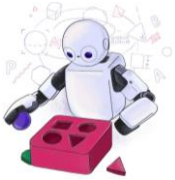
Slide 1

TP555 - Inteligência Artificial e Machine Learning:
Árvores de Decisão



Inatel

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

Slide 2

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.

Uma das muitas qualidades das Árvores de Decisão é que elas exigem muito pouca preparação de dados. Em particular, elas não necessitam de escalonamento de atributos.

As árvores de decisão são um método de aprendizado supervisionado não paramétrico usado para classificação e regressão. O objetivo é criar um modelo que prediz o valor de uma variável de saída, aprendendo regras simples de decisão inferidas a partir dos atributos de dados.

As árvores de decisão não são paramétricas porque não fazem suposições sobre a distribuição dos dados. Basicamente, isso significa que o modelo é construído apenas com base nos dados observados. Por exemplo, Classificadores Gaussian Naive Bayes são modelos paramêtricos, pois consideram que os atributos seguem distribuições Gaussianas.

Porque k-NN é um classificador não-linear.
https://stats.stackexchange.com/questions/178522/why-knn-is-a-non-linear-classifier

Slide 3

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).

Um grafo não-direcionado é um grafo, isto é, um conjunto de objetos (chamados vértices ou nós) que são conectados juntos, em que todas as arestas são bidirecionais. Um grafo não-direcionado é também chamado de rede não-direcionada. Por outro lado, um grafo em que as bordas apontam em uma direção é chamado de grafo direcionado. Grafos não-direcionados têm arestas que não têm uma direção. As arestas indicam uma relação de mão dupla, na qual cada aresta pode ser deslocada nas duas direções.

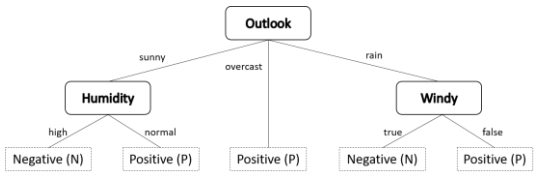
Modelos de caixa branca são o tipo de modelo que se pode entender e explicar claramente como eles se comportam, como produzem previsões e quais são as variáveis influentes.

Modelo de caixa preta: o funcionamento interno desses modelos é mais difícil de entender e não fornece uma estimativa da importância de cada atributo nas previsões do modelo, nem é fácil entender como os diferentes atributos interagem entre si. Modelos de caixa preta são altamente não-lineares.

Slide 4

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***?



Cada nó na árvore atua como um caso de teste para algum atributo, e cada extremidade descendente do nó, ou seja, uma folha, corresponde à possível classe do exemplo de teste.

Slide 5

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 ***método ID3*** é um dos métodos de indução de árvores de decisão mais utilizados.
- A metodologia do ***método 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 ***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***.

Os algoritmos da árvore de decisão transformam dados brutos em árvores de decisão baseadas em regras. O ID3 é um dos métodos de indução de árvores de decisão mais comuns. Ele foi introduzido em 1986 e é acrônimo de dicotomizador iterativo.

Dicotomização significa dividir-se em duas coisas completamente opostas.

É por isso que o método divide iterativamente os atributos em dois grupos, que são o atributo mais dominante e outros para construir uma árvore. Em seguida, calcula os ganhos de informação de cada atributo. Dessa maneira, o atributo mais dominante pode ser encontrado. Depois disso, o atributo mais dominante é colocado na árvore como um nó de decisão. Posteriormente, os valores de ganho de informação são calculados novamente entre os outros atributos. Assim, o próximo atributo mais dominante é encontrado. Finalmente, esse procedimento continua até chegar a uma decisão para esse ramo. Por isso, é chamado dicotomizador iterativo.

Slide 6

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 quantidade de ***incerteza*** ou ***aleatoriedade*** de uma variável aleatória (i.e., um conjunto de dados). Portanto, a aquisição de informação corresponde a uma redução na ***entropia***.
- Uma variável aleatória com apena 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 arremesar uma ***moeda honesta*** é igualmente provável de resultar em ***cara*** ou ***coroa***, associados aos valores 0 ou 1, respectivamente. Neste caso, esta varíavel 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

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

O ganho de informação é uma propriedade estatística que mede o quão bem um determinado atributo separa os exemplos de treinamento de acordo com sua classificação de destino.

O ganho de informação é uma diminuição da entropia. Ele calcula a diferença entre entropia antes da divisão/particionamento e a entropia média após a divisão do conjunto de dados com base nos valores do atributo fornecidos

Slide 7

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 $E_1, \ldots, E_d$. Cada subconjunto E_i possui p_i exemplos da classe positiva, *P*, e n_i exemplos da classe negativa, *N*. Um exemplo escolhido aleatóriamente 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

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

Uma ***árvore de decisão*** pode ser vista como uma ***fonte binária de informação*** com ***entropia*** igual a H(Goal).

