EA991 - Laboratório de Aprendizado de Máquina

Métodos tradicionais de classificação

Prof. Denis G. Fantinato
Prof. Levy Boccato

Regressão logística

Regressão logística

- Trata-se de uma abordagem de classificação que tenta promover a separação das classes com base em fronteiras de decisão lineares.
  - Originalmente é formulada para o caso de classificação binária, mas pode ser estendida para multi-classe.

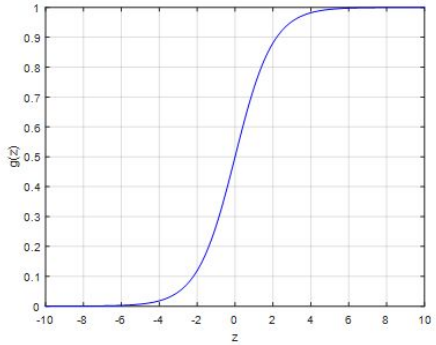
- **Caso binário:** o modelo produz uma única saída por meio do seguinte mapeamento:

$$\hat{y}(\mathbf{x}) = \frac{e^{(w_0+w_1x_1+\cdots+w_Kx_K)}}{1+e^{(w_0+w_1x_1+\cdots+w_Kx_K)}} = \frac{1}{1+e^{-(w_0+w_1x_1+\cdots+w_Kx_K)}}$$

Os coeficientes da combinação linear dos atributos de entrada são os parâmetros ajustáveis do modelo

Regressão logística

- Função logística: $g(z) = \frac{1}{1 + e^{-z}}$



- A regressão logística aplica a função logística ao resultado da combinação linear dos atributos de entrada (mais um termo adicional). Como a saída gerada está sempre entre 0 e 1, ela é interpretada como a *probabilidade de a entrada pertencer à classe positiva*, para a qual a saída desejada é *y* = 1.

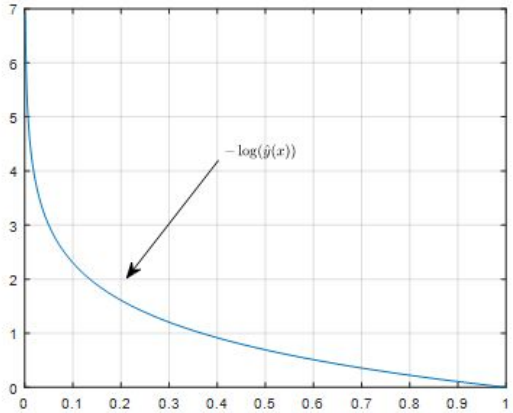
- A fronteira de decisão se manifesta quando há uma indeterminação, a saber, quando as probabilidades correspondentes às duas classes são iguais (ou seja, 0,5).

$w_0 + w_1 x_1 + \cdots + w_K x_K = 0$ → Hiperplano

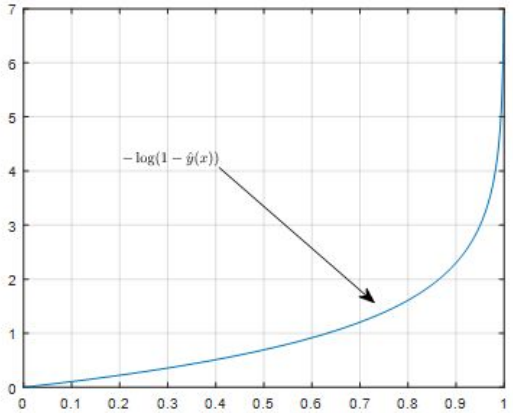
Regressão logística

- **Função de perda:**

Para uma amostra → $\text{Custo}(\hat{y}(\mathbf{x}); y) = \begin{cases} -\log \hat{y}(\mathbf{x}), & \text{se } y = 1 \\ -\log\big(1 - \hat{y}(\mathbf{x})\big), & \text{se } y = 0 \end{cases}$



$y = 1$



$y = 0$

Regressão logística

- **Função de perda:**

Reescrevendo a expressão: $\text{Custo}(\hat{y}(\mathbf{x}); y) = \underbrace{-y \log\big(\hat{y}(\mathbf{x})\big)}_{\text{Só exerce influência se } y=1} \underbrace{-(1-y) \log\big(1-\hat{y}(\mathbf{x})\big)}_{\text{Penaliza apenas se } y=0}$

Entropia cruzada:

$$J_{\text{CE}}(\mathbf{w}) = -\frac{1}{N} \sum_{i=0}^{N-1} y(i) \log\Big(\hat{y}\big(\mathbf{x}(i)\big)\Big) + (1-y(i)) \log\big(1-\hat{y}(\mathbf{x}(i))\big)$$

Regressão logística

- **Treinamento:** é feito com o auxílio de algoritmos iterativos que atualizam os parâmetros **w** à medida que os dados são apresentados ao modelo.

