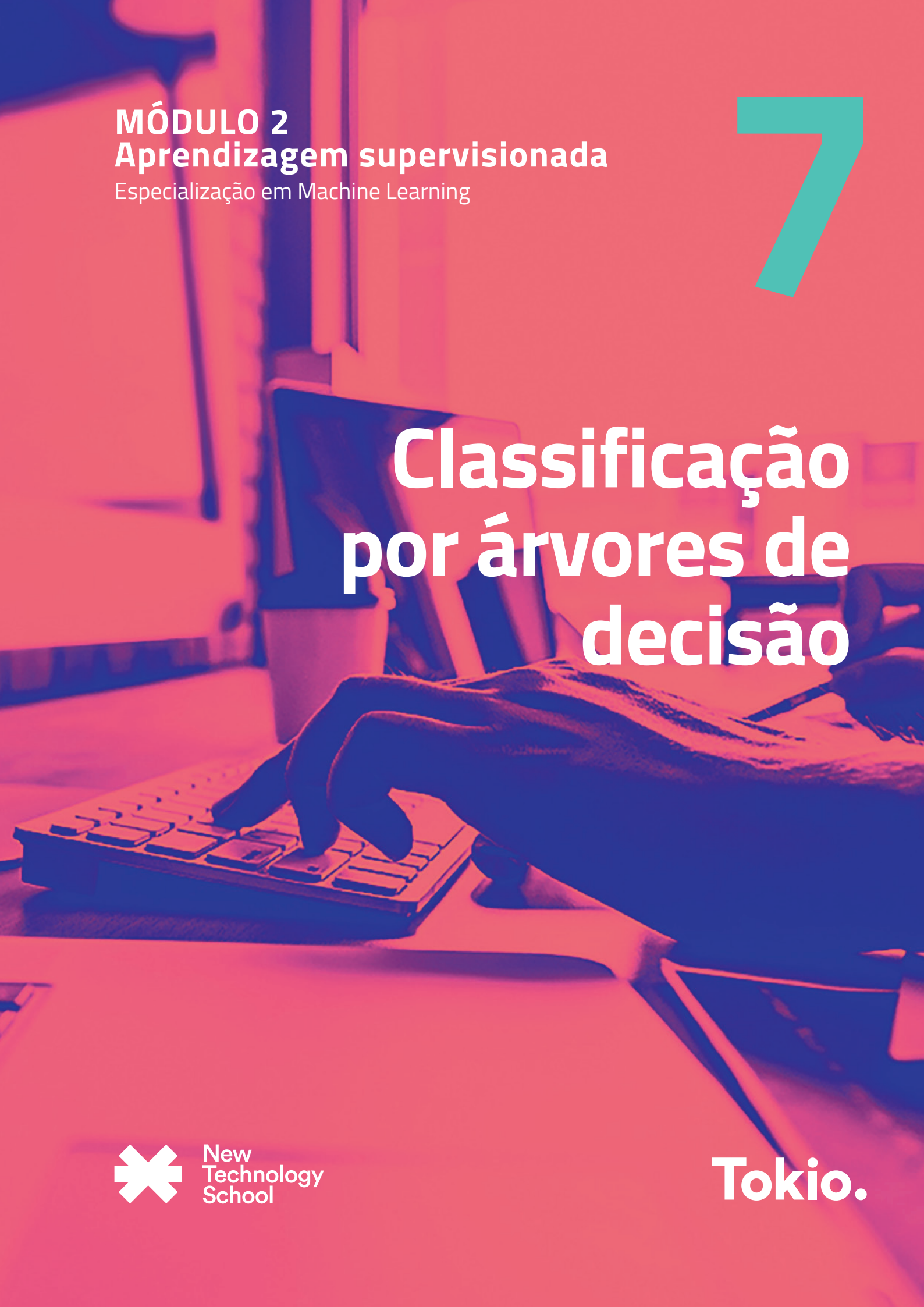
MÓDULO 2
Aprendizagem supervisionada
Especialização em Machine Learning

