FIAP ON

INTELIGÊNCIA ARTIFICIAL
E COMPUTACIONAL

APRENDIZAGEM *DE MÁQUINA*

ERICK GALANI MAZIERO

5

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

SUMÁRIO

5 APRENDIZAGEM DE MÁQUINA

Os capítulos anteriores apresentaram métodos de busca, formalismos para representação de conhecimento, raciocínio e sistemas especialistas (SE), que atuam sobre bases de conhecimento.

Mas nem sempre é possível implementar um SE para simular o comportamento humano, gerando um agente inteligente. Imagine um carro autônomo que deve navegar em uma cidade cheia de outros carros, pessoas e obstáculos. Não é possível prever todas as prováveis localizações de todos os objetos no ambiente. Na realidade, o carro autônomo deve aprender a se movimentar considerando qualquer configuração de objetos. Para isso, é necessário “ensinar” o sistema do carro autônomo a se movimentar dando exemplos de como se comportar quando algum objeto atravessa a rua, de como virar o carro, frear, etc. A montadora Tesla, por exemplo, em 2019, apresentou detalhes técnicos da tecnologia de seus carros autônomos com características semelhantes.

Em muitos cenários, há muita informação disponível para aplicar o que aprenderemos neste capítulo: **Aprendizado de máquina (AM)**. No exemplo do carro autônomo, um humano pode dirigir um carro numa cidade, tomando as decisões de frear, acelerar, virar, etc. e armazenar as configurações do ambiente a cada decisão. Essas informações serão exemplos para que o sistema autônomo aprenda a navegar em uma cidade dadas as configurações do ambiente, capturadas por sensores.

A respeito do AM, tem-se a definição a seguir:

> Diz-se que um programa de computador aprende pela **experiência E**, com respeito a algum tipo de **tarefa T** e **performance P**, se sua performance P na tarefa T melhora com a experiência E. (MITCHELL, 1997)

Mitchell (1997) definiu o aprendizado de máquina em função de uma tarefa T que é ensinada a um programa de computador (modelo de aprendizado) por meio de exemplos, ou experiências, E. Se a performance P aumenta à medida que mais exemplos E são apresentados, o programa está aprendendo a tratar a tarefa T.

Para ficar mais claro, considere as instanciações de E, T e P para diversos problemas, na tabela.

Tabela 5.1. Exemplos de tarefas tratadas por aprendizado automático

Tarefa T	Previsão meteorológica
Experiência E	Valores de diversos sensores meteorológicos e estado meteorológico (por exemplo, chuva, sol, neblina, etc.)
Performance P	Quantidade de previsões corretamente identificadas
Tarefa T	Speech Recognition (reconhecimento de fala)
Experiência E	Sons gravados e respectivos textos transcritos
Performance P	Quantidade de transcrições corretas

Fonte: Elaborado pelo autor (2017)

Diversos algoritmos, ao longo do tempo, têm sido propostos para esse fim. Desde algoritmos puramente matemáticos até probabilísticos e simbólicos. Inclusive a aprendizagem de máquina pode-se dar por meio de exemplos com a solução indicada ou por exemplos cuja solução não tenha sido indicada por humanos.

5.1 A partir de exemplos, identificando padrões

Em geral, quando se fala em aprendizado de máquina e seus diversos algoritmos de aprendizado, o que se objetiva é a identificação de padrões. Na realidade, poderíamos chamar aprendizado de máquina por aprendizado de padrões.

Para exemplificar, considere um problema em que um jogador de tênis deseja saber se deve ou não sair para jogar, dadas algumas variáveis meteorológicas, como aparência do tempo, temperatura, umidade e vento.

A tabela “Características observadas para a decisão de sair ou não para jogar tênis” apresenta diversos exemplos das características observadas e a melhor decisão: sair ou não para jogar. Note que, para as características temperatura e umidade, são apresentados valores numéricos (temperatura em graus célsius e umidade em percentual) e valores nominais (temperatura entre quente, frio ou ameno e umidade entre normal e alta). Nas próximas seções deste capítulo serão

apresentados alguns algoritmos que podem tratar valores nominais ou apenas trabalham com valores numéricos.

Tabela 5.2. Características observadas para a decisão de sair ou não para jogar tênis

Aparência do tempo	Temperatura Numérica	Temperatura Nominal	Umidade Percentual	Umidade Nominal	Vento	Jogar?
Nublado	28	Quente	86	Alta	Não	Sim
Nublado	18	Frio	65	Normal	Sim	Sim
Nublado	22	Ameno	90	Alta	Sim	Sim
Nublado	27	Quente	75	Normal	Não	Sim
Chuvoso	21	Ameno	96	Alta	Não	Sim
Chuvoso	20	Frio	80	Normal	Não	Sim
Chuvoso	18	Frio	70	Normal	Sim	Não
Chuvoso	24	Ameno	80	Normal	Não	Sim
Chuvoso	17	Ameno	91	Alta	Sim	Não
Ensolarado	29	Quente	85	Alta	Não	Não
Ensolarado	27	Quente	90	Alta	Sim	Não
Ensolarado	22	Ameno	95	Alta	Não	Não
Ensolarado	21	Frio	70	Normal	Não	Sim
Ensolarado	24	Ameno	70	Normal	Sim	Sim

Fonte: Elaborado pelo autor (2017)

Um padrão que, olhando atentamente para a tabela, poderia ser observado é que não se deve sair para jogar quando a aparência do tempo é *ensolarado* e a temperatura nominal é *quente*. Outro padrão é que todas as vezes que a aparência do tempo for *nublado*, pode-se sair para jogar.

Os diversos algoritmos de aprendizado de máquina que serão apresentados utilizarão técnicas probabilísticas, matemáticas ou simbólicas (dentre outras) para identificar tais padrões e gerarão um modelo (segundo a técnica utilizada) para ser utilizado em dados cuja resposta esperada não é conhecida. Esse modelo, portanto, imitará o comportamento dos humanos que definiram as respostas esperadas dos exemplos utilizados no treinamento do aprendizado automático e terão, assim, um comportamento “inteligente”.

5.2 Principais etapas do aprendizado automático

No desenvolvimento de uma solução com o uso de aprendizado automático, há etapas que, na maioria das vezes, serão percorridas. A figura “Principais etapas para um aprendizado automático” apresenta essas etapas, iniciando pela **seleção dos dados**, passando pela etapa de **rotulação**, **balanceamento dos dados**, **extração de *features***, **seleção de *features***, **treinamento do modelo** e **avaliação** do modelo gerado. Geralmente, **refinamentos** podem ser feitos em alguma das etapas anteriores para alcançar melhores resultados na avaliação.

