

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



Inatel

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

Árvores de decisão

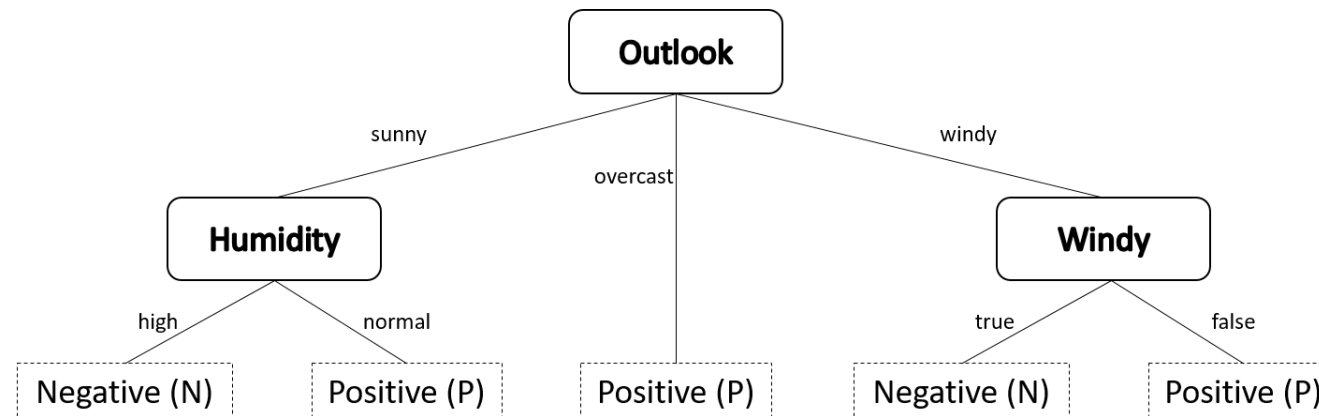
- Assim como o algoritmo k-NN, uma **árvore de decisão** (do inglês, **decision trees**), é um algoritmo de **aprendizado supervisionado não-paramétrico e não-linear** que pode ser utilizado tanto para **classificação** quanto para **regressão**.
- O objetivo é criar um modelo que prediz o valor de uma variável de saída (ou seja, uma classe), aprendendo regras simples 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 mais poderosos algoritmos 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].
- Trata-se de uma **estrutura de dados** muito importante para as áreas de computação, de aprendizado de máquina, tomada de decisão e teoria de jogos.
- A árvore possui um nó raiz, do qual parte o processo de decisão. Nesse processo, valores distintos de atributos geram arestas (i.e., ramificações) e, quando se chega a um nó folha, ocorre uma atribuição de classe.
- **Árvores de decisão** são modelos de **caixa branca**, ou seja, é possível entender e explicar facilmente como o modelo realiza a classificação de exemplos baseando-se nos atributos, sendo 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

- Na figura abaixo temos um exemplo baseado num conjunto de dados sobre se jogadores irão jogar tênis ou não. Nesse conjunto, analisam-se atributos diversos para estimar se eles jogarão ou não.
- Na figura, cada atributo (i.e., Clima, Humidade e Vento), leva a uma resposta e, para cada nó folha, atinge-se uma decisão sobre jogar ou não.
- O uso da árvore para classificar padrões é relativamente direto, mas é preciso responder uma questão crucial: ***como induzir uma árvore de decisão a partir de 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 de classificação e selecionar a mais simples (Navalha de Occam). Entretanto, essa abordagem, pode ser computacionalmente muito custosa.
- O **método ID3** (Iterative Dichotomiser 3), que discutiremos a seguir, é uma abordagem que não garante a obtenção da menor árvore, mas busca obter árvores apropriadas num período de tempo relativamente curto.
- O ID3 é um dos métodos de indução de árvores de decisão mais utilizados.
- A metodologia do **ID3** se baseia na **teoria da informação** para selecionar o atributo de cada nó.
- A ideia é escolher o **atributo** que for o mais longe possível em fornecer uma **classificação** exata dos exemplos. Um **atributo perfeito** (ou seja, muito bom) divide os exemplos em conjuntos (ou classes), cada um dos quais contendo todos exemplos positivos ou negativos do conjunto e, que portanto, serão **folhas** da **árvore de decisão**.
- Tudo o que precisamos, então, é uma medida formal de atributo "razoavelmente bom" ou "realmente inútil".
- O **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**.