- **Ideia base: gradiente descendente**

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \alpha \nabla J_e(\mathbf{w}_i)^T$$

  - Cada iteração do algoritmo corresponde a uma atualização do vetor de parâmetros; uma *época* corresponde a uma apresentação completa do conjunto de amostras de treinamento.

Regressão logística

- **Caso multi-classe:**
  - A estratégia consiste em montar um modelo que produza *Q* saídas, tal que cada saída represente a probabilidade de cada padrão pertencer a uma classe específica. Isto pode ser feito a partir da função *softmax*.

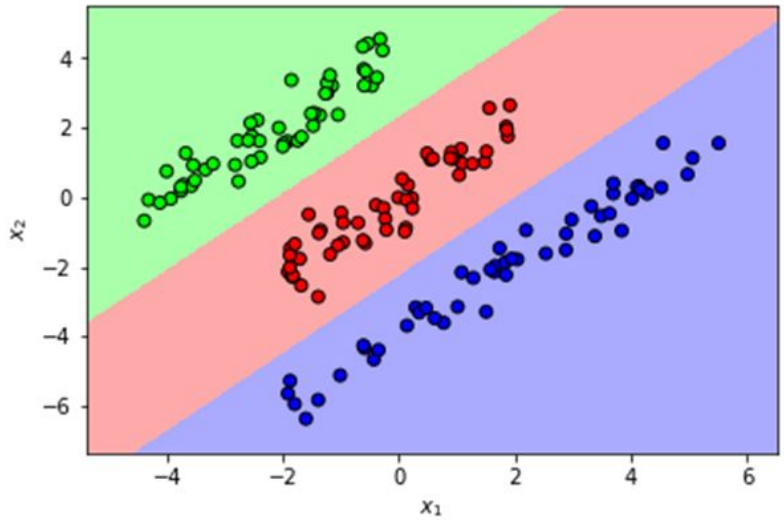
$$P(C_k|\mathbf{x}(i)) = \hat{y}_k(\mathbf{x}(i)) = \frac{e^{(\boldsymbol{\phi}(\mathbf{x}(i))^T \mathbf{w}_k)}}{\sum_j e^{(\boldsymbol{\phi}(\mathbf{x}(i))^T \mathbf{w}_j)}}$$

  - **Propriedades:**

$$\left.\begin{array}{l} \sum_{k=1}^{Q} \hat{y}_k(\mathbf{x}(i)) = 1 \\ 0 \leq \hat{y}_k(\mathbf{x}(i)) \leq 1 \end{array}\right\}$$ Preenche os requisitos de uma função probabilidade de massa

Regressão logística

- **Exemplo:**



Fronteiras de decisão obtidas com a regressão logística para um problema com três classes linearmente separáveis

Regressão logística

- **Regularização:** se refere a um conjunto de técnicas que buscam “frear” a flexibilidade do modelo a fim de reduzir as chances de ocorrer um sobreajuste aos dados de treinamento.

- **Esquema típico:** adição de um termo de penalização à função de perda que é proporcional à norma do vetor de parâmetros.
  - Implementação no scikit-learn: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

$$\min_w \frac{1}{S}\sum_{i=1}^{n} s_i \left(-y_i \log(\hat{p}(X_i)) - (1-y_i)\log(1-\hat{p}(X_i))\right) + \frac{r(w)}{SC}$$

