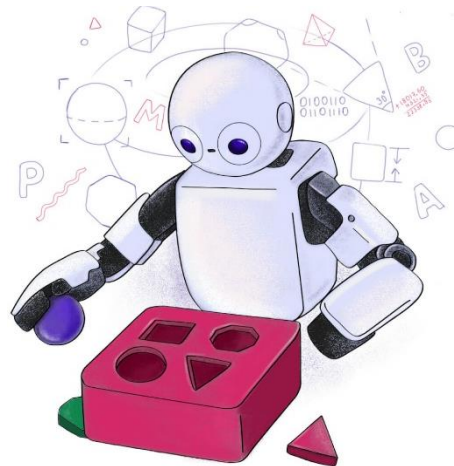


# TP555 - Inteligência Artificial e Machine Learning: *Árvores de Decisão*

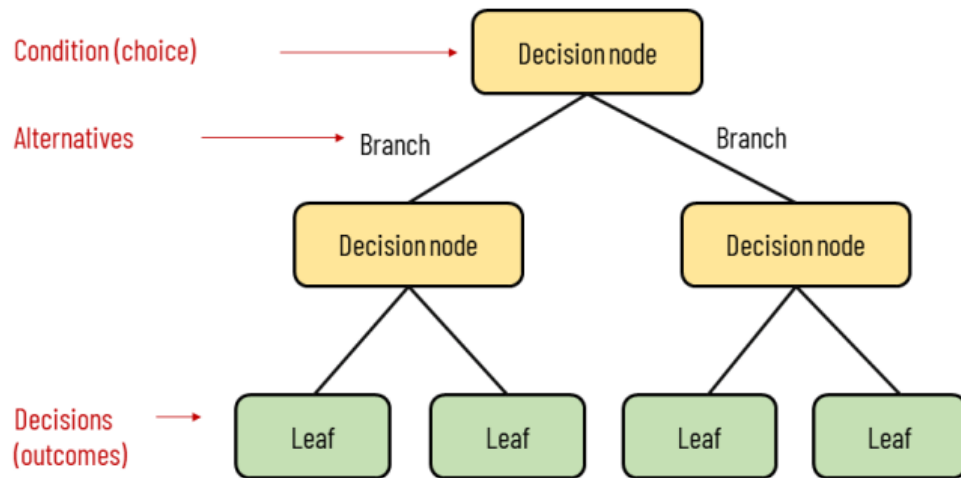


***Inatel***

Felipe Augusto Pereira de Figueiredo  
felipe.figueiredo@inatel.br

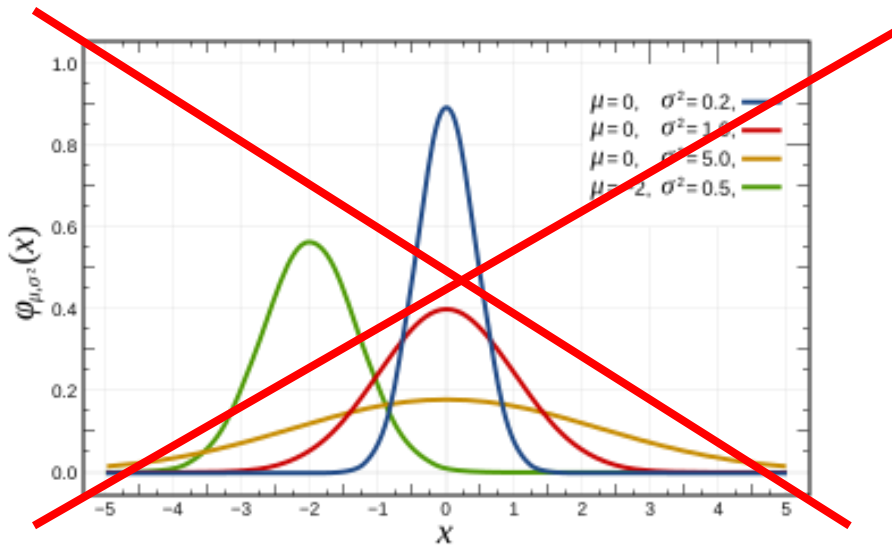
# Árvores de decisão

## Elements of a decision tree



- Assim como o k-NN, uma **árvore de decisão** (do inglês, **decision trees**), é um algoritmo de **aprendizado supervisionado não-paramétrico** e **não-linear** que pode ser utilizado tanto para **classificação** quanto para **regressão**.
- Podemos fazer uma analogia com **estruturas de controle de fluxo**.

# Árvores de decisão: modelo não-paramétrico

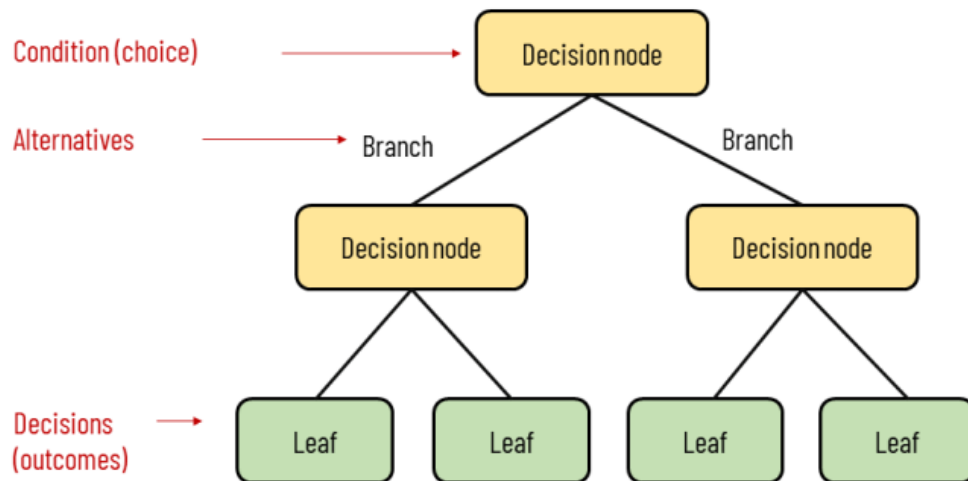


$$h(x) = a_0 + a_1 x$$

- Elas *não fazem suposições sobre a distribuição* dos dados ou *sobre a forma da função* que estão tentando aprender.
- Consequentemente, *não definem um número fixo de parâmetros* como em *modelos lineares*, que têm um *número fixo de pesos*.
- Em vez de definir um número fixo de parâmetros, as *árvores de decisão ajustam sua complexidade à medida que crescem*, dependendo da estrutura dos dados.

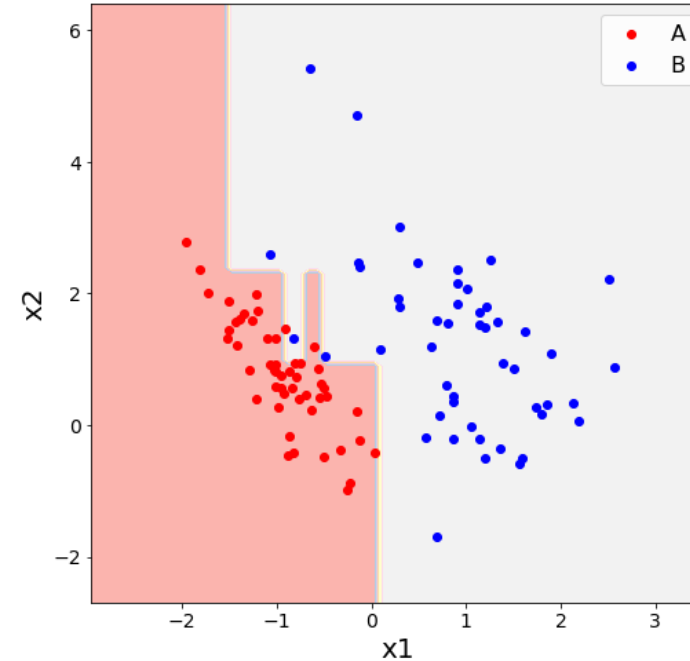
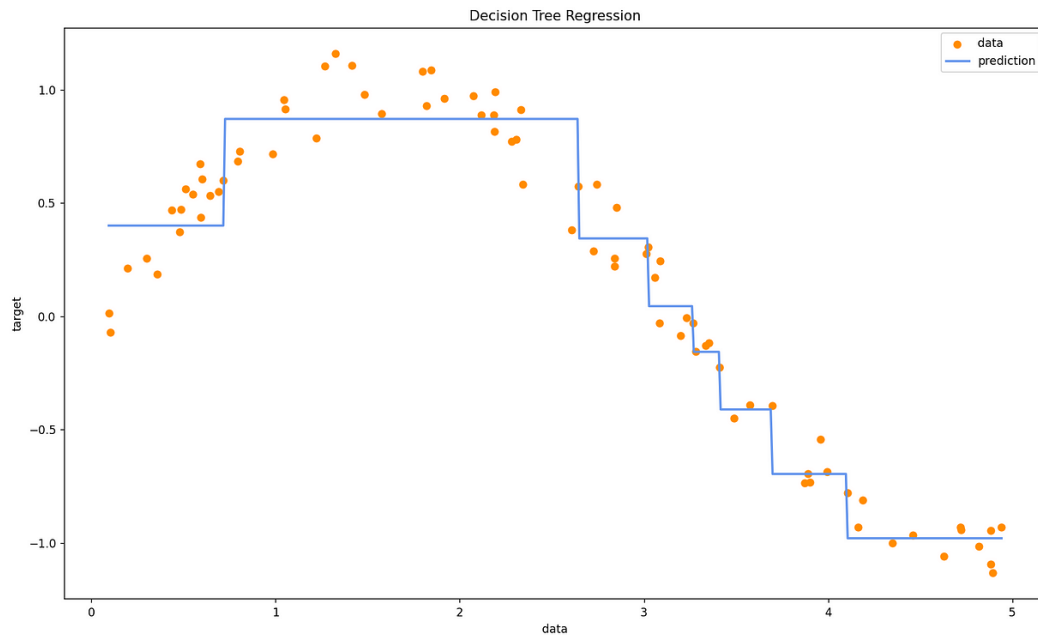
# Árvores de decisão: modelo não-paramétrico

## Elements of a decision tree



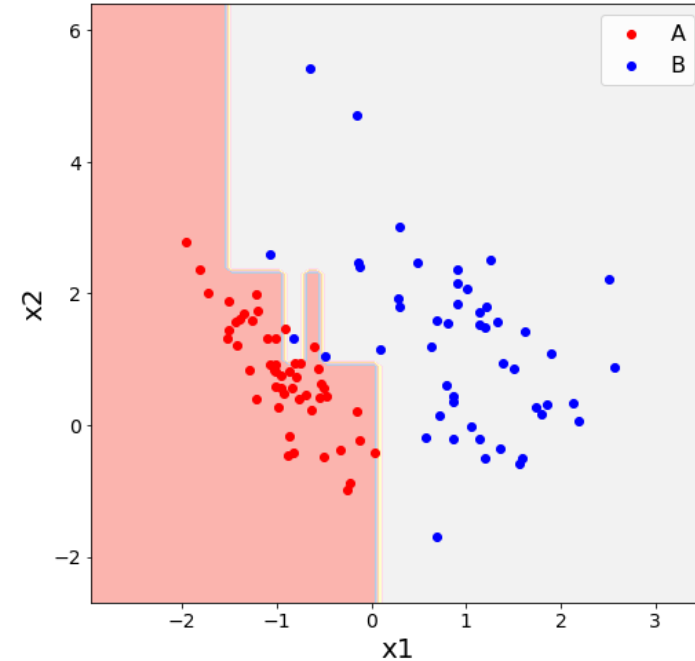
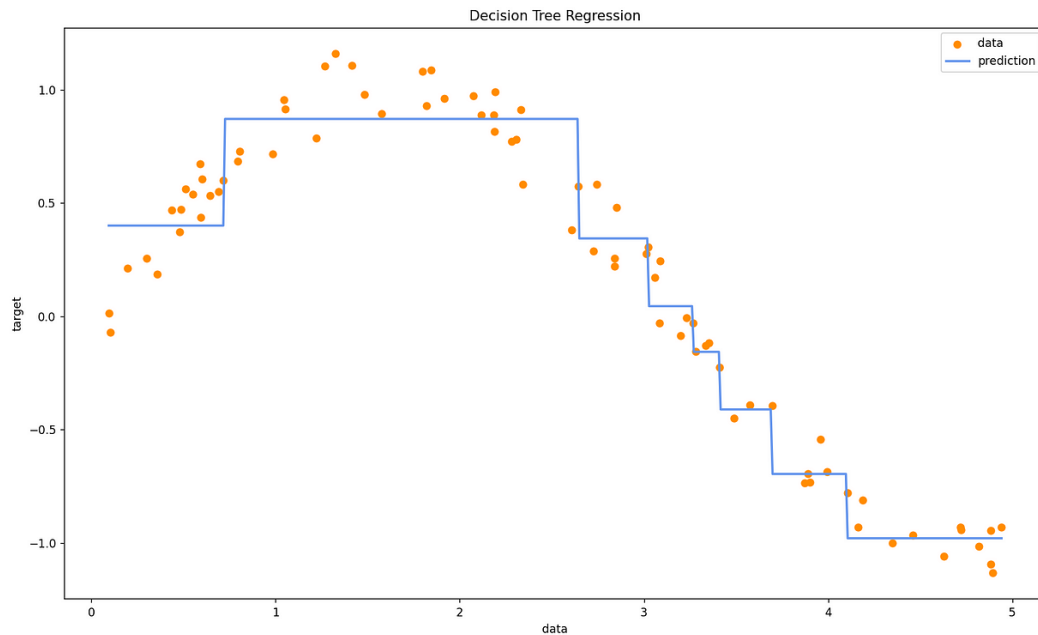
- Ou seja, os **parâmetros são criados durante o treinamento para explicar os dados.**
- O **modelo é construído apenas com base nos dados observados**
- O **tamanho da árvore**, i.e., o número de parâmetros, **tende a crescer com a base de dados.**
- **Não-linear**: funções hipótese e discriminante são **funções não-lineares**, i.e., os dados não são aproximados ou separados por um **hiperplano**.