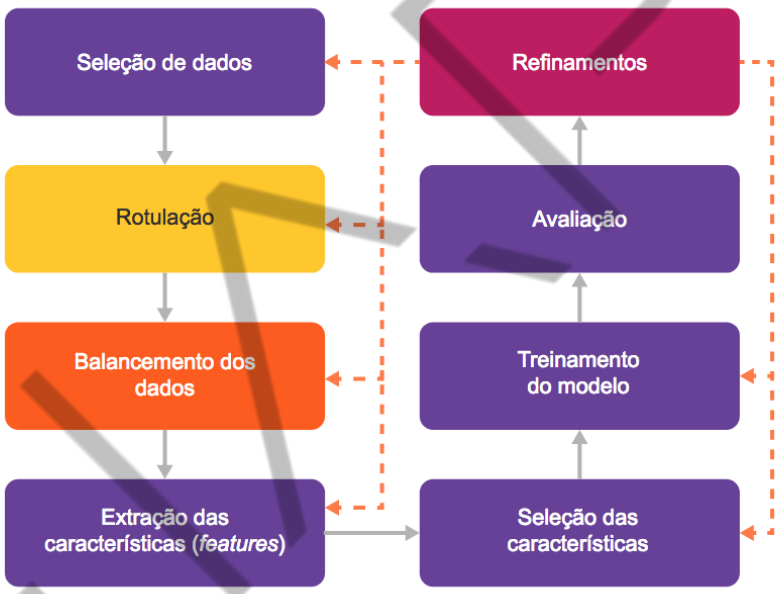

Figura 5.1. Principais etapas para um aprendizado automático
Fonte: Elaborado pelo autor (2017)

A **seleção dos dados** é uma tarefa primordial, pois, quanto mais representativos do problema forem os dados, mais chances de se obter um modelo de aprendizado que reagirá bem a novos dados do problema. Por exemplo, quando se deseja introduzir um agente inteligente proveniente do aprendizado automático em uma tarefa dantes realizada manualmente, deve-se obter o máximo possível dos dados gerados manualmente. Desta forma, tais dados possibilitarão um comportamento automático similar ao humano.

Imagine a tarefa de criar um atendente de *helpdesk* com aprendizado de máquina, os dados que devem ser obtidos são os que foram gerados tanto pelo

atendente quanto pelo usuário do *helpdesk*. Se o objetivo é gerar um modelo de predição meteorológica, dados dos mais variados sensores (temperatura, pressão, velocidade do vento, etc.) devem ser reunidos do maior tempo de medição possível.

Feita a obtenção dos dados, deve-se proceder à **rotulação** desses dados. Essa rotulação consiste, basicamente, em indicar, para cada exemplo dos dados obtidos, qual a resposta esperada. O termo muito utilizado na área para a resposta esperada é **classe** (em tarefas de classificação, como será visto a seguir, na taxonomia). Assim, para cada exemplo, deve-se dizer qual a classe esperada. No exemplo dos dados meteorológicos, para cada conjunto de medições, qual é a condição do tempo (ensolarado, nublado, chuvoso, etc.), por exemplo.

Após a rotulação dos dados, o **balanceamento** dos dados pode se mostrar necessário. Considere o caso em que os dados obtidos foram rotulados majoritariamente com apenas duas de três classes do problema. Por exemplo, o problema meteorológico, num conjunto de 10 mil exemplos, 4,9 mil são de tempo ensolarado, 5 mil de nublado e apenas 0,1 mil de chuvoso. Tem-se poucos exemplos de dados para predizer tempo chuvoso. Deve-se, portanto, balancear os dados para evitar que o modelo aprenda e prediga mais as duas classes mais frequentes, errando muito a classe menos frequente.

O balanceamento dos dados pode ser feito por duas técnicas principais: ***oversampling*** e ***undersampling.*** Na primeira técnica, deve-se obter mais dados da(s) classe(s) minoritária(s), seja buscando mais dados, seja gerando dados sintéticos. Na segunda técnica, exemplos das classes majoritárias devem ser dispensados, para ter proporção similar à(s) classes(s) minoritária(s). O *undersampling* só deve ser aplicado caso o conjunto de dados tiver quantidade suficiente para dispensar dados sem perder performance no aprendizado.

Em algoritmos tradicionais de aprendizado de máquina, a etapa de **extração de atributos** (*features*) é essencial, pois são os atributos extraídos dos dados que permitem aos algoritmos de aprendizado gerar modelos para predizer as classes desejadas.

No exemplo meteorológico, cada medida obtida por cada sensor pode ser considerada um atributo do problema. Considerando outro exemplo, o problema de dar crédito ou não a clientes de um banco, tem-se na tabela “Exemplo de atributos de

entrada para um aprendizado automático” atributos e seus possíveis valores que podem ser extraídos das fichas cadastrais dos clientes.

Tabela 5.3. Exemplo de atributos de entrada para um aprendizado automático

Atributo	Valor
Gênero	Masculino ou feminino
Idade	Número inteiro
Nível educacional	Primeiro grau incomp., primeiro grau comp., segundo grau incomp., segundo grau comp., superior incomp., superior comp., pós-graduação
Estado Civil	Solteiro, casado, divorciado, viúvo
Profissão	Nome da profissão
Trabalha há quanto tempo	Quantidade de meses
Consta em órgãos de protesto	Sim ou não
Salário	Valor decimal
Altura	Valor decimal

Fonte: Elaborado pelo autor (2017)

Nem sempre, no entanto, a extração de atributos é tão imediata, como mapear cada medição de um sensor ou campo de cadastro para um atributo. Considere o problema de detectar e-mails *spam*. Nesse caso, os atributos consistirão em *pistas* espalhadas pelo título e corpo do e-mail. Por exemplo, a ocorrência de determinadas palavras que indicam um *spam* (*blacklist*) ou palavras que indicam um e-mail confiável (*whitelist*) podem ser atributos dessa tarefa.

Um outro passo, às vezes opcional, é a **seleção de atributos**. Nem sempre um atributo que escolhemos para resolver um problema é realmente importante para a solução. Isso quando tal atributo não atrapalha a identificação da solução pelo algoritmo de aprendizado. Alguns algoritmos, como a árvore de decisão, que veremos mais adiante, tem a robustez de selecionar os atributos que melhor se adequam à solução do problema. Já outros algoritmos podem não ser robustos assim. A seleção de atributos é feita com base em alguma métrica, como correlação, ganho de informação ou entropia.

Finalmente, chega a etapa de **treinamento do modelo**. Nessa etapa, de acordo com os atributos e seus tipos, quantidade de dados, tipo de classes (discretas ou contínuas), dentre outras características, deve-se escolher o melhor algoritmo de aprendizado automático. Na próxima seção será apresentada uma taxonomia das principais técnicas de aprendizado.