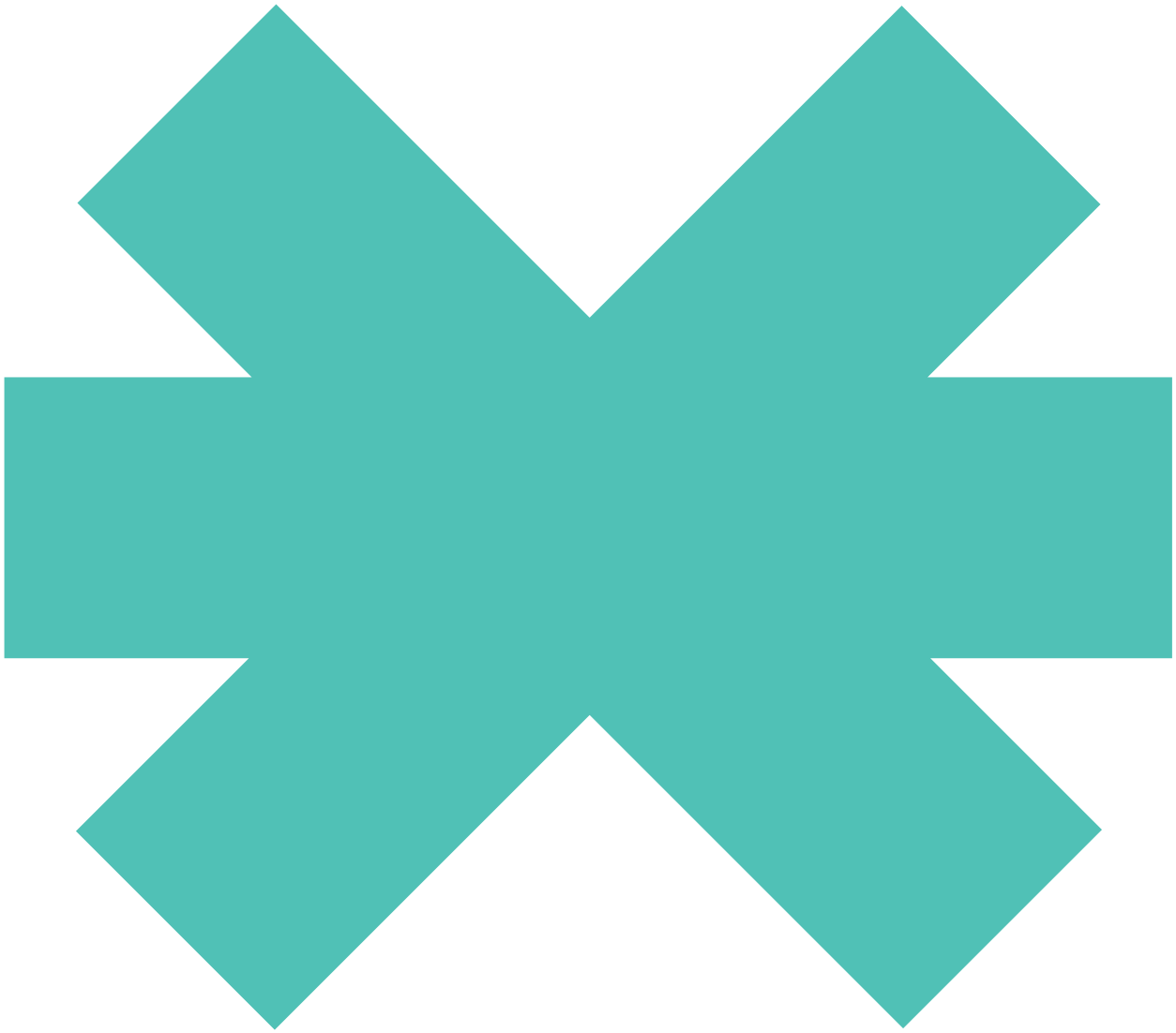
7

Classificação por árvores de decisão



New Technology School

Tokio.



