01/2015	-10	10	149
10	Acupan 30 mg	No	9/01/2015	-8	15	156

Gambar 9.1 Set data obat-obatan farmasi dibuka di Excel: 10 baris pertama data stok ditingkatkan dengan variabel sensitivitas cahaya

Seperti yang Anda lihat, informasinya adalah data deret waktu selama setahun penuh pergerakan stok, sehingga setiap obat memiliki 365 entri dalam kumpulan data. Meskipun studi kasus sudah ada dan obat-obatan dalam kumpulan data adalah nyata, nilai variabel lain yang disajikan di sini dibuat secara acak, karena data asli diklasifikasikan. Juga, kumpulan data terbatas pada 29 obat, sedikit lebih dari 10.000 baris data. Meskipun orang membuat laporan menggunakan crossfilter.js (library MapReduce Javascript) dan dc.js (library dasbor Javascript) dengan lebih dari satu juta baris data, sebagai contoh Anda akan menggunakan sebagian kecil dari jumlah ini. Selain itu, tidak disarankan untuk memuat seluruh database Anda ke dalam browser pengguna; browser akan membeku saat memuat, dan jika terlalu banyak data, browser bahkan akan macet. Biasanya data dihitung sebelumnya di server dan sebagian diminta menggunakan, misalnya, layanan REST.

Untuk mengubah data ini menjadi dasbor sebenarnya, Anda memiliki banyak opsi dan Anda dapat menemukan ikhtisar singkat tentang alat tersebut nanti di bab ini. Di antara semua opsi, untuk buku ini kami memutuskan untuk menggunakan dc.js, yang merupakan perkawinan silang antara pustaka JavaScript MapReduce Crossfilter dan pustaka visualisasi data d3.js. Crossfilter dikembangkan oleh Square Register, sebuah perusahaan yang menangani transaksi pembayaran; itu sebanding dengan PayPal tetapi fokusnya adalah pada seluler. Square mengembangkan Crossfilter untuk memungkinkan pelanggan mereka memotong dan memotong riwayat pembayaran mereka dengan sangat cepat. Crossfilter bukan satu-satunya library JavaScript yang mampu memproses Map-Reduce, tetapi yang pasti melakukan tugasnya, open source, bebas digunakan, dan dikelola oleh perusahaan mapan (Square). Contoh alternatif untuk Cross-filter adalah Map.js, Meguro, dan Underscore.js. JavaScript mungkin tidak dikenal sebagai bahasa pemecah data, tetapi pustaka ini memang memberi peramban web pukulan ekstra jika data perlu ditangani di peramban. Kami tidak akan masuk ke bagaimana Java-Script dapat digunakan untuk perhitungan besar-besaran dalam kerangka kerja terdistribusi kolaboratif, tetapi pasukan kerdil dapat menumbangkan raksasa.

d3.js dapat dengan aman disebut perpustakaan visualisasi data JavaScript paling serbaguna yang tersedia pada saat penulisan; itu dikembangkan oleh Mike Bostock sebagai penerus perpustakaan Protovis-nya. Banyak pustaka JavaScript dibangun di atas d3.js. NVD3, C3.js, xCharts, dan Dimple menawarkan hal yang kira-kira sama: lapisan abstraksi di atas d3.js, yang mempermudah menggambar grafik sederhana. Mereka terutama berbeda dalam jenis grafik yang mereka dukung dan desain standarnya. Jangan ragu untuk mengunjungi situs web mereka dan temukan sendiri:

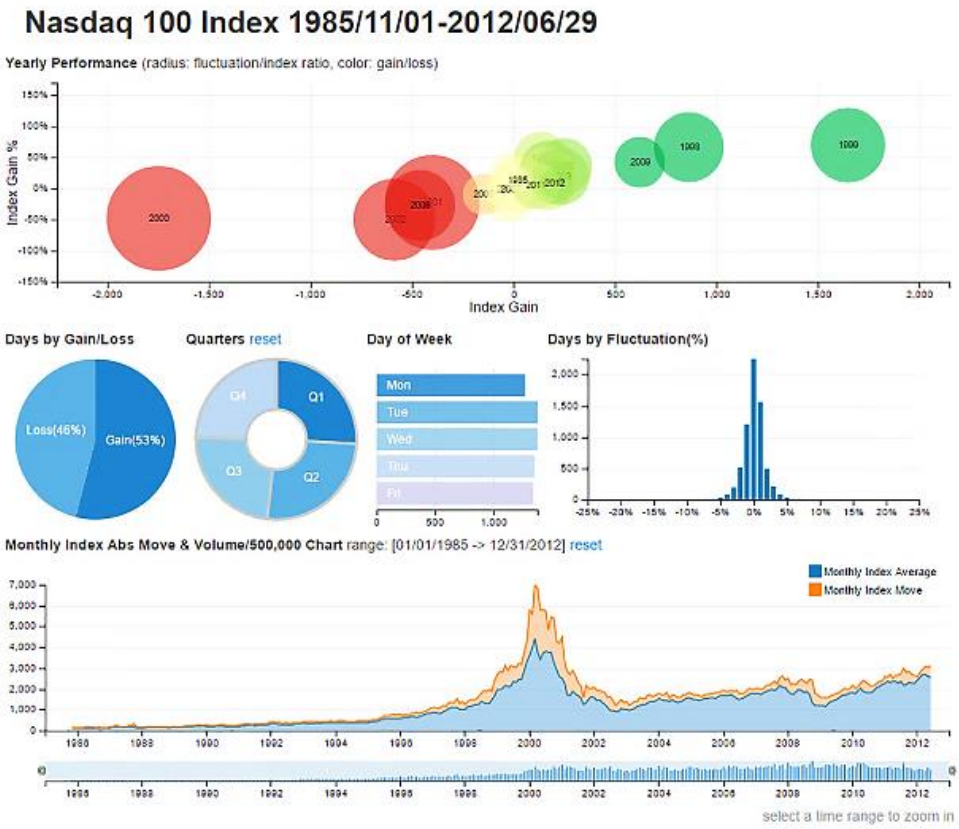
- **NVD3 —**http://nvd3.org/
- **C3.js—**http://c3js.org/
- **xChart—**http://tenxer.github.io/xcharts/
- **Lesung Pipi—**http://dimplejs.org/

Banyak pilihan yang ada. Jadi mengapa dc.js?

Alasan utama: dibandingkan dengan apa yang diberikannya, dasbor interaktif tempat mengklik satu grafik akan membuat tampilan terfilter pada grafik terkait, dc.js secara mengejutkan mudah diatur. Sangat mudah bahwa Anda akan memiliki contoh kerja pada akhir

bab ini. Sebagai seorang ilmuwan data, Anda telah mencurahkan cukup waktu untuk analisis aktual Anda; dasbor yang mudah diterapkan adalah hadiah selamat datang. Untuk mendapatkan gambaran tentang apa yang akan Anda buat, Anda dapat mengunjungi situs web berikut, http://dc-js.github.io/dc.js/, dan gulir ke bawah ke contoh NASDAQ, ditunjukkan pada gambar 9.2.

Klik di sekitar dasbor dan lihat grafik bereaksi dan berinteraksi saat Anda memilih dan membatalkan pilihan titik data. Namun, jangan menghabiskan waktu terlalu lama; saatnya untuk membuat ini sendiri. Seperti yang dinyatakan sebelumnya, dc.js memiliki dua prasyarat besar: d3.js dan crossfilter.js. d3.js memiliki kurva belajar yang curam dan ada beberapa buku tentang topik yang layak dibaca jika Anda tertarik untuk menyesuaikan sepenuhnya visualisasi Anda. Tetapi untuk bekerja dengan dc.js, tidak diperlukan pengetahuan tentangnya, jadi kami tidak akan membahasnya di buku ini. Crossfilter.js adalah masalah lain; Anda harus memiliki sedikit pemahaman tentang perpustakaan MapReduce ini untuk mengaktifkan dan menjalankan dc.js pada data Anda. Tapi karena konsep MapReduce sendiri bukanlah hal baru, ini akan terjadi berjalan lancar.