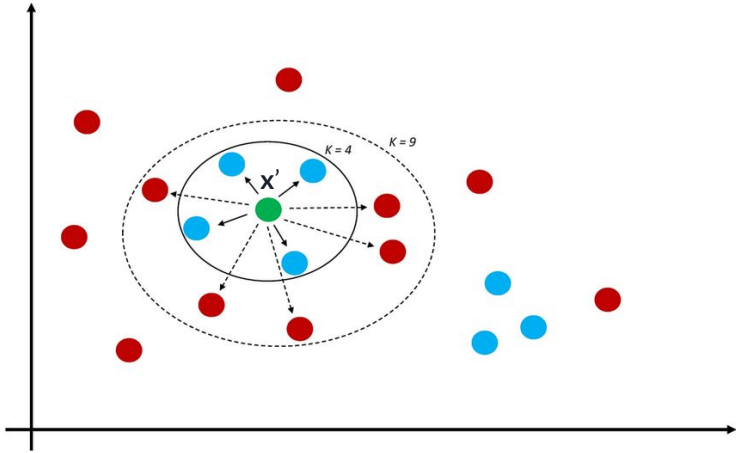
penalty	$r(w)$
None	0
ℓ_1	$\|w\|_1$
ℓ_2	$\frac{1}{2}\|w\|_2^2 = \frac{1}{2}w^T w$
ElasticNet	$\frac{1-\rho}{2}w^T w + \rho\|w\|_1$

k-nearest neighbors

k-nearest neighbors (kNN)

- Trata-se de um dos métodos mais simples para abordar os problemas de classificação e regressão, sendo de natureza *não-paramétrica*, uma vez que não há parâmetros a aprender.

- **Ideia:** para cada nova amostra de entrada **x**', a resposta gerada pelo kNN depende das saídas associadas às *k* amostras de treinamento que estão mais próximas a **x**' no espaço de atributos.



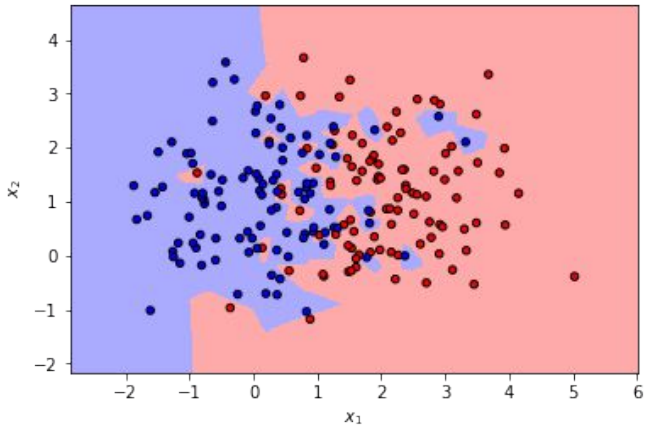
k-nearest neighbors (kNN)

- O uso do kNN envolve a definição de:
  - Uma métrica de distância a ser calculada no espaço dos atributos a fim de determinar os vizinhos mais próximos;
  - Um valor para o hiperparâmetro *k*, i.e., o número de vizinhos que são levados em consideração na geração da saída.
  - Uma estratégia de agregação das saídas (ou rótulos) dos *k* vizinhos mais próximos para determinar a resposta à nova entrada.

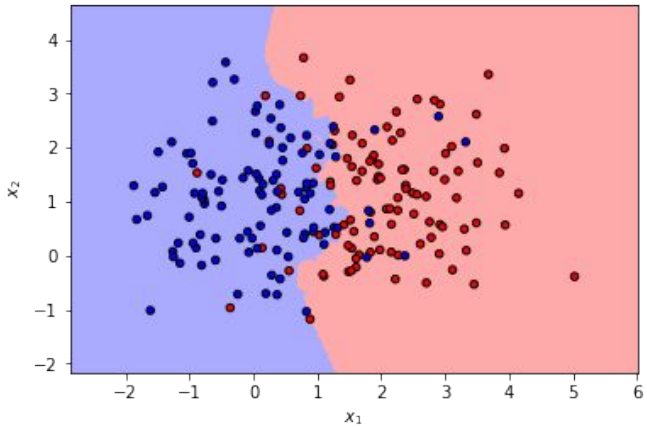
- No âmbito do problema de classificação, a saída do kNN pode ser simplesmente o voto majoritário dos *k* vizinhos mais próximos. Ou seja, um novo padrão **x**' é classificado como sendo pertencente à classe que contiver o maior número de vizinhos de **x**'.
  - É possível também atribuir pesos diferentes à contribuição de cada vizinho à decisão final; por exemplo, os pesos podem ser inversamente proporcionais às distâncias dos vizinhos ao padrão de entrada **x**'.