7 Classificação por árvores de decisão

Sumário

Uma árvore de decisão é uma representação de uma árvore invertida, com nós ou ramos intermédios divididos e "folhas" ou nós finais, que nos ajudam a alcançar uma decisão.

São habitualmente utilizadas em qualquer contexto de tomada de decisões, quer seja de negócios, resolução de problemas, procedimentos de qualquer tipo, etc. No caso de machine learning, são usadas para adotar uma decisão final sobre o valor de uma regressão ou de uma classificação; assim ao treinar o modelo, conseguimos determinar a estrutura ótima da árvore e as decisões a tomar em cada nó.

Assim, em cada nó tomaremos uma decisão sobre seguir um caminho ou "ramo" ou outro determinando a característica a considerar nesse nó, quer seja uma categoria ou outra, ou uma comparação de maior ou menor, relativamente a um valor dado.

Por último, nas "folhas" ou nós finais acabaremos por escolher uma classe ou outra ou por efetuar uma estimativa sobre um valor de regressão numérico.

APLICAÇÕES

- Indústria: avaliação de materiais para produção ou construção.
- Produção: processos e sua otimização.
- Astronomia: filtro do ruído dos telescópios.
- Biologia molecular: análise de aminoácidos em sequências de genes para a investigação de genomas.
- Farmacologia: análise da eficácia de novos fármacos no desenvolvimento de medicamentos.
- Medicina: análise da eficácia de procedimentos e predições de recuperação, escolhendo um ou outro.

7.1 Representação

Exemplo: estimativa da probabilidade de se produzir *kyphosis* ou cifose (curvatura convexa excessiva da coluna vertebral) depois da cirurgia, em função da idade do paciente e da vértebra onde ocorreu a cirurgia.

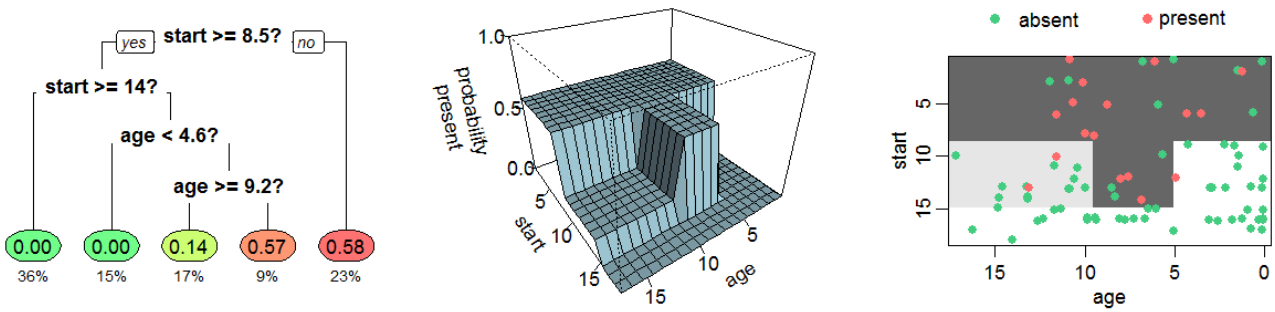


Figura 19. Stephen Milborrow (Wikipedia) - CC Attribution-Share Alike 4.0 International

- Esquerda: (nos nós terminais/"folhas") a probabilidade de *kyphosis* e a percentagem de pacientes em cada grupo/"folha".
- Meio: gráfico de perspetiva da probabilidade em função da idade e da vértebra.
- Direita: vista aérea do gráfico central com uma probabilidade mais alta nas zonas mais escuras e representação de casos em que a *kyphosis* esteve ausente ou presente.