Gambar 9.2 Contoh interaktif dc.js di website resminya

9.2 CROSSFILTER, PUSTAKA JAVASCRIPT MAPREDUCE

JavaScript bukanlah bahasa terbaik untuk mengolah data. Tapi itu tidak menghentikan orang, seperti orang-orang di Square, untuk mengembangkan pustaka MapReduce untuknya. Jika Anda berurusan dengan data, setiap peningkatan kecepatan membantu. Anda tidak ingin

mengirim banyak sekali data melalui internet atau bahkan jaringan internal Anda, karena alasan berikut:

- Mengirim sejumlah besar data akan membebani jaringan hingga mengganggu pengguna lain.
- Browser berada di sisi penerima, dan saat memuat data, browser akan membeku untuk sementara. Untuk sejumlah kecil data, ini tidak terlalu mencolok, tetapi saat Anda mulai melihat 100.000 baris, ini bisa menjadi jeda yang terlihat. Saat Anda melewati 1.000.000 baris, tergantung pada lebar data Anda, browser Anda bisa menyerah pada Anda.

Kesimpulan: ini adalah latihan keseimbangan. Untuk data yang Anda kirim, ada Crossfilter untuk menanganinya untuk Anda setelah tiba di browser. Dalam studi kasus kami, apoteker meminta server pusat untuk data stok tahun 2015 untuk 29 obat yang sangat dia minati. Kami sudah melihat datanya, jadi mari selami aplikasi itu sendiri.

9.2.1 Menyiapkan semuanya

Saatnya membangun aplikasi sebenarnya, dan bahan-bahan dari aplikasi dc.js kecil kita adalah sebagai berikut:

- **JQuery—**Untuk menangani interaktivitas
- **Crossfilter.js** —Pustaka MapReduce dan prasyarat untuk dc.js
- **d3.js**—Pustaka visualisasi data yang populer dan prasyarat untuk dc.js
- **dc.js**—Pustaka visualisasi yang akan Anda gunakan untuk membuat dasbor interaktif
- **Bootstrap**—Pustaka tata letak yang banyak digunakan yang akan Anda gunakan untuk membuat semuanya terlihat lebih baik

Anda hanya akan menulis tiga file:

- **index.html**—Halaman HTML yang berisi aplikasi Anda
- **application.js**—Untuk menampung semua kode JavaScript yang akan Anda tulis
- **application.css**—Untuk CSS Anda sendiri

Selain itu, Anda harus menjalankan kode kami di server HTTP. Anda dapat melalui upaya menyiapkan server LAMP (Linux, Apache, MySQL, PHP), WAMP (Windows, Apache, MySQL, PHP), atau XAMPP (Cross Environment, Apache, MySQL, PHP, Perl). Namun demi kesederhanaan, kami tidak akan menyiapkan salah satu dari server tersebut di sini. Sebaliknya Anda dapat melakukannya dengan satu perintah Python. Gunakan alat baris perintah Anda (Linux shell atau Windows CMD) dan pindah ke folder yang berisi index.html Anda (setelah ada). Anda harus menginstal Python untuk bab-bab lain dari buku ini sehingga perintah berikut harus meluncurkan server HTTP Python di localhost Anda.

```
python -m SimpleHTTPServer
```

Untuk Python 3.4

```
python -m http.server 8000
```

Seperti yang Anda lihat pada gambar 9.3, server HTTP dimulai pada port localhost 8000. Di browser Anda ini diterjemahkan menjadi “localhost:8000”; menempatkan "0.0.0.0:8000" tidak akan berfungsi.

Gambar 9.3 Memulai server HTTP Python sederhana

Pastikan semua file yang diperlukan tersedia di folder yang sama dengan index.html Anda. Anda dapat mengunduhnya dari situs web Manning atau dari situs web pembuatnya.

- **dc.css dan dc.min.js**—https://dc-js.github.io/dc.js/
- **d3.v3.min.js**—http://d3js.org/
- **crossfilter.min.js**—http://square.github.io/crossfilter/

Sekarang kita tahu cara menjalankan kode yang akan kita buat, jadi mari kita lihat halaman index.html, yang ditampilkan di daftar berikut.

Listing 9.1 An initial version of index.html

```
<html>
<head>
    <title>Chapter 10.  Data Science Application</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap
/3.3.0/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/
3.3.0/css/bootstrap-theme.min.css">

    <link rel="stylesheet" href="dc.css">
    <link rel="stylesheet" href="application.css">
</head>
<body>

    <main class='container'>
        <h1>Chapter 10:  Data Science Application</h1>
        <div class="row">
            <div class='col-lg-12'>
                <div id="inputtable" class="well well-sm"></div>
            </div>
        </div>
        <div class="row">
            <div class='col-lg-12'>
                <div id="filteredtable" class="well well-sm"></div>
            </div>
        </div>
    </main>
```

All CSS is loaded here.

Make sure to have dc.css downloaded from the Manning download page or from the dc website: https://dc-js.github.io/dc.js/. It must be present in the same folder as index.html file.

Main container incorporates everything visible to user.

```
    <script src="https://code.jquery.com/jquery-1.9.1.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.0/js
/bootstrap.min.js"></script>

    <script src="crossfilter.min.js"></script>
    <script src="d3.v3.min.js"></script>
    <script src="dc.min.js"></script>
    <script src="application.js"></script>
</body>
</html>
```

All Javascript is loaded here.

Make sure to have crossfilter.min.js, d3.v3.min.js, and dc.min.js downloaded from their websites or from the Manning website. Crossfilter: http://square.github.io/crossfilter/, d3.js: http://d3js.org/, dc.min.js: https://dc-js.github.io/dc.js/.

Tidak ada kejutan di sini. Header berisi semua pustaka CSS yang akan Anda gunakan, jadi kami akan memuat JavaScript di bagian akhir badan HTML. Menggunakan penangan onload JQuery, aplikasi Anda akan dimuat saat sisa halaman sudah siap. Anda memulai dengan dua placeholder tabel: satu untuk menampilkan seperti apa data input Anda, , dan yang lainnya akan digunakan dengan Crossfilter untuk menampilkan tabel yang difilter, . Beberapa kelas CSS Bootstrap digunakan, seperti "well", "container", sistem grid Bootstrap dengan "*row*" dan "*col-xx-xx*", dan seterusnya. Mereka membuat semuanya terlihat lebih bagus tetapi tidak wajib. Informasi lebih lanjut tentang kelas CSS Bootstrap dapat ditemukan di situs web mereka di http://getbootstrap.com/css/.

Sekarang setelah Anda menyiapkan HTML, saatnya menampilkan data Anda di layar. Untuk ini, alihkan perhatian Anda ke file application.js yang Anda buat. Pertama, kami membungkus seluruh kode "menjadi" dalam penangan onload JQuery.

```
$(function() {
    //All future code will end up in this wrapper
})
```

Sekarang kami yakin aplikasi kami akan dimuat hanya ketika semuanya sudah siap. Ini penting karena kami akan menggunakan pemilih JQuery untuk memanipulasi HTML. Saatnya memuat data.

```
d3.csv('medicines.csv',function(data) {
    main(data)
});
```

Anda belum memiliki layanan REST yang siap dan menunggu Anda, jadi sebagai contoh Anda akan mengambil data dari file .csv. File ini tersedia untuk diunduh di situs web Manning. d3.js menawarkan fungsi yang mudah untuk itu. Setelah memuat data, Anda menyerahkannya ke fungsi aplikasi utama Anda di fungsi callback d3.csv.

