3 Aprendizado de máquina

Este capítulo cobre

- ÿ Compreender por que os cientistas de dados usam o aprendizado de máquina
- ÿ Identificando as bibliotecas Python mais importantes para aprendizado de máquina
- ÿ Discutir o processo de construção de modelo ÿ Usar técnicas de aprendizado de máquina ÿ Ganhar experiência prática com aprendizado de máquina

Você sabe como os computadores aprendem a protegê-lo de pessoas mal-intencionadas? Os computadores filtram mais de 60% dos seus e-mails e podem aprender a fazer um trabalho ainda melhor para protegê-lo ao longo do tempo.

Você pode ensinar explicitamente um computador a reconhecer pessoas em uma imagem? É possível, mas impraticável, codificar todas as formas possíveis de reconhecer uma pessoa, mas você logo verá que as possibilidades são quase infinitas. Para ter sucesso, você precisará adicionar uma nova habilidade ao seu kit de ferramentas, *o aprendizado de máquina,* que é o tema deste capítulo.

3.1 O que é aprendizado de máquina e por que você deveria se importar sobre isso?

> "O aprendizado de máquina é um campo de estudo que dá aos computadores a capacidade de aprender sem ser explicitamente programado."
>
> —Arthur Samuel, 19591

A definição de aprendizado de máquina cunhada por Arthur Samuel é frequentemente citada e é genial em sua amplitude, mas deixa você com a questão de como o computador aprende. Para alcançar o aprendizado de máquina, os especialistas desenvolvem algoritmos de uso geral que pode ser usado em grandes classes de problemas de aprendizagem. Quando você quiser resolver um problema *específico tarefa* você só precisa alimentar o algoritmo com *dados mais específicos.* De certa forma, você está programando através do exemplo. Na maioria dos casos, um computador usará dados como fonte de informação e compare sua saída com uma saída desejada e corrija-a. Quanto mais dados ou "experiência" que o computador obtém, melhor ele se torna na função designada, como um humano faz.

Quando o aprendizado de máquina é visto como um processo, a seguinte definição é esclarecedora:

> "Aprendizado de máquina é o processo pelo qual um computador pode trabalhar mais com precisão à medida que coleta e aprende com os dados que lhe são fornecidos."
>
> —Mike Roberts2

Por exemplo, à medida que um usuário escreve mais mensagens de texto em um telefone, o telefone aprende mais sobre o vocabulário comum das mensagens e pode prever (preencher automaticamente) suas palavras com mais rapidez e precisão.

No campo mais amplo da ciência, o aprendizado de máquina é um subcampo da inteligência artificial e está intimamente relacionado à matemática aplicada e à estatística. Tudo isso pode soar um pouco abstrato, mas o aprendizado de máquina tem muitas aplicações na vida cotidiana.

3.1.1 Aplicações para aprendizado de máquina em ciência de dados

A regressão e *a classificação* são de fundamental importância para um cientista de dados. Alcançar Para atingir esses objetivos, uma das principais ferramentas que um cientista de dados usa é o aprendizado de máquina. Os usos para regressão e classificação automática são abrangentes, como os seguintes:

- Encontrar campos de petróleo, minas de ouro ou sítios arqueológicos com base em sítios existentes (classificação e regressão)
- Encontrar nomes de lugares ou pessoas no texto (classificação)
- Identificação de pessoas com base em imagens ou gravações de voz (classificação)
- Reconhecer aves com base no seu assobio (classificação)

---

1 Embora o artigo a seguir seja frequentemente citado como fonte desta citação, ele não está presente na reimpressão de 1967 de aquele papel. Os autores não conseguiram verificar ou encontrar a fonte exata desta citação. Veja Arthur L. Samuel, "Alguns estudos em aprendizado de máquina usando o jogo de damas", *IBM Journal of Research and Development* 3, não. 3 (1959):210–229.

2 Mike Roberts é o editor técnico deste livro. Obrigado, Mike.

- ÿ Identificação de clientes rentáveis (regressão e classificação)
- ÿ Identificação proativa de peças de automóveis com probabilidade de falhar (regressão)
- ÿ Identificação de tumores e doenças (classificação)
- ÿ Prever a quantidade de dinheiro que uma pessoa gastará no produto X (regressão)
- ÿ Prever o número de erupções de um vulcão em um período (regressão)
- ÿ Prever a receita anual da sua empresa (regressão)
- ÿ Prever qual time vencerá a Liga dos Campeões no futebol (classificação)

Ocasionalmente, os cientistas de dados constroem um *modelo* (uma abstração da realidade) que fornece compreensão dos processos subjacentes de um fenômeno. Quando o objetivo de um modelo não é previsão, mas interpretação, é chamada de *análise de causa raiz*. Aqui estão alguns exemplos:

- ÿ Compreender e otimizar um processo de negócios, como determinar quais produtos agregam valor a uma linha de produtos
- ÿ Descobrindo o que causa o diabetes
- ÿ Determinar as causas dos engarrafamentos

Esta lista de aplicativos de aprendizado de máquina só pode ser vista como um aperitivo porque é onipresente na ciência de dados. Regressão e classificação são duas técnicas importantes, mas o repertório e as aplicações não terminam, sendo o clustering uma outra
exemplo de uma técnica valiosa. Técnicas de aprendizado de máquina podem ser usadas durante todo o processo de ciência de dados, como discutiremos na próxima seção.

3.1.2 Onde o aprendizado de máquina é usado no processo de ciência de dados

Embora o aprendizado de máquina esteja principalmente ligado à etapa de modelagem de dados do processo de ciência de dados, ele pode ser usado em quase todas as etapas. Para refrescar sua memória dos capítulos anteriores, o processo de ciência de dados é mostrado na figura 3.1.

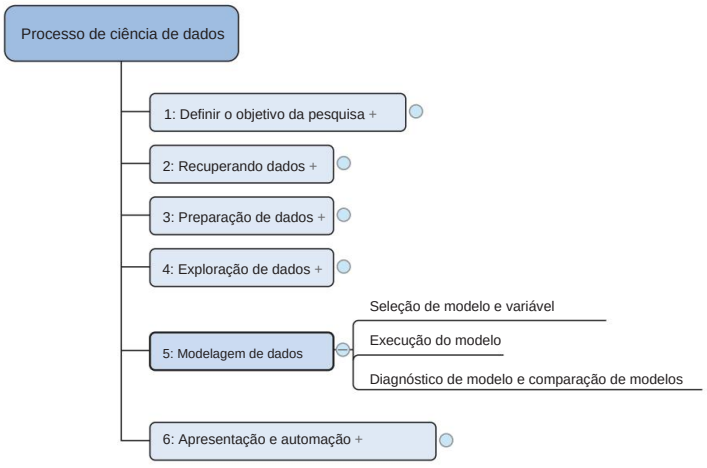


Figura 3.1 O processo de ciência de dados

A fase de modelagem de dados não pode começar até que você tenha dados brutos qualitativos que possa compreender. Mas antes disso, a fase *de preparação de dados* pode se beneficiar do uso do aprendizado de máquina. Um exemplo seria limpar uma lista de strings de texto; o aprendizado de máquina pode agrupar strings semelhantes para que seja mais fácil corrigir erros ortográficos.

O aprendizado de máquina também é útil ao *explorar dados.* Algoritmos podem erradicar problemas padrões mentirosos nos dados onde seriam difíceis de encontrar apenas com gráficos.

Dado que o aprendizado de máquina é útil em todo o processo de ciência de dados, não deveria ser surpresa que um número considerável de bibliotecas Python tenha sido desenvolvido para tornar sua vida um pouco mais fácil.

3.1.3 Ferramentas Python usadas em aprendizado de máquina

Python possui um número impressionante de pacotes que podem ser usados em um ambiente de aprendizado de máquina. O ecossistema de aprendizado de máquina Python pode ser dividido em três tipos principais de pacotes, conforme mostrado na figura 3.2.

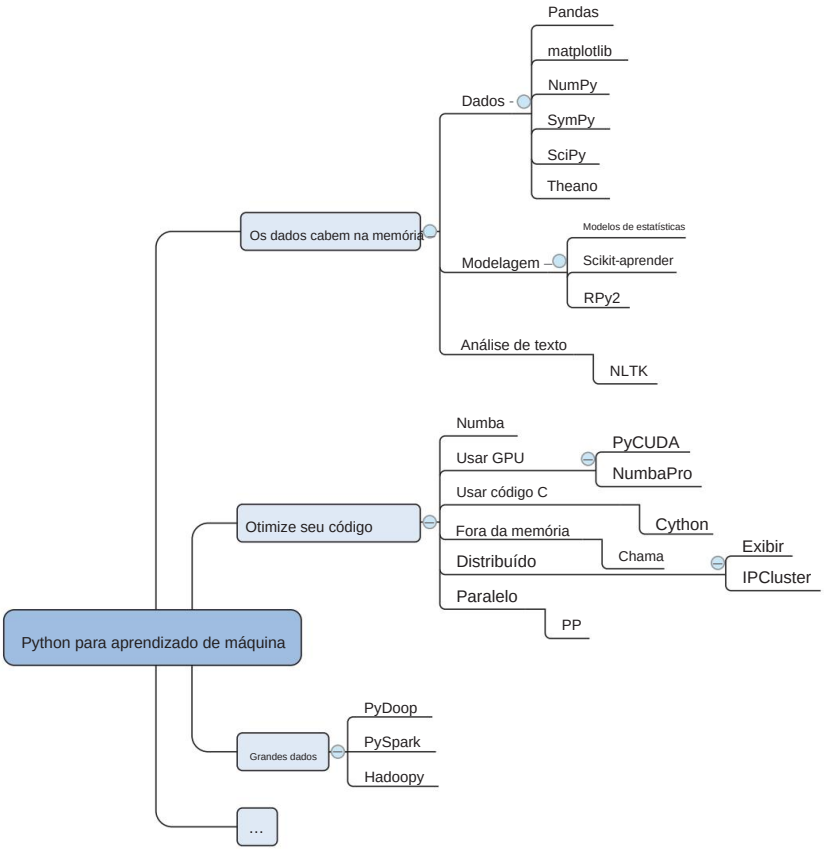


Figura 3.2 Visão geral dos pacotes Python usados durante a fase de aprendizado de máquina

O primeiro tipo de pacote mostrado na figura 3.2 é usado principalmente em tarefas simples e quando os dados cabem na memória. O segundo tipo é usado para otimizar seu código quando você termina a prototipagem e enfrenta problemas de velocidade ou memória. O terceiro tipo é específico para usar Python com tecnologias de big data.