Ganho de informação e entropia

- **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. Portanto, construir uma *árvore de decisão* tem tudo a ver com encontrar um *atributo* que retorne o maior **ganho de informação**.
- **Entropia:** é uma medida da *incerteza* de uma variável aleatória. Portanto, a aquisição de informação corresponde a uma redução na *entropia*.
- Uma variável aleatória com apenas um único valor (e.g., uma moeda que sempre que jogada cai com *cara* para cima) não tem nenhuma *incerteza* associada e, portanto, sua *entropia* é definida como sendo igual a **zero**. Isso significa que não se ganha/adquire nenhuma informação nova ao se observar o valor.
- Por outro lado, o resultado de se arremessar uma *moeda honesta* é igualmente provável de resultar em *cara* ou *coroa*, 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.
- 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 imagine um *moeda desonesta* que tenha uma probabilidade de resultar em *cara* em 99% dos arremessos. Nesse caso, a *entropia* deve ser um valor positivo muito próximo de zero, pois a incerteza do resultado é muito baixa.
- Assim, a *entropia* de uma variável aleatória V com valores v_i , onde cada um dos valores tem probabilidade $P(v_i)$, é definida como

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

Aprendizado de uma árvore de decisão

- Retornando ao problema da indução (ou aprendizado) 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 **atributo objetivo** (i.e., o rótulo ou saída desejada) para todo o conjunto de treinamento é dada por

$$H(Goal) = B\left(\frac{p}{p+n}\right) = -\left[\frac{p}{p+n}\log_2\left(\frac{p}{p+n}\right) + \left(1 - \frac{p}{p+n}\right)\log_2\left(1 - \frac{p}{p+n}\right)\right].$$

- Portanto, qualquer **árvore de decisão** correta para o conjunto de treinamento E classificará exemplos na mesma proporção de ocorrência das classes no conjunto de dados. Assim, a probabilidade de um exemplo ser da classe P é $p/(p+n)$ e a de um exemplo ser da classe N é $n/(p+n)$ ou $(1 - p/(p+n))$.
- Um teste com um único atributo x_k nós dá apenas parte da **entropia** para todo o conjunto, i.e., $H(Goal)$. Nós podemos medir exatamente o quanto cada atributo contribui 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 subconjuntos E_1, \dots, E_d . Cada subconjunto E_i possui p_i exemplos da classe positiva, P , e n_i exemplos da classe negativa, N . Um exemplo escolhido aleatoriamente do conjunto de treinamento tem o i -ésimo valor para o atributo com probabilidade $(p_i + n_i)/(p+n)$. Assim, a **entropia** restante esperada após o teste do atributo x_k é

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

- O **ganho de informação** com o atributo x_k é a redução na **entropia** total do conjunto de treinamento, que é dada por

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

Método ID3

- A ideia por trás do **método ID3** é maximizar o **ganho de informação** e então usar o procedimento recursivamente para os subconjuntos E_1, \dots, E_d . Ou seja, escolhe-se o atributo que gera a primeira ramificação e, então, se repete o processo para construir as subárvores.
- O processo por trás do **método ID3** pode ser resumido através da seguinte sequência de passos:
 - a) Cálculo da **entropia** do objetivo para todo o conjunto de treinamento.
 - b) Cálculo do **ganho de informação** de cada atributo $x_k, k = 1, \dots, K$ do conjunto de treinamento E .
 - c) Particionamento do conjunto E em subconjuntos E_1, \dots, E_d usando o atributo x_k para o qual o **ganho de informação** resultante após a divisão é maximizado.
 - d) Criação de um nó da árvore de decisão contendo o atributo que maximizou o **ganho de informação**.
 - e) Repetir os itens b) até d) em subconjuntos usando os atributos restantes. Esse processo continua até que a árvore classifique perfeitamente os exemplos de treinamento ou até que todos os atributos tenham sido utilizados.
- 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 à seguir.