Terlepas dari fungsi utama, Anda memiliki fungsi CreateTable, yang akan Anda gunakan untuk membuat tabel Anda, seperti yang ditunjukkan pada daftar berikut.

Listing 9.2 The `CreateTable` function

```
var tableTemplate = $([
        "<table class='table table-hover table-condensed table-striped'>",
        "  <caption></caption>",
        "  <thead><tr/></thead>",
        "  <tbody></tbody>",
        "</table>"
    ].join('\n'));

    CreateTable = function(data,variablesInTable,title){
        var table = tableTemplate.clone();
        var ths = variablesInTable.map(function(v) { return $("<th>").text(v)
     });
        $('caption', table).text(title);
        $('thead tr', table).append(ths);
        data.forEach(function(row) {
            var tr = $("<tr>").appendTo($('tbody', table));
            variablesInTable.forEach(function(varName) {
                var val = row, keys = varName.split('.');
                keys.forEach(function(key) { val = val[key] });
                tr.append($("<td>").text(val));
            });
        });
        return table;
    }
```

CreateTable() membutuhkan tiga argumen:

- data —Data yang perlu dimasukkan ke dalam tabel.
- variablesInTable —Variabel apa yang perlu ditampilkan.
- Title —Judul tabel. Itu selalu menyenangkan untuk mengetahui apa yang Anda lihat.

CreateTable() menggunakan variabel yang telah ditentukan sebelumnya tableTemplate, yang berisi tata letak tabel kita secara keseluruhan. CreateTable() kemudian dapat menambahkan baris data ke template ini. Sekarang setelah Anda memiliki utilitas, mari kita ke fungsi utama aplikasi, seperti yang ditunjukkan pada daftar berikut.

Listing 9.3 JavaScript `main` function

```
main = function(inputdata){

    var medicineData = inputdata ;

    var dateFormat = d3.time.format("%d/%m/%Y");
    medicineData.forEach(function (d) {
        d.Day = dateFormat.parse(d.Date);
    })
    var variablesInTable =
['MedName','StockIn','StockOut','Stock','Date','LightSen']
      var sample = medicineData.slice(0,5);
      var inputTable = $("#inputtable");

    inputTable
        .empty()
        .append(CreateTable(sample,variablesInTable,"The input table"));
}
```

Our data: normally this is fetched from a server but in this case we read it from a local .csv file

Put the variables we'll show in the table in an array so we can loop through them when creating table code

Convert date to correct format so Crossfilter will recognize date variable

Only show a sample of data

Create table

Anda memulai dengan menampilkan data Anda di layar, tetapi sebaiknya tidak semuanya; hanya lima entri pertama yang dapat melakukannya, seperti yang ditunjukkan pada gambar 9.4. Anda memiliki variabel tanggal di data Anda dan Anda ingin memastikan bahwa Crossfilter akan mengenalinya nanti, jadi pertama-tama Anda menguraikannya dan membuat variabel baru bernama Hari. Anda menunjukkan yang asli, Tanggal, untuk muncul di tabel untuk saat ini, tetapi nanti Anda akan menggunakan Hari untuk semua perhitungan Anda.

The input table

MedName	StockIn	StockOut	Stock	Date	LightSen
Acupan 30 mg	150	-7	143	1/01/2015	No
Acupan 30 mg	5	-6	142	2/01/2015	No
Acupan 30 mg	15	-9	148	3/01/2015	No
Acupan 30 mg	0	-11	137	4/01/2015	No
Acupan 30 mg	10	-8	139	5/01/2015	No

Gambar 9.4 Tabel input obat ditampilkan di browser: lima baris pertama

Inilah yang Anda dapatkan: hal yang sama yang Anda lihat di Excel sebelumnya. Sekarang setelah Anda mengetahui dasar-dasarnya berfungsi, Anda akan memasukkan Crossfilter ke dalam persamaan.

9.2.2 Unleashing Crossfilter untuk memfilter kumpulan data obat

Sekarang mari masuk ke Crossfilter untuk menggunakan pemfilteran dan MapReduce. Untuk selanjutnya Anda dapat meletakkan semua kode yang akan datang setelah kode bagian 9.2.1 di dalam fungsi main(). Hal pertama yang perlu Anda lakukan adalah mendeklarasikan instance Crossfilter dan memulainya dengan data Anda.

```
CrossfilterInstance = crossfilter(medicineData);
```

Dari sini Anda bisa mulai bekerja. Pada contoh ini Anda dapat mendaftarkan dimensi, yang merupakan kolom dari tabel Anda. Saat ini Crossfilter dibatasi hingga 32 dimensi. Jika Anda menangani data yang lebih lebar dari 32 dimensi, Anda harus mempertimbangkan untuk mempersempitnya sebelum mengirimkannya ke browser. Mari buat dimensi pertama kita, dimensi nama obat:

```
var medNameDim = CrossfilterInstance.dimension(function(d) {return
   d.MedName;});
```

Dimensi pertama Anda adalah nama obat-obatan, dan Anda sudah dapat menggunakan ini untuk memfilter set data Anda dan menampilkan data yang difilter menggunakan fungsi CreateTable()kami.

```
   var dataFiltered= medNameDim.filter('Grazax 75 000 SQ-T')
   var filteredTable = $('#filteredtable');
   filteredTable
 .empty().append(CreateTable(dataFiltered.top(5),variablesInTable,'Our
First Filtered Table'));
```

Anda hanya menampilkan lima pengamatan teratas (gambar 9.5); Anda memiliki 365 karena Anda memiliki hasil dari satu obat selama satu tahun penuh.

Our First Filtered Table

MedName	StockIn	StockOut	Stock	Date	LightSen
Grazax 75 000 SQ-T	15	0	205	31/08/2015	Yes
Grazax 75 000 SQ-T	0	-4	62	30/12/2015	Yes
Grazax 75 000 SQ-T	10	-15	66	29/12/2015	Yes
Grazax 75 000 SQ-T	15	0	71	28/12/2015	Yes
Grazax 75 000 SQ-T	10	-4	56	27/12/2015	Yes

Gambar 9.5 Data tersaring pada nama obat Grazax 75 000 SQ-T

Tabel ini tidak terlihat disortir tetapi memang begitu. Fungsi top() mengurutkannya berdasarkan nama obat. Karena Anda hanya memilih satu obat, itu tidak masalah. Menyortir tanggal cukup mudah menggunakan variabel Hari baru Anda. Mari daftarkan dimensi lain, dimensi tanggal:

```
        var DateDim = CrossfilterInstance.dimension(
function(d) {return d.Day;});
```

Sekarang kita dapat mengurutkan berdasarkan tanggal, bukan nama obat:

```
      filteredTable
             .empty()
             .append(CreateTable(DateDim.bottom(5),variablesInTable,'Our
   First Filtered Table'));
```

Hasilnya sedikit lebih menarik, seperti yang ditunjukkan pada gambar 9.6.