O ganho de informação é uma diminuição na entropia. Ele nós dá uma medida a diferença entre a entropia antes da divisão e entropia após a divisão do conjunto de dados com base nos valores de atributo fornecidos. Assim, podemos escrever que o ganho de informação é dado por

Gain = entropiaAntesDaDivisão – entropiaApósADivisão

Observe que o valor do ganho de informação sempre será maior ou igual a 0:
Gain(Y|X) = H(Y) - H(Y|X) >= 0,
Dado que H(Y) >= H(Y|X). O pior caso é quando X e Y são independentes, e portanto, desta forma, H(Y|X) = H(Y), fazendo com que Gain(Y|X) = 0.

Referência:
https://stackoverflow.com/questions/3289589/can-the-value-of-information-gain-be-negative

Slide 8

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 gere 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 o conjunto de treinamento corrente (o conjunto é alterado a cada nova iteração do método ID3 de acordo com o(s) atributo(s) sendo testado(s)).
  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 de treinamento** 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.

Referência:
https://iq.opengenus.org/id3-algorithm/

Algumas características do método ID3 são:

- O ID3 usa uma abordagem gananciosa (greed), por isso não garante uma solução ideal; ele pode ficar preso em mínimos locais.
- O ID3 pode sobreajustar aos dados de treinamento (para evitaro sobreajuste, árvores de decisão menores devem ser preferidas ao invés das maiores).
- Esse método geralmente produz árvores pequenas, mas nem sempre produz a menor árvore possível.
- O ID3 é mais difícil de usar com dados contínuos (se os valores de qualquer atributo específico forem contínuos, haverá muito mais lugares para dividir os dados nesse atributo, e a busca pelo melhor valor a ser dividido pode ser demorada).

Características do método ID3

- O ID3 usa uma abordagem gananciosa (greed), por isso não garante uma solução ideal; ele pode ficar preso em mínimos locais.
- O ID3 pode sobreajustar aos dados de treinamento (para evitaro sobreajuste, árvores de decisão menores devem ser preferidas ao invés das maiores).
- Esse método geralmente produz árvores pequenas, mas nem sempre produz a menor árvore possível.
- O ID3 é mais difícil de usar com dados contínuos (se os valores de qualquer atributo específico forem contínuos, haverá muito mais lugares para dividir os dados nesse atributo, e a busca pelo melhor valor a ser dividido pode ser demorada).

Slide 10

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 prunning).
  - Ou utilizar ***florestas aleatórias***.

Deve-se usar índice gini ou entropia? A verdade é que, na maioria das vezes, não faz grande diferença: eles levam a árvores semelhantes. O índice gini é um pouco mais rápido para executar os cálculos, por isso é um bom padrão. No entanto, quando as árvores diferem, o índice gini tende a isolar a classe mais frequente em seu próprio ramo da árvore, enquanto a entropia tende a produzir árvores um pouco mais equilibradas

As árvores de decisão fazem muito poucas suposições sobre os dados de treinamento (em oposição aos modelos lineares, que obviamente assumem que os dados são lineares, por exemplo). Se deixada sem restrições, a estrutura da árvore se adaptará aos dados de treinamento, se ajustando a eles muito bem e, provavelmente, sobreajustando.

Outro tipo de mátrica também usada é a redução da variância. A redução de variância é frequentemente empregada nos casos em que a variável alvo é contínua. A redução da variação de um nó é definida como a redução total da variância da variável objetivo devido à divisão neste nó.

A poda é uma técnica utilizada para reduzir o tamanho das árvores de decisão, removendo seções da árvore que fornecem pouco poder/informação para classificar exemplos. A poda reduz a complexidade do classificador final e, portanto, melhora a precisão preditiva (ou sej,a o poder de generalizar) através da redução do sobreajuste.

Florestas aleatórias são modelos de ensemble (conjunto) criados à partir de muitas árvores de decisão usando subconjuntos aleatórios de atributos e votação majoritária para realizar predições. A idéia fundamental por trás de uma floresta aleatória é combinar muitas árvores de decisão em um único modelo. Individualmente, as

predições feitas por árvores de decisão podem não ser precisas, mas combinadas, as predições ficarão mais próximas do valor real, na média.

Slide 11

Exemplo de Árvore de Decisão com método ID3

- Neste exemplo, vamos construir uma árvore de decisão para predizer se jogadores irão ou não praticar um determinado esporte baseado em algumas condições meteorológicas.
- Vamos considerar um conjunto de dados da forma $(\boldsymbol{x}_i, d_i)$, onde $\boldsymbol{x}_i$ 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}.