PACOTES PARA TRABALHAR COM DADOS NA MEMÓRIA

Durante a prototipagem, os seguintes pacotes podem ajudá-lo a começar, fornecendo funcionalidades avançadas com algumas linhas de código:

ÿ *SciPy* é uma biblioteca que integra pacotes fundamentais frequentemente usados em computação científica, como NumPy, matplotlib, Pandas e SymPy. ÿ *NumPy* oferece acesso a
poderosas funções de array e funções de álgebra linear. ÿ *Matplotlib* é um pacote popular de plotagem 2D com algumas funcionalidades 3D . ÿ *Pandas* é um pacote de processamento de dados de alto desempenho, mas fácil de usar. Ele introduz dataframes no Python, um tipo de tabela de dados na memória. É um conceito que deve soar familiar aos usuários regulares do R.

ÿ *SymPy* é um pacote usado para matemática simbólica e álgebra computacional. ÿ *StatsModels* é um pacote para métodos e algoritmos estatísticos. ÿ *Scikit-learn* é uma biblioteca repleta de algoritmos de aprendizado de máquina. ÿ *RPy2* permite chamar funções R de dentro do Python. R é um popular programa de estatísticas de código aberto. ÿ *NLTK* (Natural Language Toolkit) é um kit de ferramentas
Python com foco em análise de texto.

Essas bibliotecas são boas para começar, mas quando você toma a decisão de executar um determinado programa Python em intervalos frequentes, o desempenho entra em jogo.

OTIMIZANDO OPERAÇÕES

Depois que seu aplicativo entrar em produção, as bibliotecas listadas aqui poderão ajudá-lo a fornecer a velocidade necessária. Às vezes, isso envolve a conexão com infraestruturas de big data, como Hadoop e Spark. ÿ *Numba e NumbaPro* — usam compilação just-in-time
para acelerar aplicativos escritos diretamente em Python e algumas anotações. NumbaPro também permite que você use o poder da sua unidade de processador gráfico (GPU). ÿ *PyCUDA* — Permite escrever código que será executado na GPU em vez de na CPU e, portanto, é ideal para
aplicativos com muitos cálculos. Funciona melhor com problemas que podem ser paralelizados e precisam de pouca entrada em comparação com o número de ciclos de computação necessários. Um exemplo é estudar a robustez de suas previsões calculando milhares de resultados diferentes com base em um único estado inicial.

ÿ *Cython, ou C para Python* – traz a linguagem de programação C para Python. C é uma linguagem de nível inferior, portanto o código está mais próximo do que o computador eventualmente usa (bytecode). Quanto mais próximo o código estiver de bits e bytes, mais rápido ele será executado. Um computador também é mais rápido quando conhece o tipo de uma variável (chamada *digitação estática).* Python não foi projetado para fazer isso e Cython ajuda você a superar essa deficiência.

ÿ Blaze—O Blaze oferece estruturas de dados que podem ser maiores que as do seu computador memória principal, permitindo trabalhar com grandes conjuntos de dados.

ÿ *Dispy e IPCluster — Esses* pacotes permitem escrever código que pode ser distribuído em um cluster de computadores. ÿ PP—Python é

executado como um processo único por padrão. Com a ajuda do PP você pode paralelizar cálculos em uma única máquina ou em clusters. ÿ *Pydoop e Hadoopy—Eles* conectam Python ao

Hadoop, um sistema comum de big data estrutura.

ÿ *PySpark* — conecta Python e Spark, uma estrutura de big data na memória.

Agora que você teve uma visão geral das bibliotecas disponíveis, vamos dar uma olhada no processo de modelagem em si.

3.2 O processo de modelagem A fase de

modelagem consiste em quatro etapas:

1 Engenharia de recursos e seleção de modelo

2 Treinando o modelo

3 Validação e seleção do modelo

4 Aplicando o modelo treinado a dados não vistos

Antes de encontrar um bom modelo, você provavelmente irá repetir as três primeiras etapas.

A última etapa nem sempre está presente porque às vezes o objetivo não é a previsão, mas a explicação (análise da causa raiz). Por exemplo, você pode querer descobrir as causas da extinção de espécies, mas não necessariamente prever qual delas será a próxima a deixar nosso planeta.

É possível *encadear* ou *combinar* múltiplas técnicas. Ao encadear vários modelos, a saída do primeiro modelo se torna uma entrada para o segundo modelo. Ao combinar vários modelos, você os treina de forma independente e combina seus resultados. Esta última técnica também é conhecida como *aprendizagem em conjunto.*

Um modelo consiste em construções de informações chamadas *recursos* ou *preditores* e uma variável *de destino* ou *resposta.* O objetivo do seu modelo é prever a variável alvo, por exemplo, a alta temperatura de amanhã. As variáveis que ajudam você a fazer isso e são (geralmente) conhecidas por você são os recursos ou variáveis preditoras, como a temperatura atual, movimentos das nuvens, velocidade atual do vento e assim por diante. Os melhores modelos são aqueles que representam a realidade com precisão, de preferência concisos e interpretáveis. Para conseguir isso, a engenharia de recursos é a parte mais importante e possivelmente mais interessante da modelagem. Por exemplo, uma característica importante num modelo que tentou explicar a extinção de grandes animais terrestres nos últimos 60 mil anos na Austrália revelou-se o número populacional e a dispersão dos seres humanos.

3.2.1 Recursos de engenharia e seleção de modelo

Com recursos de engenharia, você deve criar e criar possíveis preditores para o modelo. Esta é uma das etapas mais importantes do processo porque um modelo

recombina esses recursos para alcançar suas previsões. Freqüentemente, você pode precisar consultar um especialista ou a literatura apropriada para encontrar recursos significativos.

Certas características são as variáveis que você obtém de um conjunto de dados, como é o caso dos conjuntos de dados fornecidos em nossos exercícios e na maioria dos exercícios escolares. Na prática, você mesmo precisará encontrar os recursos, que podem estar espalhados entre diferentes conjuntos de dados. Em vários projetos tivemos que reunir mais de 20 fontes de dados diferentes antes de termos os dados brutos necessários. Muitas vezes você precisará aplicar uma transformação a uma entrada antes que ela se torne um bom preditor ou combinar múltiplas entradas. Um exemplo de combinação de múltiplos inputs seriam *as variáveis de interacção:* o impacto de qualquer uma das variáveis é baixo, mas se ambas estiverem presentes o seu impacto torna-se imenso. Isto é especialmente verdadeiro em ambientes químicos e médicos. Por exemplo, embora o vinagre e a água sanitária sejam produtos domésticos comuns bastante inofensivos por si só, misturá-los resulta em gás cloro venenoso, um gás que matou milhares de pessoas durante a Primeira Guerra Mundial.

Na medicina, a farmácia clínica é uma disciplina dedicada a pesquisar o efeito da interação dos medicamentos. Este é um trabalho importante e não precisa nem envolver dois medicamentos para produzir resultados potencialmente perigosos. Por exemplo, misturar um medicamento antifúngico como o Sporanox com toranja tem efeitos colaterais graves.

Às vezes é necessário usar técnicas de modelagem para derivar recursos: a saída de um modelo torna-se parte de outro modelo. Isso não é incomum, especialmente na mineração de texto. Os documentos podem primeiro ser anotados para classificar o conteúdo em categorias, ou você pode contar o número de locais geográficos ou pessoas no texto. Esta contagem é muitas vezes mais difícil do que parece; os modelos são aplicados primeiro para reconhecer certas palavras como uma pessoa ou um lugar. Todas essas novas informações são então inseridas no modelo que você deseja construir. Um dos maiores erros na construção de modelos é o *viés de disponibilidade:* seus recursos são apenas aqueles que você poderia facilmente obter e seu modelo, consequentemente, representa essa “verdade” unilateral. Os modelos que sofrem de viés de disponibilidade muitas vezes falham quando são validados porque fica claro que não são uma representação válida da verdade.

Na Segunda Guerra Mundial, após bombardeios em território alemão, muitos dos aviões ingleses voltaram com buracos de bala nas asas, ao redor do nariz e perto da cauda do avião. Quase nenhum deles tinha buracos de bala na cabine, no leme traseiro ou no bloco do motor, então a engenharia decidiu que uma blindagem extra deveria ser adicionada às asas. Isto parecia uma boa ideia até que um matemático chamado Abraham Wald explicou a obviedade do seu erro: eles apenas levaram em conta os aviões que regressaram. Os buracos de bala nas asas eram, na verdade, a menor preocupação deles, porque pelo menos um avião com esse tipo de dano poderia voltar para casa para reparos.