# Árvores de decisão: modelos não-lineares



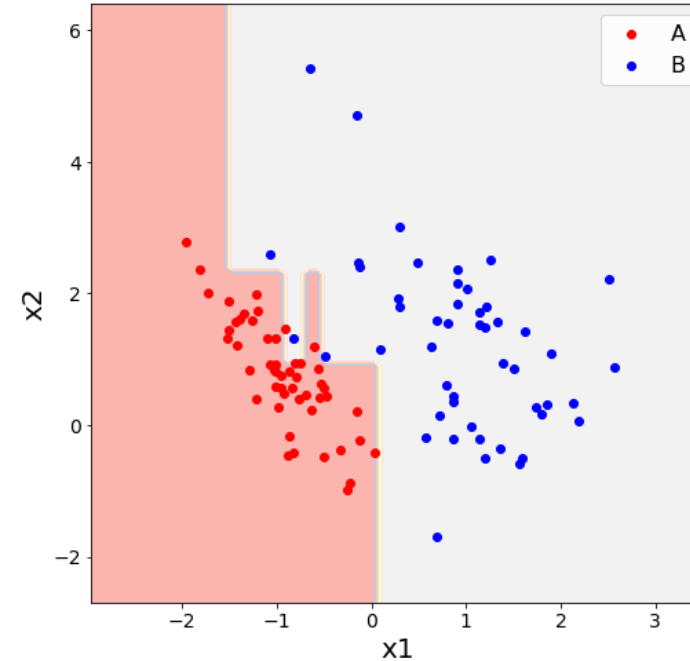
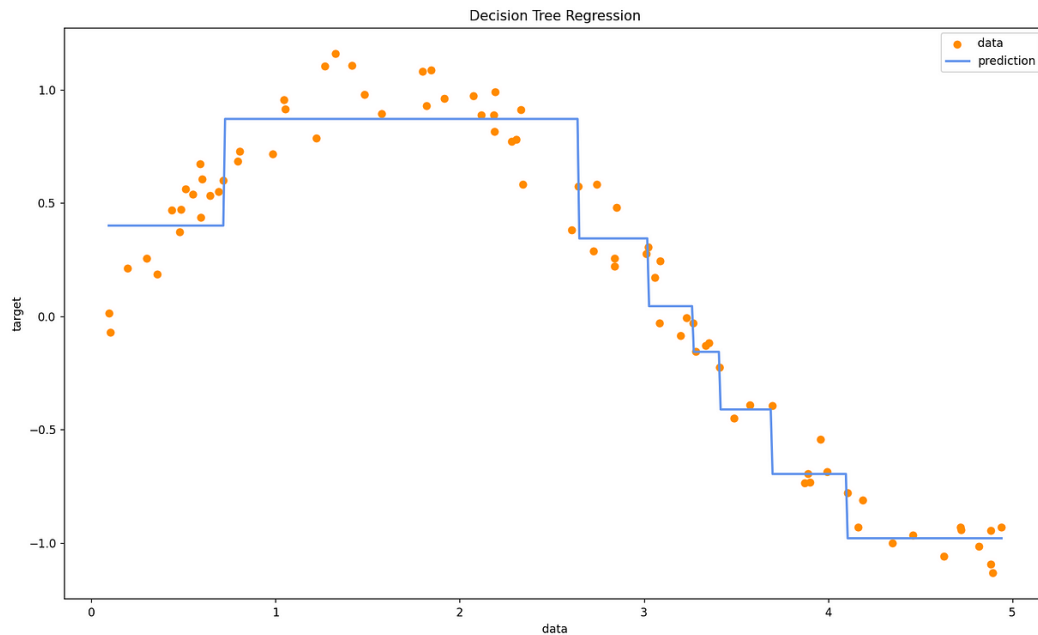
- As funções hipótese e discriminante são ***funções não-lineares***, i.e., os dados ***são aproximados ou separados por conjuntos de hiperplanos***.

# Árvores de decisão: modelos não-lineares



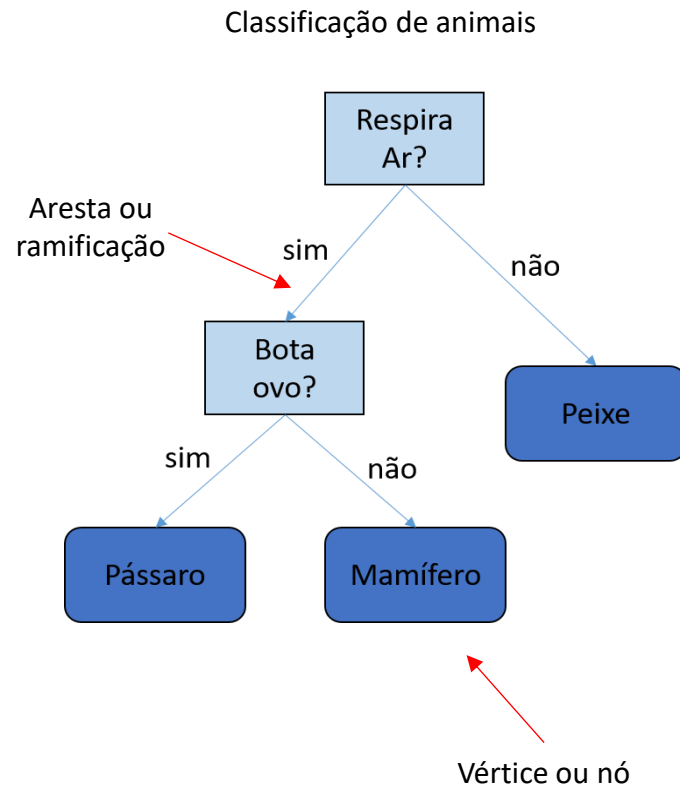
- As árvores criam uma estrutura (i.e., função) *não-linear* e *discreta*, *baseada nas regras* de decisão aplicadas em cada nó da árvore.
- A função é composta por trechos perpendiculares aos eixos dos atributos.

# Árvores de decisão: modelos não-lineares



- Quanto *mais profunda* (mais divisões) for a árvore, *mais complexa/não-linear* será a função.
- Isso pode levar a *funções que se ajustam muito bem aos dados de treinamento*, mas também corre o risco de **overfitting**.

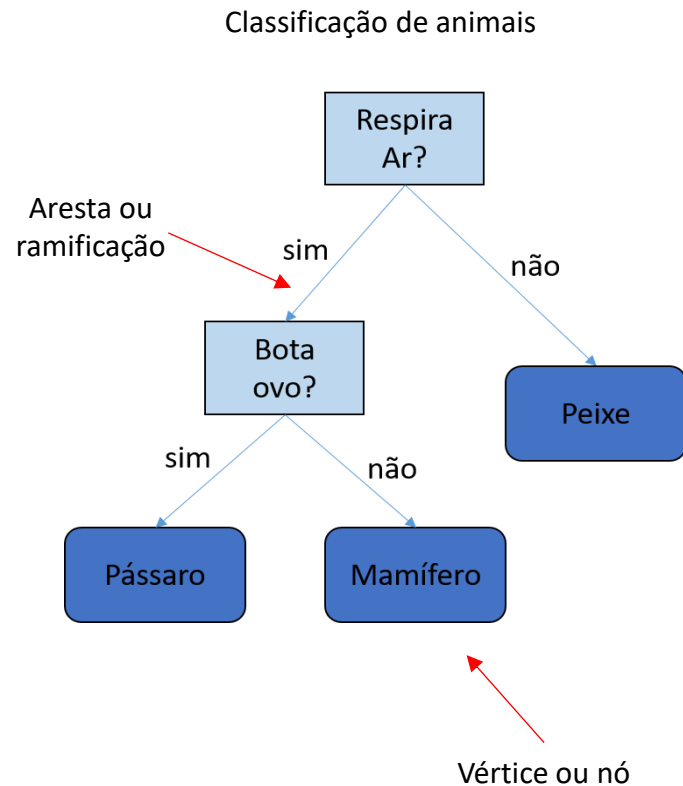
# Árvores de decisão



- O objetivo é criar um modelo que realize previsões aprendendo **regras de decisão inferidas** a partir dos **atributos do conjunto de treinamento**.
- As **árvores de decisão** são os componentes fundamentais das **florestas aleatórias** (do inglês, **random forests**) que estão entre os **algoritmos mais poderosos de aprendizado de máquina** disponíveis atualmente.

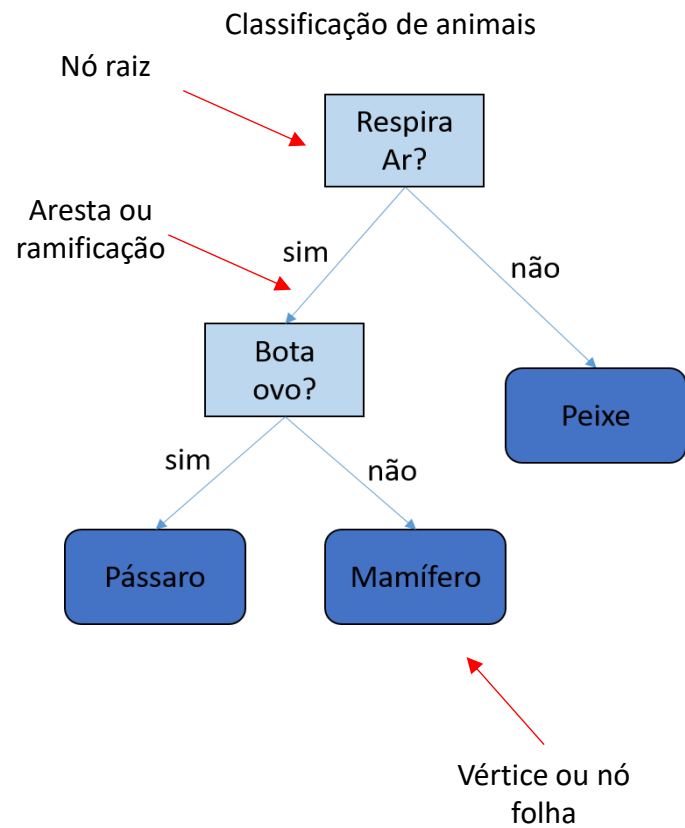


# Árvores de decisão



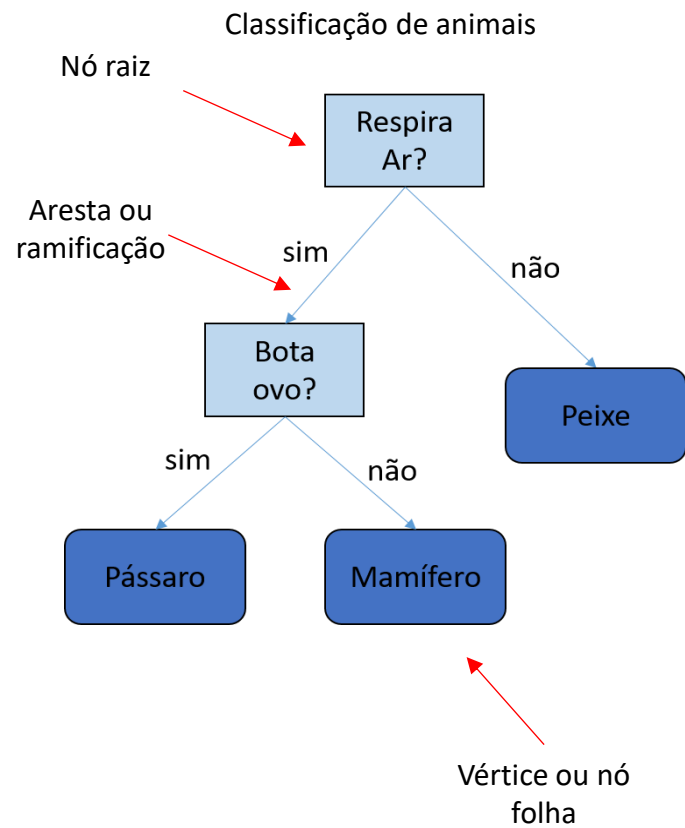
- Formalmente, uma árvore é um **grafo não-direcionado** no qual dois **vértices** quaisquer se conectam por um único caminho (ou seja, um **grafo acíclico não-direcionado**) [Wikipedia, 2019].
  - Um ciclo é um caminho em que o primeiro e o último vértice coincidem.
- Por exemplo, o vértice “Pássaro” só se conecta com o vértice “Mamífero” através do vértice “Bota ovo?” e não existe um caminho que termine no mesmo vértice de início.

# Árvores de decisão



- A árvore possui um **nó raiz**, do qual **parte o processo de decisão**.
- No processo de decisão, **valores distintos dos atributos geram arestas** (i.e., **ramificações**) e, quando se chega a um **nó folha**, ocorre uma **atribuição de uma classe ou valor**.

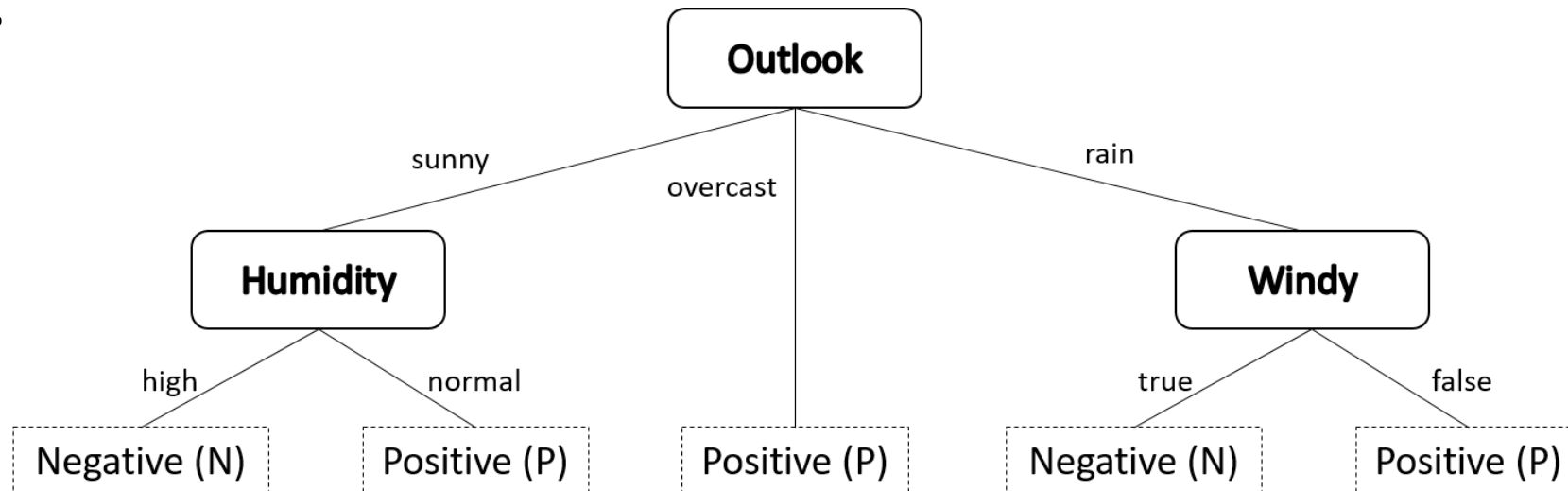
# Árvores de decisão



- **Árvores de decisão** são modelos de **caixa branca**, ou seja, é possível entender e **explicar facilmente como o modelo realiza a classificação ou regressão** baseando-se nos atributos.
- Elas são o oposto dos modelos de **caixa preta**, onde os resultados são difíceis de interpretar e **não é fácil entender como os diferentes atributos interagem entre si para gerar a saída** (e.g., redes neurais artificiais).