O treinamento do modelo consiste na aplicação do algoritmo de treinamento nos dados que foram tratados nas etapas anteriores. Esses algoritmos, basicamente, encontrarão, para cada classe, o padrão que a indica. Posteriormente, com dados não vistos durante a etapa de treinamento, o modelo gerado poderá ser aplicado e as classes “identificadas”.

Nem sempre treinar um modelo é o fim da solução. Uma etapa de **avaliação** se mostra essencial para definir uma *performance* para a solução. A avaliação é feita utilizando-se algumas métricas, como *precisão*, *cobertura*, *medida-F*, *acurácia*. Outro recurso muito útil em alguns casos é a *matriz de confusão*. É importante salientar que muitos algoritmos de aprendizado apresentam, ao final do treinamento, muitas dessas métricas, mas nem sempre baseados em dados específicos para testar a solução.

Antes do treinamento, os dados devem ser separados em conjunto de treinamento (para efetivamente treinar o modelo) e teste. O conjunto de teste, que não foi utilizado durante o treinamento, deve ser utilizado na etapa de avaliação para a geração das métricas.

A avaliação assim como o uso efetivo do modelo de aprendizado no problema ao qual foi proposto devem ser analisados para identificar possíveis melhorias no processo. Por exemplo, ao se identificar que uma classe do problema está sendo confundida com outra, deve-se retornar à etapa de extração de *features* e escolher atributos que permitam distinguir melhor as classes confundidas.

Tendo uma visão geral do processo de aprendizado automático, na próxima seção será apresentada uma taxonomia, ou organização, dos principais paradigmas e técnicas de aprendizado automático.

5.3 Taxonomia do aprendizado automático

As diversas formas de aprendizado automático podem ser categorizadas conforme a taxonomia da figura “Taxonomia das tarefas de aprendizado de máquina”. Os três principais paradigmas de aprendizado são: **supervisionado**, **semissupervisionado** e **não supervisionado**. A grande diferença está nos dados utilizados durante o treinamento do modelo que encapsulará a “inteligência”.

No supervisionado, os dados são rotulados, ou seja, têm a indicação da resposta esperada, dada as características de cada exemplo. No não supervisionado, os dados não têm essa rotulação. E no semissupervisionado, parte dos dados são rotulados e parte não são rotulados.

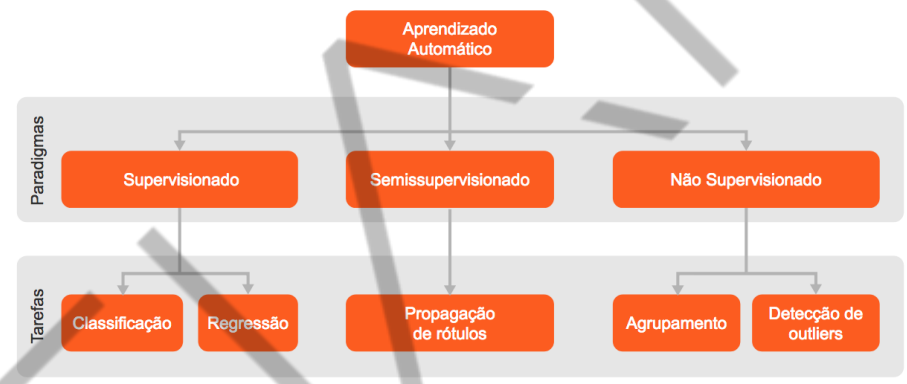

Figura 5.2. Taxonomia das tarefas de aprendizado de máquina
Fonte: Elaborado pelo autor (2017)

No paradigma supervisionado tem-se dois grandes grupos de tarefas: **classificação** e **regressão**. Na primeira tarefa objetiva-se, dadas as características de uma instância, definir uma classe (ou resposta esperada) dentre um número finito de possibilidades (conjunto discreto de valores). Na regressão, no entanto, esse conjunto de classes é infinito, como, por exemplo, um número real entre -1,0 e 1,0.

No paradigma não supervisionado, há também diversas tarefas dentre as quais podemos citar a principal, que é o **agrupamento** e também a **detecção de *outliers***. No agrupamento, o objetivo é identificar grupos nos exemplos (dados) segundo alguma métrica de similaridade dadas as características extraídas dos exemplos.

No paradigma semissupervisionado, quaisquer das tarefas mencionadas podem ser abordadas, iniciando tanto pela supervisão quanto pela não supervisão. A semissupervisão é indicada para os casos em que se tem uma parte dos dados rotulados e uma grande quantidade de dados não rotulados.

Para ficar mais claros esses paradigmas e suas principais tarefas, considere a seguir alguns exemplos de algoritmos de aprendizado de máquina, seus fundamentos e aplicabilidades.

5.4 Aprendizagem supervisionada

No aprendizado supervisionado, os dados servem como um *instrutor* do algoritmo de treinamento, indicando para cada exemplo a resposta esperada. Nos dados apresentados na tabela “Características observadas para a decisão de sair ou não para jogar tênis”, a última coluna (Jogar?) é a resposta esperada, que foi definida por humanos, ao observar as caraterísticas listadas nas outras colunas.

A seguir, serão apresentadas noções dos principais algoritmos de aprendizado supervisionado para a tarefa de classificação: Árvores de Decisão, *Naive Bayes* e SVM. Para a tarefa de regressão linear será apresentado um algoritmo de regressão linear.

5.4.1 Árvores de decisão, um paradigma simbólico

Uma árvore de decisão, a exemplo dos outros algoritmos de aprendizado em uma tarefa de classificação, é uma função que toma como entrada um conjunto de valores e retorna uma decisão. A árvore de decisão toma uma série de passos baseados nos valores de entrada até chegar a uma decisão. No Capítulo 2 foi definida a estrutura de dados em formato de árvore, como na figura, em que se tem basicamente dois tipos de nós: internos (incluindo o nó raiz) e raiz. Instanciando uma estrutura de árvore para tornar em uma árvore de decisão deve-se, a cada nó interno, definir uma decisão a ser tomada em relação aos valores de entrada (atributos ou *features*). Na figura tem-se a ilustração de uma árvore de decisão que toma decisões (D1 a D3) sobre os valores de três atributos (A1 a A3).