7.2 Conceitos principais

- **Nó raiz:** representa a população completa que, posteriormente, vamos dividir em vários subconjuntos, segundo os ramos que sigamos.
- **Divisão/*splitting*:** processo de dividir um nó, em dois subnós ou ramos, na continuação do mesmo. Trata-se de converter uma "folha" ou nó final em dois "ramos" com duas "folhas" ou nós, na continuação do mesmo.
- "**Folha"/nó terminal**: nó que não é mais subdividido e representa uma decisão final sobre a variável objetivo, ou seja, sobre o valor a estimar na regressão ou sobre a classe a atribuir na classificação.
- **Poda/*pruning*:** eliminar determinados subnós ou "ramos" da árvore de decisão para simplificar, tentando preservar a maior precisão possível. É o contrário da divisão ou *splitting*.
- "**Ramo"/subárvore:** uma subsecção completa da árvore, a partir de um nó comum.
- **Nós pai-filho:** hierarquia de nós; um pais subdivide-se em vários filhos que podem, por sua vez, ter filhos próprios.

7.3 Variáveis objetivo categóricas e contínuas

Podemos usar árvores de decisão para realizar regressão numérica ou classificação, isto é, para estimar variáveis dependentes ou objetivas categóricas e contínuas.

Assim, por exemplo, uma árvore de decisão para classificação com uma variável objetivo-categórica poderia ajudar-nos a predizer se o aluno quererá jogar basquetebol na equipa da escola ou não.

Da mesma forma, por exemplo, uma árvore de decisão, para regressão com uma variável objetivo contínua, poderá ajudar-nos a predizer o nível de rendimentos de um cliente, segundo a informação com a qual contamos, para o converter no objetivo de uma ação comercial ou não.

7.4 Divisão de nós ou *splitting*

Inicialmente, considera-se que todo o conjunto de dados está incluído no nó raiz. A partir daí, começamos a dividir este conjunto de dados em vários subconjuntos através de vários nós, que partem do nó raiz.

Considera-se, também, que cada nó possui um subconjunto de dados que, por sua vez, se divide nos seus nós "filhos" a partir da decisão que tomemos nele até às "folhas", que não se dividem mais e apresentam um conjunto de dados praticamente homogéneo (da mesma classe ou valor de regressão).

Assim, podemos imaginar que todos os exemplos num nó estão contidos nele, de forma que o nosso objetivo é encontrar um nó final ou "folha", onde todos os exemplos contidos apresentem a mesma categoria ou atributo que procuramos, o que implicará a maior precisão possível.

Para dividir nós, usamos valores de variáveis categóricas ("é louro ou moreno?") ou variáveis contínuas discretas ("tem mais de 120 m^2?").

Deste modo, os exemplos do conjunto de dados vão-se distribuindo consoante avançamos nos nós, em função dos seus atributos. A ordem na qual são distribuídos, ou seja, na qual os atributos são usados para dividir os dados, determina-se seguindo uma análise estatística da sua importância.

Consequentemente, procuramos um método que nos ajude a dividir os nossos dados em nós ou subconjuntos, o mais homogeneamente possível, entre si. Desta forma, a árvore será mais simples, pois terá menos estrutura ou número de nós.

Por outro lado, se o resultado de um dos ramos, que partem de um nó, não for tão homogéneo (tiver 60%-40% de exemplos de cada classe em vez de 95%-5%), necessitaremos de continuar a subdividi-lo até chegar a "folhas", o mais "puras" ou homogéneas possível.

Métodos para realizar *splitting*

Assim, quando realizamos *splitting*, procuramos:

- Que os nós finais sejam o mais homogéneos possível, já que um nó "folha" que só contenha uma classe proporcionará 100% de precisão, enquanto um que contenha exemplos de duas classes, na mesma proporção, apenas nos oferecerá 50%.
- Que os nós intermédios dividam o seu conjunto de dados, segundo a característica que mais os diferencie, no momento de encontrar a categoria ou valor de regressão final.
- Em suma, a ordem de consideração dessas características nos nós iniciais não só contribuirá para uma maior precisão, como também contribuirá, especialmente, para que a nossa árvore seja o mais simples possível, logo, fácil de treinar.