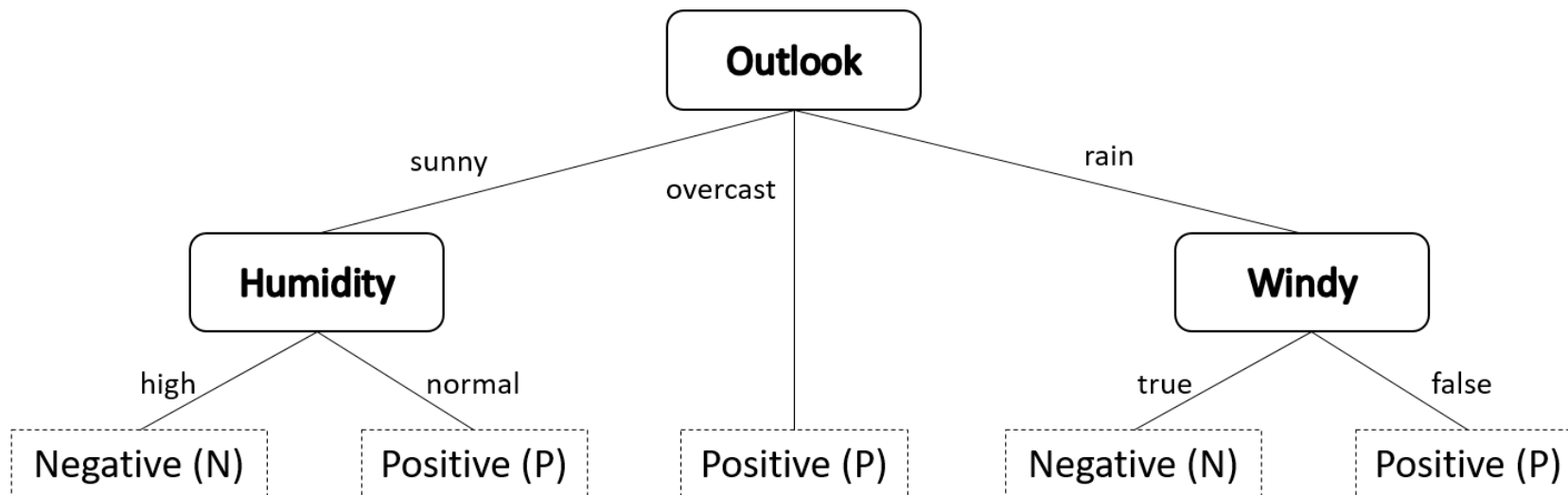
# Árvores de decisão

- A figura abaixo mostra uma árvore de decisão criada a partir de informações se *jogadores jogarão tênis ou não dependendo dos valores dos atributos Clima, Humidade e Vento*.
- Cada *nó da árvore atua como um caso de teste* para algum atributo, e cada extremidade, ou seja, uma terminação ou folha, corresponde a uma classe.



# Árvores de decisão

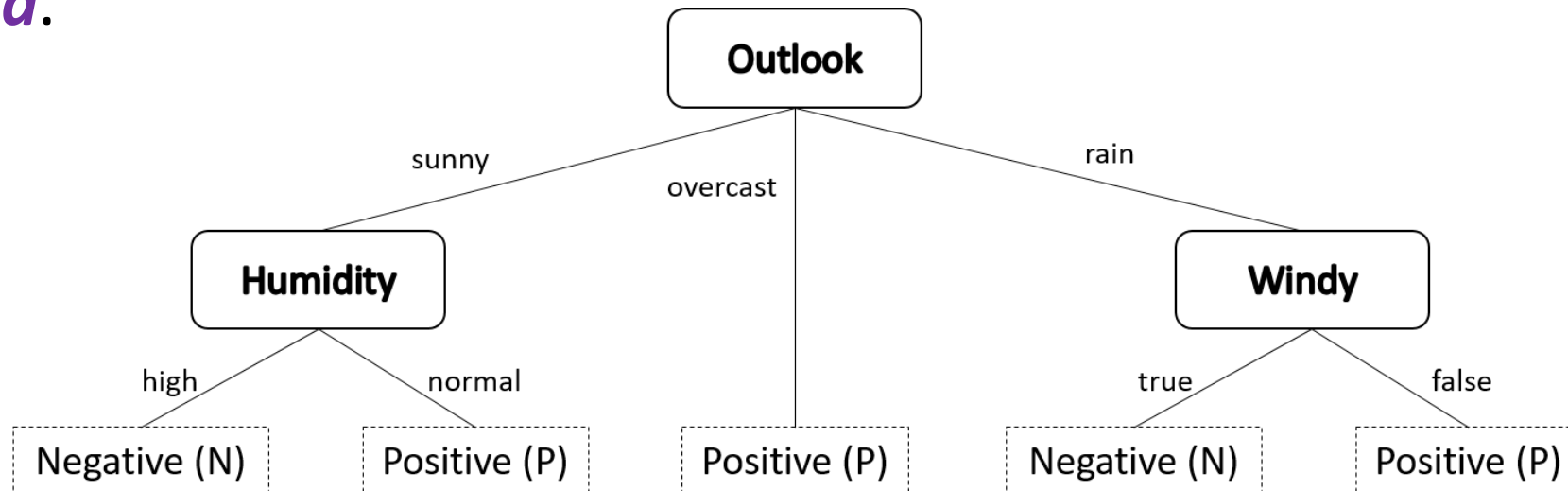
- Na figura, cada atributo leva a uma ramificação até que se atinja um nó folha, onde decide-se sobre jogar ou não.
- O *uso da árvore para classificar padrões (e regressão) é direto*, mas é preciso responder uma questão crucial.



***Como induzimos uma árvore de  
decisão a partir dos dados de  
treinamento?***

# O processo de indução de uma árvore de decisão

- Uma primeira abordagem para induzir uma árvore poderia ser **construir, de maneira exaustiva**, todas as árvores **capazes de resolver o problema** (classificação ou regressão) e **selecionar a mais simples** (i.e., menos complexa) utilizando a **regra da Navalha de Occam**.
- Entretanto, essa **abordagem pode ser computacionalmente muito custosa**.



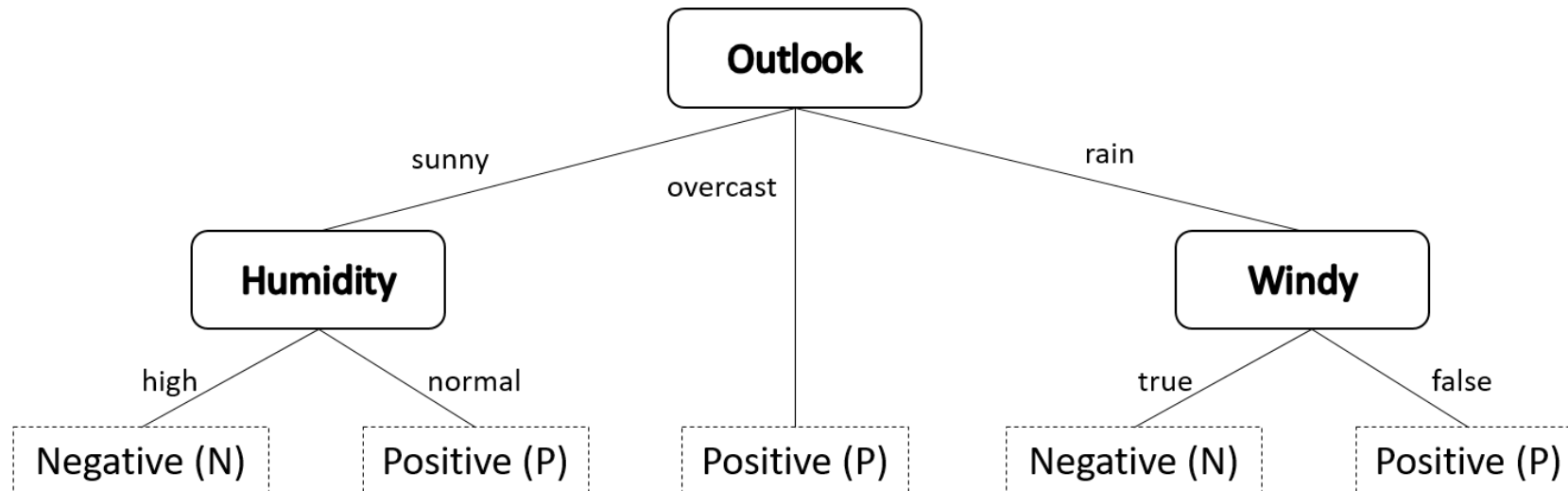
# O processo de indução de uma árvore de decisão

- Encontrar uma árvore de decisão ótima é um problema de **tempo polinomial não-determinístico completo** (NP-completo).
- São problemas:
  - Cujas soluções podem ser **verificadas** em tempo polinomial.
    - Um algoritmo roda em tempo polinomial se o tempo necessário para resolver um problema de tamanho  $n$  é limitado por uma função polinomial de  $n$ .
    - Por exemplo, a inversa de uma matriz  $n \times n$  pode ser calculada em  $n^3$ .
  - Não se sabe se existe um algoritmo eficiente (polinomial) para **resolver** o problema.
- Exemplos incluem problemas clássicos como a Satisfatibilidade Booleana, Caixeiro Viajante e o Caminho Hamiltoniano, etc.



# O processo de indução de uma árvore de decisão

- Como a indução de árvores são problemas NP-completos, em geral, usa-se **abordagens heurísticas** para induzir árvores de decisão.
  - **Heurística**: estratégia **prática** para encontrar uma **solução rápida e aproximada**.
- Dentre os mais conhecidos podemos citar: **ID3**, C4.5, C5.0 e **CART (scikit-learn)**.



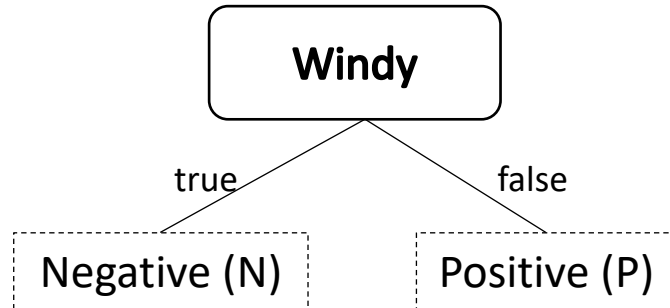
# O processo de indução de uma árvore de decisão

- Um dos métodos *heurísticos* de indução de árvores de decisão mais utilizados é o **ID3** (do inglês, *Iterative Dichotomiser 3*).
- A *dicotomização iterativa* *divide recursivamente* o *conjunto de dados* em *duas ou mais partes*, baseando-se em um *atributo de maior relevância* para separar os exemplos.
- O *método ID3* *não garante* a obtenção da *menor árvore possível*, mas busca obter *árvores razoáveis num intervalo de tempo relativamente curto*.

# O processo de indução de uma árvore de decisão

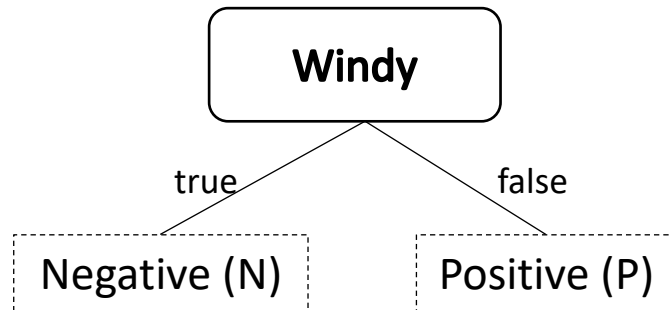
- O ID3 usa uma **estratégia top-down gananciosa**.
- Ou seja, a árvore é **criada de cima para baixo** e um **nó é escolhido** selecionando-se o **atributo que melhor divide o conjunto de dados naquele momento**.
- Mas como se decide qual é o melhor atributo?
- Qual é a métrica usada para essa escolha?
- O **método ID3** se baseia na **teoria da informação** para selecionar o atributo de cada nó da árvore.

# O processo de indução de uma árvore de decisão



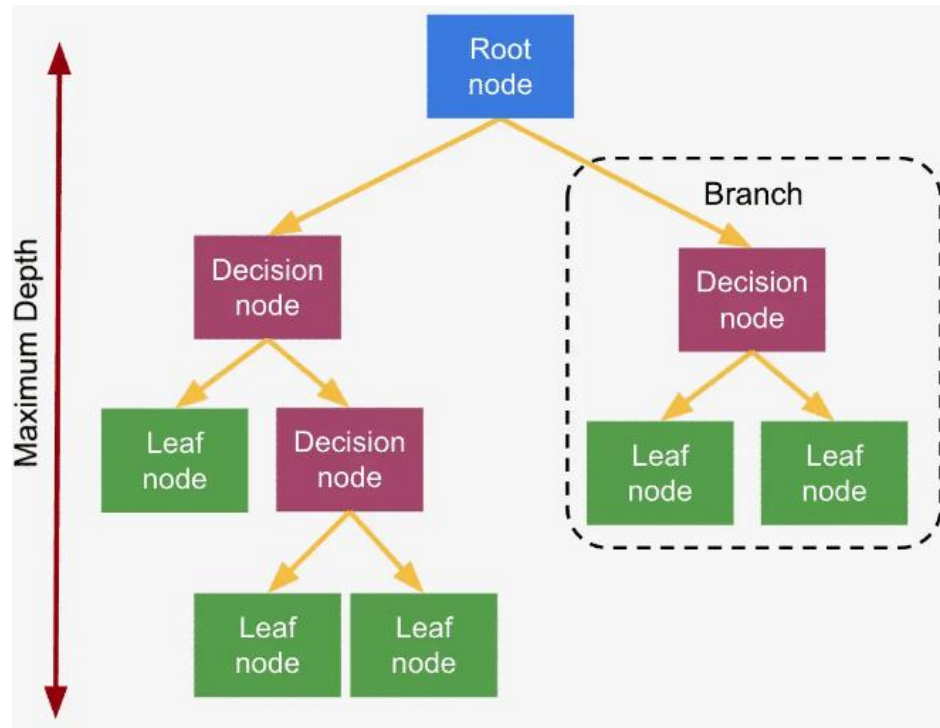
- A ideia do **método ID3** é escolher como **nó** o **atributo que for o mais longe possível em dividir os exemplos**.
- Um **atributo perfeito** divide perfeitamente todos os exemplos em grupos (ou classes).
- Ou seja, é o atributo que consegue **explicar perfeitamente o conjunto de dados**.
- Por exemplo, o atributo **Windy** divide as exemplos perfeitamente em duas classes, i.e., não precisamos de mais ramificações.