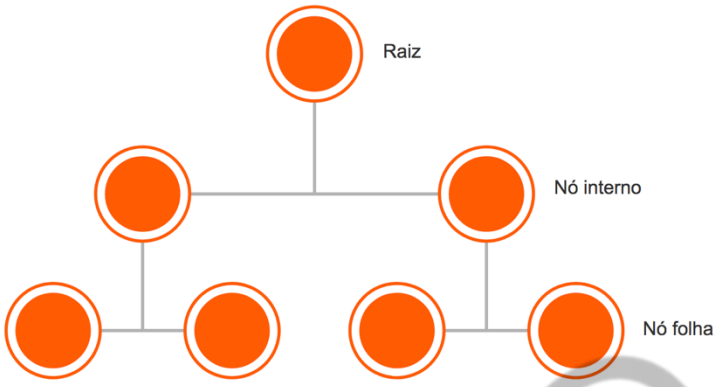

Figura 5.3. Representação dos principais elementos de uma árvore: raiz, nó interno e nó folha
Fonte: Elaborado pelo autor (2017)

Por exemplo, para se tomar a decisão 2 (D2), o atributo A1 deve ter valor menor ou igual a α e A2 deve ser maior que β. Já a decisão 1 (D1) pode ser escolhida para qualquer valor de A1, mas A2 deve ser menor ou igual a β ou A3 deve ser menor ou igual a γ.

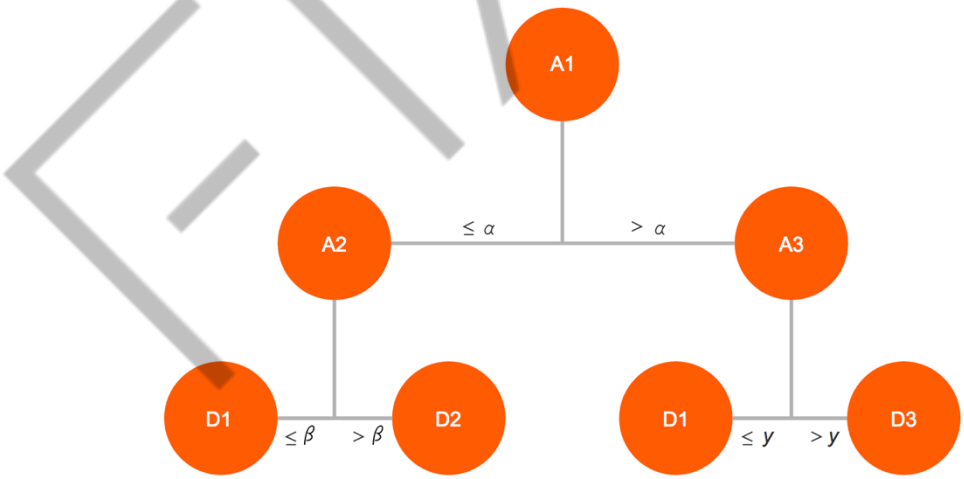

Figura 5.4. Exemplo genérico de uma árvore de decisão que utiliza atributos A1 a A3 e pode chegar a 3 possíveis decisões
Fonte: Elaborado pelo autor (2017)

Vale notar que uma árvore de decisão atua tanto sobre valores de atributos numéricos, como no exemplo anterior, como sobre valores nominais. Na figura, nos nós internos, as decisões foram alteradas para o caso de atributos nominais. Por exemplo, para a decisão 1 (D1), A1 deve ser igual a “alta” e A2 deve ser “p”.

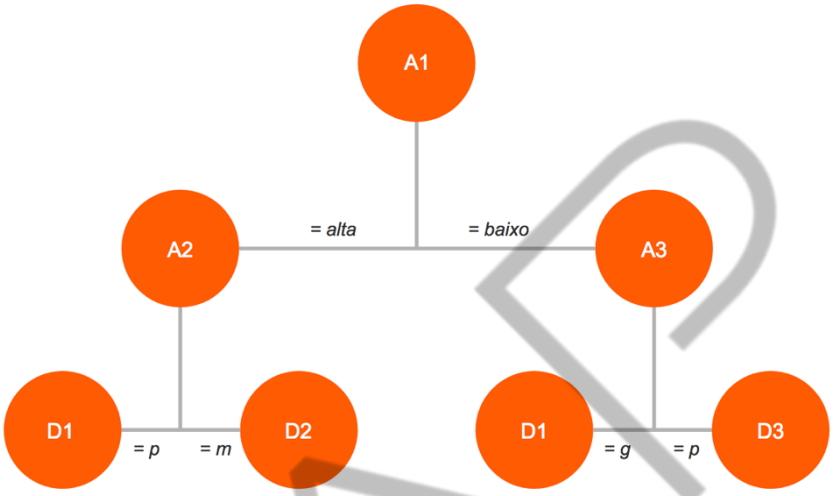

Figura 5.5. Árvore de decisão que atua sobre atributos com valores nominais
Fonte: Elaborado pelo autor (2017)

O processo de criação de uma árvore de decisão é também chamado de indução. Os algoritmos de indução objetivam a criação de árvores que tenham a menor profundidade possível sem perder seu poder e acurácia de classificação. Isso garante, quando o conjunto de atributos é muito grande, que a decisão seja tomada com maior eficiência.

Assim como encontrar a menor árvore de decisão para um problema, uma heurística simples pode encontrar uma solução aproximada: estratégia gulosa. Nessa estratégia, utilizam-se os atributos mais importantes primeiro para dividir o problema em problemas menores, até que todos os exemplos de treinamento da árvore sejam corretamente classificados pela árvore gerada.

Considerando os atributos da tabela (aparência do tempo, temperatura, umidade e vento) para decidir se devo ou não sair para jogar tênis (decisão SIM ou NÃO), podemos gerar a árvore de decisão ilustrada na figura “Árvore de decisão gerada sobre os dados da”. Há outras possíveis árvores de decisão? Mais ou menos eficientes?