k-nearest neighbors (kNN)

- **Exemplo:**



$k = 1$



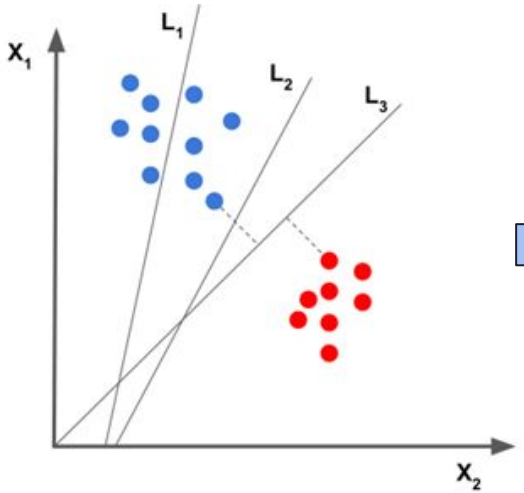
$k = 5$

Fronteiras de decisão obtidas com o kNN para dois valores de *k*. À medida que *k* aumenta, a fronteira tende a ficar mais suave e menos regiões isoladas são criadas para cada classe.

Máquinas de vetores-suporte

Máquinas de vetores-suporte (SVMs)

- **Problematização:** considere um conjunto de amostras representadas por dois atributos, x_1 e x_2, que pertencem a uma de duas classes que são linearmente separáveis.



- Reta L_1: incapaz de separar corretamente as duas classes.
- Reta L_2: está bem distante da classe vermelha, mas "perigosamente" próxima aos dados da classe azul.
- Reta L_3: paira de maneira mais segura entre as duas classes .

Máquinas de vetores-suporte (SVMs)

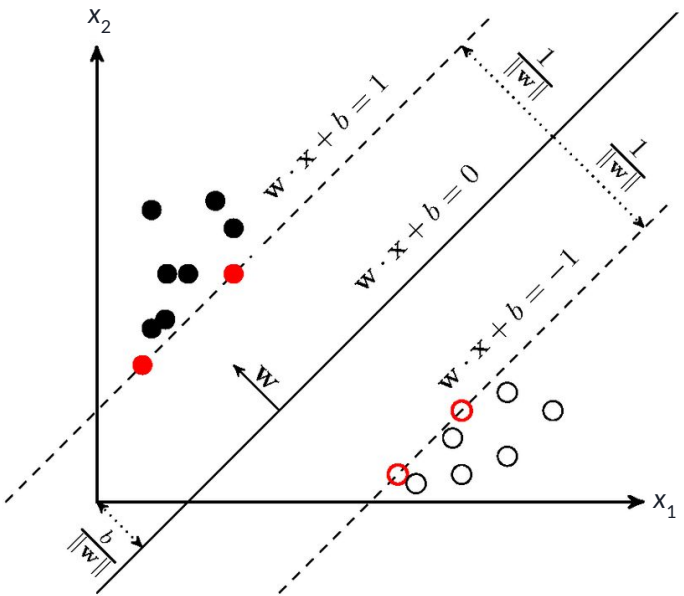
- A preferência pela reta L_3 traz à tona a noção de que quanto maior for a folga entre a fronteira e as amostras conhecidas das duas classes, mais “protegido” estará o classificador para lidar com novas amostras de cada classe que eventualmente se aproximem da região correspondente à outra classe.

- Essa ideia está no cerne do projeto das SVMs e é formalmente conhecida como *maximização da margem*.

- **Objetivo:** projetar um classificador linear (i.e., um hiperplano de separação das classes) de máxima margem.