# O processo de indução de uma árvore de decisão



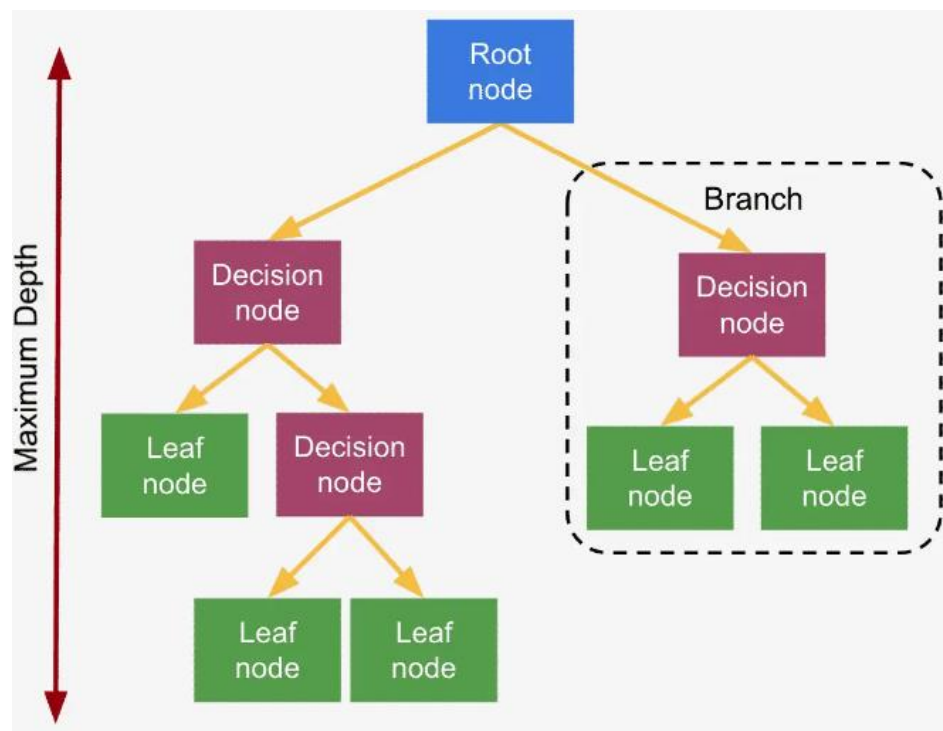
- Então, tudo o que precisamos é uma medida formal de **atributo razoavelmente bom** ou **realmente inútil**.
- O **método ID3** utiliza a noção de **ganho de informação**, o qual é **definido em termos da entropia**, que é uma quantidade fundamental em **teoria da informação**.
- A seguir, veremos alguns conceitos úteis para inferirmos árvores de decisão com o **método ID3**.

# Ganho de informação



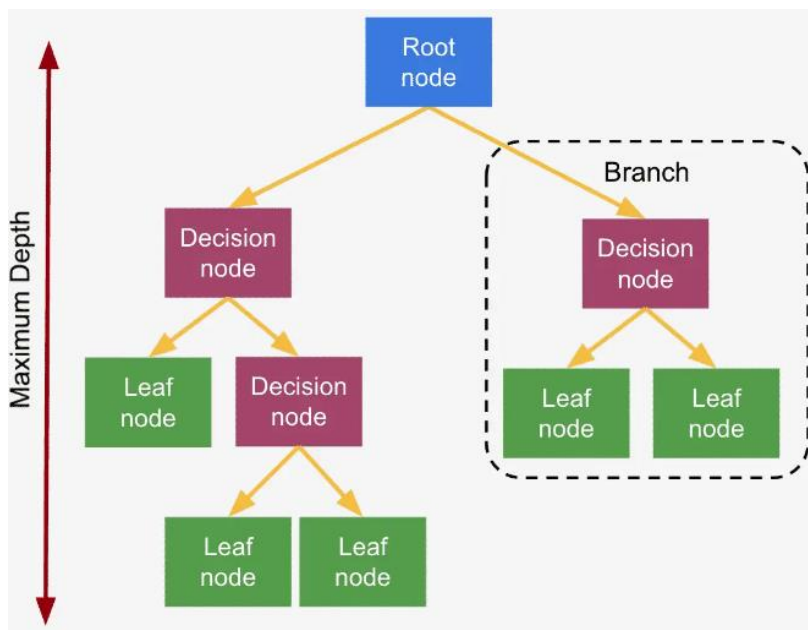
- É uma *propriedade estatística* que *mede o quão bem um determinado atributo separa os exemplos* de treinamento de acordo com suas classes.
- O *ganho de informação* significa uma *redução da aleatoriedade*, ou seja, da *entropia*.
- Portanto, construir uma *árvore de decisão* significa *encontrar*, a cada momento, um *atributo* que *reduza a aleatoriedade*.

# Entropia



- É uma medida da **quantidade de incerteza ou aleatoriedade** de uma variável aleatória (e.g., um conjunto de dados).
- Portanto, **um ganho de informação** corresponde a uma **redução na entropia**.
- Quando **ganhamos informação** sobre o conjunto, a **incerteza diminui**.
- No caso do ID3, o ganho é o quanto conseguimos **explicar do conjunto com aquele atributo**.

# Entropia



- O **ganho de informação** nos dá a **diferença entre a entropia antes e após a divisão do conjunto de dados** com base nos valores de um determinado atributo.
- Assim, podemos escrever que o **ganho de informação** é dado por

$$Gain = entropiaAntesDaDivisão - entropiaApósADivisão$$

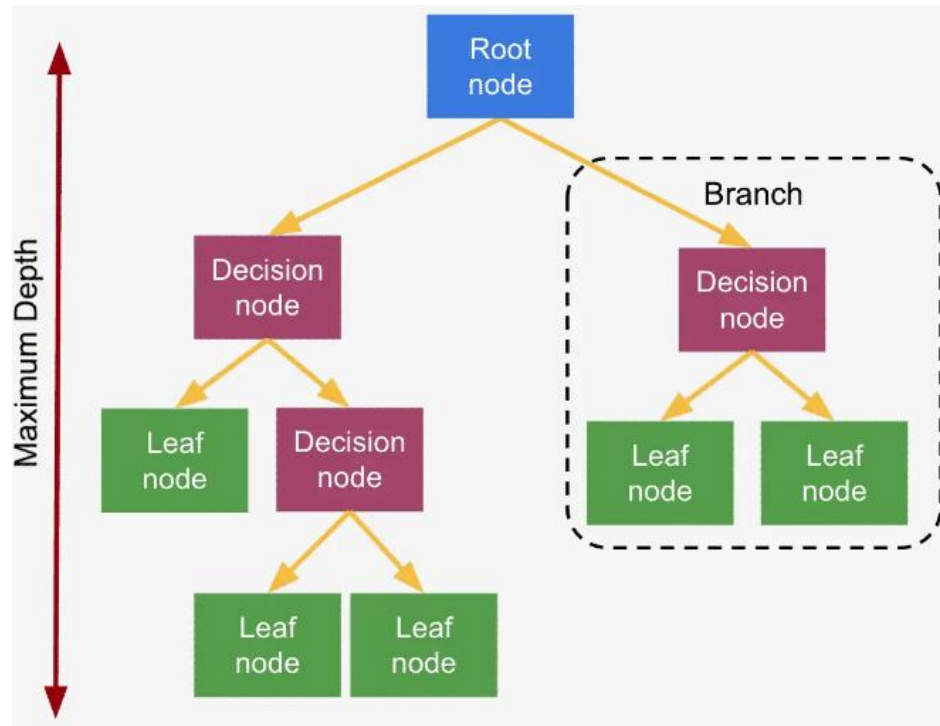
ou

$$Gain = H(y) - H(y|x)$$

- Vejamos alguns exemplos.

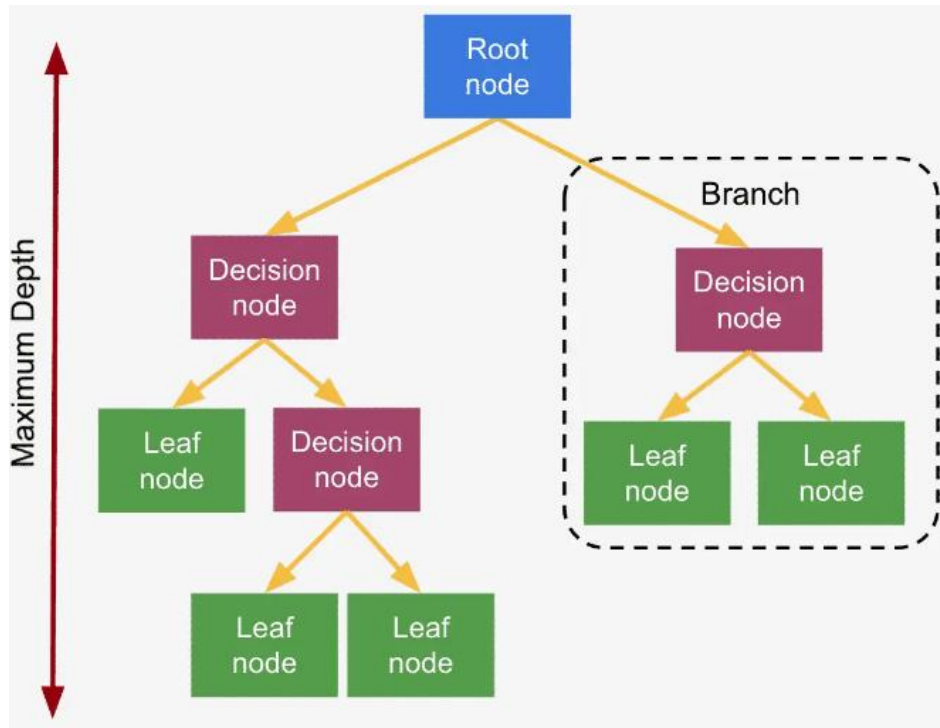


# Exemplos de entropia



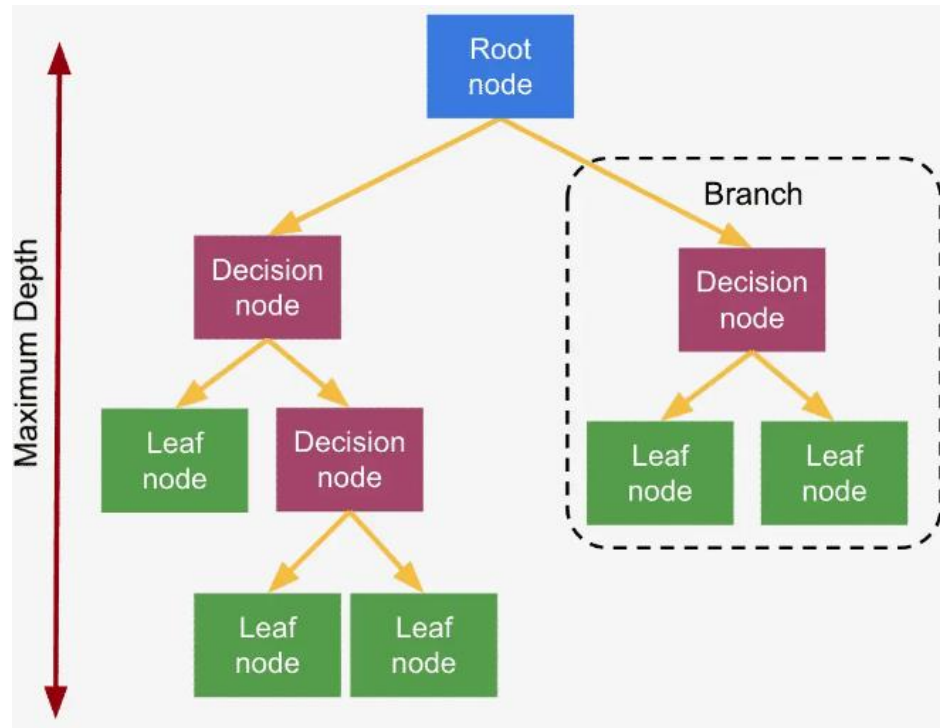
- Uma variável com apenas um único valor (e.g., o arremesso de uma moeda com ***cara*** em ambos os lados) não tem nenhuma ***incerteza*** associada
- Portanto, sua ***entropia é igual a zero***.
- Isso significa que ***não se ganha nenhuma informação nova ao se observar o valor***.

# Exemplos de entropia



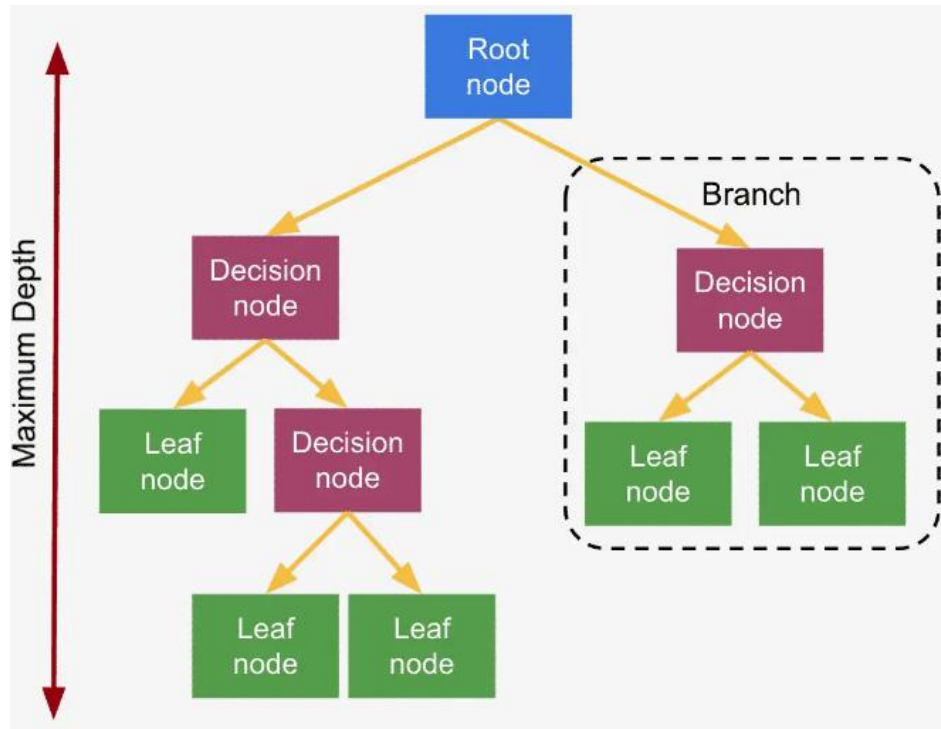
- Por outro lado, o resultado de se arremessar uma **moeda honesta** é igualmente provável de resultar em **cara** ou **coroa**, os quais são associados aos valores 0 ou 1, respectivamente.
- Neste caso, esta variável tem **1 bit de entropia**, significando que se necessita de 1 bit para representar os 2 possíveis resultados.