Our First Filtered Table

MedName	StockIn	StockOut	Stock	Date	LightSen
Grazax 75 000 SQ-T	65	-12	53	1/01/2015	Yes
Grazax 75 000 SQ-T	15	-11	57	2/01/2015	Yes
Grazax 75 000 SQ-T	5	-9	53	3/01/2015	Yes
Grazax 75 000 SQ-T	5	-4	54	4/01/2015	Yes
Grazax 75 000 SQ-T	0	-14	40	5/01/2015	Yes

Gambar 9.6 Data yang difilter berdasarkan nama obat Grazax 75 000 SQ-T dan disortir berdasarkan hari

Tabel ini memberi Anda tampilan jendela dari data Anda, tetapi belum meringkasnya untuk Anda. Di sinilah kapabilitas Crossfilter MapReduce berperan. Katakanlah Anda ingin mengetahui berapa banyak pengamatan yang Anda lakukan per obat. Logika menyatakan

bahwa Anda harus mendapatkan angka yang sama untuk setiap obat: 365, atau 1 pengamatan per hari pada tahun 2015.

```
        var countPerMed = medNameDim.group().reduceCount();
        variablesInTable = ["key","value"]
        filteredTable
                .empty()
    .append(CreateTable(countPerMed.top(Infinity),
variablesInTable,'Reduced Table'));
```

Crossfilter hadir dengan dua fungsi MapReduce: reduceCount() dan reduceSum(). Jika Anda ingin melakukan sesuatu selain menghitung dan menjumlahkan, Anda perlu menulis fungsi pengurangan untuk itu. Variabel countPerMed sekarang berisi data yang dikelompokkan menurut dimensi obat-obatan dan jumlah garis untuk setiap obat dalam bentuk kunci dan nilai. Untuk membuat tabel, Anda perlu mengalamatkan kunci variabel alih-alih medName dan nilai untuk hitungan (gambar 9.7).

Reduced Table

key	value
Adoport 1 mg	365
Atenolol EG 100 mg	365
Ceftriaxone Actavis 1 g	365
Cefuroxim Mylan 500 mg	365
Certican 0.25 mg	365

Gambar 9.7 Tabel MapReduced dengan obat sebagai grup dan jumlah baris data sebagai nilainya

Dengan menentukan .top(Infinity) Anda meminta untuk menampilkan semua 29 obat di layar, tetapi demi menghemat kertas, gambar 9.7 hanya menampilkan lima hasil pertama. Oke, Anda bisa tenang; data berisi 365 baris per obat. Perhatikan bagaimana Crossfilter mengabaikan filter pada "Grazax". Jika dimensi digunakan untuk pengelompokan, filter tidak berlaku untuk dimensi tersebut. Hanya filter pada dimensi lain yang dapat mempersempit hasil.

Bagaimana dengan kalkulasi yang lebih menarik yang tidak dibundel dengan Crossfilter, seperti rata-rata, misalnya? Anda masih bisa melakukannya tetapi Anda perlu menulis tiga fungsi dan memasukkannya ke metode .reduce(). Katakanlah Anda ingin mengetahui stok rata-rata per obat. Seperti yang disebutkan sebelumnya, hampir semua logika MapReduce perlu ditulis oleh Anda. Rata-rata tidak lebih dari pembagian jumlah dengan hitungan, jadi Anda membutuhkan keduanya; bagaimana Anda melakukan ini? Selain fungsi reduce- Count() dan reduceSum(), Crossfilter memiliki fungsi reduce() yang lebih umum. Fungsi ini membutuhkan tiga argumen:

- **Fungsi reduceAdd()—**Fungsi yang menjelaskan apa yang terjadi saat pengamatan tambahan ditambahkan.
- **Fungsi reduceRemove()—**Fungsi yang menjelaskan apa yang perlu terjadi saat observasi menghilang (misalnya, karena filter diterapkan).

- **Fungsi reduceInit()—**Yang ini menetapkan nilai awal untuk semua yang dihitung. Untuk penjumlahan dan hitungan titik awal yang paling logis adalah 0.

Mari kita lihat masing-masing fungsi pengurangan yang Anda perlukan sebelum mencoba memanggil metode Crossfilter .reduce() , yang menggunakan ketiga komponen ini sebagai argumen. Fungsi pengurangan kustom memerlukan tiga komponen: inisiasi, fungsi tambah, dan fungsi hapus. Fungsi pengurangan awal akan menetapkan nilai awal dari objek p:

```
var reduceInitAvg = function(p,v){
    return {count: 0, stockSum : 0, stockAvg:0};
}
```

Seperti yang Anda lihat, fungsi pengurangan itu sendiri mengambil dua argumen. Ini secara otomatis diberikan kepada mereka dengan metode Crossfilter .reduce() :

- p adalah objek yang berisi situasi kombinasi sejauh ini; itu bertahan selama semua pengamatan. Variabel ini melacak jumlah dan hitungan untuk Anda dan dengan demikian mewakili tujuan Anda, hasil akhir Anda.
- v mewakili catatan data input dan semua variabelnya tersedia untuk Anda. Berlawanan dengan p, itu tidak bertahan tetapi digantikan oleh baris data baru setiap kali fungsi dipanggil. ReduceInit() dipanggil hanya sekali, tetapi reduceAdd() dipanggil setiap kali record ditambahkan dan reduceRemove() setiap kali satu baris data dihapus.
- Fungsi reduceInit(), di sini disebut reduceInitAvg() karena Anda akan menghitung rata-rata, pada dasarnya menginisialisasi objek p dengan mendefinisikan komponennya (jumlah, jumlah, dan rata-rata) dan menyetel nilai awalnya. Mari kita lihat penguranganAddAvg():

```
var reduceAddAvg = function(p,v){
    p.count += 1;
    p.stockSum  = p.stockSum  + Number(v.Stock);
    p.stockAvg = Math.round(p.stockSum  / p.count);
    return p;
}
```

reduceAddAvg() mengambil argumen p dan v yang sama tetapi sekarang Anda benar-benar menggunakan v; Anda tidak memerlukan data Anda untuk menyetel nilai awal p dalam kasus ini, meskipun Anda bisa jika mau. Stok Anda dijumlahkan untuk setiap catatan yang Anda tambahkan, lalu rata-rata dihitung berdasarkan jumlah akumulasi dan jumlah catatan:

```
var reduceRemoveAvg = function(p,v){
    p.count -= 1;
    p.stockSum  = p.stockSum  -  Number(v.Stock);
    p.stockAvg = Math.round(p.stockSum  / p.count);
    return p;
}
```

Fungsi reduceRemoveAvg() terlihat mirip tetapi kebalikannya: saat record dihapus, hitungan dan jumlah diturunkan. Rata-rata selalu menghitung dengan cara yang sama, jadi tidak perlu mengubah rumus itu. Momen kebenaran: Anda menerapkan fungsi MapReduce buatan sendiri ini ke kumpulan data:

```
dataFiltered = medNameDim.group().reduce(reduceAddAvg,
reduceRemoveAvg,reduceInitAvg)

variablesInTable = ["key","value.stockAvg"]
filteredTable
        .empty()
    .append(CreateTable(dataFiltered.top(Infinity),
variablesInTable,'Reduced Table'));
```

Business as usual: draw result table.

reduce() takes the 3 functions (reduceInitAvg(), reduceAddAvg(), and reduceRemoveAvg()) as input arguments.

Perhatikan bagaimana nama variabel output Anda berubah dari value menjadi value.stockAvg. Karena Anda mendefinisikan sendiri fungsi pengurangan, Anda dapat menampilkan banyak variabel jika Anda mau. Oleh karena itu, nilai telah berubah menjadi objek yang berisi semua variabel yang Anda hitung; stockSum dan count juga ada di sana. Hasilnya berbicara sendiri, seperti yang ditunjukkan pada gambar 9.8. Tampaknya kami telah meminjam Cimalgex dari rumah sakit lain, dengan rata-rata stok negatif. Ini semua Crossfilter yang perlu Anda ketahui untuk bekerja dengan dc.js, jadi mari lanjutkan dan tampilkan grafik interaktif tersebut.