Observações

- Além do **ganho de informação**, existem outras métricas que podem ser usadas para definir as partições. Uma possibilidade é usar métricas de distância/divergência, como o **índice de Gini**.
- Caso haja exemplos ruidosos, ou seja, exemplos que não são totalmente “consistentes”, passa a ser necessária uma análise estatística mais ampla, incluindo, por exemplo, **testes de hipóteses**.
- Outro ponto importante é, se for o caso, deve-se buscar metodologias para se lidar com atributos faltantes.
- 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 atributos e classes diferentes, precisará ser reconsiderado (os atributos podem não ser suficientes, por exemplo, precisando de mais atributos).
- Um problema muito comum das **árvores de decisão**, especialmente quando se tem um número muito grande de atributos, é o **sobreajuste**. Existem duas formas para se minimizar este problema:
 - Podar as árvores de decisão (tree pruning).
 - Ou utilizar **florestas aleatórias**.

Exemplo de Árvore de Decisão com método ID3

- Neste exemplo, 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 (x_i, d_i) , onde x_i é um vetor de atributos e d_i é um rótulo. Nesse conjunto de dados, cada entrada diz respeito à condição meteorológica de um dia. Os atributos são todos categóricos:
 - **Tempo:** {ensolarado, nublado, chuvoso}
 - **Temperatura:** {frio, agradável, quente}
 - **Umidade:** {alta, normal}
 - **Vento:** {presente, ausente}
- Os rótulos são apenas 2: 'positivo' (P), ou seja, jogar, e 'negativo' (N), ou seja, não jogar, denotando um problema genérico de duas classes.
- Um exemplo de condição meteorológica de um dia poderia ser descrito por: {nublado, dia frio, normal, ausente}.

Exemplo de Árvore de Decisão com 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 Árvore de Decisão com 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

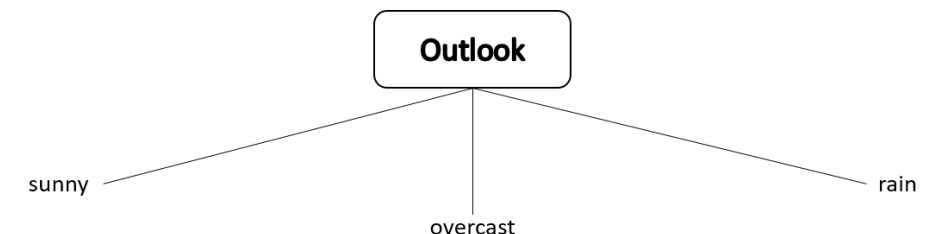
$$Gain(\text{outlook}) = 0.9403 - \left[\frac{5}{14} H\left(\frac{2}{5}\right) + \frac{4}{14} H(1) + \frac{5}{14} H\left(\frac{3}{5}\right) \right] = 0.247$$

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

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

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

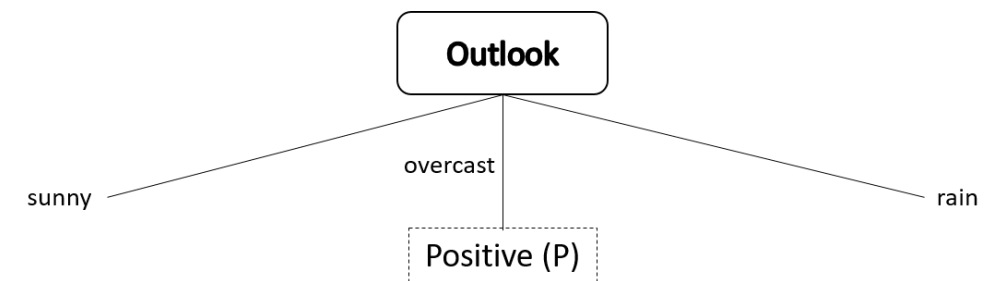
O **ganho de informação** é maximizado com o atributo **outlook**, que é, portanto, escolhido como o nó raiz da árvore.