# Exemplos de entropia



- Dessa forma, a variável aleatória que representa o resultado de se rolar um ***dado honesto de 4 lados, tem 2 bits de entropia***, pois necessita-se de 2 bits para se representar os 4 possíveis valores.
- Agora imaginem um ***moeda desonesta*** que tenha uma probabilidade de resultar em ***cara*** em 99% dos arremessos.
- Nesse caso, a ***entropia*** deve ser um ***valor muito próximo de zero***, pois a incerteza do resultado é muito baixa.

# Cálculo da entropia



- Assim, a **entropia** de uma **variável aleatória discreta**,  $V$ , com valores  $v_i$ , onde cada um dos valores tem probabilidade  $P(v_i)$ , é definida como

$$H(V) = - \sum_i P(v_i) \log_2(P(v_i)).$$

- OBS.:** a entropia é sempre **maior ou igual a 0**.
- Na sequência, vamos entender como o método ID3 funciona com um problema de classificação binário.

# Indução de uma árvore de decisão

- Retornando ao problema da indução de **árvores de decisão** nós temos que se um conjunto de treinamento,  $E$ , contém  $p$  exemplos pertencentes à classe positiva ( $P$ ) e  $n$  exemplos pertencentes à classe negativa ( $N$ ), então a **entropia do objetivo** (i.e., o rótulo) para todo o conjunto de treinamento é dada por

$$\begin{aligned} H(\text{Goal}) &= B\left(\frac{p}{p+n}\right) \\ &= -\left[\frac{p}{p+n} \log_2\left(\frac{p}{p+n}\right) + \left(1 - \frac{p}{p+n}\right) \log_2\left(1 - \frac{p}{p+n}\right)\right], \end{aligned}$$

onde  $B(q)$  é a **entropia de uma variável booleana que é verdadeira com probabilidade igual a  $q$** , ou seja

$$B(q) = -(q \log_2(q) + (1 - q) \log_2(1 - q)), \text{ onde } q = \frac{p}{p+n}.$$

# Indução de uma árvore de decisão

- Um teste com um único atributo,  $x_k$ , pode nos dar **toda ou parte** da **entropia do objetivo**,  $H(\text{Goal})$ .
- Nós podemos medir exatamente o **quanto cada atributo contribui para a entropia do objetivo** através do cálculo da **entropia restante após o teste do atributo**.
- Um **atributo**  $x_k$  **com  $d$  valores distintos divide o conjunto de treinamento  $E$  em  $d$  subconjuntos:  $E_1, \dots, E_d$** .
- Cada subconjunto  $E_i, i \in \{1, \dots, d\}$ , possui  $p_i, i \in \{1, \dots, d\}$ , exemplos da classe positiva,  $P$ , e  $n_i, i \in \{1, \dots, d\}$ , exemplos da classe negativa,  $N$ .

# Indução de uma árvore de decisão

- Um exemplo escolhido aleatoriamente do conjunto de treinamento tem probabilidade igual a  $(p_i + n_i)/(p + n)$  de pertencer ao  $i$ -ésimo subconjunto,  $E_i$ , do atributo  $x_k$ .

- Assim, a **entropia restante** após o teste do atributo  $x_k$  é dada por

$$Remainder(x_k) = H(y|x_k) = \sum_{i=1}^d \frac{p_i + n_i}{p + n} B\left(\frac{p_i}{p_i + n_i}\right).$$

- O **ganho de informação** com o atributo  $x_k$  é a **redução na entropia do objetivo**,  $H(y)$ , que é dado por

$$Gain(x_k) = B\left(\frac{p}{p + n}\right) - Remainder(x_k) \geq 0.$$

# Indução de uma árvore de decisão

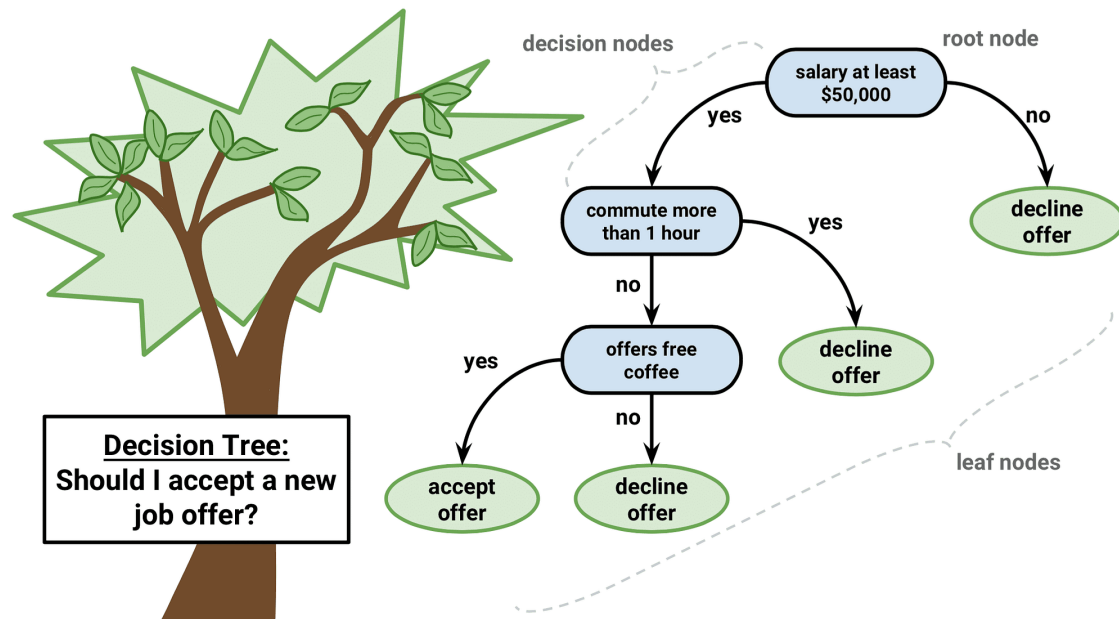
- $Gain(x_k)$  é maximizado o quão menor for  $Remainder(x_k)$ .
- $Gain(x_k)$  atinge seu valor máximo se  $Remainder(x_k) = 0$ , portanto

$$Gain(x_k) = B \left( \frac{p}{p+n} \right) \geq 0.$$

- Ou seja, se  $x_k$  **explicar totalmente o conjunto de dados**, a **incerteza** quanto àquele conjunto **se torna igual a 0**.



# Método ID3

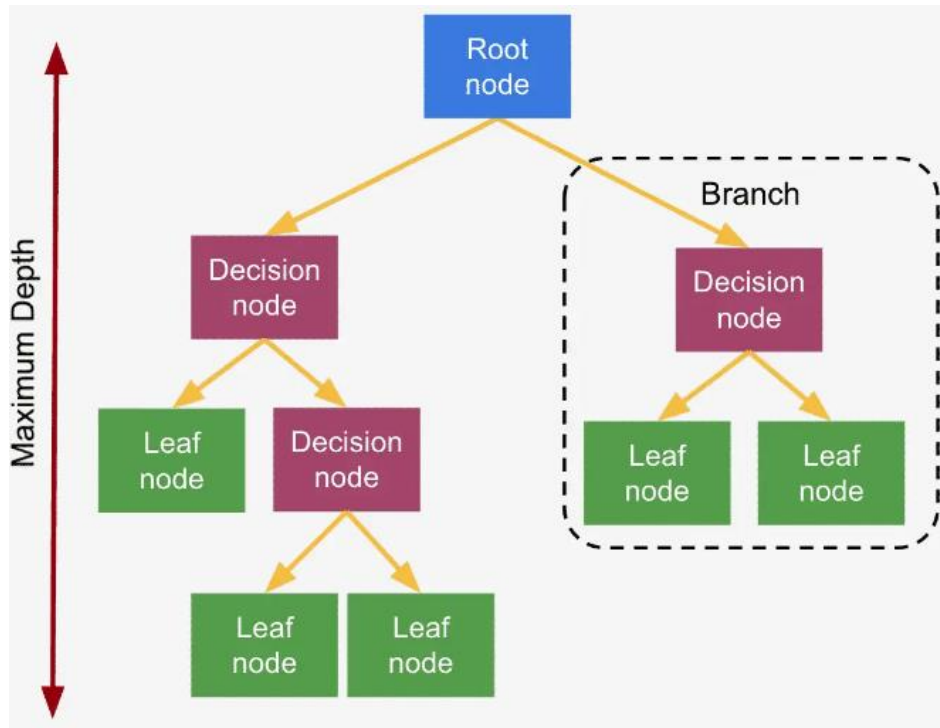


- A ideia por trás do **método ID3** é encontrar o **atributo  $x_k$  que maximiza o ganho de informação** e, então, usar o procedimento **recursivamente com os demais atributos**.
- Ou seja, escolhe-se o atributo que gera a primeira ramificação e, então, se repete o processo usando os demais atributos para construir o restante da árvore.

# Sequência de passos para inferência com o método ID3

- a) Cálculo da **entropia do objetivo** para o (sub)conjunto de treinamento **corrente**.
  - **OBS.:** um novo (sub)conjunto é criado a cada iteração de acordo com os valores do nó anterior (atributo).
- b) Cálculo do **ganho de informação** com cada atributo,  $x_k, \forall k$ , do conjunto de treinamento,  $E$ .
- c) Criação de um nó com o atributo  $x_k$  que maximiza o **ganho de informação**.
- d) **Particionamento do (sub)conjunto em subconjuntos** usando os valores de  $x_k$  que maximizou o **ganho de informação**.
- e) Repetir os itens a) até d) em **subconjuntos de treinamento** usando os atributos restantes. Essa sequência **continua até que a árvore classifique perfeitamente todos os exemplos de treinamento** ou **até que todos os atributos tenham sido utilizados**.

# Sequência de passos do método ID3

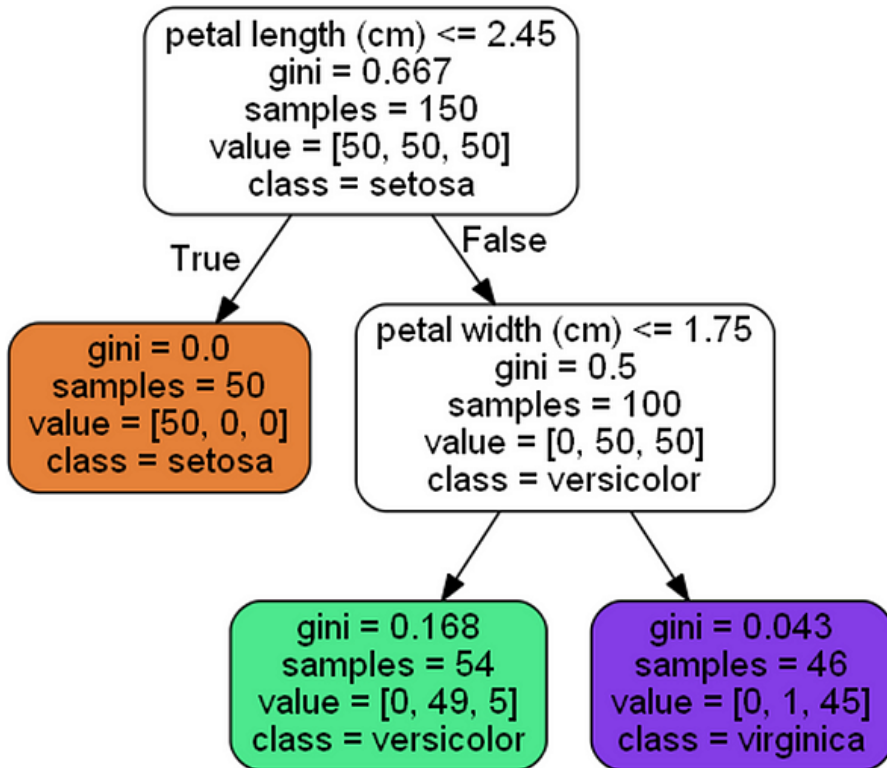


- O **ID3** segue a regra: um **ramo** com uma **entropia igual a zero** é uma **folha** e um **ramo** com **entropia maior do que zero precisa de partição adicional**.
- O **método ID3** será exemplificado através do exemplo apresentado em breve.

# Características do método ID3

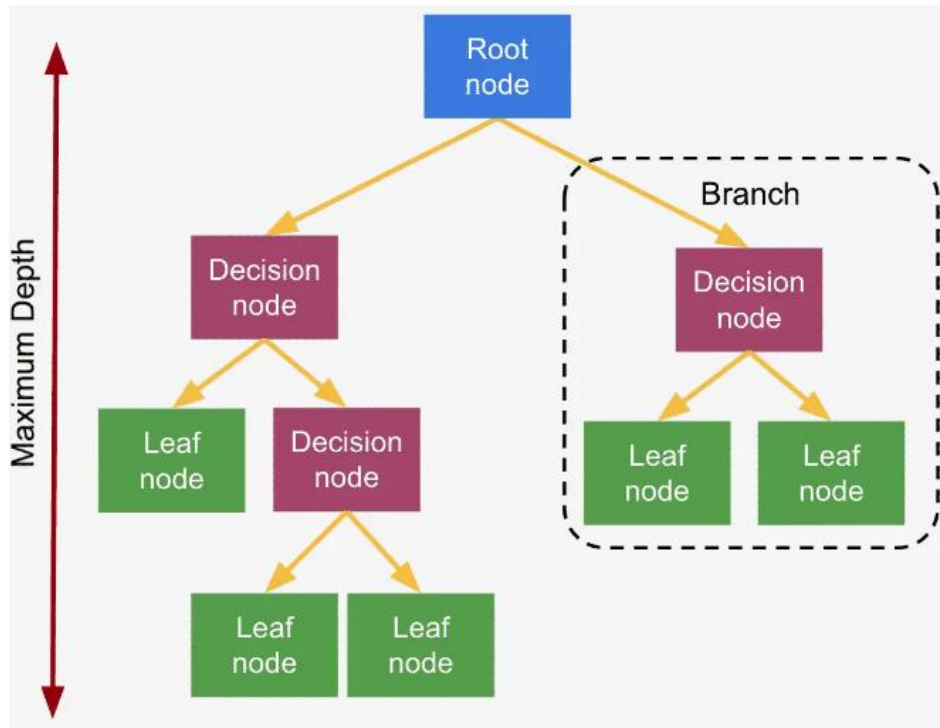
- Por usar uma abordagem baseada em uma *heurística gananciosa*, o ID3 *não garante uma solução ideal*.
- Pode sobreajustar aos dados de treinamento
  - Para evitar o sobreajuste, árvores de decisão menores devem ser preferidas ao invés das maiores.
- O método ID3 geralmente produz árvores pequenas, *mas nem sempre produz a menor árvore possível*.
- É mais *difícil de usar com dados contínuos*.
- Se os valores de um atributo forem contínuos, podem haver muitos pontos para se dividir os dados com esse atributo, e a busca pelo melhor valor a ser dividido pode se tornar demorada.