Reduced Table

key	value.stockAvg
Adoport 1 mg	36
Atenolol EG 100 mg	49
Ceftriaxone Actavis 1 g	207
Cefuroxim Mylan 500 mg	118
Certican 0.25 mg	156
Cimalgex 8 mg	-24

Gambar 9.8 Tabel MapReduced dengan rata-rata stok per obat

9.3 MEMBUAT DASBOR INTERAKTIF DENGAN DC.JS

Sekarang setelah Anda mengetahui dasar-dasar Crossfilter, saatnya mengambil langkah terakhir: membuat dasbor. Mari mulai dengan membuat tempat untuk grafik Anda di halaman index.html. Tubuh baru terlihat seperti daftar berikut. Anda akan melihat tampilannya mirip dengan penyiapan awal kami selain dari tag placeholder grafik yang ditambahkan dan tombol reset tag <button>

Listing 9.4 A revised index.html with space for graphs generated by dc.js

```
Layout:
        Title
        | input table                          | (row 1)
        | filtered table                       | (row 2)
        [ reset button ]
        | stock-over-time chart | stock-per-medicine chart | (row 3)
        | light-sensitive chart |                          | (row 4)
          (column 1)            (column 2)

<body>
    <main class='container'>

        <h1>Chapter 10:  Data Science Application</h1>
        <div class="row">
            <div class='col-lg-12'>
                <div id="inputtable" class="well well-sm">
</div>                                          <-- This is a placeholder <div> for input data table inserted later.
            </div>
        </div>
        <div class="row">
            <div class='col-lg-12'>
                <div id="filteredtable" class="well well-sm">
</div>                                          <-- This is a placeholder <div> for filtered table inserted later.
            </div>
        </div>

This is new: reset button.
        <button class="btn btn-success">Reset Filters</button>
        <div class="row">
            <div class="col-lg-6">
                <div id="StockOverTime" class="well well-sm"></div>   <-- This is new: time chart placeholder.

This is new: light sensitivity pie-chart placeholder.
                <div id="LightSensitiveStock" class="well well-sm"></div>
            </div>
            <div class="col-lg-6">
                <div id="StockPerMedicine" class="well well-sm"></div>   <-- This is new: stock per medicine bar-chart placeholder.
            </div>
        </div>
    </main>

    <script src="https://code.jquery.com/jquery-1.9.1.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap
/3.3.0/js/bootstrap.min.js"></script>

    <script src="crossfilter.min.js"></script>
    <script src="d3.v3.min.js"></script>
    <script src="dc.min.js"></script>          Crossfilter, d3, and dc libraries can be downloaded from their respective websites.

    <script src="application.js"></script>     <-- Our own application JavaScript code.
</body>

Standard practice. JS libraries are last to speed page load.
        jQuery:       vital HTML-JavaScript interaction
        Bootstrap:    simplified CSS and layout from folks at Twitter
        Crossfilter:  our JavaScript MapReduce library of choice
        d3:           the d3 script, necessary to run dc.js
        DC:           our visualization library
        application:  our data science application; here we store all the logic
        *.min.js denotes minified JavaScript for our 3rd party libraries
```

Kami memiliki pemformatan Bootstrap, tetapi elemen yang paling penting adalah tiga tag dengan ID dan tombol. Yang ingin Anda buat adalah representasi dari total stok dari waktu ke waktu, , dengan kemungkinan memfilter obat-

obatan, , dan apakah sensitif terhadap cahaya atau tidak, . Anda juga menginginkan tombol untuk menyetel ulang semua filter, <button class="btn btn-success">Setel Ulang Filter</button>. Elemen tombol setel ulang ini tidak diperlukan, tetapi berguna.

Sekarang alihkan perhatian Anda kembali ke application.js. Di sini Anda dapat menambahkan semua kode yang akan datang di fungsi main() Anda seperti sebelumnya. Namun, ada satu pengecualian untuk aturan tersebut: dc.renderAll(); adalah perintah dc untuk menggambar grafik. Anda perlu menempatkan perintah render ini hanya sekali, di bagian bawah fungsi main() Anda. Grafik pertama yang Anda butuhkan adalah "total stok dari waktu ke waktu", seperti yang ditunjukkan pada daftar berikut. Anda sudah memiliki dimensi waktu yang dinyatakan, jadi yang Anda butuhkan hanyalah menjumlahkan stok Anda dengan dimensi waktu.

Listing 9.5 Code to generate "total stock over time" graph

```
var SummatedStockPerDay =
DateDim.group().reduceSum(function(d){return d.Stock;})

var minDate = DateDim.bottom(1)[0].Day;
var maxDate = DateDim.top(1)[0].Day;
var StockOverTimeLineChart = dc.lineChart("#StockOverTime");

StockOverTimeLineChart
        .width(null) // null means size to fit container
        .height(400)
        .dimension(DateDim)
        .group(SummatedStockPerDay)

          .x(d3.time.scale().domain([minDate,maxDate]))
          .xAxisLabel("Year 2015")
          .yAxisLabel("Stock")
          .margins({left: 60, right: 50, top: 50, bottom: 50})

dc.renderAll();
```

Stock over time data

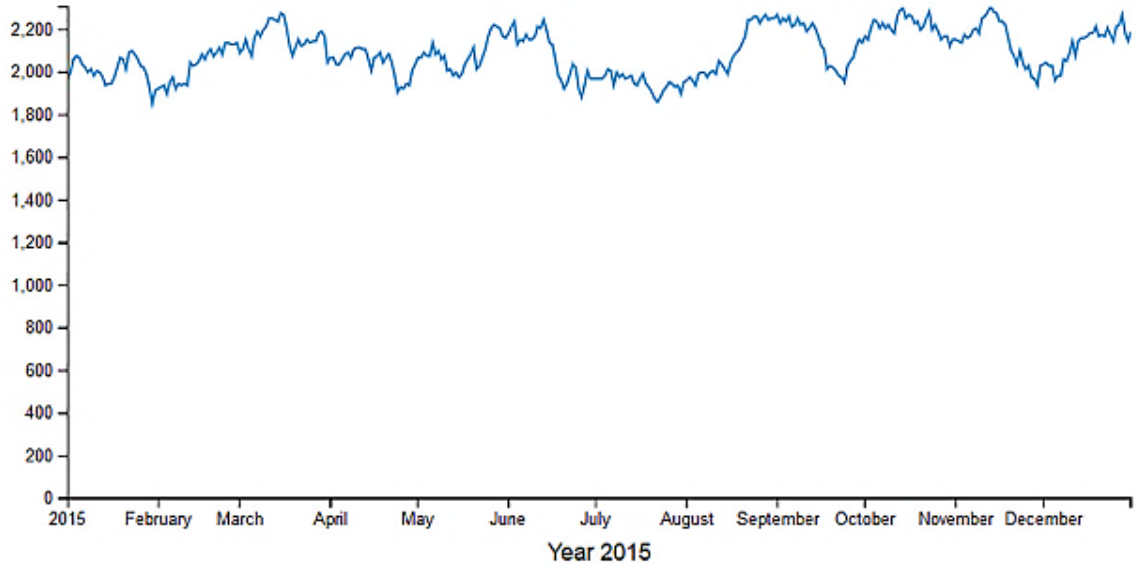
Line chart

Deliveries per day graph

Deliveries per day graph

Render all graphs

Lihatlah semua yang terjadi di sini. Pertama, Anda perlu menghitung rentang sumbu x sehingga dc.js akan mengetahui dari mana harus memulai dan mengakhiri bagan garis. Kemudian bagan garis diinisialisasi dan dikonfigurasi. Metode yang paling tidak jelas di sini adalah .group() dan .dimension(). .group() mengambil dimensi waktu dan mewakili sumbu x. .dimension() adalah mitranya, yang mewakili sumbu y dan menggunakan data ringkasan Anda sebagai input. Gambar 9.9 terlihat seperti bagan garis yang membosankan, tetapi tampilannya bisa menipu.