A fortificação dos aviões foi, portanto, aumentada nos locais que permaneceram ilesos no retorno dos aviões. O raciocínio inicial sofria de viés de disponibilidade: os engenheiros ignoraram uma parte importante dos dados porque eram mais difíceis de obter. Neste caso tiveram sorte, porque o raciocínio poderia ser invertido para obter o resultado pretendido sem obter os dados dos aviões acidentados.

Quando os recursos iniciais são criados, um modelo pode ser *treinado* para os dados.

3.2.2 Treinando seu modelo

Com os preditores corretos implementados e uma técnica de modelagem em mente, você pode avançar para o treinamento de modelo. Nesta fase você apresenta ao seu modelo os dados dos quais ele pode aprender.

As técnicas de modelagem mais comuns têm implementações prontas para a indústria em quase todas as linguagens de programação, incluindo Python. Isso permite que você treine seus modelos executando algumas linhas de código. Para uma ciência de dados mais avançada técnicas, você provavelmente acabará fazendo cálculos matemáticos pesados e implementando-os com técnicas modernas de ciência da computação.

Depois que um modelo for treinado, é hora de testar se ele pode ser extrapolado para a realidade: validação do modelo.

3.2.3 Validando um modelo

A ciência de dados tem muitas técnicas de modelagem, e a questão é qual delas é a mais adequada. certo para usar. Um bom modelo tem duas propriedades: tem bom poder preditivo e generaliza bem para dados que não viu. Para conseguir isso, você define uma medida de erro (quão errado está o modelo) e uma estratégia de validação.

Duas *medidas de erro* comuns no aprendizado de máquina são a *taxa de erro de classificação* para problemas de classificação e o *erro quadrático médio* para problemas de regressão. A taxa de erro de classificação é a porcentagem de observações no conjunto de dados de teste que seu modelo rotulado incorretamente; menor é melhor. O erro quadrático médio mede o tamanho da média erro de sua previsão é. Elevar o erro médio ao quadrado tem duas consequências: você não pode cancelar uma previsão errada em uma direção com uma previsão defeituosa na outra direção. Por exemplo, superestimar o faturamento futuro para o próximo mês em 5.000 não cancela a subestimação em 5.000 para o mês seguinte. Como um segundo consequência da quadratura, erros maiores ganham ainda mais peso do que de outra forma seria. Erros pequenos permanecem pequenos ou podem até diminuir (se <1), enquanto erros grandes são ampliado e com certeza chamará sua atenção.

Existem muitas *estratégias de validação ,* incluindo as seguintes: ÿ *Dividir seus dados em um conjunto de treinamento com X% das observações e manter o restante como um conjunto de dados de validação* (um conjunto de dados que nunca é usado para criação de modelo) — Este é o técnica mais comum.

ÿ *Validação cruzada K-folds - Esta* estratégia divide o conjunto de dados em k partes e usa cada parte uma vez como um conjunto de dados de teste enquanto usa as outras como um conjunto de dados de treinamento. Isso tem a vantagem de usar todos os dados disponíveis no conjunto de dados.

ÿ *Deixar-1* de fora—Essa abordagem é igual às k-dobras, mas com k=1. Você sempre deixe uma observação de fora e treine com o restante dos dados. Isso é usado apenas em pequenos conjuntos de dados, por isso é mais valioso para pessoas que avaliam experimentos de laboratório do que para analistas de big data.

Outro termo popular em aprendizado de máquina é *regularização.* Ao aplicar a regularização, você incorre em uma penalidade para cada variável extra usada para construir o modelo. Com *L1*

regularização você pede um modelo com o menor número possível de preditores. Isto é importante para a robustez do modelo: soluções simples tendem a ser verdadeiras em mais situações. *A regularização L2* visa manter a variância entre os coeficientes dos preditores tão pequena quanto possível. A sobreposição de variações entre preditores torna difícil determinar o impacto real de cada preditor. Evitar que suas variações se sobreponham aumentará a interpretabilidade. Para simplificar: a regularização é usada principalmente para impedir que um modelo use muitos recursos e, assim, evitar ajustes excessivos.

A validação é extremamente importante porque determina se o seu modelo funciona em condições da vida real. Para ser franco, a questão é se o seu modelo vale um centavo. Mesmo assim, de vez em quando as pessoas enviam artigos para revistas científicas respeitadas (e às vezes até conseguem publicá-los) com validação deficiente. O resultado disso é que eles são rejeitados ou precisam retirar o artigo porque está tudo errado. Situações como esta são prejudiciais à sua saúde mental, por isso tenha sempre isto em mente: teste os seus modelos em dados que o modelo construído nunca viu e certifique-se de que estes dados são uma representação verdadeira do que encontraria quando aplicados em novas observações por outros. pessoas.

Para modelos de classificação, instrumentos como a matriz de confusão (introduzida no capítulo 2, mas explicada detalhadamente mais adiante neste capítulo) são ideais; abraçá-los.

Depois de construir um bom modelo, você pode (opcionalmente) usá-lo para prever o futuro.

3.2.4 Previsão de novas observações

Se você implementou as três primeiras etapas com sucesso, agora você tem um modelo de desempenho que generaliza para dados invisíveis. O processo de aplicação do seu modelo a novos dados é chamado de pontuação do modelo. Na verdade, a pontuação do modelo é algo que você fez implicitamente durante a validação, só que agora você não sabe o resultado correto. Agora você deve confiar em seu modelo o suficiente para usá-lo de verdade.

A pontuação do modelo envolve duas etapas. Primeiro, você prepara um conjunto de dados que possui recursos exatamente como definidos pelo seu modelo. Isso se resume a repetir a preparação de dados feita na primeira etapa do processo de modelagem, mas para um novo conjunto de dados. Em seguida, você aplica o modelo a esse novo conjunto de dados e isso resulta em uma previsão.

Agora vamos examinar os diferentes tipos de técnicas de aprendizado de máquina: uma abordagem diferente problema requer uma abordagem diferente.

3.3 Tipos de aprendizado de máquina

Em termos gerais, podemos dividir as diferentes abordagens de aprendizado de máquina pela quantidade de esforço humano necessário para coordená-las e como elas usam *dados rotulados – dados* com uma categoria ou um número de valor real atribuído a eles que representa o resultado de observações anteriores. .

- As técnicas *de aprendizagem supervisionada* tentam discernir resultados e aprender tentando encontrar padrões num conjunto de dados rotulado. A interação humana é necessária para rotular os dados.
- As técnicas *de aprendizagem não supervisionadas* não dependem de dados rotulados e tentam encontrar padrões em um conjunto de dados sem interação humana.

ÿ As técnicas *de aprendizagem semissupervisionadas* precisam de dados rotulados e, portanto, de dados humanos interação, para encontrar padrões no conjunto de dados, mas eles ainda podem progredir em direção a um resultado e aprender mesmo se também forem transmitidos dados não rotulados.

Nesta seção, veremos todas as três abordagens, veremos para quais tarefas cada uma é mais apropriada e usaremos uma ou duas das bibliotecas Python mencionadas anteriormente para fornecer uma ideia geral.
sinta o código e resolva uma tarefa. Em cada um desses exemplos, trabalharemos com um conjunto de dados baixável que já foi limpo, então pularemos direto para a etapa de modelagem de dados do processo de ciência de dados, conforme discutido anteriormente neste capítulo.

3.3.1 Aprendizagem supervisionada

Como afirmado anteriormente, a aprendizagem supervisionada é uma técnica de aprendizagem que só pode ser aplicada em dados rotulados. Um exemplo de implementação disso seria discernir dígitos de
imagens. Vamos mergulhar em um estudo de caso sobre reconhecimento de números.

ESTUDO DE CASO: DISCERNINDO DÍGITOS DE IMAGENS

Uma das muitas abordagens comuns na web para impedir que computadores invadam contas de usuários é a verificação Captcha – uma imagem de texto e números que o
o usuário humano deve decifrar e inserir um campo de formulário antes de enviar o formulário de volta
para o servidor web. Algo como a figura 3.3 deve parecer familiar.

Figura 3.3 Um simples controle Captcha pode ser usado para evitar que spam automatizado seja enviado através de um formulário online.

Com a ajuda do *classificador Naïve Bayes,* um algoritmo simples, mas poderoso para categorizar
observações em classes explicadas com mais detalhes na barra lateral, você pode reconhecer dígitos a partir de imagens textuais. Essas imagens não são diferentes das verificações do Captcha em muitos
sites existem para garantir que você não seja um computador tentando invadir o usuário
contas. Vamos ver como é difícil deixar um computador reconhecer imagens de números.

Nosso objetivo de pesquisa é permitir que um computador reconheça imagens de números (primeiro passo do
o processo de ciência de dados).

Os dados nos quais trabalharemos são o conjunto de dados MNIST , que é frequentemente usado na análise de dados literatura científica para ensino e benchmarking.

Apresentando classificadores Naïve Bayes no contexto de um filtro de spam Nem todo e-mail que você recebe tem intenções honestas. Sua caixa de entrada pode conter e-mails comerciais ou em massa não solicitados, também conhecidos como spam. O spam não é apenas irritante, mas também é frequentemente usado em golpes e como portador de vírus. A Kaspersky3 estima que mais de 60% dos e-mails no mundo são spam. Para proteger os usuários contra spam, a maioria dos clientes de e-mail executa um programa em segundo plano que classifica os e-mails como spam ou seguros.