Exemplo de Árvore de Decisão com método ID3

- Agora, precisamos testar o conjunto de treinamento para subconjuntos específicos do atributo do **Outlook**.
- Quando **Outlook = overcast**, vemos na tabela abaixo que os valores dos outros atributos não importam, sendo a classe escolhida sempre a Positiva (P), ou seja, a decisão será sempre pela classe Positiva se o tempo estiver nublado.
- 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 Árvore de Decisão com 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

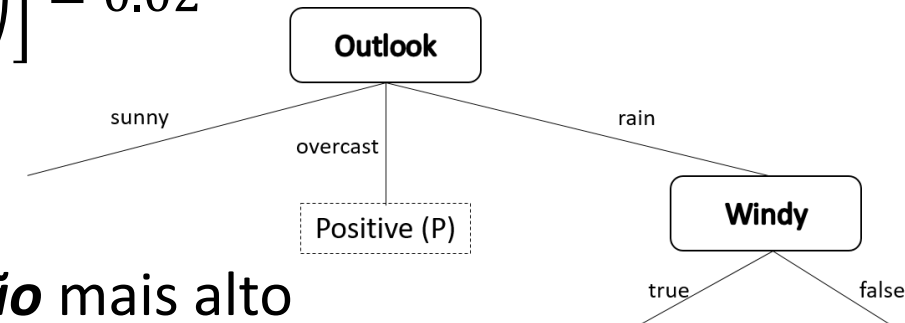
Outlook=rain		Jogar?		
		P	N	
Windy	true	0	2	2
	false	3	0	3
				5

$$Gain(\text{temperature}) = 0.9403 - \left[\frac{0}{5} H\left(\frac{0}{0}\right) + \frac{3}{5} H\left(\frac{2}{3}\right) + \frac{2}{5} H\left(\frac{1}{2}\right) \right] = 0.02$$

$$Gain(\text{humidity}) = 0.9403 - \left[\frac{2}{5} H\left(\frac{1}{2}\right) + \frac{3}{5} H\left(\frac{2}{3}\right) \right] = 0.02$$

$$Gain(\text{windy}) = 0.9403 - \left[\frac{2}{5} H\left(\frac{0}{2}\right) + \frac{3}{5} H\left(\frac{3}{3}\right) \right] = \boxed{0.971}$$

- Aqui, o atributo **windy** resulta no **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**.



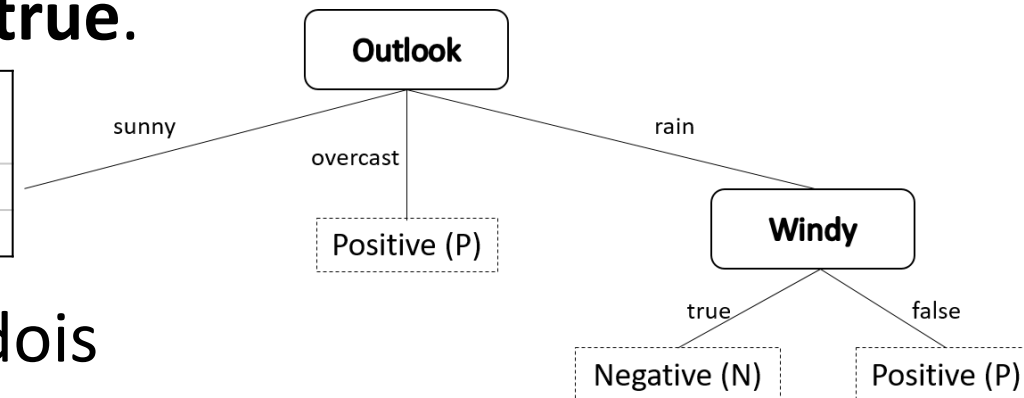
Exemplo de Árvore de Decisão com método ID3

- Se analisarmos a tabela com **Outlook = rain** e **windy = false** percebemos que a decisão será sempre pela classe Positiva (*P*).