Gambar 9.9 grafik dc.js: jumlah stok obat selama tahun 2015

Banyak hal berubah secara drastis setelah Anda memperkenalkan elemen kedua, jadi mari buat bagan baris yang mewakili stok rata-rata per obat, seperti yang ditampilkan di daftar berikutnya.

Listing 9.6 Code to generate "average stock per medicine" graph

```
var AverageStockPerMedicineRowChart = dc.rowChart("#StockPerMedicine");
var AvgStockMedicine =  medNameDim.group().reduce(reduceAddAvg,
reduceRemoveAvg,reduceInitAvg);

AverageStockPerMedicineRowChart
                .width(null)
                .height(1200)
                .dimension(medNameDim)
                .group(AvgStockMedicine)
                .margins({top: 20, left: 10, right: 10, bottom: 20})
                .valueAccessor(function (p) {return p.value.stockAvg;});
```

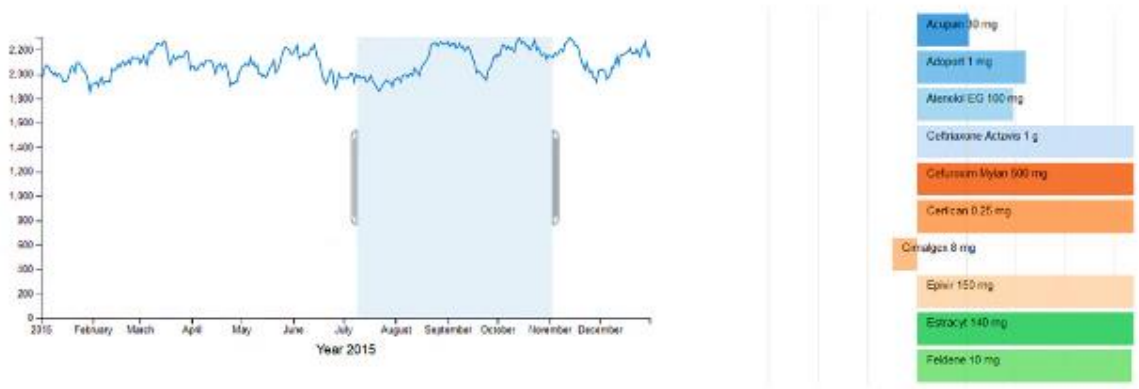
Null means "size to fit container"

Average stock per medicine row chart

Ini seharusnya sudah tidak asing lagi karena merupakan representasi grafik dari tabel yang Anda buat sebelumnya. Satu hal yang menarik: karena Anda menggunakan fungsi pengurangan () yang ditentukan khusus kali ini, dc.js tidak tahu data apa yang harus diwakili. Dengan metode .valueAccessor() Anda dapat menentukan p.value.stockAvg sebagai nilai pilihan Anda. Warna font label bagan baris dc.js berwarna abu-abu; ini membuat bagan baris Anda agak sulit dibaca. Anda dapat memperbaikinya dengan menimpa CSS-nya di file application.css Anda:

```
.dc-chart g.row text {fill: black;}
```

Satu garis sederhana dapat membuat perbedaan antara grafik yang jelas dan tidak jelas (gambar 9.10).



Gambar 9.10 bagan garis dc.js dan interaksi bagan baris

Sekarang saat Anda memilih area di bagan garis, bagan baris secara otomatis diadaptasi untuk mewakili data untuk periode waktu yang tepat. Sebaliknya, Anda dapat memilih satu atau beberapa obat pada bagan baris, yang menyebabkan bagan garis disesuaikan. Terakhir, mari kita tambahkan dimensi kepekaan cahaya sehingga apoteker dapat membedakan antara stok obat peka cahaya dan yang tidak peka cahaya, seperti yang ditunjukkan pada daftar berikut.

Listing 9.7 Adding the light-sensitivity dimension

```
        var lightSenDim = CrossfilterInstance.dimension(
function(d){return d.LightSen;});
        var SummatedStockLight =  lightSenDim.group().reduceSum(
function(d) {return d.Stock;});

        var LightSensitiveStockPieChart = dc.pieChart("#LightSensitiveStock");

    LightSensitiveStockPieChart
            .width(null) // null means size to fit container
            .height(300)
            .dimension(lightSenDim)
            .radius(90)
            .group(SummatedStockLight)
```

Kami belum memperkenalkan dimensi ringan, jadi Anda harus mendaftarkannya ke instans Crossfilter Anda terlebih dahulu. Anda juga dapat menambahkan tombol setel ulang, yang menyebabkan semua filter disetel ulang, seperti yang ditunjukkan pada daftar berikut.

Listing 9.8 The dashboard reset filters button

```
resetFilters = function(){
    StockOverTimeLineChart.filterAll();
    LightSensitiveStockPieChart.filterAll();
    AverageStockPerMedicineRowChart.filterAll();
    dc.redrawAll();
}
$('.btn-success').click(resetFilters);
```

When an element with class btn-success is clicked (our reset button), resetFilters() is called.

resetFilters() function will reset our dc.js data and redraw graphs.

Metode .filterAll() menghapus semua filter pada dimensi tertentu; dc.redraw- All() kemudian secara manual memicu semua diagram dc untuk menggambar ulang. Hasil akhirnya adalah dasbor interaktif (gambar 9.11), siap digunakan oleh apoteker kami untuk mengetahui perilaku stoknya.

Gambar 9.11 Dashboard dc.js yang sepenuhnya interaktif tentang obat-obatan dan stoknya di apotek rumah sakit

9.4 ALAT PENGEMBANGAN DASBOR

Kami sudah memiliki dasbor kami yang luar biasa, tetapi kami ingin mengakhiri bab ini dengan ikhtisar singkat (dan jauh dari lengkap) tentang pilihan perangkat lunak alternatif dalam hal menyajikan nomor Anda dengan cara yang menarik.

Anda dapat menggunakan paket perangkat lunak yang terbukti dan benar dari pengembang terkenal seperti Tableau, MicroStrategy, Qlik, SAP, IBM, SAS, Microsoft, Spotfire, dan sebagainya. Semua perusahaan ini menawarkan alat dasbor yang patut diselidiki. Jika Anda bekerja di perusahaan besar, kemungkinan besar Anda memiliki setidaknya satu alat berbayar yang dapat Anda gunakan. Pengembang juga dapat menawarkan versi publik gratis dengan fungsi terbatas. Pasti periksa Tableau jika Anda belum melakukannya di http://www.tableausoftware.com/public/download.

Perusahaan lain setidaknya akan memberi Anda versi percobaan. Pada akhirnya Anda harus membayar untuk versi lengkap dari salah satu paket ini, dan mungkin sepadan, terutama untuk perusahaan besar yang mampu membelinya. Namun, fokus utama buku ini adalah pada alat gratis. Saat melihat alat visualisasi data gratis, Anda dengan cepat berakhir di dunia HTML,

yang berkembang biak dengan pustaka JavaScript gratis untuk memplot data apa pun yang Anda inginkan. Lanskapnya sangat besar:

- **HighCharts—**Salah satu pustaka grafik berbasis browser yang paling matang. Lisensi gratis hanya berlaku untuk pengejaran nonkomersial. Jika Anda ingin menggunakannya dalam konteks komersial, harganya berkisar antara $90 hingga $4000. Lihat http://toko.highsoft.com/highcharts.html.
- **Chartkick—**Pustaka charting JavaScript untuk penggemar Ruby on Rails. Lihat http://ank-ane.github.io/chartkick/.
- **Google Charts—**Pustaka charting gratis dari Google. Seperti banyak produk Google lainnya, ini gratis untuk digunakan, bahkan secara komersial, dan menawarkan berbagai macam grafik. Lihat https://developers.google.com/chart/.
- **d3.js—**Ini aneh karena ini bukan pustaka grafik tetapi pustaka visualisasi data. Perbedaannya mungkin terdengar halus tetapi implikasinya tidak. Sedangkan pustaka seperti HighCharts dan Google Charts dimaksudkan untuk menggambar bagan tertentu yang telah ditentukan sebelumnya, d3.js tidak memberikan batasan seperti itu. d3.js saat ini adalah perpustakaan visualisasi data JavaScript paling serbaguna yang tersedia. Anda hanya perlu mengintip sekilas contoh interaktif di situs web resmi untuk memahami perbedaan dari perpustakaan pembuat grafik biasa. Lihat http://d3js.org/.