Uma técnica popular na filtragem de spam é empregar um classificador que usa as palavras do e-mail como preditores. Ele gera a chance de um e-mail específico ser spam, dadas as palavras que o compõem (em termos matemáticos, P(spam | words)). Alcançar esta conclusão utiliza três cálculos:

- ÿ P(spam) — A taxa média de spam sem conhecimento das palavras. De acordo com a Kaspersky, um e-mail é spam 60% das vezes.
- ÿ P(palavras) — com que frequência essa combinação de palavras é usada, independentemente de spam.
- ÿ P(palavras | spam) — a frequência com que essas palavras são vistas quando um e-mail de treinamento foi rotulado como spam.

Para determinar a chance de um novo e-mail ser spam, você usaria a seguinte fórmula:

P(spam|palavras) = P(spam)P(palavras|spam) / P(palavras)

Esta é uma aplicação da regra P(B|A) = P(B) P(A|B) / P(A), que é conhecida como regra de Bayes e que dá nome a este classificador. A parte "ingênua" vem da suposição do classificador de que a presença de um recurso não diz nada sobre outro recurso (independência de recurso, também chamada de ausência de multicolinearidade). Na realidade, os recursos estão frequentemente relacionados, especialmente no texto. Por exemplo, a palavra "comprar" será frequentemente seguida por "agora". Apesar da suposição irrealista, o classificador ingênuo funciona surpreendentemente bem na prática.

Com um pouco de teoria na barra lateral, você está pronto para realizar a modelagem propriamente dita. Certifique-se de executar todo o próximo código no mesmo escopo, pois cada parte requer o anterior. Um arquivo IPython pode ser baixado para este capítulo na página de download de Manning deste livro.3

As imagens MNIST podem ser encontradas no pacote de conjuntos de dados do Scikit-learn e já estão normalizadas para você (todas dimensionadas para o mesmo tamanho: 64x64 pixels), portanto não precisaremos de muita preparação de dados (etapa três do processo de ciência de dados). Mas vamos primeiro buscar nossos dados como etapa dois do processo de ciência de dados, com a listagem a seguir.

Listagem 3.1 Etapa 2 do processo de ciência de dados: buscando os dados da imagem digital

```
de sklearn.datasets importar load_digits importar pylab como
pl dígitos = load_digits()
```

Importa banco de dados de dígitos.

Carrega dígitos.

3 Relatório trimestral de estatísticas de spam de 2014 da Kaspersky, http://usa.kaspersky.com/internet-security-center/threats/spam-statistics-report-q1-2014#.VVym9blViko .

Trabalhar com imagens não é muito diferente de trabalhar com outros conjuntos de dados. No caso de uma imagem cinza, você coloca um valor em cada entrada da matriz que representa o valor cinza para ser mostrado. O código a seguir demonstra esse processo e é a etapa quatro da análise de dados processo científico: exploração de dados.

Listagem 3.2 Etapa 4 do processo de ciência de dados: usando o Scikit-learn

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

dígitos.imagens[0]
```

Mostra as primeiras imagens.

Transforma a imagem em valores de escala de cinza.

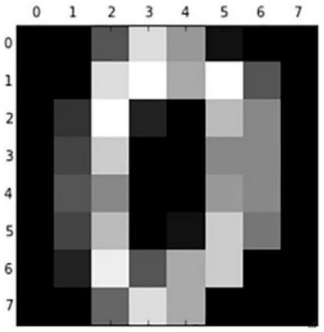
Mostra o matriz correspondente.

A Figura 3.4 mostra como uma imagem “0” borrada se traduz em uma matriz de dados.

A Figura 3.4 mostra a saída real do código, mas talvez a Figura 3.5 possa esclarecer isso um pouco, porque mostra como cada elemento do vetor é um pedaço da imagem.

Fácil até agora, não é? Há, naturalmente, um pouco mais de trabalho a fazer. O Ingênuo Bayes classificador está esperando uma lista de valores, mas pl.matshow() retorna um valor bidimensional array (uma matriz) refletindo a forma da imagem. Para achatá-lo em uma lista, precisamos chame reshape() em digits.images. O resultado líquido será uma matriz unidimensional que parece algo assim:

```
matriz([[ 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.]])
```

Figura 3.4 Representação em tons de cinza desfocados do número 0 com sua matriz correspondente. Quanto maior o número, mais próximo está do branco; quanto menor o número, mais próximo está do preto.

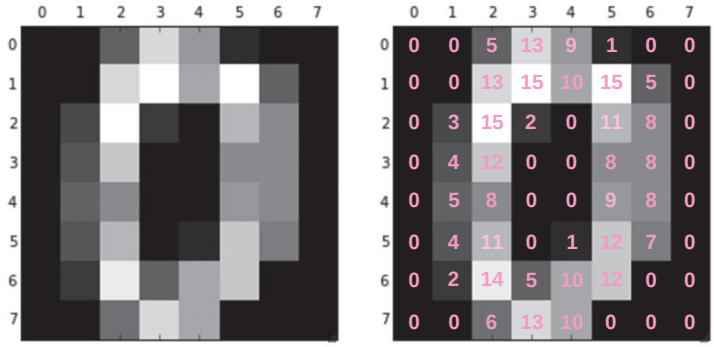


Figura 3.5 Transformaremos uma imagem em algo utilizável pelo classificador Naïve Bayes obtendo o valor da escala de cinza para cada um de seus pixels (mostrados à direita) e colocando esses valores em uma lista.

O trecho de código anterior mostra a matriz da figura 3.5 achatada (o número de dimensões foi reduzido de duas para uma) em uma lista Python. Desse ponto em diante, trata-se de um problema de classificação padrão, que nos leva à quinta etapa do processo de ciência de dados: construção de modelo.

Agora que temos uma maneira de passar o conteúdo de uma imagem para o classificador, precisamos passar um conjunto de dados de treinamento para que ele possa começar a aprender como prever os números nas imagens. Mencionamos anteriormente que o Scikit-learn contém um subconjunto do banco de dados MNIST (1.800 imagens), então usaremos isso. Cada imagem também é rotulada com o número que realmente mostra. Isto irá construir um modelo probabilístico na memória do dígito mais provável mostrado em uma imagem, dados seus valores em escala de cinza.

Depois que o programa tiver passado pelo conjunto de treinamento e construído o modelo, podemos então passá-lo no conjunto de dados de teste para ver se ele aprendeu a interpretar as imagens usando o modelo.

A listagem a seguir mostra como implementar essas etapas no código.

Listagem 3.3 Problema de classificação de dados de imagem em imagens de dígitos

```
de sklearn.cross_validation importar train_test_split de sklearn.naive_bayes
importar GaussianNB de sklearn.metrics importar confusão_matrix
importar pylab como plt


y = dígitos.alvo

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

imprimir X

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

Etapa 1: selecione a variável de destino.

Etapa 2: preparar os dados. Reshape adapta a forma da matriz. Este método poderia, por exemplo, transformar uma matriz 10x10 em 100 vetores.

Etapa 3: Divida em conjunto de teste e conjunto de treinamento.

Etapa 5:
Ajustar dados.

```
gnb = GaussianoNB()
ajuste = gnb.fit(X_train,y_train)
previsto = fit.predict(X_test)
confusão_matriz(y_test, previsto)
```

Etapa 4: Selecione um classificador Naïve Bayes; use uma distribuição gaussiana para estimar a probabilidade.

Etapa 6: prever dados para dados não vistos.

Etapa 7: criar matriz de confusão.

O resultado final deste código é chamado de *matriz de confusão,* como a mostrada na figura 3.6. Retornado como uma matriz bidimensional, mostra quantas vezes o número previsto foi o número correto na diagonal principal e também na entrada da matriz (i,j), onde j foi previsto, mas a imagem mostrou i. Olhando para a figura 3.6 podemos ver que o modelo previu o número 2 corretamente 17 vezes (nas coordenadas 3,3), mas também que o modelo previu o número 8 15 vezes quando na verdade era o número 2 em a imagem (em 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]])
```

Figura 3.6 Matriz de confusão produzida pela previsão de qual número é representado por uma imagem borrada

Matrizes de confusão

Uma matriz de confusão é uma matriz que mostra quão errado (ou correto) um modelo previu, o quanto ficou “confuso”. Na sua forma mais simples será uma tabela 2x2 para modelos que tente classificar as observações como sendo A ou B. Digamos que temos um modelo de classificação que prevê se alguém comprará nosso mais novo produto: pudim de cereja frito. Podemos prever: “Sim, esta pessoa comprará” ou “Não, este cliente não comprar." Depois de fazermos a nossa previsão para 100 pessoas, podemos compará-la com a sua comportamento real, mostrando-nos quantas vezes acertamos. Um exemplo é mostrado em tabela 3.1.

Tabela 3.1 Exemplo de matriz de confusão

Matriz de confusão	Previsto “A pessoa comprará”	Previsto “A pessoa não comprará”
Pessoa **comprou** o pudim de cereja frito	35 (verdadeiro positivo)	10 (falso negativo)
A pessoa **não comprou** o pudim de cereja frito	15 (falso positivo)	40 (verdadeiro negativo)

O modelo estava correto em (35+40) 75 casos e incorreto em (15+10) 25 casos, resultando em uma precisão de 75% (75 observações corretas/100 observações totais).

Todas as observações corretamente classificadas são somadas na diagonal (35+40) enquanto todo o resto (15+10) é classificado incorretamente. Quando o modelo prevê apenas dois classes (binário), nossas suposições corretas são dois grupos: verdadeiros positivos (previstos para comprar e o fizeram) e verdadeiros negativos (previram que não comprariam e não compraram). Nosso suposições incorretas são divididas em dois grupos: falsos positivos (previstos que seriam compraram, mas não compraram) e falsos negativos (prevê-se que não comprariam, mas compraram). O Matrix é útil para ver onde o modelo está tendo mais problemas. Neste caso tendemos ter excesso de confiança em nosso produto e classificar os clientes como futuros compradores com muita facilidade (falso positivo).

A partir da matriz de confusão, podemos deduzir que para a maioria das imagens as previsões são muito preciso. Em um bom modelo, você esperaria que a soma dos números da tabela principal diagonal da matriz (também conhecida como *traço de matriz)* é muito alta em comparação com a soma de todas as entradas da matriz, indicando que as previsões estavam corretas para o na maior parte.

Vamos supor que queremos mostrar nossos resultados de uma forma mais facilmente compreensível ou queremos inspecionar várias imagens e as previsões que nosso programa tem made: podemos usar o seguinte código para exibir um ao lado do outro. Então nós podemos veja onde o programa deu errado e precisa de um pouco mais de treinamento. Se estivermos satisfeitos com os resultados, a construção do modelo termina aqui e chegamos ao sexto passo: apresentar os resultados.

Listagem 3.4 Inspecionando previsões versus números reais

Adiciona uma subtrama extra em uma grade de plotagem 6x3. Este código poderia ser simplificado como: plt.subplot (3, 2 ,index) mas parece visualmente mais atraente.

Armazena matriz de imagem numérica e sua previsão (como um número) juntos em matriz.

Percorre as primeiras 7 imagens