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**.

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 Árvore de Decisão com 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

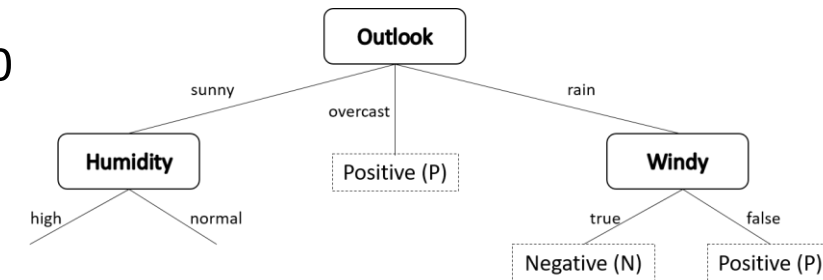
Outlook=sunny		Jogar?		
		P	N	
Windy	true	1	1	2
	false	1	2	3
				5

$$\text{Gain}(\text{temperature}) = 0.9403 - \left[\frac{2}{5} H\left(\frac{0}{2}\right) + \frac{2}{5} H\left(\frac{1}{2}\right) + \frac{1}{5} H\left(\frac{1}{1}\right) \right] = 0.570$$

$$\text{Gain}(\text{humidity}) = 0.9403 - \left[\frac{3}{5} H\left(\frac{0}{3}\right) + \frac{2}{5} H\left(\frac{2}{2}\right) \right] = \boxed{0.970}$$

$$\text{Gain}(\text{windy}) = 0.9403 - \left[\frac{2}{5} H\left(\frac{1}{2}\right) + \frac{3}{5} H\left(\frac{1}{3}\right) \right] = 0.019$$

- 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**.



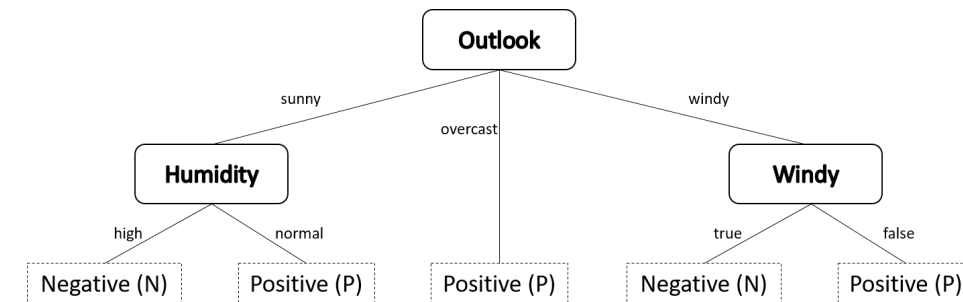
Exemplo de Árvore de Decisão com método ID3

- Após analisarmos a tabela com **Outlook = sunny** e **humidity = normal** percebemos que a decisão será sempre pela classe Positiva (P).

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**.

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.

Considerações

- Uma **árvore de decisão** transforma os exemplos do conjunto de treinamento em uma sequência de regras que classifica os exemplos de entrada. Portanto, elas são fáceis de serem interpretadas.
- Embora as **árvores de decisão** sejam poderosos algoritmos de **classificação**, elas apresentam um longo tempo de treinamento.
- 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
- Entretanto, quando as classes não são bem separadas, as árvores são suscetíveis a **sobreajustar** ao conjunto de treinamento, de modo que a **fronteira de decisão linear** dos **classificadores lineares** separe melhor as classes, apresentando melhor desempenho de **classificação**.
 - **Exemplo:** DTTwoOverlappingClassesClassification.ipynb