De uma forma ou de outra, a maioria dos métodos baseia-se na entropia do conjunto de dados resultante, que nos indica o quão homogéneo é, ou seja, como está "ordenado" ou "desordenado".

ÍNDICE GINI

Trata-se de uma medida de "pureza" do conjunto de dados. Mede a probabilidade de que um elemento do subconjunto de dados numa "folha", selecionado aleatoriamente, seja incorretamente catalogado, de acordo com a sua classificação.

Logo, quanto mais homogéneo for o conjunto de dados dessa folha e maior for a percentagem de exemplos, com a classe correta prevista por essa folha, maior será este índice.

Consequentemente, usamos o índice Gini para comparar várias características, no momento de escolher a mais adequada para subdividir os dados nesse nó, assim como para estimar se devemos subdividir esse "folha", convertendo-a num nó intermédio, em duas "folhas" finais.

Apenas pode ser utilizado para divisões binárias de nós.

GANHO DE INFORMAÇÃO

Os nós mais "puros" requerem menos informação para os descrever, menos características, regras e subnós, enquanto os nós mais "impuros" requerem mais.

A entropia também é um conceito que usamos nas teorias da informação. Quanto mais desordenado se encontrar um conjunto de dados, mais informação sobre eles necessitaremos para o subdividir.

Neste método, consideramos o ganho de informação de um nó pai e dos seus filhos, ao mesmo tempo, e calculamos todas as divisões possíveis, usando diferentes características para encontrar a mais ótima em cada nó.

O ganho de informação será, então, a diferença entre a entropia do pai e a soma das entropias dos filhos.

Continuamos o cálculo do ganho de informação para toda a árvore até que, finalmente, a correspondente a possíveis subdivisões ou *splits* seja 0 ou todas as folhas sejam completamente "puras".

7.5 Vantagens e desvantagens das árvores de decisão

Em comparação com outros métodos de regressão ou classificação, as árvores de decisão apresentam as seguintes vantagens e desvantagens.

VANTAGENS

- São mais fáceis de visualizar e compreender, logo, de avaliar ou usar para explicar.
- Por serem mais fáceis de visualizar, são mais úteis para explorar os dados de outros métodos. Assim, por exemplo, ajuda-nos a definir em quantos subconjuntos de dados se subdividiria o modelo, ou que características são as que mais influenciam, no momento de estimar a variável objetivo e em que ordem.
- Requerem relativamente pouca preparação dos dados, assim como uma escassa transformação destes em outros formatos ou tipos de variável.
- Podem usar-se características numéricas e categóricas no mesmo modelo, o que nem sempre é fácil noutros, ou requerem o pré-processamento dos dados.
- É um método não paramétrico, isto é, pode trabalhar com dados que não sigam uma distribuição paramétrica.
- É capaz de modelar relações não lineares entre os parâmetros.
- O número de hiperparâmetros a considerar e otimizar é quase nulo.

DESVANTAGENS

- Classifica os exemplos apenas em partições retangulares: imaginemos um diagrama 2D com duas dimensões segundo duas características; o modelo apenas proporá áreas retangulares que dividam o espaço do conjunto de dados em duas ou mais classes, de forma que não poderá criar uma linha divisória que não seja reta e não esteja baseada num retângulo; para além disso cada retângulo pressuporá um novo nó e uma maior complexidade do modelo.
- Por ter de discretizar as variáveis contínuas, perde informação com cada nó de decisão, que as use, assim como nas "folhas" finais.
- Pela própria estrutura e método de treino da árvore, pequenas variações nos dados podem originar árvores completamente diferentes, por isso existe uma alta variância entre modelos.
- Pode cair facilmente em mínimos locais e não encontrar o mínimo global ou estrutura ótima, por se tratar de um modelo treinado nó a nó e não em conjunto.
- Geralmente, pode proporcionar menor precisão que outros algoritmos.
- Com frequência, pode resultar numa árvore demasiado grande ou complexa perante conjuntos de dados ou modelos complexos.
- Pela sua alta complexidade e as razões expostas anteriormente, pode sofrer facilmente sobreajuste, um aspeto que, por isso, devemos monitorizar sempre.