# Observações



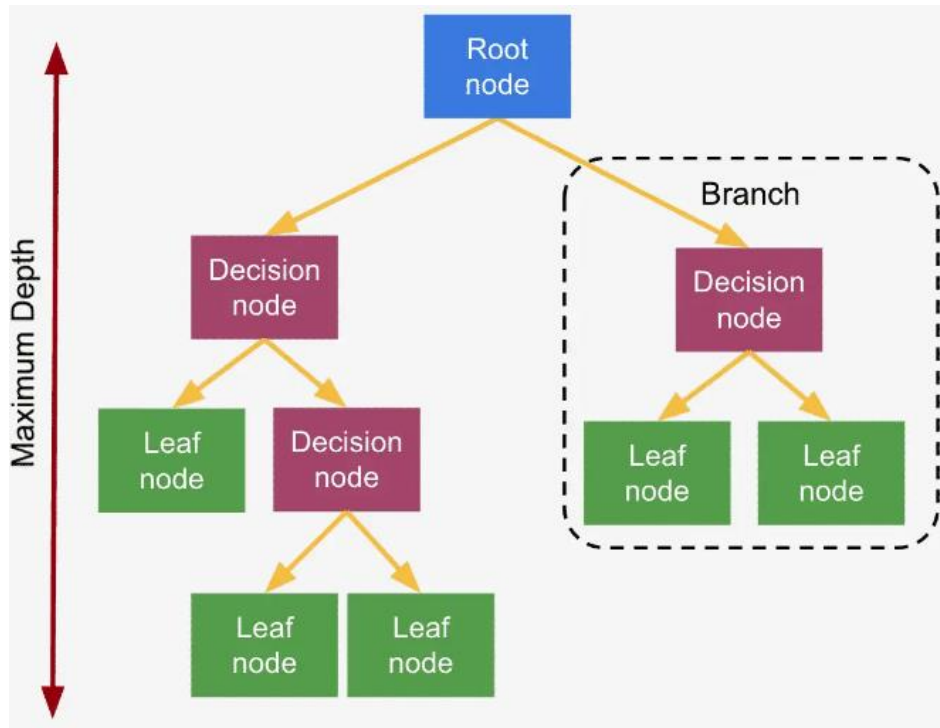
- Além do ***ganho de informação***, existem outras métricas que podem ser usadas para definir as partições.
- Uma possibilidade é usar ***métricas de distância ou divergência***, como o ***índice de Gini***.
- Outro ponto importante é, se for o caso, deve-se buscar metodologias para se lidar com atributos faltantes.
- Existem várias soluções, como por exemplo:
  - Remover o exemplo.
  - Atribuir ao atributo a média de seus valores.

# Observações



- O conjunto de treinamento é a base para definirmos a ***árvore de decisão***.
- Um conjunto que contenha inconsistências, como, por exemplo, ***dois exemplos com os mesmos valores de atributos, porém de classes diferentes***, precisará ser reconsiderado, pois os atributos podem não ser suficientes, por exemplo, ***precisando de mais atributos***.

# Observações



- Um problema muito comum das **árvores de decisão**, especialmente quando se tem um número muito grande de atributos, é o **sobreajuste (alta variância)**.
- Existem duas formas para se minimizar este problema
  - Podar as árvores de decisão (**tree pruning**).
  - Ou utilizar **florestas aleatórias**.

# Exemplo de inferência de árvore de decisão com o método ID3

- Vamos construir uma árvore de decisão para ***prever se jogadores irão ou não praticar um determinado esporte*** baseado em algumas ***condições meteorológicas***.
- Vamos considerar um conjunto de dados da forma  $(\mathbf{x}_i, d_i)$ , onde  $\mathbf{x}_i$  é um vetor de atributos e  $d_i$  é um rótulo correspondente ao vetor de atributos.
- Nesse conjunto, cada entrada diz respeito à condição meteorológica de um dia e se os jogadores jogaram ou não.



# Exemplo de inferência de árvore de decisão com o método ID3

- Existem **quatro** atributos categóricos:
  - **Tempo**: {ensolarado, nublado, chuvoso}
  - **Temperatura**: {frio, agradável, quente}
  - **Umidade**: {alta, normal}
  - **Vento**: {presente, ausente}
- Os rótulos são apenas dois: **positivo** ( $P$ ), ou seja, jogar, e **negativo** ( $N$ ), ou seja, não jogar, denotando um problema binário.
- Um exemplo de condição meteorológica de um determinado dia poderia ser descrito por: {nublado, frio, normal, ausente}.

# Exemplo de inferência de árvore de decisão com o método ID3

- O conjunto de treinamento do exemplo é dado pela tabela abaixo.

Day	Attributes				Class (y)
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

# Exemplo de inferência de árvore de decisão com o método ID3

- A **entropia do objetivo**, i.e.,  $y$ , para todo o conjunto de treinamento é

$$H(y) = - \left[ \frac{9}{14} \log_2 \left( \frac{9}{14} \right) + \left( 1 - \frac{9}{14} \right) \log_2 \left( 1 - \frac{9}{14} \right) \right] = 0.9403.$$

- Encontrando o nó raiz: o **ganho de informação** de cada atributo é calculado como

		Jogar?		
		P	N	
Outlook	sunny	2	3	5
	overcast	4	0	4
	rain	3	2	5
				14

		Jogar?		
		P	N	
Temperature	hot	2	2	4
	mild	4	2	6
	cool	3	1	4
				14

		Jogar?		
		P	N	
Humidity	high	3	4	7
	normal	6	1	7
				14

		Jogar?		
		P	N	
Windy	true	3	3	6
	false	6	2	8
				14

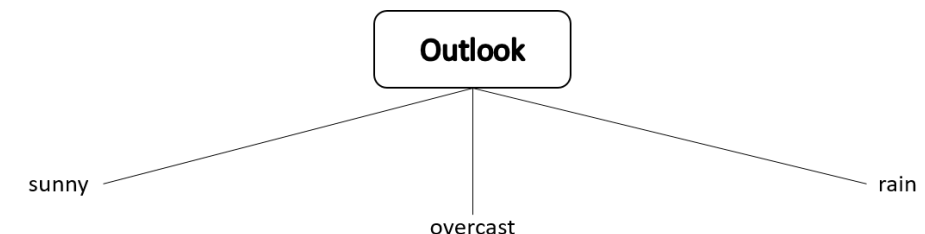
$$Gain(\text{outlook}) = 0.9403 - \left[ \frac{5}{14} B\left(\frac{2}{5}\right) + \frac{4}{14} B(1) + \frac{5}{14} B\left(\frac{3}{5}\right) \right] = 0.247$$

$$Gain(\text{temperature}) = 0.9403 - \left[ \frac{4}{14} B\left(\frac{2}{4}\right) + \frac{6}{14} B\left(\frac{4}{6}\right) + \frac{4}{14} B\left(\frac{3}{4}\right) \right] = 0.029$$

$$Gain(\text{humidity}) = 0.9403 - \left[ \frac{7}{14} B\left(\frac{3}{7}\right) + \frac{7}{14} B\left(\frac{6}{7}\right) \right] = 0.1518$$

$$Gain(\text{windy}) = 0.9403 - \left[ \frac{6}{14} B\left(\frac{3}{6}\right) + \frac{8}{14} B\left(\frac{6}{8}\right) \right] = 0.04813$$

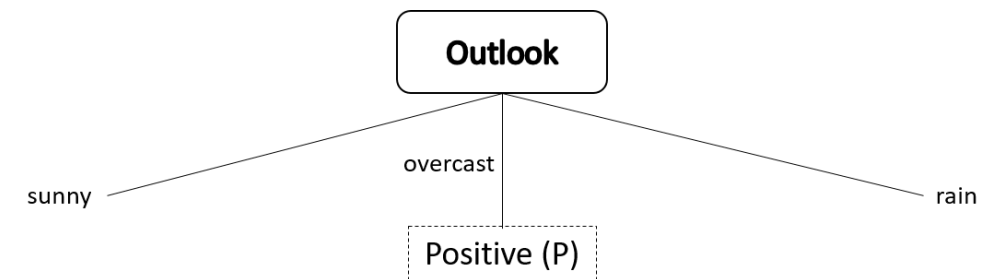
O **ganho de informação** é maximizado com o atributo **outlook**, que é, portanto, escolhido como o nó raiz da árvore.



# Exemplo de inferência de árvore de decisão com o método ID3

- Agora, criamos subconjuntos para cada um dos valores do atributo **Outlook** e refazemos os cálculos do **ganho de informação** para cada um deles.
- Quando **Outlook = overcast**, vemos no subconjunto abaixo que os valores dos outros atributos não importam, sendo a classe escolhida **sempre** a **Positiva (P)**, i.e., a decisão será sempre pela classe Positiva se o tempo estiver nublado.
- A **entropia** nesse caso é igual a **zero**, pois não há incerteza, indicando uma **folha** da árvore.
- Portanto, encontramos a folha deste ramo.

Day	Attributes				Class (y)
	Outlook	Temperature	Humidity	Windy	
3	overcast	hot	high	false	P
7	overcast	cool	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P



# Exemplo de inferência de árvore de decisão com o método ID3

- Quando **Outlook = rain**

Outlook = rain		Jogar?		
		P	N	
Temperature	hot	0	0	0
	mild	2	1	3
	cool	1	1	2
				5

Outlook = rain		Jogar?		
		P	N	
Humidity	high	1	1	2
	normal	2	1	3
				5

Outlook = rain		Jogar?		
		P	N	
Windy	true	0	2	2
	false	3	0	3
				5

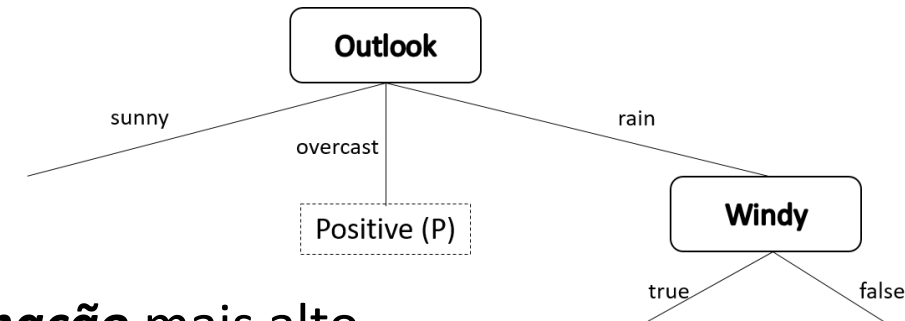
**Entropia do objetivo** para o subconjunto **Outlook = rain**:  $H(y \mid \text{Outlook} = \text{rain}) = 0.971$

$$\text{Gain}(\text{temperature}) = 0.971 - \left[ \frac{0}{5} B\left(\frac{0}{0}\right) + \frac{3}{5} B\left(\frac{2}{3}\right) + \frac{2}{5} B\left(\frac{1}{2}\right) \right] = 0.02$$

$$\text{Gain}(\text{humidity}) = 0.971 - \left[ \frac{2}{5} B\left(\frac{1}{2}\right) + \frac{3}{5} B\left(\frac{2}{3}\right) \right] = 0.02$$

$$\text{Gain}(\text{windy}) = 0.971 - \left[ \frac{2}{5} B\left(\frac{0}{2}\right) + \frac{3}{5} B\left(\frac{3}{3}\right) \right] = \boxed{0.971}$$

- Aqui, o atributo **windy** resulta no valor de **ganho de informação** mais alto quando o tempo estiver chuvoso (i.e., **Outlook = rain**).
- Por isso, o atributo **windy** será o nó do 2º nível da árvore, no ramo **rain** de **Outlook**.



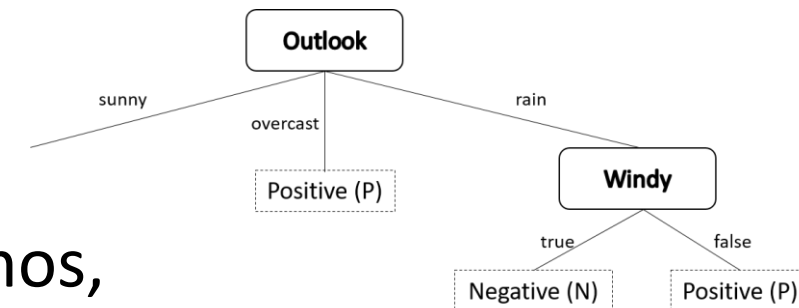
# Exemplo de inferência de árvore de decisão com o método ID3

- Se analisarmos o subconjunto onde **Outlook = rain** e **windy = false**, percebemos que a decisão será sempre pela classe **Positiva (P)**. A entropia  $H(y | \text{Outlook} = \text{rain}, \text{windy} = \text{false}) = 0$ .

Day	Attributes				Class (y)
	Outlook	Temperature	Humidity	Windy	
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
10	rain	mild	normal	false	P

- Além disso, a decisão sempre será pela classe Negativa (N) se **Outlook = rain** e **windy = true**. A entropia  $H(y | \text{Outlook} = \text{rain}, \text{windy} = \text{true}) = 1$ .

Day	Attributes				Class (y)
	Outlook	Temperature	Humidity	Windy	
6	rain	cool	normal	true	N
14	rain	mild	high	true	N



- Portanto, encontramos as folhas para os dois ramos, **true** e **false** do nó **windy**.