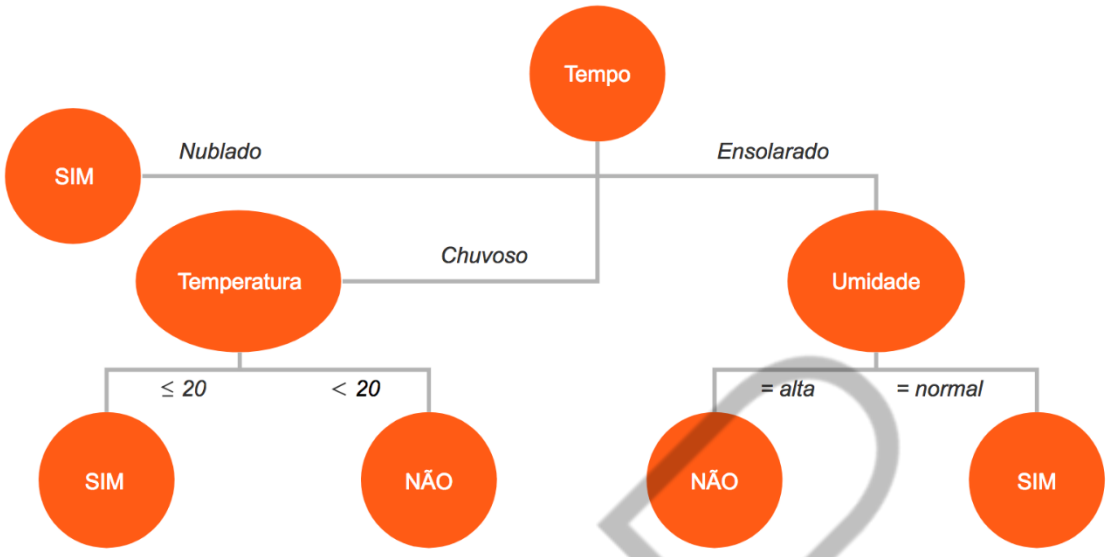

Figura 5.6. Árvore de decisão gerada sobre os dados da Tabela 5.2“Características observadas para a decisão de sair ou não para jogar tênis”
Fonte: Elaborado pelo autor (2017)

5.4.2 Support Vector Machines (SVM), um paradigma matemático

O SVM (*Support Vector Machines*) é um paradigma matemático para a criação de classificadores. É uma das técnicas mais utilizadas em aprendizado supervisionado, por basicamente 3 razões (uma ideia básica, um truque hábil e não é paramétrico):

- Generaliza bem para dados não vistos, pois define um **separador de margem máxima** em espaço multidimensional (cada atributo do problema define uma dimensão do espaço).
- Embora crie uma separação linear no espaço multidimensional, pode utilizar **kernels** (truque hábil) que transformam o problema num espaço de dimensão superior. Assim, os problemas que não têm separação linear nos dados de entrada (atributos) podem ter separação linear num espaço de maior dimensão (gerado pelo kernel utilizado).

- Não é paramétrico e armazena os dados de treinamento que são os mais indicativos para o separador de margem máxima (são os *support vector machines*).

Uma das limitações do SVM é que ele não trata diretamente de valores nominais, pois tem de gerar um espaço multidimensional, com cada atributo em um dos eixos desse espaço. Como representar valores nominais em um espaço numérico? Assim, problemas com valores nominais devem ter esses valores convertidos devidamente para valores numéricos. Há diversas técnicas de conversão, sendo a ***one-hot-encoding*** uma das mais utilizadas.

Considere o atributo tempo, que pode assumir três possíveis valores: *chuvoso*, *nublado* e *ensolarado*. Utilizando o *one-hot-encoding*, geraremos tantos atributos quantos forem os possíveis valores do atributo. Nesse caso, três. Cada um dos atributos gerados será binário, ou seja, assumirá apenas os valores 1 ou 0. Veja o exemplo em que, para representar um valor do atributo tempo utilizam-se três atributos numéricos (binários).

Figura 5.7. Conversão de valores nominais do atributo "tempo" para valores numéricos com a técnica "one-hot-encoding"
Fonte: Elaborado pelo autor (2017)

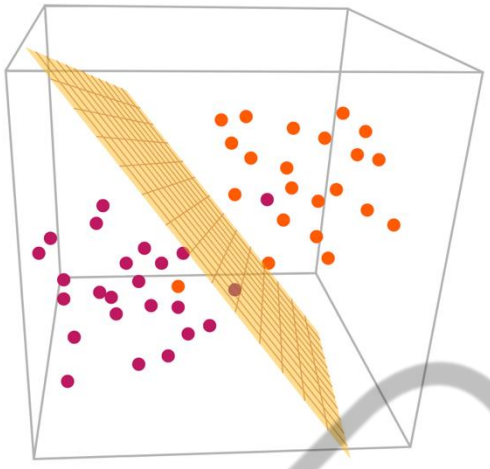

Figura 5.8. Hiperplano de separação entre duas classes em um espaço multidimensional
Fonte: Elaborado pelo autor (2017)

O SVM é essencialmente uma técnica de classificação binária, pois ele busca obter um hiperplano de separação entre duas classes, como ilustrado na Figura “Hiperplano de separação entre duas classes em um espaço multidimensional”. Mas o que fazer quando meu problema tem mais de duas classes? Nesses casos, treinam-se diversos classificadores binários e utiliza-se alguma métrica para escolher uma das classes.

Imagine um problema com três classes, representadas por cores: azul, verde e vermelho. Para decidir entre essas classes tem-se, por exemplo, duas formas de combinar classificadores binários SVM, a saber: **Um contra Todos** ou **Um Contra Um**. A figura “Formas de combinar classificadores binários em problemas multiclasse” ilustra essas duas formas de combinação.

Figura 5.9. Formas de combinar classificadores binários em problemas multiclasse
Fonte: Elaborado pelo autor (2017)

Vale salientar que a maioria das implementações SVM, em diversos pacotes de algoritmos de aprendizado de máquina, tratam automaticamente os problemas multiclasse.

5.4.3 Regressão linear, problema de classes contínuas

Tanto as árvores de decisão quanto o SVM tratam de problemas cujas classes são discretas, ou seja, quando já se sabe a priori a quantidade de classes a serem tratadas. Mas e quando o problema não tem um número predefinido de classes? Isto é, a classe é definida como um valor contínuo?

Por exemplo, em grandes portais de vendas pela internet, em que se tem milhões de produtos à venda, como tratar o preço desses produtos de forma dinâmica? Seria interessante monitorar a concorrência e definir automaticamente o preço dos produtos com o intuito de se obter o maior volume de vendas.

Nesse problema, a classe é contínua, pois o preço pode variar continuamente em um intervalo de valores. Imagine o produto X, ele pode custar desde $ 0,01 até $ 100,00.

Uma técnica muito utilizada nesses tipos de problema é a **Regressão Linear**, que é uma técnica matemática que visa minimizar o erro de uma equação cuja representação geométrica se adapte a um conjunto de pontos. Esse conjunto de pontos são os exemplos do problema a ser solucionado.

Em outras palavras, a regressão linear é a definição de uma equação que gera uma linha que melhor se adequa ao relacionamento entre os atributos de entrada X (dados de treinamento) e as classes Y (contínuas) do problema. Isso é feito obtendo-se coeficientes para os atributos de entrada X. Por exemplo, o preço de um produto Y (classe) é dado de acordo com seus atributos de entrada X (como nome, marca, modelo, etc., que, por serem valores nominais, devem ser convertidos em valores numéricos).