7.6 Limitações ao tamanho da árvore

Para conseguir uma árvore menos complexa e/ou reduzir a variância e o possível sobreajuste, podemos estabelecer uma série de limitações ao treino do modelo, relativas ao tamanho ou estrutura da árvore final. Todos estes métodos, ao focarmo-nos em reduzir a complexidade ou sobreajuste, necessitam de uma avaliação em validação cruzada adicional, para estabelecer os valores ótimos.

Entre outros aspetos, podemos limitar, por exemplo:

- O n.º mínimo de exemplos para dividir um nó: se o nó conta com menos exemplos que este número, não poderá ser subdividido.
- O n.º mínimo de exemplos por folha: se uma folha, resultante de uma subdivisão, apresentar menos exemplos, o seu nó pai não se subdividirá em várias folhas e será uma folha que irá conter todos os exemplos.
- O n.º máximo de níveis: estabelecendo um n.º máximo de nós hierárquicos, em todos os ramos, podemos limitar a profundidade máxima da árvore.
- O n.º máximo de folhas: pode usar-se em vez do n.º máximo de níveis.
- O n.º máximo de atributos a considerar para dividir: ao considerar dividir cada nó, podemos escolher qualquer atributo ou característica do conjunto de dados, exceto aqueles usados previamente nesse ramo, ou os que sejam homogéneos para todos os exemplos desse nó. Contudo, se tivermos em conta todos os atributos, pode produzir-se um sobreajuste. Idealmente, selecionamos um subconjunto aleatório, para cada nó de 30-40% das características totais, e apenas consideraremos estas características no momento de escolher uma para subdividir o nó.

7.7 *Pruning* ou poda da árvore

Podar a árvore permite-nos recortar alguns nós ou ramos para obter uma árvore menos complexa e com menor tendência a sobreajustar os dados.

O objetivo é recortar nós e reagrupar os que previamente dividimos, perdendo a mínima precisão total possível.

Para isso, treinamos o modelo criando uma árvore de decisão até um nível de profundidade ou nº de folhas máximo. Depois percorremos a árvore, desde as folhas até à raiz "podando" ou eliminando ramos, que usem características de menor importância.

Começando pelas folhas, eliminamos os nós, cuja classe mais popular entre os seus exemplos seja a mais popular dessa folha, com o objetivo de perder a menor precisão possível.

7.8 Árvores de decisão vs. modelos lineares

Habitualmente, se a relação entre os exemplos e a variável objetivo se pode aproximar através de um modelo linear, a regressão linear ou logística (para classificação) será capaz de obter maior precisão e sofrer menos sobreajuste.

Da mesma forma, se não existir uma relação linear, entre uma ou mais variáveis independentes e a variável objetivo, ou for muito complexa, uma árvore de decisão será capaz de treinar um modelo com maior precisão.

Como foi referido anteriormente, uma árvore de decisão é muito mais simples de visualizar e explicar, de modo que se procurarmos um modelo que possamos interpretar, é mais fácil utilizar uma árvore de decisão que um modelo linear ou logístico e a sua função hipótese.

7.9 *Bootstrap*

O *bootstrap* consiste em criar múltiplas árvores imperfeitas, assim como um modelo que prediga o valor médio (para uma variável contínua) ou a categoria mais comum (para uma variável categórica) de entre as suas respostas, como a resposta final que iremos predizer ou classificar.

Em parte deve-se ao facto de múltiplas árvores complexas, profundas, sem *pruning* e, portanto, com alta variância, poderem diminuir, em grande medida, o seu sobreajuste total se os combinarmos entre si.