# Exemplo de inferência de árvore de decisão com o método ID3

- Quando **Outlook = sunny**

Outlook = sunny		Jogar?		
		P	N	
Temperature	hot	0	2	2
	mild	1	1	2
	cool	1	0	1
				5

Outlook = sunny		Jogar?		
		P	N	
Humidity	high	0	3	3
	normal	2	0	2
				5

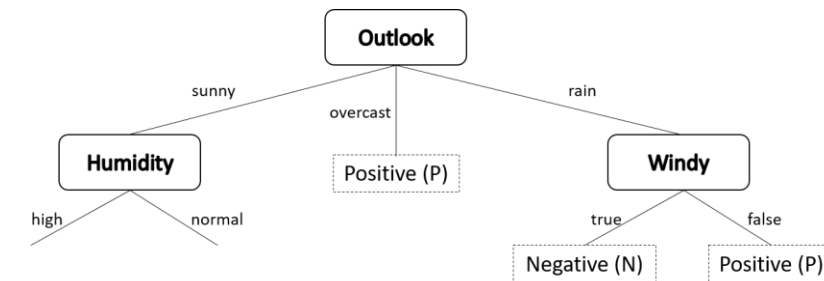
Outlook = sunny		Jogar?		
		P	N	
Windy	true	1	1	2
	false	1	2	3
				5

**Entropia do objetivo** para o subconjunto dado por **Outlook=sunny**:  $H(y \mid \text{Outlook} = \text{sunny}) = 0.971$

$$\text{Gain}(\text{temperature}) = 0.971 - \left[ \frac{2}{5} B\left(\frac{0}{2}\right) + \frac{2}{5} B\left(\frac{1}{2}\right) + \frac{1}{5} B\left(\frac{1}{1}\right) \right] = 0.570$$

$$\text{Gain}(\text{humidity}) = 0.971 - \left[ \frac{3}{5} B\left(\frac{0}{3}\right) + \frac{2}{5} B\left(\frac{2}{2}\right) \right] = \boxed{0.971}$$

$$\text{Gain}(\text{windy}) = 0.971 - \left[ \frac{2}{5} B\left(\frac{1}{2}\right) + \frac{3}{5} B\left(\frac{1}{3}\right) \right] = 0.02$$



- Aqui, o atributo **humidity** resulta no **ganho de informação** mais alto quando o tempo estiver ensolarado (i.e., **Outlook = sunny**).
- Por isso, o atributo **humidity** será o nó do 2º nível da árvore no ramo **sunny**.

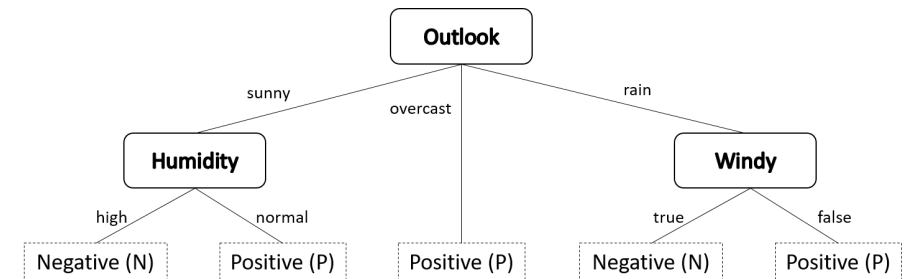
# Exemplo de inferência de árvore de decisão com o método ID3

- Analisando o subconjunto onde **Outlook = sunny** e **humidity = normal**, percebemos que a decisão será sempre pela classe **Positiva (P)**. A entropia  $H(y | \text{Outlook} = \text{sunny}, \text{humidity} = \text{normal}) = 0$ .

Day	Attributes				Class (y)
	Outlook	Temperature	Humidity	Windy	
9	sunny	cool	normal	false	P
11	sunny	mild	normal	true	P

- Além disso, a decisão sempre será pela classe **Negativa (N)** se **Outlook = sunny** e **humidity = high**. A entropia  $H(y | \text{Outlook} = \text{sunny}, \text{humidity} = \text{high}) = 0$

Day	Attributes				Class (y)
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
8	sunny	mild	high	false	N



- Portanto, encontramos as folhas para os dois ramos, **normal** e **high** do nó **humidity**.
- Com isso, a construção da **árvore de decisão** se encerra e podemos usar as regras encontradas por ela para classificar novos exemplos.
- **Algun atributo ficou de fora?** Sim, **Temperature**, indicando que ele não traz informação.



# Para casa

- Lista #7:
  - Exercícios #1, #4 e #7



# Considerações sobre as Árvores de Decisão

- Uma **árvore de decisão** *inferre através dos exemplos do conjunto de treinamento uma sequência de regras* que classifica os exemplos de entrada.
  - Portanto, elas são *fáceis de serem interpretadas* (modelos de **caixa branca**).
  - Podem ser facilmente interpretadas como uma **estrutura de controle de fluxo**.
- Embora as **árvores de decisão** sejam poderosos algoritmos de **classificação**, elas apresentam um *longo tempo de treinamento, principalmente quando os atributos são contínuos*.
- Em casos onde as classes são separadas por **fronteiras de decisão não-lineares**, as *árvores de decisão apresentam um desempenho de classificação superior ao apresentado por classificadores lineares*.
  - **Exemplo:** [DTTwoConcentricClassesClassification.ipynb](#)

# Considerações sobre as Árvores de Decisão

- Entretanto, quando as **classes não são bem separadas** (ou seja, se elas se sobrepõem), **as árvores são suscetíveis a sobreajustar** ao conjunto de treinamento, de modo que a **fronteira de decisão linear** dos **classificadores lineares** separa melhor as classes, apresentando melhor desempenho de **classificação**.
  - **Exemplo:** [DTTwoOverlappingClassesClassification.ipynb](#)
- Para evitar **sobreajuste**, existem duas maneiras:
  1. Limitamos sua profundidade e/ou o número mínimo de amostras necessárias em um nó folha.
  2. Geramos primeiro uma árvore completa e, em seguida, eliminamos alguns ramos (i.e., podamos a árvore).
- **Árvores de decisão precisam de muito pouco pré-processamento dos dados**. Em particular, elas não necessitam de **escalonamento dos atributos**, mas **valores faltantes devem ser tratados de alguma forma**.

# Considerações sobre as Árvores de Decisão

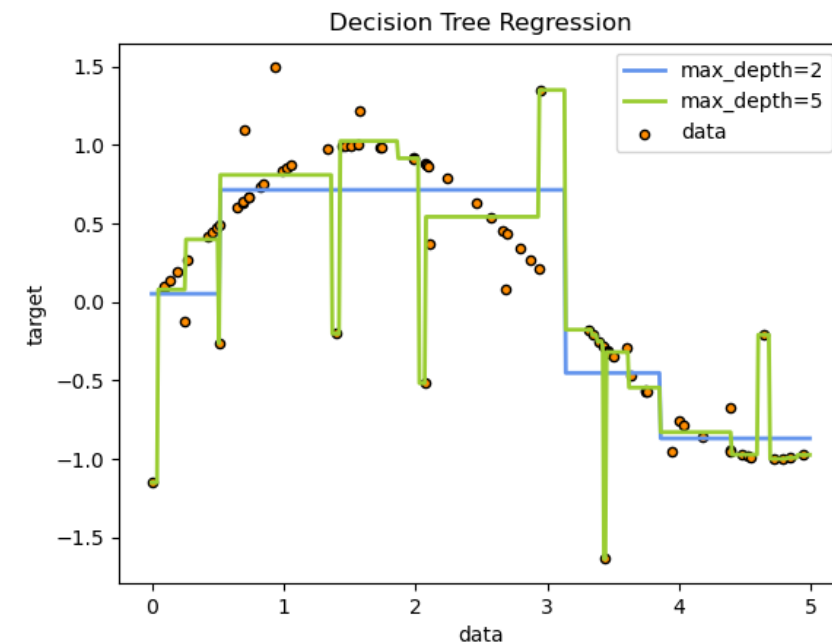
- O **principal problema** das **árvores de decisão** é que elas **são muito sensíveis a pequenas variações nos dados de treinamento** (e.g., rotação).
- **Árvores de decisão** criam **fronteiras de decisão ortogonais** (i.e., todas as **fronteiras de decisão** são perpendiculares aos eixos), o que as torna **sensíveis à rotação do conjunto de treinamento**.
  - **Exemplo:** [DTSensitivityToTrainingSetRotation.ipynb](#)
  - Uma maneira para minimizar esse problema é usar a técnica conhecida como **Análise de Componentes Principais (PCA)**, que rotaciona e dimensiona linearmente a matriz de atributos.
- Alguns **algoritmos para inferência de árvores de decisão são estocásticos** (e.g., o CART seleciona aleatoriamente o conjunto de atributos para avaliar em cada nó) e, portanto, **podem gerar árvores completamente diferentes com o mesmo modelo e conjunto de dados, mas sementes aleatórias diferentes**.
  - **Exemplo:** [DTSensitivityToTrainingSetDetails.ipynb](#)
  - As **florestas aleatórias** podem limitar essa instabilidade calculando a média das previsões feitas por diversas **árvores de decisão**.

# Considerações sobre as Árvores de Decisão

- Se não for restringida, a estrutura de uma **árvore de decisão** se adaptará aos dados de treinamento, ajustando-se muito bem e, provavelmente, se **sobreajustando** a eles.
  - Esse modelo é frequentemente chamado de modelo **não-paramétrico**, não porque não tenha nenhum parâmetro (ele geralmente tem muitos), mas porque o **número de parâmetros não é determinado antes do treinamento**, de modo que a estrutura do modelo é livre para se adaptar aos dados.
  - Para evitar o **sobreajuste** do modelo aos dados de treinamento, nós precisamos restringir (pode ser visto como **uma forma de regularização**) a liberdade da árvore de decisão durante o treinamento.
  - No Scikit-Learn, a regularização pode ser controlada pelos hiperparâmetros: ***max\_depth, min\_samples\_split, min\_samples\_leaf, min\_weight\_fraction\_leaf, max\_leaf\_nodes*** e ***max\_features***.
  - **Exemplo:** [DTRegularizationHyperparameters.ipynb](#)

# Considerações sobre as Árvores de Decisão

- **Árvores de decisão** também podem ser utilizadas para **regressão**.
  - Assim como em tarefas de **classificação**, as **árvores de decisão** tendem a se **sobreajustar** ao conjunto de treinamento ao lidar com tarefas de **regressão**.
  - **Exemplo:** [DTNoisyQuadraticDatasetRegression.ipynb](#)
- Predições feitas por **árvores de decisão** não são suaves nem contínuas, mas **aproximações constantes por partes/trechos**, conforme visto na figura ao lado.
- O valor previsto para cada trecho é sempre o valor médio dos exemplos nesse trecho.



# Classificação com árvores de decisão e SciKit-Learn

[Exemplo: DTTwoConcentricClassesClassification.ipynb](#)

# Import all necessary libraries.

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
from sklearn.metrics import accuracy_score
```

Importa a classe  
DecisionTreeClassifier.

Cria duas classes concêntricas  
com a função *make\_circles*.

# Define the number of examples.

N = 1000

# Create the dataset.

```
x, y = make_circles(n_samples=N, random_state=42, noise=0.1, factor=0.2)
```

Divide o conjunto em  
subconjuntos de  
treinamento (80%) e  
teste (20%).

# Split array into random train and test subsets.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=23, test_size=0.2)
```

# Instantiate classifier.

```
clf = DecisionTreeClassifier(criterion='entropy')
```

Instancia classificador.  
Não suporta atributos categóricos,  
apenas contínuos

# Fit the classifier on the training features and labels.

```
clf.fit(x_train, y_train)
```

Treina o classificador.

# Use the trained classifier to predict labels for the test features.

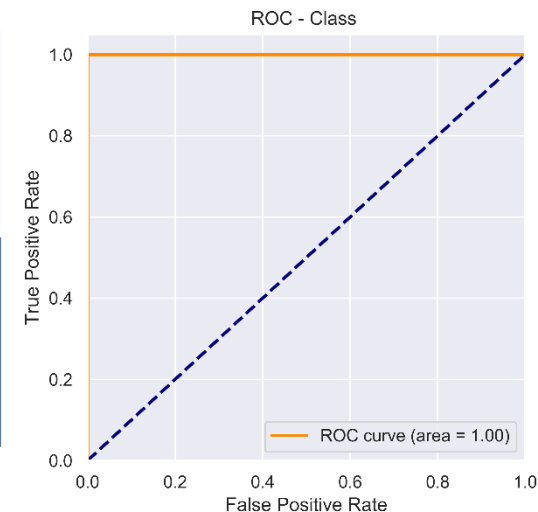
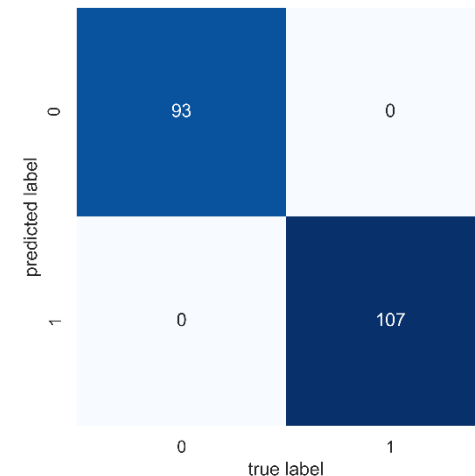
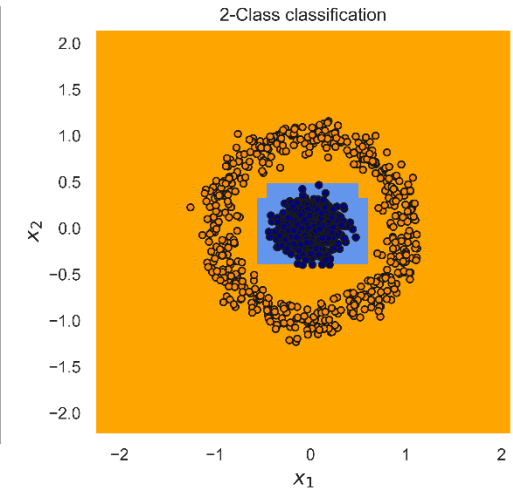
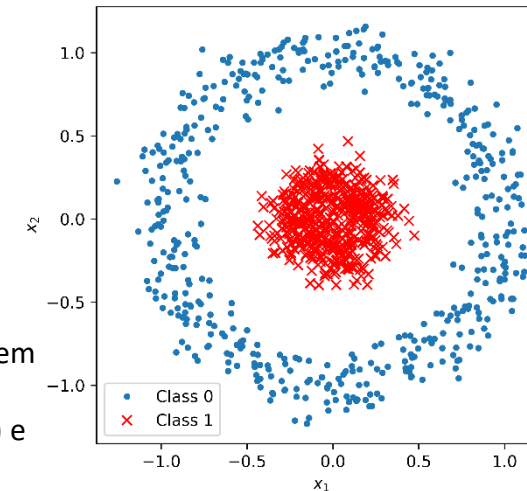
```
y_pred = clf.predict(x_test)
```

Realiza predição com  
conjunto de testes.