Máquinas de vetores-suporte (SVMs)

- **Formulação original:**



Maximizar a margem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w}$$

Restrição: separar corretamente as classes

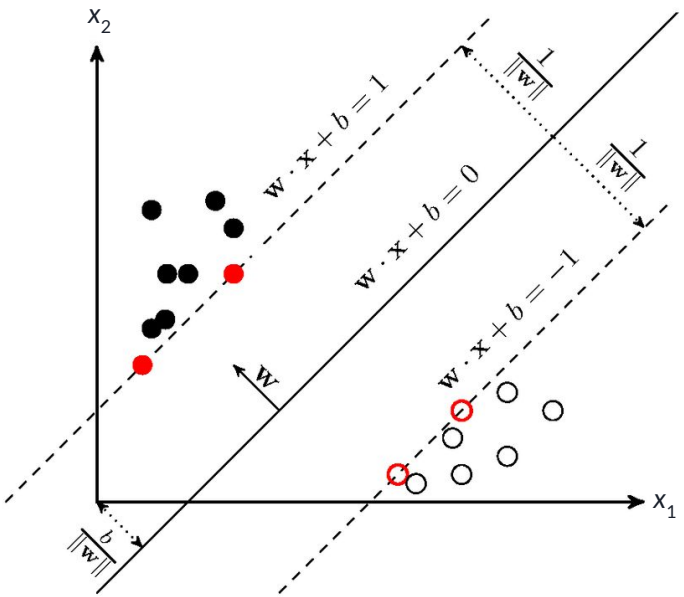
- $\mathbf{w}^T\mathbf{x}_i + b \geq 1$, se $y_i = 1$
- $\mathbf{w}^T\mathbf{x}_i + b \leq -1$, se $y_i = -1$

Juntando as duas condições:

- $y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Máquinas de vetores-suporte (SVMs)

- **Formulação original:**



Problema dual

$$\text{Maximizar } L(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i{}^T \mathbf{x}_j$$

$$\text{Sujeito a } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ e } \forall_{i=1}^{n} \alpha_i \geq 0$$

Máquinas de vetores-suporte (SVMs)

- **Solução:** $\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$

  - O vetor ótimo de pesos é uma combinação linear de amostras do conjunto de treinamento; mais especificamente, daquelas para as quais $\alpha_i \neq 0$. Somente estas amostras de treinamento determinam o hiperplano de máxima margem e, por isso, são chamadas de **vetores-suporte**.

- Resolvido o problema dual, a saída da SVM para uma nova amostra **x** é dada por:

$$\hat{y}(\mathbf{x}) = \sum_{i \in SV} y_i \alpha_i \mathbf{x}_i^T \mathbf{x} + b$$

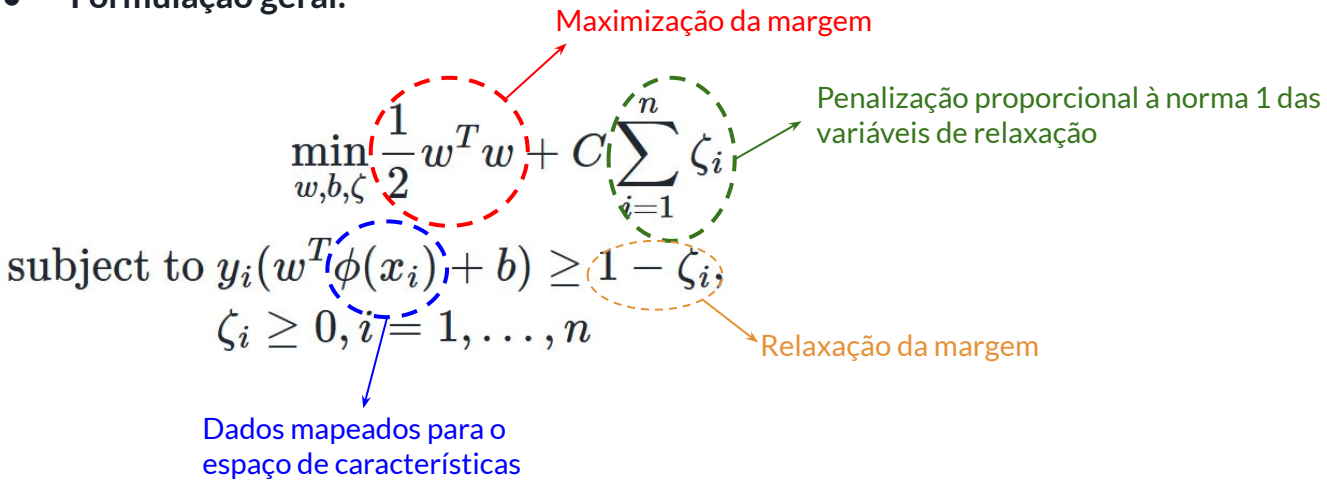
Máquinas de vetores-suporte (SVMs)

- **Extensões:**
  - Permitir a violação da margem e até mesmo erros de classificação nas amostras de treinamento.
    - Isto é realizado por meio da introdução de variáveis de relaxação, as quais devem ser minimizadas.
  - Conceder ao modelo flexibilidade suficiente para gerar fronteiras não-lineares de separação das classes.
    - Isto é realizado através do uso de funções *kernel* para implicitamente mapear os dados para outro espaço de características no qual o problema de classificação se torne mais linear.