```
imagens_e_predições = lista(zip(digits.images, fit.predict(X)))
para índice, (imagem, previsão) em enumerate(images_and_predictions[:6]):
    plt.subplot(6, 3, índice + 5) plt.axis('off')
    plt.imshow(imagem,
    cmap=plt.cm.gray_r, interpolation='mais próximo') plt.title('Predição: %i' % previsão) plt.show()
```

Não mostra um eixo.

Programas imagem em escala de cinza.

Mostra o previsto valor como o título para a imagem mostrada.

Mostra o enredo completo que está agora preenchido com 6 subparcelas.

A Figura 3.7 mostra como todas as previsões parecem estar corretas, exceto o dígito número 2, que é rotulado como 8. Devemos perdoar esse erro, pois este 2 compartilha semelhanças visuais

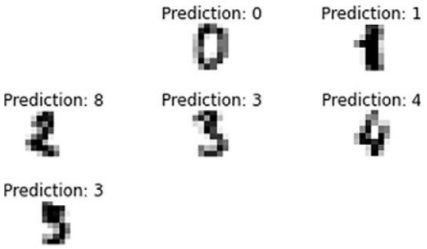


Figura 3.7 Para cada imagem borrada é previsto um número; apenas o número 2 é mal interpretado como 8. Então, prevê-se que um número ambíguo seja 3, mas também poderia ser 5; mesmo para os olhos humanos isso não está claro.

com 8. O número inferior esquerdo é ambíguo, até mesmo para humanos; é um 5 ou um 3? Isso é discutível, mas o algoritmo pensa que é um 3.

Ao discernir quais imagens foram mal interpretadas, podemos treinar ainda mais o modelo rotulando-os com o número correto que eles exibem e inserindo-os de volta o modelo como um novo conjunto de treinamento (etapa 5 do processo de ciência de dados). Isto fará com que modelo mais preciso, de modo que o ciclo de aprender, prever e corrigir continua e as previsões se tornam mais precisas. Este é um conjunto de dados controlado que estamos usando no exemplo. Todos os exemplares são do mesmo tamanho e estão todos em 16 tons de cinza. Expanda isso até as imagens de tamanho variável de strings de comprimento variável de tons variáveis de caracteres alfanuméricos mostrados no controle Captcha, e você pode entender por que um modelo preciso o suficiente para prever que qualquer imagem Captcha ainda não existe.

Neste exemplo de aprendizagem supervisionada, fica evidente que sem os rótulos associados com cada imagem informando ao programa qual número aquela imagem mostra, um modelo não pode ser construído e previsões não podem ser feitas. Por outro lado, uma aprendizagem não supervisionada abordagem não precisa que seus dados sejam rotulados e pode ser usada para dar estrutura a um conjunto de dados não estruturados.

3.3.2 Aprendizagem não supervisionada

Geralmente é verdade que a maioria dos grandes conjuntos de dados não possui rótulos em seus dados, portanto, a menos que você classificar tudo e atribuir rótulos, a abordagem de aprendizagem supervisionada dos dados não trabalhar. Em vez disso, devemos adotar a abordagem que funcionará com estes dados porque

- Podemos estudar a *distribuição dos dados* e inferir verdades sobre os dados de diferentes maneiras. partes importantes da distribuição.
- Podemos estudar a *estrutura e os valores dos dados* e inferir novas e mais significativas dados e estrutura dele.

Existem muitas técnicas para cada uma dessas abordagens *de aprendizagem não supervisionada* . No entanto, em do mundo real, você está sempre trabalhando em direção ao objetivo de pesquisa definido no primeiro fase do processo de ciência de dados, portanto, você pode precisar combinar ou experimentar diferentes técnicas antes que um conjunto de dados possa ser rotulado, permitindo, talvez, técnicas de aprendizagem supervisionada, ou até mesmo que o objetivo em si seja alcançado.

DISCERNINDO UMA ESTRUTURA LATENTE SIMPLIFICADA DE SEUS DADOS

Nem tudo pode ser medido. Quando você conhece alguém pela primeira vez você
você pode tentar adivinhar se eles gostam de você com base no comportamento deles e como eles
responder. Mas e se eles tiveram um dia ruim até agora? Talvez o gato deles tenha sido atropelado
ou ainda não compareceram a um funeral na semana anterior? A questão é que certas variáveis podem estar
imediatamente disponíveis, enquanto outras só podem ser inferidas e são
portanto, ausente em seu conjunto de dados. O primeiro tipo de variáveis são conhecidas como *observáveis variáveis* e o segundo tipo são conhecidos como *variáveis latentes.* No nosso exemplo, o estado emocional do seu novo amigo é uma variável latente. Definitivamente influencia o julgamento que eles fazem de você, mas seu valor não é claro.

Derivando ou inferindo variáveis latentes e seus valores com base no conteúdo real
de um conjunto de dados é uma habilidade valiosa porque

- ÿ As variáveis latentes podem substituir diversas variáveis existentes já no conjunto de dados.
- ÿ Ao reduzir o número de variáveis no conjunto de dados, o conjunto de dados se torna mais gerenciável, quaisquer algoritmos adicionais executados nele funcionam mais rápido e as previsões podem tornar-se mais preciso.
- ÿ Porque as variáveis latentes são projetadas ou direcionadas para a pesquisa definida objetivo, você perde poucas informações importantes ao usá-los.

Se pudermos reduzir um conjunto de dados de 14 variáveis observáveis por linha para 5 ou 6 variáveis latentes, por exemplo, teremos mais chances de atingir nosso objetivo de pesquisa por causa de
a estrutura simplificada do conjunto de dados. Como você verá no exemplo abaixo, não é o caso
de reduzir o conjunto de dados existente ao menor número possível de variáveis latentes. Você precisará
encontre o ponto ideal onde o número de variáveis latentes derivadas retorna mais
valor. Vamos colocar isso em prática com um pequeno estudo de caso.

ESTUDO DE CASO: ENCONTRANDO VARIÁVEIS LATENTES NUM CONJUNTO DE DADOS DE QUALIDADE DE VINHO

Neste breve estudo de caso, você usará uma técnica conhecida como Análise de Componentes Principais (PCA) para encontrar variáveis latentes num conjunto de dados que descreve a qualidade do vinho. Então
você comparará o quão bem um conjunto de variáveis latentes funciona na previsão da qualidade do vinho
contra o conjunto observável original. Você vai aprender

1. Como identificar e derivar essas variáveis latentes.
2. Como analisar onde está o ponto ideal – quantas novas variáveis retornam o mais utilidade - gerando e interpretando um *scree plot* gerado pelo PCA. (Bem veja os gráficos de cascalho em um momento.)

Vejamos os principais componentes deste exemplo.

- ÿ Conjunto *de dados* — A Universidade da Califórnia, Irvine (UCI) possui um repositório on-line de 325 conjuntos de dados para exercícios de aprendizado de máquina em http://archive.ics.uci.edu/ml/. Utilizaremos o conjunto de dados de qualidade do vinho para vinhos tintos criado por P. Cortez, A. Cerdeira, F. Almeida, T. Matos e J. Reis[4] . Possui 1.600 linhas e 11 variáveis por linha, conforme mostrado na tabela 3.2.

[4] Você pode encontrar detalhes completos do conjunto de dados de qualidade do vinho em https://archive.ics.uci.edu/ml/datasets/Wine+Quality.

Tabela 3.2 As três primeiras linhas do conjunto de dados de qualidade do vinho tinto

Acidez fixa	Acidez volátil	Ácido Cítrico	Residual açúcar	Cloretos	Dióxido de enxofre livre	Dióxido de enxofre total	Densidade	pH	Sulfatos	Qualidade do Álcool	
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

- *Análise de Componentes Principais – Uma* técnica para encontrar as variáveis latentes em seu conjunto de dados, mantendo o máximo de informações possível.
- *Scikit-learn—Usamos* esta biblioteca porque ela já implementa PCA para nós e é uma forma de gerar o scree plot.

A primeira parte do processo de ciência de dados é definir nosso objetivo de pesquisa: Queremos explicar o feedback subjetivo da “qualidade do vinho” usando as diferentes propriedades do vinho.

Nosso primeiro trabalho é baixar o conjunto de dados (etapa dois: adquirir dados), conforme mostrado na listagem a seguir e prepare-o para análise (etapa três: preparação dos dados).
Então podemos executar o algoritmo PCA e visualizar os resultados para ver nossas opções.

Listagem 3.5 Aquisição de dados e padronização de variáveis

X é uma matriz de variáveis preditoras. Essas variáveis são propriedades do vinho, como densidade e presença de álcool.