Para isso, criamos múltiplos subconjuntos do conjunto de treino com substituição (os exemplos podem aparecer em vários conjuntos simultaneamente) e, posteriormente, treinamos um modelo para cada subconjunto.

Finalmente, para novas predições, calculamos a estimativa de cada modelo e encontramos o valor médio ou mais comum, que usaremos como predição final.

Adaptive boosting ou *AdaBoost*

Trata-se de um método de *bootstrap* de árvores de decisão, onde adaptamos esse *boost* ou melhora a predição das árvores, com o objetivo de transformar múltiplos modelos "débeis" num modelo "forte".

Um modelo "débil" é aquele que, ainda que apresente uma baixa precisão, melhora um modelo simplesmente aleatório.

- Começamos por treinar um modelo com o conjunto de dados de treino.
- Verificamos em que exemplos, esse modelo, se enganou e em quais acertou.
- Treinamos um segundo modelo, mas dando maior ênfase aos exemplos em que o modelo anterior se enganou e menor aos que acertou.
- Continuamos a treinar vários modelos por ordem, segundo este método, contando com pesos distintos em cada ocasião.
- Finalmente, classificamos o conjunto de dados, agrupando as regras, em função das áreas onde a maioria dos modelos classifica esses exemplos, em cada classe.

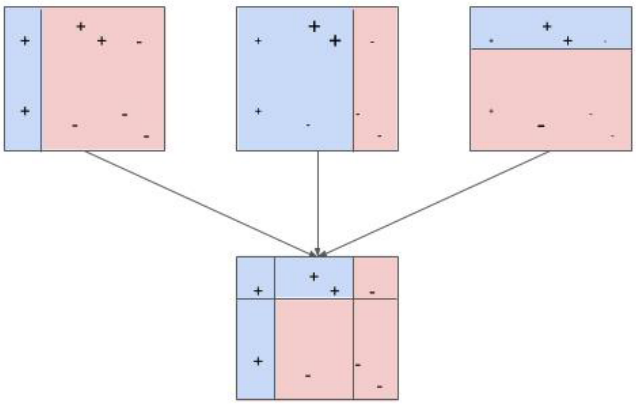


Figura 20. Classificação por árvores de decisão – AdaBoost.

Random-forest

Um dos problemas possíveis, que podem surgir com o *bootstrap* é que, no final, ainda que criemos árvores de decisão múltiplas, a maioria será bastante parecida, uma vez que se baseará nas mesmas características preditivas do conjunto de dados, para realizar o *splitting*, naquelas que mais influenciem a variável dependente ou objetivo.

Deste modo, através do *random-forest*, tentamos forçar uma maior variedade de preditores, escolhendo apenas um subconjunto aleatório para estimar em cada modelo.

Como rácio entre o subconjunto de preditores para estimar e o número total, pode escolher-se a raiz quadrada do número total de preditores.

A partir daqui, podemos usar qualquer algoritmo de *boosting*.

7.10 Algoritmo de treino

O algoritmo de treino completo, para uma classificação por árvores de decisão será:

1. Compilar os exemplos *X* e os seus resultados previamente conhecidos *Y*.
2. Dividir conjunto de dados em subconjuntos de treino, validação e teste.
3. Começamos no nó raiz.
4. Selecionamos o melhor preditor, usando algum dos métodos disponíveis e dividimos o conjunto de dados segundo aquele.
5. Continuamos a criar ramos e subconjuntos de exemplos em novos nós.
6. Repetimos até que todos os ramos sejam puros ou que tenhamos alcançado algum dos limites imposto ao tamanho da árvore.
7. A partir das "folhas", recuamos, podando a árvore e seguindo uma validação cruzada.
8. Verificamos a possível variância ou sobreajuste, no subconjunto de validação cruzada.
9. Finalmente, avaliamos a precisão final da árvore, no subconjunto de dados de teste.