Tentu saja, yang lain tersedia yang belum kami sebutkan. Anda juga bisa mendapatkan perpustakaan visualisasi yang hanya datang dengan masa percobaan dan tidak ada edisi komunitas gratis, seperti Wijmo, Kendo, dan FusionCharts. Mereka layak diperhatikan karena mereka juga memberikan dukungan dan menjamin pembaruan rutin.

Tetapi mengapa atau kapan Anda bahkan mempertimbangkan untuk membangun antarmuka Anda sendiri dengan HTML5 alih-alih menggunakan alternatif seperti BusinessObjects SAP, SAS JMP, Tableau, Clickview, atau salah satu dari banyak lainnya? Berikut beberapa alasannya:

- **Tanpa anggaran—**Saat Anda bekerja di startup atau perusahaan kecil lainnya, biaya lisensi yang menyertai perangkat lunak semacam ini bisa tinggi.
- **Aksesibilitas tinggi—**Aplikasi ilmu data dimaksudkan untuk merilis hasil ke semua jenis pengguna, terutama orang yang mungkin hanya memiliki browser—pelanggan Anda sendiri, misalnya. Visualisasi data dalam HTML5 berjalan dengan lancar di perangkat seluler.
- **Sejumlah besar talenta di luar sana—**Meskipun tidak banyak developer Tableau, banyak orang yang memiliki keterampilan pengembangan web. Saat merencanakan sebuah proyek, penting untuk mempertimbangkan apakah Anda dapat mengelolanya.
- **Rilis cepat—**Melalui seluruh siklus TI mungkin memakan waktu terlalu lama di perusahaan Anda, dan Anda ingin orang-orang menikmati analisis Anda dengan cepat. Setelah antarmuka Anda tersedia dan digunakan, TI dapat menghabiskan waktu yang mereka inginkan untuk mengindustrialisasi produk.

- **Pembuatan Prototipe—**Semakin baik Anda menunjukkan kepada TI tujuannya dan apa yang seharusnya mampu dilakukannya, semakin mudah bagi mereka untuk membangun atau membeli aplikasi berkelanjutan yang melakukan apa yang Anda inginkan.
- **Dapat dikustomisasi—**Meskipun paket perangkat lunak yang dibuat sangat bagus dalam fungsinya, aplikasi tidak akan pernah dapat dikustomisasi seperti saat Anda membuatnya sendiri.

Dan mengapa Anda tidak melakukan ini?

- **Kebijakan perusahaan—**Ini yang terbesar: tidak diperbolehkan. Perusahaan besar memiliki tim cadangan TI yang hanya mengizinkan sejumlah alat tertentu untuk digunakan sehingga peran pendukung mereka tetap terkendali.
- **Anda memiliki tim reporter yang berpengalaman—**Anda akan melakukan pekerjaan mereka, dan mereka mungkin mendatangi Anda dengan garpu rumput.
- **Alat Anda memungkinkan penyesuaian yang cukup untuk memenuhi selera Anda—**Beberapa platform yang lebih besar adalah antarmuka browser dengan JavaScript yang berjalan di bawah terpal. Tableau, BusinessObjects Webi, SAS Visual Analytics, dan seterusnya semuanya memiliki antarmuka HTML; toleransi mereka terhadap penyesuaian mungkin tumbuh seiring waktu.

Ujung depan aplikasi apa pun dapat memenangkan hati banyak orang. Semua kerja keras yang Anda lakukan dalam persiapan data dan analitik mewah yang Anda terapkan hanya bernilai sebanyak yang dapat Anda sampaikan kepada mereka yang menggunakannya. Sekarang Anda berada di jalur yang benar untuk mencapai ini. Pada catatan positif ini kami akan menyimpulkan bab ini.

9.5 RINGKASAN

- Bab ini berfokus pada bagian terakhir dari proses ilmu data, dan tujuan kami adalah membangun aplikasi ilmu data di mana pengguna akhir dilengkapi dengan dasbor interaktif. Setelah melalui semua langkah proses ilmu data, kami disajikan dengan data yang bersih, seringkali padat atau padat informasi. Dengan cara ini kami dapat meminta lebih sedikit data dan mendapatkan wawasan yang kami inginkan.
- Dalam contoh kami, data stok apotek dianggap telah dibersihkan dan disiapkan secara menyeluruh dan hal ini harus selalu dilakukan pada saat informasi tersebut sampai ke pengguna akhir.
- Dasbor berbasis JavaScript sempurna untuk memberikan akses cepat ke hasil ilmu data Anda karena hanya mengharuskan pengguna untuk memiliki browser web. Ada alternatif lain, seperti Qlik (bab 5).
- Crossfilter adalah pustaka MapReduce, salah satu dari banyak pustaka JavaScript MapReduce, tetapi telah terbukti stabilitasnya dan sedang dikembangkan serta digunakan oleh Square, sebuah perusahaan yang melakukan transaksi moneter. Menerapkan MapReduce efektif, bahkan pada satu node dan di browser; itu meningkatkan kecepatan perhitungan.

- dc.js adalah pustaka bagan yang dibangun di atas d3.js dan Crossfilter yang memungkinkan pembuatan dasbor browser dengan cepat.
- Kami menjelajahi kumpulan data apotek rumah sakit dan membuat dasbor interaktif untuk apoteker. Kekuatan dasbor adalah sifatnya yang melayani sendiri: mereka tidak selalu membutuhkan reporter atau ilmuwan data untuk memberikan wawasan yang mereka dambakan.
- Tersedia alternatif visualisasi data, dan sebaiknya luangkan waktu untuk menemukan alternatif yang paling sesuai dengan kebutuhan Anda.
- Ada beberapa alasan mengapa Anda membuat laporan ubahsuaian sendiri alih-alih memilih alat perusahaan (yang seringkali lebih mahal) di luar sana:
  – Tanpa anggaran—Startup tidak selalu mampu membeli setiap alat
  – Aksesibilitas tinggi—Setiap orang memiliki browser
  – Bakat yang tersedia—(Relatif) akses mudah ke pengembang JavaScript
  – Rilis cepat—siklus TI dapat memakan waktu cukup lama
  – Pembuatan Prototipe—Aplikasi prototipe dapat menyediakan dan memberikan waktu bagi TI untuk membuat versi produksi
  – Dapat disesuaikan—Kadang-kadang Anda hanya menginginkannya persis seperti yang Anda bayangkan.
- Tentu saja ada alasan untuk tidak mengembangkan aplikasi Anda sendiri:
  – Kebijakan perusahaan—Proliferasi aplikasi bukanlah hal yang baik dan perusahaan mungkin ingin mencegahnya dengan membatasi pengembangan lokal.
  – Tim pelaporan yang matang—Jika Anda memiliki departemen pelaporan yang baik, mengapa Anda masih repot?
  – Kustomisasi memuaskan —Tidak semua orang menginginkan barang yang mengkilap; dasar bisa cukup.