Máquinas de vetores-suporte (SVMs)

- **Formulação geral:**

Problema primal



Máquinas de vetores-suporte (SVMs)

- **Formulação geral:**

Problema dual

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

$$\text{subject to } y^T\alpha = 0$$

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

$$e = [\underbrace{1 \cdots 1}_{n \text{ vezes}}]$$

$$Q_{ij} \equiv y_i y_j K(x_i, x_j)$$

Truque do *kernel*

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

Trunfo: os produtos escalares no espaço de características podem ser diretamente obtidos, isto é, sem o uso explícito do mapeamento $\phi(\cdot)$, a partir do espaço original em que se encontram os dados.

Máquinas de vetores-suporte (SVMs)

- Uma vez resolvido o problema de otimização dual, a saída da SVM para uma nova amostra **x** é dada por:

$$\sum_{i \in SV} y_i \alpha_i K(x_i, x) + b$$

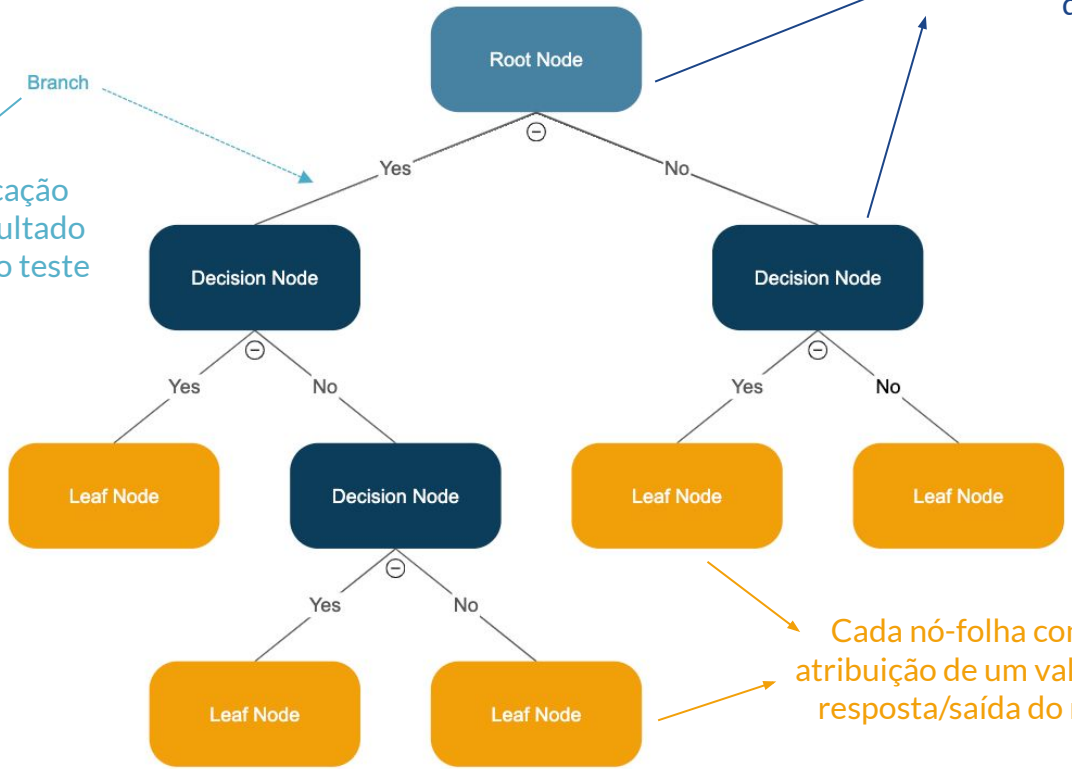
Árvores de decisão e *Random Forest*

Árvores de decisão

Cada nó interno, incluindo o nó raiz, define um teste envolvendo um atributo dos dados