# Calculate and return the accuracy on the test data

```
accuracy = accuracy_score(y_test, y_pred)
```

Calcula a performance do  
classificador no conjunto de teste.



Exemplo de classificação de 2 classes concêntricas. As figuras mostram a distribuição das classes, fronteira de decisão, matriz de confusão e curva ROC. Conforme podemos ver a classificação do conjunto de testes é perfeita.



# Regressão com árvores de decisão e SciKit-Learn

# Import the necessary modules and libraries.

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import GridSearchCV

from sklearn.datasets import make\_regression

Importa a classe

DecisionTreeRegressor.

Cria dados para a regressão com a função make\_regression.

# Create dataset.

X, y = make\_regression(n\_samples=1000, n\_features=1, n\_informative=1, random\_state=42, noise=5)

# Split the dataset.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Divide o conjunto em subconjuntos de treinamento (80%) e teste (20%).

# Set parameters for grid-search.

param\_grid = [{'max\_depth': [1, 2, 3, 4, 5, 6, None], 'min\_samples\_leaf': [1, 2, 3, 4, 5, 6, 7, 8, 9]}]

Lista de valores a serem testados.

# Instantiate DT class.

reg = DecisionTreeRegressor(random\_state=42)

grid\_search = GridSearchCV(reg, param\_grid, cv=5, verbose=3, n\_jobs=-1)

Executa o grid search.

# Find best hyperparameters.

grid\_search.fit(X\_train, y\_train)

Imprime os valores ótimos dos hiperparâmetros.

# Print best parameters.

print(grid\_search.best\_params\_)

Realiza predição com conjunto de testes.

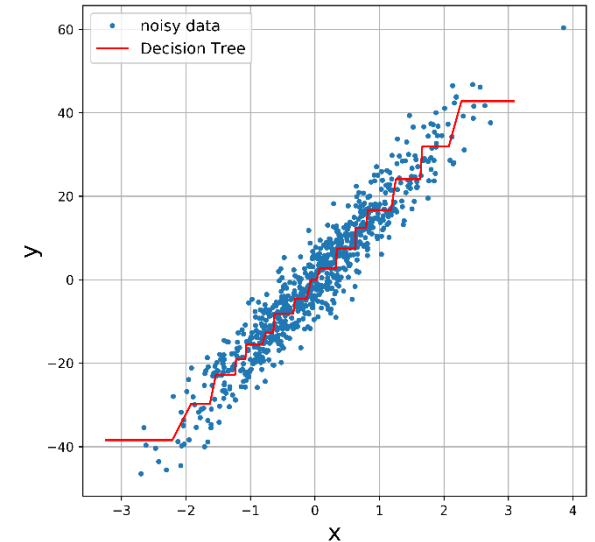
# Predicting with test set.

y\_pred = grid\_search.predict(X\_test)

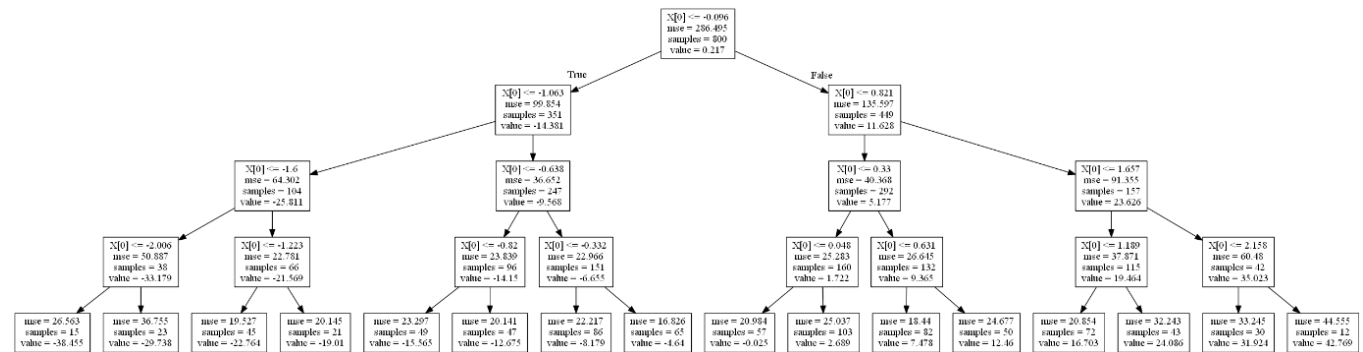
Calcula a performance com conjunto de teste.

# Calculate MSE.

mse = mean\_squared\_error(y\_test, y\_pred)



[Exemplo: DTMakeRegression.ipynb](#)



Exemplo de **regressão** utilizando **GridSearch** para encontrar os valores ótimos para os hiperparâmetros 'max\_depth' e 'min\_samples\_leaf'. As figuras acima mostram os dados ruidosos, a curva de regressão e a árvore de decisão do regressor.

# Avisos

- Nossa próxima aula será um estudo dirigido sobre ***florestas aleatórias*** (Lista #8) e ***k-Means*** (Lista #9).
- Material se encontra no github.

## Material das aulas

---

O material das aulas pode ser encontrado na pasta **slides** e pode ser acessado através dos links abaixo.

1. [Introdução](#)

2. [Regressão Linear](#)

3. [Regressão Não-Linear](#)

4. [Classificação](#)

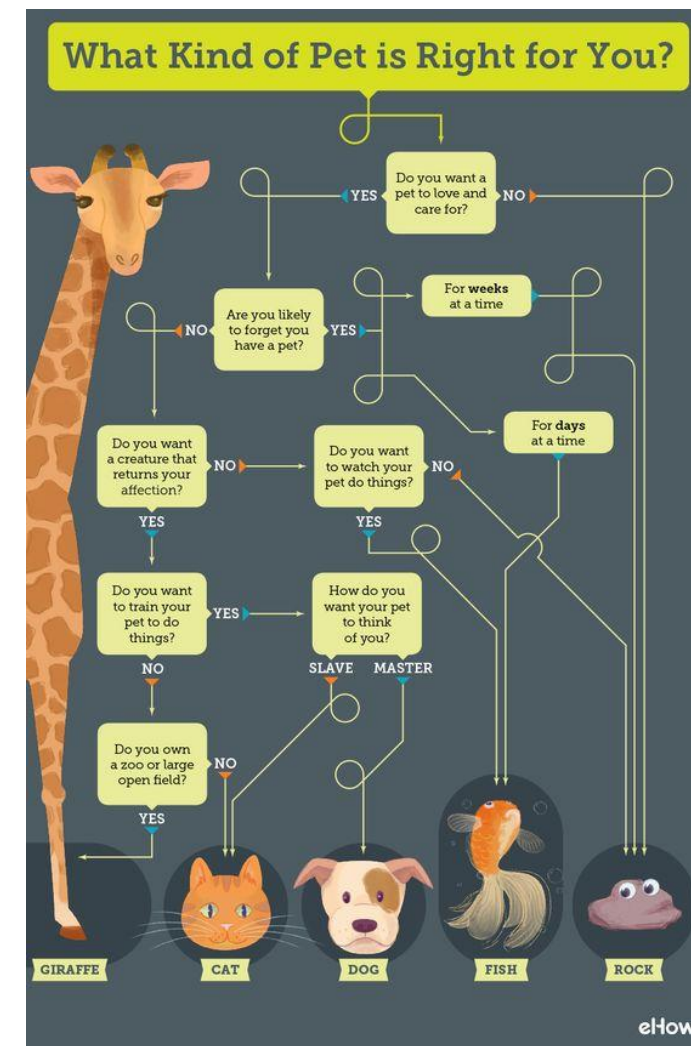
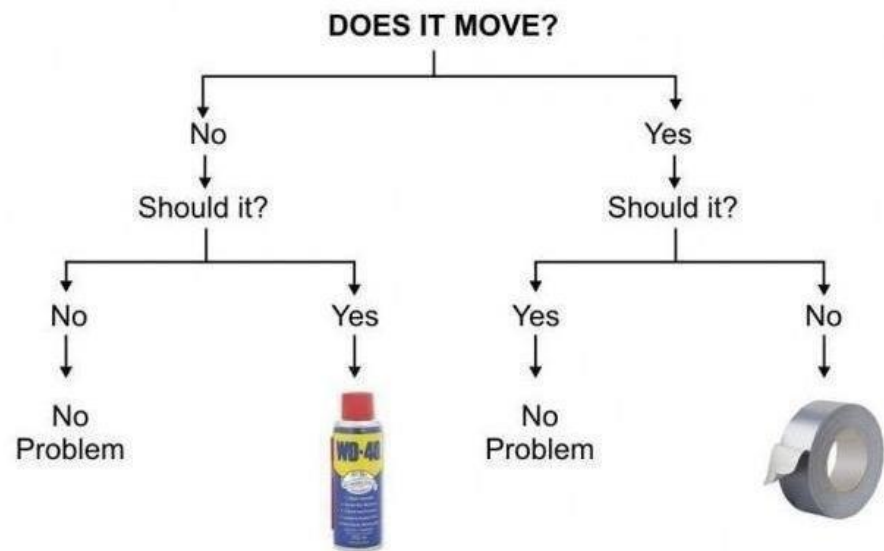
5. [kNN](#)

6. [Decision Trees](#)

→ 7. [Ensemble Learning e Random Forests](#)

→ 8. [k-Means](#)

Obrigado!



# Anexo I: Cálculo da entropia para mais de duas classes

# Cálculo da entropia para mais de duas classes

- Veja a tabela ao lado com os exemplos de treinamento.
- Problema com três classes: 0, 1, e 2. Portanto, o cálculo da entropia terá três termos.
- A **entropia** do objetivo, i.e.,  $y$ , para o conjunto de treinamento é

$$H(y) = - \left[ \frac{2}{4} \log_2 \left( \frac{2}{4} \right) + \left( \frac{1}{4} \right) \log_2 \left( \frac{1}{4} \right) + \left( \frac{1}{4} \right) \log_2 \left( \frac{1}{4} \right) \right] = 1.5$$

- Separando os atributos em tabelas separadas, temos

		Classes (y)			
		0	1	2	
x1	0	1	0	0	1
	1	0	0	1	1
	2	1	1	0	2
					4

		Classes (y)			
		0	1	2	
x2	0	2	0	0	2
	1	0	1	1	2
					4

Atributos		Rótulos
x1	x2	Classe (y)
0	0	0
2	1	1
1	1	2
3	0	0

- O **ganho de informação** de cada atributo é calculado como:

$$Gain(x1) = H(y) - \left\{ \frac{1}{4} H(x1 = 0) + \frac{1}{4} H(x1 = 1) + \frac{2}{4} H(x1 = 2) \right\}$$

$$Gain(x2) = H(y) - \left\{ \frac{2}{4} H(x2 = 0) + \frac{2}{4} H(x2 = 1) \right\}$$

# Cálculo da entropia para mais de duas classes

Como exemplo do cálculo da entropia, vejamos como  $H(x1 = 0)$  deve ser calculado.

$$\begin{aligned}
 H(x1 = 0) &= - \sum_{i=0}^2 p(y = i | x1 = 0) \log_2(p(y = i | x1 = 0)) \\
 &= -[p(y = 0 | x1 = 0) \log_2(p(y = 0 | x1 = 0)) + p(y = 1 | x1 = 0) \log_2(p(y = 1 | x1 = 0)) + \\
 &\quad p(y = 2 | x1 = 0) \log_2(p(y = 2 | x1 = 0))]
 \end{aligned}$$

Assim, os ganhos de informação para  $x1$  e  $x2$  são calculados como

$$\begin{aligned}
 Gain(x1) &= H(y) - \left\{ \frac{1}{4} H(x1 = 0) + \frac{1}{4} H(x1 = 1) + \frac{2}{4} H(x1 = 2) \right\} \\
 &= H(y) - \left\{ \frac{1}{4} \left[ \frac{1}{1} \log_2 \left( \frac{1}{1} \right) \right] + \frac{1}{4} \left[ \frac{1}{1} \log_2 \left( \frac{1}{1} \right) \right] + \frac{2}{4} \left[ \frac{1}{2} \log_2 \left( \frac{1}{2} \right) + \frac{1}{2} \log_2 \left( \frac{1}{2} \right) \right] \right\}
 \end{aligned}$$

$$\begin{aligned}
 Gain(x2) &= H(y) - \left\{ \frac{2}{4} H(x2 = 0) + \frac{2}{4} H(x2 = 1) \right\} \\
 &= H(y) - \left\{ \frac{2}{4} \left[ \frac{2}{2} \log_2 \left( \frac{2}{2} \right) \right] + \frac{2}{4} \left[ \frac{1}{2} \log_2 \left( \frac{1}{2} \right) + \frac{1}{2} \log_2 \left( \frac{1}{2} \right) \right] \right\}
 \end{aligned}$$

**Obs.:**

- $H(0) = 0$
- $H(1) = 0$
- $0 * \log_2(0) = 0$

		Classes (y)			
		0	1	2	
x1	0	1	0	0	1
	1	0	0	1	1
	2	1	1	0	2
					4

		Classes (y)			
		0	1	2	
x2	0	2	0	0	2
	1	0	1	1	2
					4

# Anexo II: Qual a entropia de 0?



# Qual a entropia de 0?

- À primeira vista, pode parecer que  $-x \log_2(x)$  para  $x = 0$  é um valor indeterminado, pois resulta em  $0 \times -\infty$ .
- Porém, se fizermos uma pequena modificação na equação,

$$H(x) = -x \log_2(x) = -\frac{\log_2(x)}{\frac{1}{x}},$$

e, na sequência, aplicarmos a regra de L'Hôpital, conseguimos encontrar o valor quando  $x$  se aproxima de zero do lado positivo ( $0_+$ ):

$$\lim_{x \rightarrow 0_+} H(x) = \lim_{x \rightarrow 0_+} \frac{1}{-\frac{1}{x^2}} = \lim_{x \rightarrow 0_+} \frac{x}{\log_2(2)} = 0.$$

Assim,  $H(x) = 0$ .

Figuras