```
importar pandas como pd
do pré-processamento de importação do sklearn
de sklearn.decomposition importar PCA
importar pylab como plt
do pré-processamento de importação do sklearn

url = http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
      winequality-red.csv dados =
pd.read_csv(url, set= ";")
X = dados[[u'acidez fixa', u'acidez volátil', u'ácido cítrico', u'açúcar residual', u'cloretos', u'dióxido de enxofre
      livre', u'dióxido de enxofre total', u' densidade', u'pH', u'sulfatos', u'álcool']] y = dados.qualidade X=
      pré-processamento.StandardScaler().fit(X).transform(X)
```

Local de download de conjunto de dados de qualidade do vinho.

Lê os dados CSV. É separado por ponto e vírgula.

y é um vetor e representa a variável dependente (variável alvo). y é a qualidade percebida do vinho.

Ao padronizar dados, a seguinte fórmula é aplicada a todos os dados ponto: z = (x-ÿ)/ÿ, onde z é o novo valor de observação, x o antigo, ÿ é a média e ÿ o desvio padrão. O PCA de uma matriz de dados é mais fácil para interpretar quando as colunas foram centralizadas pela primeira vez por suas médias.

Com a preparação inicial dos dados concluída, você pode executar o PCA. O resultado
O gráfico de scree (que será explicado em breve) é mostrado na figura 3.8. Como o PCA é um
técnica exploratória, chegamos agora à etapa quatro do processo de ciência de dados: dados
exploração, conforme mostrado na listagem a seguir.

Listagem 3.6 Executando a análise de componentes principais

```
modelo = PCA()
resultados = model.fit(X)
Z = resultados.transformação(X)
plt.plot(resultados.explained_variance_) plt.show()
```

Cria instância da classe de análise de componentes principais

Aplica PCA em variáveis preditoras para ver se elas podem ser compactadas em menos variáveis

Transforma o resultado em array para que possamos usar os dados recém-criados

Os gráficos explicaram a variância nas variáveis; este enredo é um enredo de seixos

Mostra o enredo

Agora vamos dar uma olhada no gráfico do scree na figura 3.8.

O gráfico gerado a partir do conjunto de dados do vinho é mostrado na figura 3.8. O que você espera
veja o formato de um cotovelo ou taco de hóquei no enredo. Isso indica que algumas variáveis podem
representam a maioria das informações no conjunto de dados, enquanto o restante apenas adiciona um pouco
mais. Em nosso gráfico, o PCA nos diz que reduzir o conjunto a uma variável pode capturar
aproximadamente 28% do total de informações do conjunto (o gráfico é baseado em zero, então variável

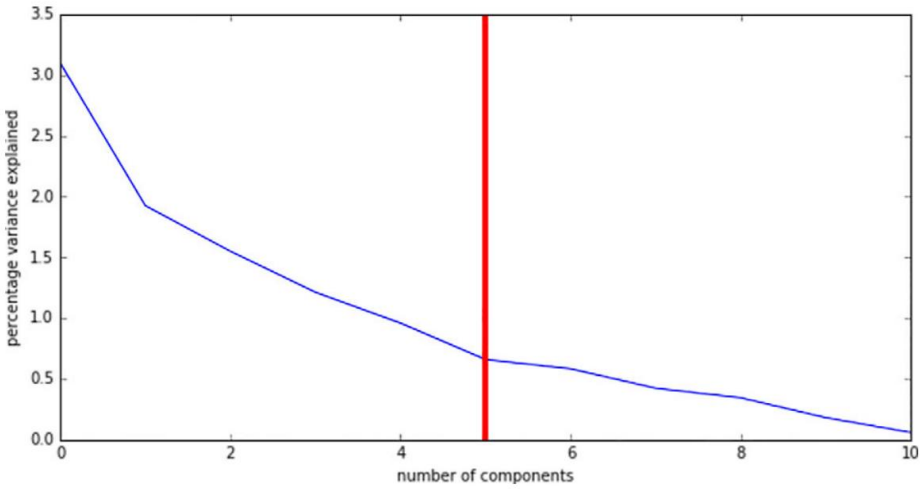


Figura 3.8 Gráfico de scree do PCA mostrando a quantidade marginal de informação de cada nova variável que o PCA pode criar. As primeiras variáveis explicam aproximadamente 28% da variância dos dados, a segunda variável explica outros 17%, a terceira aproximadamente 15% e assim por diante.

uma está na posição zero no eixo x), duas variáveis capturarão aproximadamente 17% mais ou 45% do total e assim por diante. A Tabela 3.3 mostra a leitura completa.

Tabela 3.3 As conclusões do PCA

Número de variáveis	Informações extras capturadas	Total de dados capturados
1	28%	28%
2	17%	45%
3	14%	59%
4	10%	69%
5	8%	77%
6	7%	84%
7	5%	89%
8 - 11	…	100%

Um formato de cotovelo no gráfico sugere que cinco variáveis podem conter a maior parte das informações encontradas nos dados. Você poderia defender um limite de seis ou sete variáveis em vez disso, mas vamos optar por um conjunto de dados mais simples em vez de um com menos variação dados em relação ao conjunto de dados original.

Neste ponto, poderíamos prosseguir e ver se o conjunto de dados original foi recodificado com cinco variáveis latentes é bom o suficiente para prever com precisão a qualidade do vinho, mas antes disso, veremos como podemos identificar o que eles representam.

INTERPRETANDO AS NOVAS VARIÁVEIS

Com a decisão inicial de reduzir o conjunto de dados de 11 variáveis originais para 5 variáveis latentes, podemos verificar se é possível interpretá-las ou nomeá-las com base em suas relações com os originais. Nomes reais são mais fáceis de trabalhar do que códigos como lv1, lv2 e assim por diante. Podemos adicionar a linha de código na listagem a seguir para gerar uma tabela que mostra como os dois conjuntos de variáveis se correlacionam.

Listagem 3.7 Mostrando componentes PCA em um data frame do Pandas

```
pd.DataFrame(resultados.components_, colunas=lista(
ÿ [u'acidez fixa', u'acidez volátil', u'ácido cítrico', u'açúcar residual',
ÿ u'cloretos', u'dióxido de enxofre livre', u'dióxido de enxofre total', u'densidade',
ÿ u'pH', u'sulfatos', u'álcool']))
```

As linhas da tabela resultante (tabela 3.4) mostram a correlação matemática. Ou, em Inglês, a primeira variável latente lv1, que captura aproximadamente 28% do total informações do conjunto, tem a seguinte fórmula.

Lv1 = (acidez fixa * 0,489314) + (acidez volátil * -0,238584) + (álcool * -0,113232) … +

Tabela 3.4 Como o PCA calcula a correlação das 11 variáveis originais com 5 variáveis latentes

	Acidez fixa	Acidez volátil	Ácido Cítrico	Residual açúcar	Cloretos	Dióxido de enxofre livre	Dióxido de enxofre total	Densidade	pH	Álcool Sulfatado	
0	0,489314	-0,238584	0,463632	0,146107	0,212247	-0,036158	0,023575	0,395353	-0,438520	0,242921	-0,113232
1	-0,110503	0,274930	-0,151791	0,272080	0,148052	0,513567	0,569487	0,233575	0,006711	-0,037554	-0,386181
2	0,123302	0,449963	-0,238247	-0,101283	0,092614	-0,428793	-0,322415	0,338871	-0,057697	-0,279786	-0,471673
3	-0,229617	0,078960	-0,079418	-0,372793	0,666195	-0,043538	-0,034577	-0,174500	-0,003788	0,550872	-0,122181
4	0,082614	-0,218735	0,058573	-0,732144	-0,246501	0,159152	0,222465	-0,157077	-0,267530	-0,225962	-0,350681

Dar um nome utilizável para cada nova variável é um pouco mais complicado e provavelmente seria requerem consulta com um verdadeiro especialista em vinhos para precisão. Contudo, como não Se tiver um especialista em vinhos por perto, iremos chamá-lo da seguinte forma (tabela 3.5).

Tabela 3.5 Interpretação das variáveis de qualidade do vinho criadas pela PCA

Variavel latente	Interpretação possível
0	Acidez persistente
1	Sulfetos
2	Acidez volátil
3	Cloretos
4	Falta de açúcar residual

Agora podemos recodificar o conjunto de dados original apenas com as cinco variáveis latentes. Fazendo isso Esta é a preparação de dados novamente, então revisitamos a terceira etapa do processo de ciência de dados: preparação de dados. Conforme mencionado no capítulo 2, o processo de ciência de dados é recursivo e isso é especialmente verdadeiro entre a etapa três: preparação de dados e a etapa 4: exploração de dados.

A Tabela 3.6 mostra as três primeiras linhas com isso feito.

Tabela 3.6 As três primeiras linhas do Conjunto de Dados de Qualidade do Vinho Tinto recodificadas em cinco variáveis latentes

	Acidez persistente	Sulfetos	Acidez volátil	Cloretos	Falta de açúcar residual
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**

Já podemos observar valores elevados para o vinho 0 em acidez volátil, enquanto o vinho 2 é particularmente rico em acidez persistente. Não parecem bons vinhos!

COMPARANDO A PRECISÃO DO CONJUNTO DE DADOS ORIGINAIS COM VARIÁVEIS LATENTES

Agora que decidimos que nosso conjunto de dados deveria ser recodificado em 5 variáveis latentes, em vez do que os 11 originais, é hora de ver se o novo conjunto de dados funciona bem para prever a qualidade do vinho quando comparado com o original. Usaremos o Classificador Naïve Bayes algoritmo que vimos no exemplo anterior para aprendizagem supervisionada para ajudar.

Vamos começar vendo até que ponto as 11 variáveis originais poderiam prever bem os índices de qualidade do vinho. A listagem a seguir apresenta o código para fazer isso.

Listagem 3.8 Previsão da pontuação do vinho antes da análise dos componentes principais