Considere a equação simplificada a seguir. A classe Y_i é definida como o resultado do somatório da constante α (que indica a interceptação da linha com o eixo vertical, numa representação geométrica) com os atributos X_i multiplicados pelo coeficiente β.

$$Y_i = \alpha + \beta X_i$$

Desta forma, consegue-se um valor contínuo Y, de acordo com os valores de X. Essa equação define uma reta, como ilustrada na figura “Exemplo de regressão linear, ilustrada geometricamente”, em que os pontos vermelhos são as instâncias de treinamento da regressão. A reta azul (regressão) é utilizada para, dado um valor de X, definir o valor Y (classe).

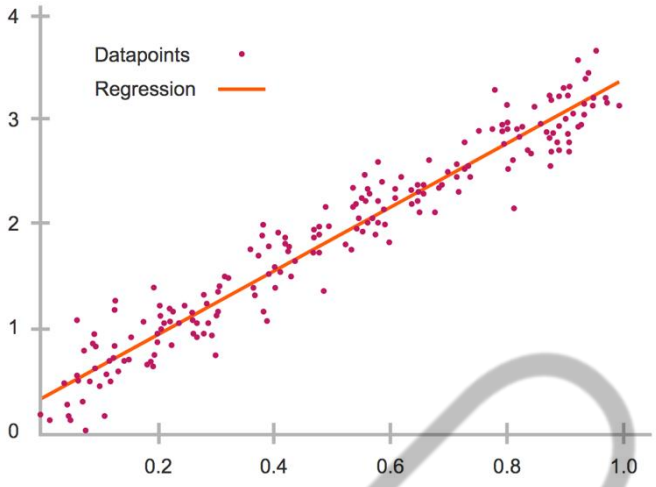

Figura 5.10. Exemplo de regressão linear, ilustrada geometricamente
Fonte: Elaborado pelo autor (2017)

5.5 Aprendizagem não supervisionada

Até agora, tratou-se de aprender a resolução de problemas quando se sabe quais são as possíveis respostas esperadas. Por exemplo, decidir se saio ou não para praticar esportes, qual será o preço do produto, etc. Não podemos nos esquecer, no entanto, de uma classe de problemas em que não se sabe quais classes esperar, ou seja, não temos exemplos de solução do problema.

Imagine um conjunto enorme de listas de compras realizadas em um supermercado. Será que podemos aprender alguma coisa com essa informação? Será que posso otimizar alguma coisa no supermercado, baseado no comportamento dos compradores, obtido de suas compras?

Imagine então a quantidade de logs gerado em um sistema bancário? E as informações coletadas por sensores meteorológicos em um país?

Esse tipo de aprendizado é dito não supervisionado, pois não se tem a definição ou anotação das classes do problema, mas ainda assim deseja-se aprender algo.

No aprendizado não supervisionado, podemos citar duas grandes tarefas: **associação** e **agrupamento**, dentre outras.

5.5.1 K-means, um método de agrupamento

O **K-means** é um dos métodos mais conhecidos do aprendizado não supervisionado. Esse é um método de agrupamento que, aplicando alguma técnica de similaridade vetorial, busca identificar K grupos nas instâncias de treinamento.

A figura “Aplicação do método K-means, com K = 3” ilustra a aplicação de um algoritmo de agrupamento K-means, com K = 3 (grupos azul, verde e vermelho, à direita). Para a definição de um grupo, o algoritmo escolhe um ponto que é chamado **centroide**. Definidos os centroides de cada grupo, um novo ponto (instância composta por atributos) é alocado no grupo a que estiver mais próximo, de acordo com a medida de similaridade escolhida.

Dentre as medidas de similaridade, podemos citar a distância **Euclidiana**, distância de **Manhattan**, e distância de **Mahalanobis**.

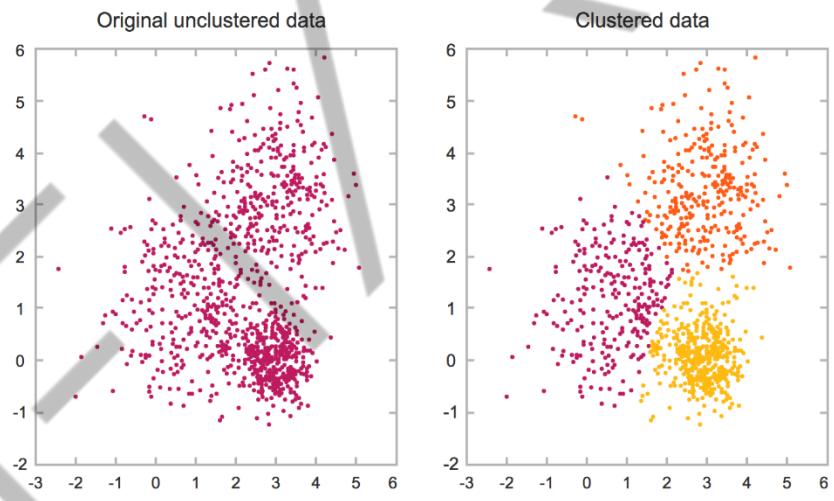

Figura 5.11. Aplicação do método K-means, com K = 3
Fonte: Elaborado pelo autor (2017)

Veja que aqui deve-se, a priori, definir a quantidade de grupos (K). Há outras técnicas que não necessitam que se defina a quantidade a priori, como os **métodos aglomerativos** ou **hierárquicos.**

5.6 Aprendizagem semissupervisionada

Entre o aprendizado supervisionado e o aprendizado não supervisionado há uma abordagem chamada de semissupervisionada. Nessa abordagem há diversos métodos que tomam vantagem de dados anotados, mas não deixam de aproveitar também os dados que não estão anotados e, geralmente, se tem em abundância.

Uma forma simples de aplicar o aprendizado semissupervisionado é utilizar os dados rotulados para treinar um modelo inicial. Esse modelo inicial será, então, aplicado aos dados rotulados com a finalidade de gerar novos dados rotulados. Esses novos dados rotulados podem conter erros, obviamente, pois provêm de uma anotação (classificação) automática. Dessa forma, a escolha de um método de treinamento que indique a confiabilidade de suas classificações é muito importante, pois permite escolher apenas as anotações que forem feitas com alta confiabilidade. Os exemplos com alta confiabilidade podem se juntar aos exemplos gerados manualmente e um novo modelo será treinado com mais dados anotados.

O algoritmo SVM, por exemplo, fornece, para cada classificação, a distância ao hiperplano de separação entre as classes do problema. Quanto maior for essa distância, maior a confiabilidade de classificação (consegue dizer o porquê?).