Selamat! Anda telah mencapai bagian akhir buku ini dan awal sebenarnya dari karier Anda sebagai ilmuwan data. Kami harap Anda sangat senang membaca dan mempelajari contoh dan studi kasus. Sekarang setelah Anda memiliki wawasan dasar tentang dunia ilmu data, terserah Anda untuk memilih jalan. Cerita berlanjut, dan kami semua berharap Anda sukses besar dalam pencarian Anda untuk menjadi ilmuwan data terhebat yang pernah hidup.

DAFTAR PUSTAKA

A. M. Law and W. D. Kelton. Simulation Modeling and Analysis. McGraw-Hill, New York, third edition, 2000

Anonymous, â€œBig Data Analytics in Cyber Defenseâ€, Ponemon Institute Research Report, 2013.

B. D. Ripley. Stochastic Simulation. John Wiley & Sons, New York, 1987.

C. D. Manning, P. Raghavan, and H. Schutze, An Introduction to Information Retrieval. Cambridge, England: Cambridge University Press, 2009.

C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning. MIT Press, Cambridge, 2006.

C. Hayashi, K. Yajima, H. Bock, N, Ohsumi, Y, Tanaka, and Y, Baba, (Eds.), â€œData Science, Classification, and Related Methodsâ€, Proceedings of the Fifth Conference of the International Federation of Classification Societies (IFCS-96), Kobe, Japan, March 27â€“30, 1996, Springer Science & Business Media, 2013.

C. P. Chen & C. Y. Zhang, â€œData-intensive applications, challenges, techniques and technologies: A survey on Big Dataâ€, Information Sciences Vol. 275, 2014.

D. Normawati and S. A. Prayogi, “Implementasi Naïve Bayes Classifier Dan Confusion Matrix Pada Analisis Sentimen Berbasis Teks Pada Twitter,” J-SAKTI (Jurnal Sains Komput. dan Inform., vol. 5, no. 2, pp. 697–711, Sep. 2021.

D. P. Kroese and J. C. C. Chan. Statistical Modeling and Computation. Springer, 2014.

Doug Cackett,"Information Management and Big data: A Reference Architecture", Oracle: Redwood City, CA, USA, 2013.

Faisal, Irwan Budiman, Erick Kurniawan. Belajar Data SciencePengenalan Azure Machine Learning Studio. Vol. Cetakan 1. Banjarbaru, Kalimantan Selatan, Indonesia: Scripta Cendekia, 2019.

G. H. Golub and C. F. Van Loan. Matrix Computations. Johns Hopkins University Press, Baltimore, fourth edition, 2013.

G. Strang. Linear Algebra and Learning from Data. Wellesley–Cambridge Press, Cambridge, 2019.

H. Wendland. Scattered Data Approximation. Cambridge University Press, Cambridge, 2005.

I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, Cambridge, 2016.

J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, & A. H. Byers, â€œBig data: The next frontier for innovation, competition, and productivityâ€, McKinsey Publication, 2011.

K. P. Murphy. Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge, 2012.

L Xu, C. Jiang, J. Wang, J. Yuan, & Y. Ren, â€œInformation security in big data: privacy and data miningâ€, IEEE Access, 2, 2014.

M. Lutz. Learning Python. O'Reilly, fifth edition, 2013.

M. Reza Faisal, Dodon T. Nugrahadi. Belajar Data ScienceKlasifikasi Dengan Bahasa Pemrograman R. Scripta Cendekia Cetakan I. Banjarbaru, Kalimantan Selatan, Indonesia: Scripta Cendekia, 2019

Müller, A Introduction to Machine Learning with Python: A Guide for Data Scientists 1st Edition. In A. Müller Introduction to MachineLearning with Python: A Guide for Data Scientists 1st Edition (p. 19).O'Reilly Media; 2016

N Falessi, R Gavrila, MR Klejnstrup, K Moulinos.National cyber security strategies: practical guide on development and execution. European Network and Information Security Agency (ENISA) Publication, 2012.

P. Zikopoulos, & C. Eaton, â€œUnderstanding big data: Analytics for enterprise class hadoop and streaming dataâ€, McGraw-Hill Osborne Media, 2011.

R. Agrawal, A. Imran, C. Seay, & J. Walker, â€œA layer based architecture for provenance in big dataâ€, In Big Data (Big Data), 2014 IEEE International Conference on Big Data, 2014.

R. Alguliyev & Y. Imamverdiyev, â€œBig data: big promises for information security, In Application of Information and Communication Technologies (AICT), IEEE 8th International Conference on Oct 15, 2014.

R. J. A. Little and D. B. Rubin. Statistical Analysis with Missing Data. John Wiley & Sons, Hoboken, second edition, 2002.

R. Schutt, and C. O'Neil, â€œDoing data science: Straight talk from the frontline", O'Reilly Media, Inc.", 2013.

S Curry, E Kirda, E Schwartz, WH Stewart, and A Yoran, A. Big data fuels intelligence-driven security. RSA Security Brief Report, 2013.

S. B Siewert, â€œBig data in the cloud: Data velocity, volume, variety, veracityâ€, IBM Developerworks, 2013.

Staffan TruvÃ©, â€Machine Learning in Cyber Security: Age of the Centaursâ€, Threat Intelligence

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, â€œFrom data mining to knowledge discovery in databasesâ€ AI magazine, 1996.

U. Ganugula, & A. Saxena, â€œHigh Performance Cryptography: Need of the Hourâ€ CSI Communications Magazine, 2012.

Vashant Dhar, "Data science and prediction", Communications of the ACM, Vol. 56, No.12, 2013.

Veronika S. Moertini, Mariskha T. Adithia. Pengantar Data Science Dan Aplikasinya Bagi Pemula. Bandung: Unpar Press, Bandung Indonesia,2020

W. McKinney. Python for Data Analysis. O'Reilly Media, Inc., second edition, 2017.

Whitepaper, Recorded Future, 2017.

Wu, X., X. Zhu, X., G. Q. Wu, and W. Ding, â€œData Mining With Big Data.â€ IEEE Transactions On Knowledge And Data Engineering 26 (1), 2014.

Y. Sari, M. Maulida, E. Gunawan, and J. Wahyudi, "Artificial Intelligence Approach for BAZNAS Website Using K-Nearest Neighbor (KNN)," 2021 6th Int. Conf. Informatics Comput. ICIC 2021.

ILMU DATA
(Data Science)

Oleh:
Dr. Joseph Santoso, S.Kom, M.Kom

BIODATA PENULIS

Dr. Joseph Teguh Santoso, S.Kom, M.Kom adalah Rektor dari Universitas Sains & Teknologi Komputer (Universitas STEKOM) Semarang yang memiliki banyak pengalaman praktis dalam bidang *e-commerce*sejak Tahun 2002. Beliau mempunyai 3 (tiga) toko *Official Online Store* di China untuk merek Sepeda Raleigh, dengan omzet tahunan pada Tahun 2019 mencapai lebih dari Rp. 35 Milyar rupiah dan terus meningkat. Dr. Joseph T.S memiliki lisensi tunggal sepeda merek "Raleigh" untuk penjualan *Online* di seluruh China. Di samping itu beliau juga memiliki pabrik sepeda dan sepeda listrik merek "Fengjiu", yaitu Pabrik Sepeda Listrik yang masih tergolong kecil di China. Pengalaman beliau malang melintang di dunia *online store* di China seperti Alibaba, Tmall, Taobao, JD, Aliexpress sangat membantu mahasiswa untuk memiliki pengalaman teknis dan praktis untuk membuka toko *online* bersama beliau.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8120-50-5 (PDF)