```
de sklearn.cross_validation importar train_test_split
de sklearn.naive_bayes importar GaussianNB
de sklearn.metrics importar confusão_matrix
importar pylab como plt

gnb = GaussianNB() ajuste =
gnb.fit(X,y) pred = fit.predict(X)
imprimir confusão_matriz(pred,y)

imprimir confusão_matrix(pred,y).trace()
```

Use o classificador Naïve Bayes de distribuição gaussiana para estimativa.

Ajustar dados.

Preveja dados para dados não vistos.

Estude a matriz de confusão.

Contagem de todos os casos classificados corretamente: todos contam com traço ou diagonal resumido após análise de confusão matriz. Podemos ver as pontuações do classificador Naïve Bayes 897 previsões corretas de 1.599.

Agora executaremos o mesmo teste de previsão, mas começando com apenas 1 variável latente em vez de o 11 original. Depois adicionaremos outro, veremos como ficou, adicionaremos outro e assim por diante para ver como o desempenho preditivo melhora. A listagem a seguir mostra como isso é feito.

Listagem 3.9 Previsão da pontuação do vinho com número crescente de componentes principais

```
previsto_correto = []
para i no intervalo (1,10):
        modelo = PCA (n_componentes = i)
        resultados = modelo.fit(X)
        Z = resultados.transform(X) fit =
        gnb.fit(Z,y) pred = fit.predict(Z)

        predito_correto.append(confusion_matrix(pred,y).trace())
        imprimir previsto_correto
plt.plot(predito_correto) plt.show()
```

Ajustar modelo PCA em variáveis x (características)

A matriz será preenchida corretamente observações previstas

Faz um loop pelos primeiros 10 componentes principais detectados

Instancie o modelo PCA com 1 componente (primeira iteração) até 10 componentes (na 10ª iteração)

Z é o resultado em forma de matriz (na verdade, uma matriz preenchida com matrizes)

Use distribuição gaussiana Naïve Classificador Bayes para estimativa

O real predição em si usando o equipado modelo

Gráfico mostrado

Mais facil de veja quando variedade traçado

Imprimindo esta matriz podemos ver como após cada iteração, uma nova contagem de observações classificadas corretamente é anexada

No final de cada iteração, acrescentamos o número de observações classificadas corretamente

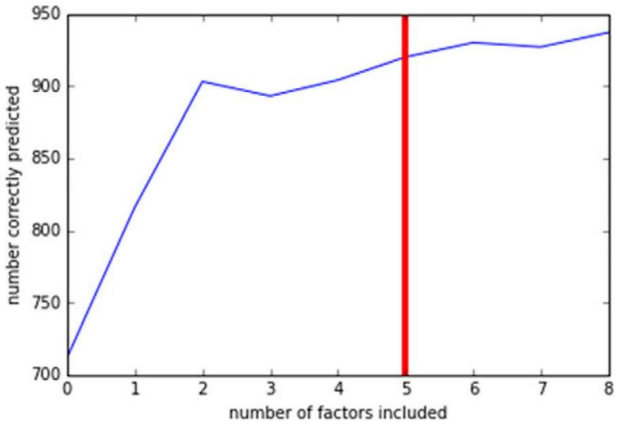


Figura 3.9 O gráfico de resultados mostra que adicionar mais variáveis latentes a um modelo (eixo x) aumenta muito o poder preditivo (eixo y) até certo ponto, mas depois diminui. O ganho em poder preditivo resultante da adição de variáveis eventualmente

O gráfico resultante é mostrado na figura 3.9.

O gráfico da figura 3.9 mostra que com apenas 3 variáveis latentes, o classificador faz uma melhor trabalho de previsão da qualidade do vinho do que com o 11 original. Além disso, adicionar mais variáveis latentes além de 5 não adicionam tanto poder preditivo quanto os primeiros 5. Isso mostra que nossa escolha de cortar em 5 variáveis foi boa, como esperávamos.

Vimos como agrupar variáveis semelhantes, mas também é possível agrupar observações.

AGRUPANDO OBSERVAÇÕES SEMELHANTES PARA OBTER INFORMAÇÕES SOBRE A DISTRIBUIÇÃO DE SEUS DADOS

Suponha por um momento que você esteja criando um site que recomenda filmes aos usuários com base nas preferências inseridas e nos filmes que assistiram. As chances são é alto que, se assistirem a muitos filmes de terror, é provável que queiram saber sobre novos filmes de terror e não tanto sobre novos filmes de romance adolescente. Ao agrupar usuários que assistiram mais ou menos aos mesmos filmes e definiram mais ou menos as mesmas preferências, você pode obter uma boa visão sobre o que mais eles gostariam de ter recomendado.

A técnica geral que descrevemos aqui é conhecida como *agrupamento.* Neste processo, nós tentativa de dividir nosso conjunto de dados em subconjuntos de observação, ou *clusters,* onde as observações devem ser semelhantes aos do mesmo cluster, mas diferem muito das observações em outros aglomerados. A Figura 3.10 dá uma ideia visual do que o clustering pretende alcançar. O os círculos no canto superior esquerdo da figura estão claramente próximos uns dos outros, embora estejam mais distantes longe dos outros. O mesmo se aplica às cruzes no canto superior direito.

Scikit-learn implementa vários algoritmos comuns para agrupar dados em seu Módulo sklearn.cluster , incluindo o algoritmo k-means, propagação de afinidade e agrupamento espectral. Cada um tem um ou dois casos de uso para os quais é mais adequado,5 embora

[5] Você pode encontrar uma comparação de todos os algoritmos de cluster no Scikit-learn em http://scikit-learn.org/stable/módulos/clustering.html.

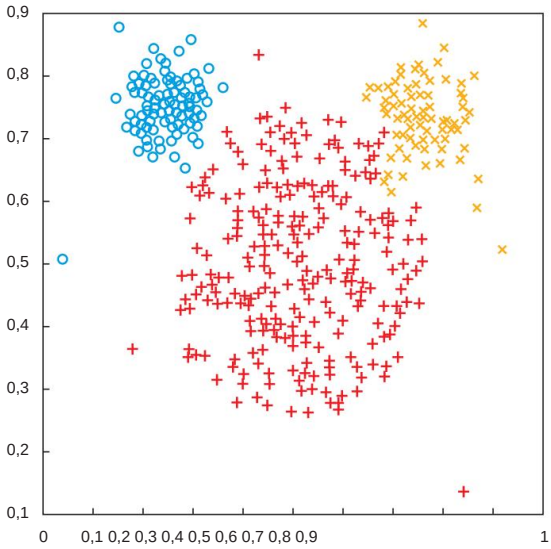


Figura 3.10 O objetivo do agrupamento é dividir um conjunto de dados em subconjuntos "suficientemente distintos". Neste gráfico, por exemplo, as observações foram divididas em três grupos.

k-means é um bom algoritmo de uso geral para começar. No entanto, como todos os algoritmos de clustering, você precisa especificar o número de clusters desejados em avanço, o que necessariamente resulta em um processo de tentativa e erro antes de chegar a um conclusão decente. Também pressupõe que todos os dados necessários para a análise já estejam disponíveis. E se não fosse?

Vejamos o caso real de agrupamento de íris (a flor) por suas propriedades (comprimento e largura da sépala, comprimento e largura da pétala e assim por diante). Neste exemplo usaremos o algoritmo k-means. É um bom algoritmo para obter uma impressão dos dados, mas é sensível aos valores iniciais, para que você possa acabar com um cluster diferente toda vez que executar o algoritmo, a menos que você defina manualmente os valores iniciais especificando uma semente (constante para o gerador de valor inicial). Se você precisa detectar uma hierarquia, é melhor usando um algoritmo da classe de técnicas de agrupamento hierárquico.

Uma outra desvantagem é a necessidade de especificar o número de clusters desejados em avançar. Isso geralmente resulta em um processo de tentativa e erro antes de chegar a um resultado satisfatório. conclusão.

A execução do código é bastante simples. Segue a mesma estrutura de todos os outros análises, exceto que você não precisa passar uma variável de destino. Cabe ao algoritmo aprenda padrões interessantes. A listagem a seguir usa um conjunto de dados de íris para ver se o algoritmo pode agrupar os diferentes tipos de íris.

Listagem 3.10 Exemplo de classificação de íris

Imprimir as primeiras 5 observações dos dados quadro para tela; agora podemos ver claramente 4 variáveis: comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala.

Carregue dados de íris (flores) do Scikit-learn.