O processo descrito anteriormente é conhecido como **self-training**, um método de treinamento semissupervisionado. Existem outros algoritmos consagrados no paradigma semissupervisionado, dentre os quais podemos citar o **co-training** e suas variações (conhecidas como multivisão). Na realidade esses métodos são encapsuladores de outros algoritmos de aprendizado automático.

5.7 Redes neurais: matematizando o biológico

Redes neurais (RN) é um tópico estudado há muitas décadas (desde a década de 40, com a definição matemática de um neurônio: o Perceptron), mas seu início não foi favorecido dado o poder computacional da época. Já recentemente, com o aumento do poder computacional, principalmente com a popularização e barateamento do processamento paralelo em uma mesma máquina (GPUs), as redes

neurais emergiram e têm sido aplicadas com incrível sucesso em diversos problemas, com um novo rótulo: **Deep Learning**.

Dada a grande proeminência do assunto, é orientado que o aluno que lê este material busque maior aprofundamento no tema. Neste capítulo, as RNs estão abordadas muito superficialmente, por estar sendo abordadas com outras técnicas de aprendizado automático.

Basicamente, uma rede neural é composta por uma grande quantidade de neurônios artificiais, formalizados matematicamente. Uma formalização matemática é exemplificada na figura “Formalização matemática de um neurônio”. Um neurônio recebe (nos dendritos) e passa (ou não) impulsos elétricos (pelos axônios) no contato com outros neurônios. O que define se o sinal será repassado e com qual intensidade é o núcleo do neurônio.

Na formalização, cada neurônio tem n entradas (X_0 a X_n), e cada entrada é multiplicada (ponderada) por um peso w. Essas entradas ponderadas são somadas e uma função de ativação é aplicada, definindo a saída do neurônio, que será repassado para outros neurônios.

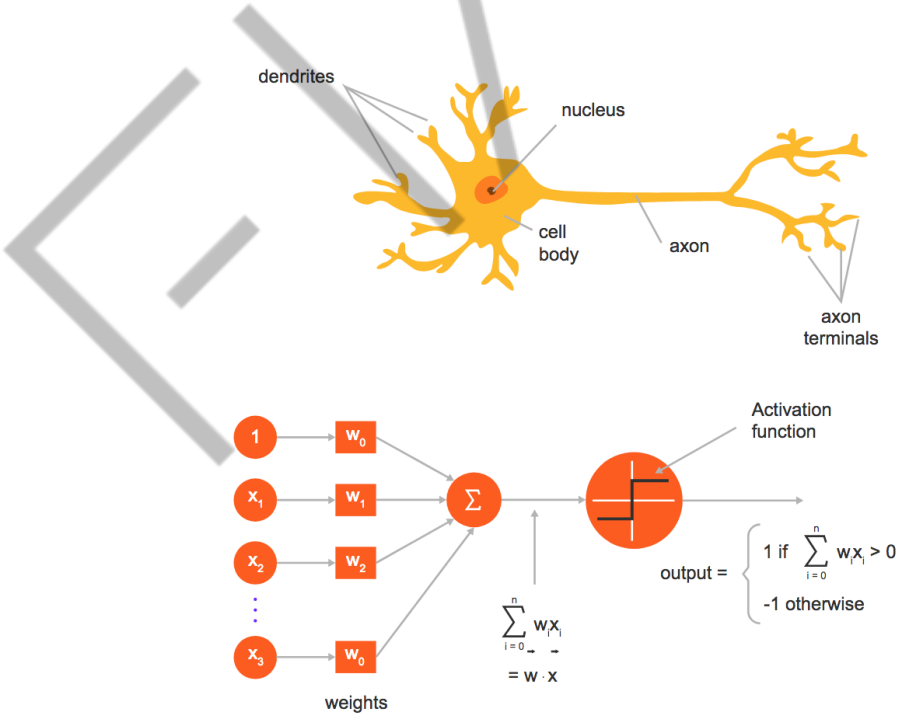

Figura 5.12. Formalização matemática de um neurônio
Fonte: Elaborado pelo autor (2017)

Há diversas funções de ativação. Na figura “Formalização matemática de um neurônio”, uma função de ativação *degrau* foi utilizada (há também as funções *sigmoide, tangente hiperbólica, softmax,* dentre outras). Nessa função, caso o somatório seja maior que zero, o sinal é repassado para o próximo neurônio. Caso o somatório seja menor ou igual a zero, o sinal é retido.

O aprendizado, na maioria das arquiteturas de redes neurais, é realizado no ajuste dos pesos aplicados às entradas X. O ajuste dos pesos W é realizado no processo chamado **backpropagation**. Nesse processo, os atributos X são inseridos na rede e a resposta obtida pela rede é comparada à resposta esperada (no caso de um aprendizado supervisionado). Caso a resposta esperada seja diferente da resposta predita, os pesos são ajustados com o intuito de que a rede prediga a resposta esperada. Um dos métodos mais conhecidos e utilizados no backpropagation é conhecido como **gradiente descent** (gradiente descendente).

Há diversas arquiteturas de redes neurais. Entenda arquitetura como a forma de combinação de milhares, milhões ou bilhões de neurônios com o intuito de realizar o aprendizado automático. Podemos citar as seguintes arquiteturas (as breves descrições apresentadas podem parecer um tanto complexas, mas à medida que o leitor tomar familiaridade com redes neurais, irá entender mais cada arquitetura):

- Multilayer Perceptrons: baseada no agrupamento de diversos Perceptrons em camadas, permite o tratamento de problemas não lineares. O Perceptron trata apenas de problemas lineares, isto é, cujas classes do problema sejam separáveis por um hiperplano no espaço de atributos.
- Redes Convolucionais: uma arquitetura amplamente utilizada no tratamento de imagens. Nessa arquitetura, os dados de entrada são tratados como matrizes numéricas tridimensionais cujas dimensões são alteradas a cada convolução na rede até gerar uma distribuição de probabilidades na camada de saída da rede, para tomada de uma decisão.
- Redes Recorrentes: muito utilizada no tratamento de dados sequenciais, como sequência de palavras, sons e dados temporais, por exemplo. Nessa arquitetura, a predição de um dado é utilizada na predição do próximo dado (série temporal, como os exemplos citados).