Considerações

- **Árvores de decisão** precisam de muito pouco pré-processamento dos dados. Em particular, elas não necessitam de **escalonamento dos atributos**.
- **Árvores de decisão** adoram **fronteiras de decisão ortogonais** (observando os exemplos, vocês vão perceber que todas as **fronteiras de decisão** são perpendiculares a um dos 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 (do inglês, Principal Component Analysis (PCA)).
- De maneira geral, o principal problema das **árvores de decisão** é que elas são muito sensíveis a pequenas variações nos dados de treinamento. Estas variações nos dados podem gerar árvores completamente diferentes.
 - **Exemplo:** DTSensitivityToTrainingSetDetails.ipynb
 - As **florestas aleatórias** podem limitar essa instabilidade calculando a média das previsões feitas por diversas **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

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='gini')
```

Instancia classificador.

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.

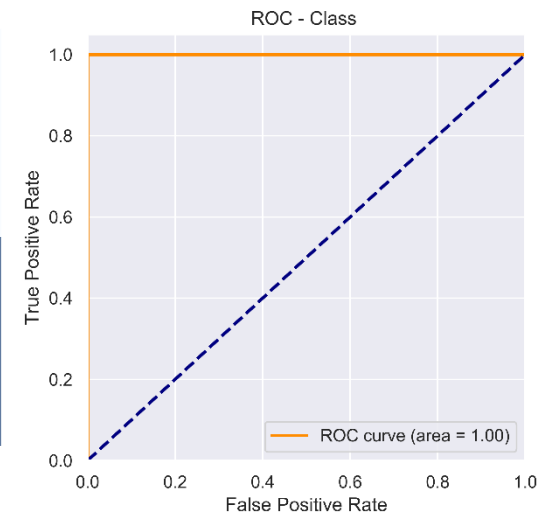
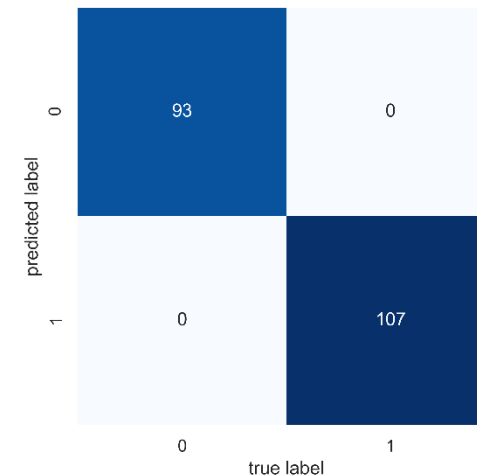
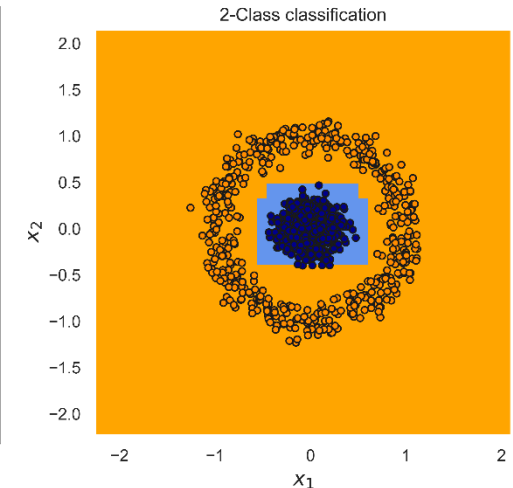
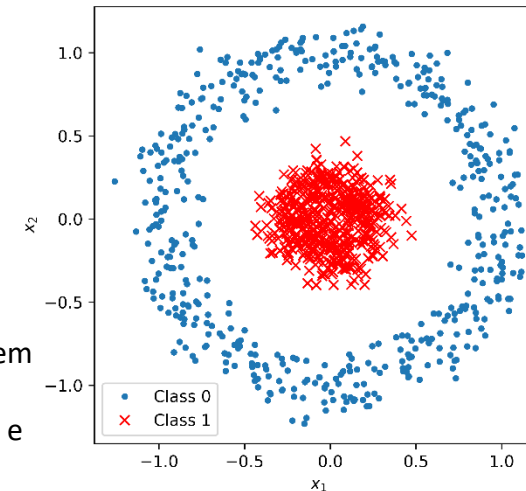
```
y_pred = clf.predict(x_test)
```

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
```

Importa a classe
DecisionTreeRegressor.

```
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.metrics import mean_squared_error
```

```
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.datasets import make_regression
```

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.