```
importar sklearn
do cluster de importação do sklearn
importar pandas como pd

dados=sklearn.datasets.load_iris()
X = pd.DataFrame(data.data, colunas = lista(data.feature_names)) imprimir X[:5] modelo =

cluster.KMeans(n_clusters=3, random_state=25) resultados = model.fit(X)

X["cluster"] = resultados.prever(X)
X["target"] = data.target X["c"] =
"lookatmelamimportant" imprimir X[:5]

resultado_classificação = X[["cluster",
        "destino","c"]].groupby(["cluster","destino"]).agg("contagem")
imprimir(classificação_resultado)
```

Adicione outra variável chamada "cluster" aos dados quadro. Isso indica a associação do cluster de cada flor no conjunto de dados.

Ajustar o modelo aos dados. Todas as variáveis são considerados independentes variáveis; aprendizagem não supervisionada não tem variável de destino (y).

Transforme dados da íris em Quadro de dados do Pandas.

Vamos finalmente adicionar uma variável de destino (y) ao quadro de dados.

Adicionar uma variável c é apenas um pequeno truque que usamos para fazer uma contagem posteriormente. O valor aqui é arbitrário porque precisamos de uma coluna para contar as linhas.

Três partes para este código. Primeiro selecionamos o cluster, alvo e colunas c. Então agrupamos pelo cluster e colunas de destino. Finalmente, agregamos a linha de o grupo com uma agregação de contagem simples.

Inicialize um modelo de cluster k-means com 3 clusters. O random_state é uma semente aleatória; se você não colocar, a semente também será aleatória. Optamos por 3 clusters porque vimos na listagem anterior que este pode ser um bom compromisso entre complexidade e desempenho.

A matriz que este resultado de classificação representa nos dá uma indicação se nosso agrupamento foi bem-sucedido. Para o cluster 0, estamos certos. Nos clusters 1 e 2 houve uma ligeira confusão, mas no total obtivemos apenas 16 (14+2) erros de classificação em 150.

A Figura 3.11 mostra o resultado da classificação da íris.

Esta figura mostra que mesmo sem usar uma etiqueta você encontrará clusters semelhantes à íris oficial classificação com resultado 134 (50+48+36) corretos classificações de 150.

Você nem sempre precisa escolher entre supervisionado e não supervisionado; às vezes combinando-os é uma opção.

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

Figura 3.11 Resultado da classificação da íris

3.4 Aprendizagem semissupervisionada

Não deveria ser surpresa para você saber que, embora gostaríamos que todos os nossos dados fossem rotulados para que possamos podemos usar as técnicas mais poderosas de aprendizado de máquina supervisionado, na realidade, muitas vezes comece com apenas dados minimamente rotulados, se é que estão rotulados. Podemos usar nosso não supervisionado técnicas de aprendizado de máquina para analisar o que temos e talvez adicionar rótulos ao conjunto de dados, mas será proibitivamente dispendioso rotular tudo. Nosso objetivo então é treinar nossos modelos preditores com o mínimo possível de dados rotulados. É aqui que entram as técnicas de aprendizagem semissupervisionada – híbridos das duas abordagens que já vimos.

Tomemos por exemplo o gráfico da figura 3.12. Neste caso, os dados possuem apenas dois rótulos observações; normalmente, isso é muito pouco para fazer previsões válidas.

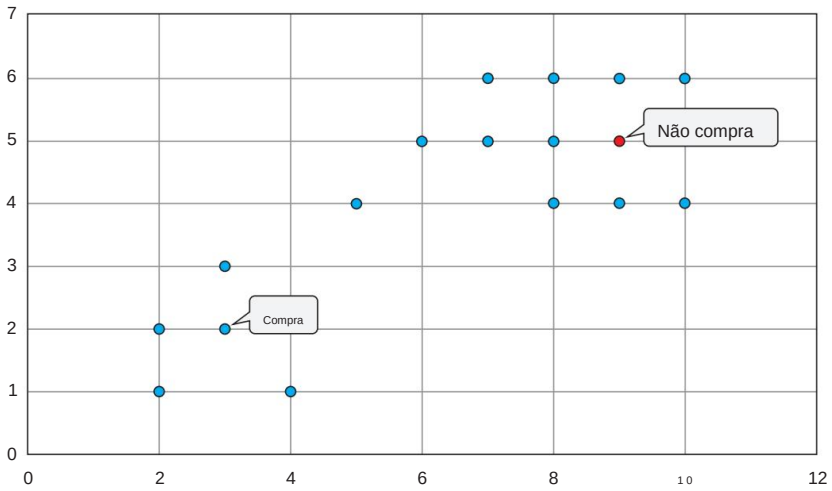


Figura 3.12 Este gráfico tem apenas duas observações rotuladas – poucas para observações supervisionadas, mas suficientes para começar com uma abordagem não supervisionada ou semissupervisionada.

Uma técnica comum de aprendizagem semissupervisionada é *a propagação de rótulos.* Nesta técnica, você começa com um conjunto de dados rotulado e atribui o mesmo rótulo a pontos de dados semelhantes. Isso é semelhante a executar um algoritmo de cluster no conjunto de dados e rotular cada cluster com base nos rótulos que eles contêm. Se aplicássemos esta abordagem ao conjunto de dados em figura 3.12, podemos acabar com algo como a figura 3.13.

Uma abordagem especial à aprendizagem semissupervisionada que vale a pena mencionar aqui é *a aprendizado.* Na aprendizagem ativa, o programa aponta as observações que deseja ver rotulado para sua próxima rodada de aprendizagem com base em alguns critérios que você especificou. Para Por exemplo, você pode configurá-lo para tentar rotular as observações sobre as quais o algoritmo tem menos certeza ou pode usar vários modelos para fazer uma previsão e selecionar o pontos onde os modelos mais discordam.

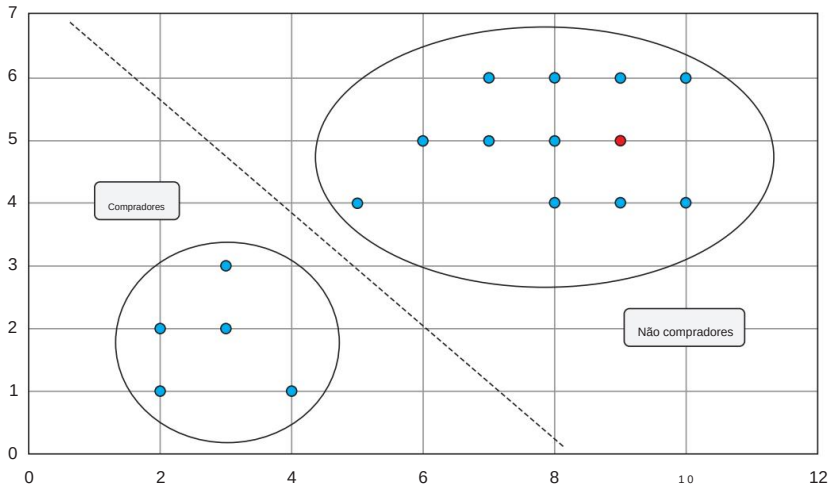


Figura 3.13 A figura anterior mostra que os dados têm apenas duas observações rotuladas, muito poucas para aprendizagem supervisionada. Esta figura mostra como você pode explorar a estrutura do conjunto de dados subjacente para aprender classificadores melhores do que apenas com os dados rotulados. Os dados são divididos em dois clusters pela técnica de clustering; temos apenas dois valores rotulados, mas se estivermos em negrito, podemos assumir que outros dentro desse cluster têm o mesmo rótulo (comprador ou não comprador), conforme ilustrado aqui. Esta técnica não é perfeita; é melhor obter os rótulos reais, se possível.

Com os fundamentos do aprendizado de máquina à sua disposição, o próximo capítulo discute o uso aprendizado de máquina dentro das restrições de um único computador. Isso tende a ser desafiador quando o conjunto de dados é grande demais para ser carregado inteiramente na memória.

3.5 Resumo

Neste capítulo você aprendeu que

- ÿ Os cientistas de dados dependem fortemente de técnicas que vão desde estatística e aprendizado de máquina até realizar sua modelagem. Existe um bom número de aplicações da vida real para máquinas aprendizagem, desde classificar o assobio dos pássaros até prever erupções vulcânicas.
- ÿ O processo de modelagem consiste em quatro fases:
  - **1** *Engenharia de atributos, preparação de dados e parametrização de modelos — Definimos* o parâmetros de entrada e variáveis para nosso modelo.
  - **2** *Treinamento do modelo – O* modelo é alimentado com dados e aprende os padrões ocultos nos dados.
  - **3** *Seleção e validação de modelo – Um* modelo pode ter um desempenho bom ou ruim; baseado em seu desempenho selecionamos o modelo que faz mais sentido.
  - **4** *Pontuação do modelo – Quando* nosso modelo é confiável, ele é liberado em novos dados. Se fizemos bem o nosso trabalho, isso irá fornecer-nos informações adicionais ou dar-nos uma boa previsão do que o futuro nos reserva.

ÿ Os dois grandes tipos de técnicas de aprendizado de máquina

1 *Supervisionado – Aprendizagem* que requer dados rotulados.

2 *Não supervisionado – Aprendizagem* que não requer dados rotulados, mas geralmente é menos preciso ou confiável do que o aprendizado supervisionado.

ÿ A aprendizagem semissupervisionada está entre essas técnicas e é usada quando apenas uma pequena parte dos dados é rotulada.

ÿ Dois estudos de caso demonstraram aprendizagem supervisionada e não supervisionada, respectivamente:

1 Nosso primeiro estudo de caso utilizou um classificador Naïve Bayes para classificar imagens de números como o número que eles representam. Também demos uma olhada na confusão matriz como um meio de determinar o desempenho do nosso modelo de classificação.

2 Nosso estudo de caso sobre técnicas não supervisionadas mostrou como poderíamos usar a análise de componentes principais para reduzir as variáveis de entrada para posterior construção do modelo, mantendo a maior parte das informações.