- Long Short-Term Memory (LSTM): nessa, tem-se a introdução de unidades LSTM que ajudam a rede a aprender correlação “mais espaçadas” entre dados em uma série temporal. Esse espaço pode ser até mais de 1000 passos de distância na sequência dos dados. As LSTMs são um avanço sobre as redes recorrentes para muitos problemas com longas séries temporais, como o Processamento da Linguagem Natural.
- Máquinas de Boltzmann: é um tipo de rede neural recorrente com uma característica mais estatística (estocástica). São úteis para resolver problemas combinatórios mais difíceis.
- Deep Belief Network: é uma opção para algumas limitações ao processo de aprendizado do *backpropagation*, tais como a necessidade de que os dados estejam todos rotulados (aprendizado supervisionado), tempo de aprendizado (muito demorado para redes muito profundas). O Deep Belief Network é uma opção de aprendizado não supervisionado em que os dados de entrada são agrupados. Os grupos aprendidos são as classes aprendidas. Um uso é o reconhecimento, agrupamento e geração de imagens, sequências de vídeos e dados de captura de movimento e o Processamento da Linguagem Natural.
- Deep Auto-Encoders: é uma arquitetura composta por duas redes simétricas, uma chamada de *encoder* e a outra, de *decoder*. São redes úteis para mapear uma representação em outra, como, por exemplo, mapear uma entrada textual em uma representação numérica.
- Generative Adversarial Network: é uma arquitetura bem profunda, composta por duas redes que são adversárias. Seu uso é no aprendizado da geração de alguma representação, seja imagem, texto ou música. Já imaginou uma rede que imite um compositor musical? Nessa arquitetura, em vez de prever uma classe Y com os dados de entrada X, ela tenta gerar o dados X para uma dada classe Y.

Essas arquiteturas podem ser combinadas, inclusive, para resolver os mais variados tipos de problemas, dependendo do tipo de dado a ser tratado, do tipo de predição (classes), etc.

Para um iniciante das redes neurais é importante a experimentação dessas diversas arquiteturas, desde as mais simples às mais complexas. Essa prática possibilitará a melhor escolha em face de um problema do mundo real.

5.8 Medidas de avaliação da aprendizagem

Por fim, como verificar o quanto um método automático de aprendizado aprendeu ou está aprendendo? Como comparar dois modelos gerados para uma tarefa específica? Há diversas medidas de avaliação que servem para essa finalidade. Veja, a seguir, algumas delas para um problema de classificação.

Considere Q a quantidade de instâncias disponíveis para realizar o aprendizado automático. Cada instância de Q tem uma classe y, que é a resposta esperada. Feito o aprendizado, cada instância q de Q, terá uma classe predita (y_p) pelo modelo gerado. A classe predita pode ser igual (acerto) ou diferente da esperada (erro).

Dadas as definições anteriores, podemos definir as seguintes medidas de avaliação:

Precisão: que é o número de acertos (a) sobre a quantidade de instâncias preditas pelo modelo (Q_p). Visto que nem sempre o modelo é capaz de predizer uma classe, nessa medida consideram-se apenas as respostas dadas pelo modelo gerado.

A precisão (P) é dada pela Equação 5.1.

$$P = \frac{a}{Q_p}$$

Equação 5.1. Cálculo da Precisão

Cobertura: número de acertos (a) sobre todas as predições esperadas pelo modelo (Q_e). Nessa medida, diferentemente da precisão, considera-se como denominador todas as instâncias fornecidas na avaliação do modelo, mesmo que o modelo não consiga dar alguma predição para alguma delas.

A cobertura (C) é dada pela Equação 5.2.

$$C = \frac{a}{Q_e}$$

Equação 5.2. Cálculo da Cobertura

Medida-F: é uma média harmônica que combina precisão e cobertura. Veja que algumas tarefas podem prezar mais pela precisão que pela cobertura. Outras, o inverso.

Para algumas tarefas, se o classificador der alguma resposta, é altamente desejável que essa resposta esteja correta. Então a precisão é mais importante que a cobertura. Para a maioria dos problemas, no entanto, é interessante um balanço entre a precisão e a cobertura, portanto, a medida-F é muito importante.

A medida-F (F) é dada pela Equação 5.3.

$$F = \frac{2\ P\ C}{P + C}$$

Equação 5.3. Cálculo da Medida-F

Matriz de confusão: é uma forma matricial de análise das respostas de um modelo. Na matriz de confusão, verifica-se qualitativamente o comportamento do classificador.

Para isso, considere um modelo de classificação que distinga entre as classes *a*, *b*, *c* e *d*. O modelo pode predizer todas as instâncias da classe a corretamente, mas pode predizer algumas instâncias da classe a como sendo da classe b. Esses erros são contabilizados nas medidas de P, C e F, mas ficam implícitos. Pela matriz de confusão é possível analisar essas “confusões” feitas pelo classificador.

A tabela “Exemplo de Matriz de Confusão” ilustra uma matriz de confusão. Pela matriz de confusão, o modelo predisse corretamente todas as instâncias com classe esperada *a*. Já para a classe *b*, 6 instâncias estão corretas, 2 foram confundidas com a classe *a*, e 1 instância confundida com a classe *d*. As da classe *c*, 20 estão corretas e 1 errada para a classe *a* e 1 errada para a classe *d*. O que podemos concluir sobre as instâncias da classe *d*?

Tabela 5.4. Exemplo de Matriz de Confusão

Esperada (ao lado) Predita (abaixo)	***a***	***B***	***c***	***d***
a	50	0	0	0
b	2	6	0	1
c	1	0	20	1
d	0	1	1	10

Fonte: Elaborado pelo autor (2017)

Outra informação importante é que os acertos estão na diagonal principal da matriz (células sombreadas) e os erros, nas outras células.

REFERÊNCIAS

ESTADÃO. **Tesla apresenta tecnologia de carros autônomos**. 22 abr. 2019. Disponível em: <https://link.estadao.com.br/noticias/empresas,tesla-apresenta-tecnologia-de-carros-autonomos,70002799995>. Acesso em: 25 jun. 2020.

HAN, Jiawei.; KAMBER, Michele. **Data Mining: Concepts and Techniques**, 2nd edition, Morgan Kaufmann, 2006.

MITCHELL, Thomas M. **Machine Learning.** New York, NY, USA: McGraw-Hill, Inc., 1997.

RUSSEL, Stuart; NORVIG, Peter. **Inteligência Artificial.** Rio de Janeiro: Elsevier, 2013.

GLOSSÁRIO

Entropia	Grau de pureza de um conjunto. Conceito que vem da Teoria da Informação, que define a medida de “falta de informação”, mais precisamente, o número de bits necessários para representar uma informação em falta.
Estocástico	Da Teoria Probabilística, um processo estocástico é aquele cujos eventos são determinados aleatoriamente.
GPU	*Graphics Processing Unit* ou Unidade de Processamento Gráfico.
Heurística	Estratégia dependente de um problema definida na busca por uma solução viável, em casos de problemas altamente complexos (computacionalmente) em termos de tempo e espaço.
Outlier	Um ponto, ou instância, ou exemplo, muito diferente dos demais considerados em um problema.