Branch

Cada ramificação indica um resultado possível para o teste



Cada nó-folha contém a atribuição de um valor para a resposta/saída do modelo

Árvores de decisão

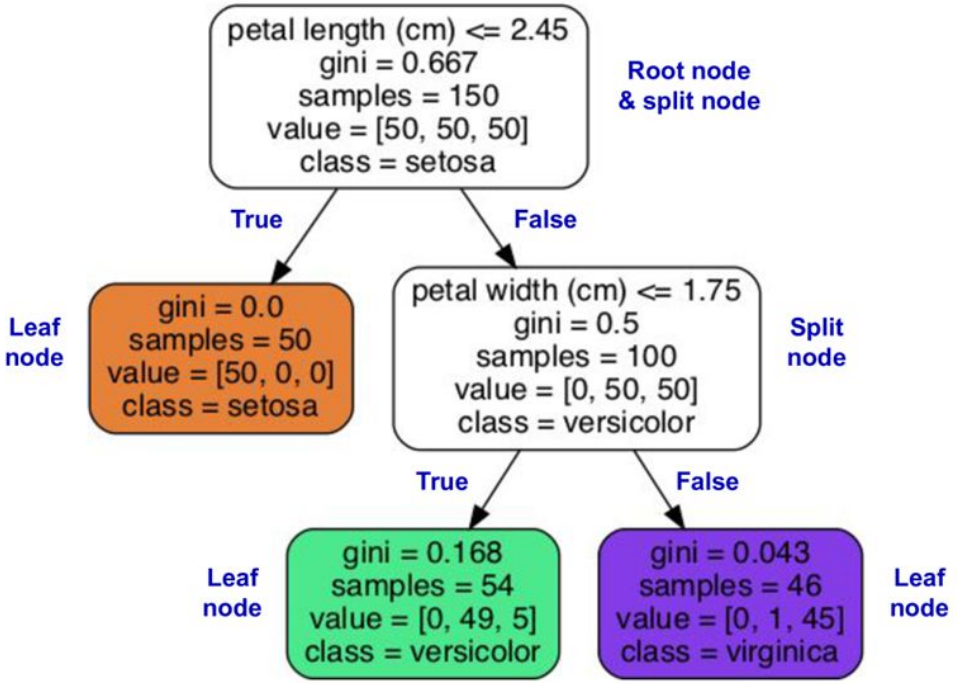
- As árvores de decisão são modelos não-paramétricos criados a partir de dados rotulados.

- Esse tipo de estrutura reflete uma forma intuitiva e bastante humana de analisar informações e tomar decisões, a qual se dá por meio de uma *sequência de perguntas*, em que a próxima pergunta é aplicada sobre um subconjunto diferente das amostras dependendo da resposta dada à pergunta anterior.

- O caminho percorrido desde a raiz até um nó-folha gera uma *regra de decisão*.

Árvores de decisão

- **Exemplo:**



Árvores de decisão

Como podemos derivar (induzir ou, como frequentemente mencionado na literatura, expandir (*grow*)) uma árvore de decisão a partir de um conjunto de amostras?

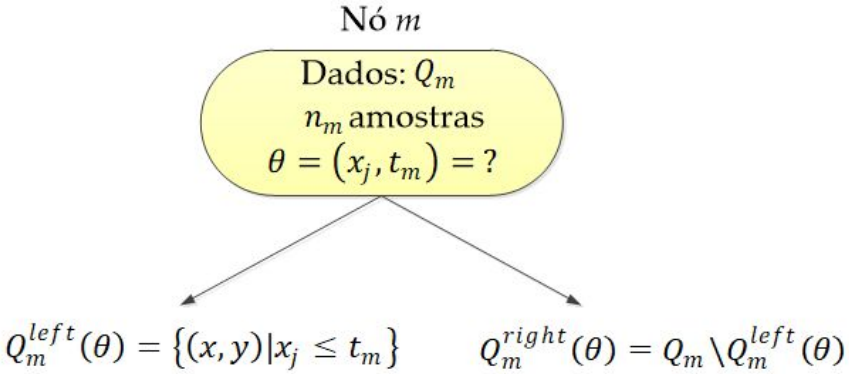
- Trata-se de um processo recursivo e guloso: em cada passo, há um subconjunto de amostras a considerar e é preciso escolher qual variável fará a divisão, bem como o valor do limiar para esse particionamento.