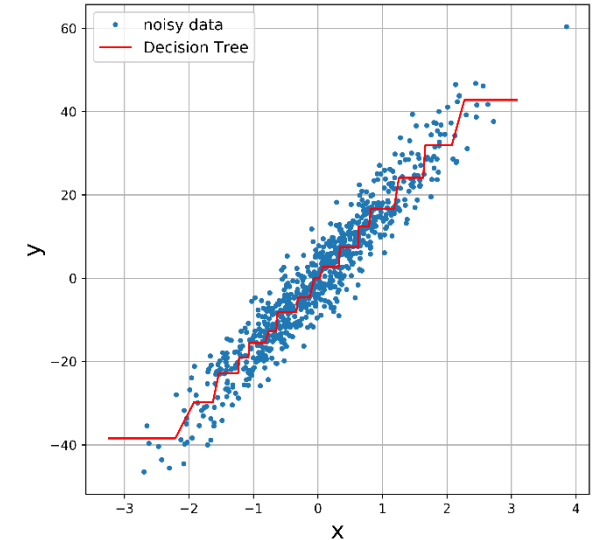
Predicting with test set.

```
y_pred = grid_search.predict(X_test)
```

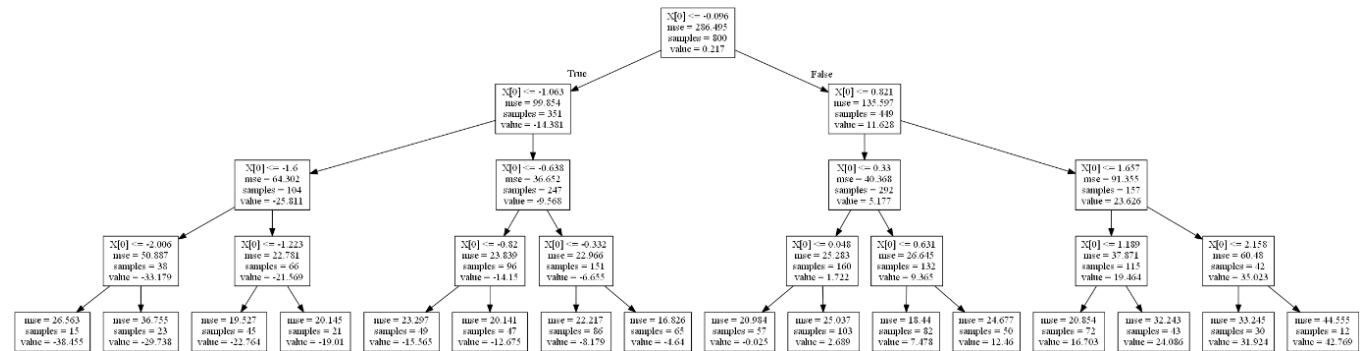
Calcula a performance
com conjunto de teste.