- Idealmente, estando em um determinado nó da árvore, gostaríamos de encontrar o teste – isto é, o atributo e o valor (limiar) correspondente – que tornasse os subconjuntos de amostras para os dois nós subsequentes tão "puros" quanto possível.

- **Impureza:** expressa por uma métrica *i(m)* que deve ser nula quando todos os padrões que chegam ao nó *m* possuem o mesmo rótulo (i.e., pertencem à mesma classe), e deve assumir valores elevados se as classes estão igualmente representadas neste nó (o que significa que o teste não teve um bom poder de discriminação).

Árvores de decisão

- **Resumo do algoritmo CART:**



Qualidade de um particionamento candidato no nó *m*: $G(Q_m, \theta) = \frac{n_m^{left}}{n_m} i\left(Q_m^{left}(\theta)\right) + \frac{n_m^{right}}{n_m} i\left(Q_m^{right}(\theta)\right)$

Problema: $\theta^* = \arg\min_{\theta} G(Q_m, \theta)$

Árvores de decisão

- **Medidas de impureza:**
  - A proporção de amostras na região R_m, correspondente ao nó m, que pertencem à classe k é:

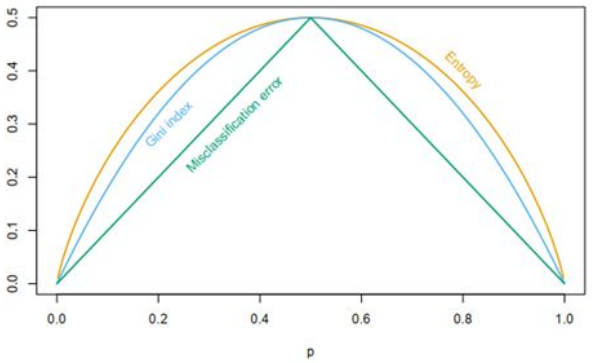
$$\hat{p}_{mk} = \frac{1}{n_m} \sum_{x_i \in R_m} I(y_i = k)$$

  - **Índice de Gini:**

$$i_{\text{Gini}}(m) = \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{Q} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

  - **Entropia:**

$$i_{\text{entropia}}(m) = -\sum_{k=1}^{Q} \hat{p}_{mk} \log \hat{p}_{mk}$$



Árvores de decisão

Pontos positivos	
1	São fáceis de compreender e de interpretar - modelos “caixa-branca”
2	Quase não exigem preparação prévia dos dados, podendo lidar com atributos numéricos e categóricos, além de faltantes
3	A hierarquia dos atributos em uma árvore de decisão reflete a importância que cada um possui para a tarefa
4	O custo de inferência é logarítmico em relação ao número de amostras usadas para treinar a árvore

Pontos negativos
O processo de indução pode criar árvores excessivamente complexas que não generalizam bem
O processo de indução é suscetível ao desbalanceamento de classes, podendo criar árvores enviesadas caso haja uma classe dominante
O algoritmo de indução toma decisões localmente ótimas (isto é, para cada nó), mas não garante a obtenção da árvore ótima
Alta variância / Instabilidade: pequenas variações nos dados ou nos hiperparâmetros podem levar a grandes mudanças na estrutura da árvore obtida

Random Forest (RF)

- Trata-se de uma extensão que explora várias árvores de decisão combinadas em um comitê (*ensemble*).

- **Ideia:**
  - Cada árvore é induzida considerando um subconjunto de amostras do *dataset* escolhidas aleatoriamente com reposição (*bootstrap*).
  - Em cada nó, o particionamento é definido por meio de uma busca exaustiva em um subconjunto aleatório de atributos.

- Essas duas fontes de aleatoriedade contribuem para reduzir a variância da floresta obtida.
  - Espera-se que os erros cometidos pelas diferentes árvores tenham baixo acoplamento.
  - Assim, ao considerar uma espécie de “média das predições”, alguns erros são cancelados.

Random Forest (RF)

Training Data
(Sample size, N= 6, No. of features, F = 4)

F1	F2	F3	F4	Y
2.1	0	400	-9	A
3.0	1	890	-42	B
2.2	1	929	0	B
4.0	0	324	-23	A
3.5	1	333	-15	A
6.0	0	215	-9	A