Calculate MSE.

```
mse = mean_squared_error(y_test, y_pred)
```



Exemplo: DTMakeRegression.ipynb

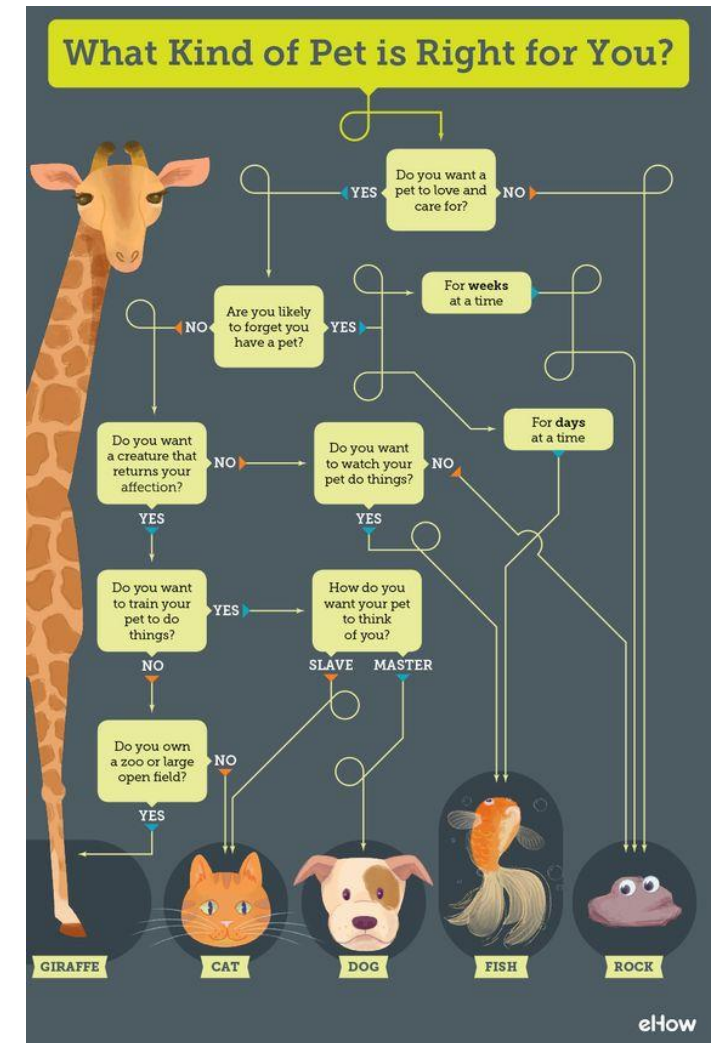
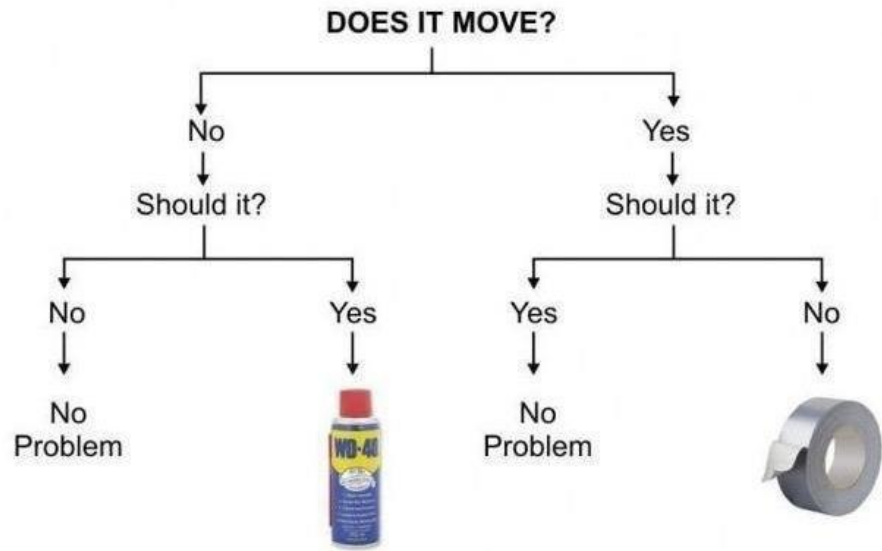


Exemplo de **regressão** utilizando **GridSearch** para encontrar os valores ótimos para os hiperparâmetros 'max_depth' e 'min_samples_leaf'. As figuras acima mostram os dados ruidosos, a curva de regressão e a árvore de decisão do regressor.

Avisos

- Exemplos já estão disponíveis no site.
- Lista #7 já está disponível no site.
- Lista #5 pode ser entregue até dia 15/05.
- Estudo dirigido sobre ***florestas aleatórias*** na próxima semana.

Obrigado!



Figuras