Slide 12

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

Referência:
https://iq.opengenus.org/id3-algorithm/

Slide 13

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 raíz: 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

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

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

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

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

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

Slide 14

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.
- A ***entropia*** nessa 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

Slide 15

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

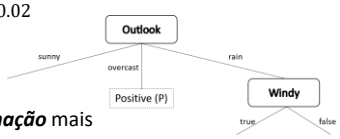
Outlook=rain		Jogar?		
		P	N	
Windy	true	0	2	2
	false	3	0	3
				5

Entropia para o subconjunto dado por **Outlook=rain**: $H(y \mid \textbf{Outlook} = \textbf{rain}) = 0.971$

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

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

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



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

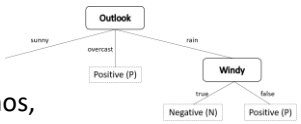
Exemplo de Árvore de Decisão com método ID3

- Se analisarmos a tabela com ***Outlook = rain*** e **windy = false** percebemos que a decisão será sempre pela classe Positiva (*P*). A entropia H(S|Outlook = rain, windy = 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(S|Outlook = rain, windy = false) = 0.

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***.

Slide 17

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

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

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

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

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

- 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 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*). A entropia H(S|Outlook = sunny, humidity = 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(S|Outlook = sunny, humidity = 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.

Slide 19

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

Entretanto, quando as classes não são bem separadas, as árvores são suscetíveis a sobreajustar aos dados de treinamento, de modo que a fronteira de decisão linear dos classificadores lineares generalize melhor.

Slide 20

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

Entretanto, quando as classes não são bem separadas, as árvores são suscetíveis a sobreajustar aos dados de treinamento, de modo que a fronteira de decisão linear dos classificadores lineares generalize melhor.

Florestas de decisão aleatória (ou florestas aleatórias) são um método de aprendizado conjunto para classificação e regressão, que operam através da construção de uma infinidade de árvores de decisão no momento do treinamento e gerando a classe que tem o maior númerode votos (classificação) ou a média dos valores de saída das várias árvores (regressão). As florestas de decisão aleatória corrigem o hábito das árvores de decisão de se sobreajustar ao conjunto de treinamento.

Slide 21

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

# 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)

# 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')

# Fit the classifier on the training features and labels.
clf.fit(x_train, y_train)

# Use the trained classifier to predict labels for the test features.
y_pred = clf.predict(x_test)

# Calculate and return the accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
```

Importa a classe DecisionTreeClassifier.

Cria duas classes concêntricas com a função ***make_circles***.

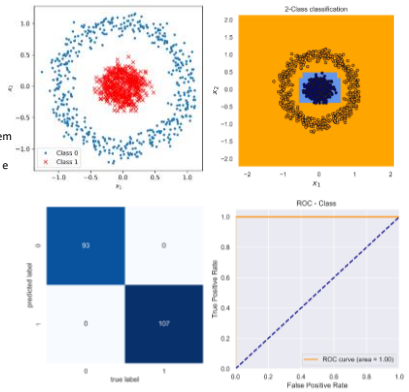
Divide o conjunto em subconjuntos de treinamento (80%) e teste (20%).

Instancia classificador.

Treina o classificador.

Realiza predição com conjunto de testes.

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.

Observação: A biblioteca scikit-learn usa uma versão otimizada do algoritmo CART, porém, a implementação do scikit-learn não suporta variáveis categóricas por enquanto.

Slide 22

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

# Create datase.
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)

# 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]}]

# Instantiate DT class.
reg = DecisionTreeRegressor(random_state=42)
grid_search = GridSearchCV(reg, param_grid, cv=5, verbose=3, n_jobs=-1)

# Find best hyperparameters.
grid_search.fit(X_train, y_train)

# Print best parameters.
print(grid_search.best_params_)

# Predicting with test set.
y_pred = grid_search.predict(X_test)

# Calculate MSE.
mse = mean_squared_error(y_test, y_pred)
```

Importa a classe DecisionTreeRegressor.

Cria dados para a regressão com a função make_regression.

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

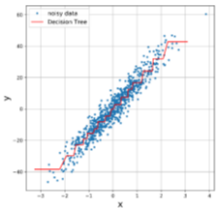
Lista de valores a serem testados.

Executa o grid search.

Imprime os valores ótimos dos hiperparâmetros.

Realiza predição com conjunto de testes.

Calcula a performance com conjunto de teste.



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.

Documentação da classe DecisionTreeRegressor: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

Slide 23

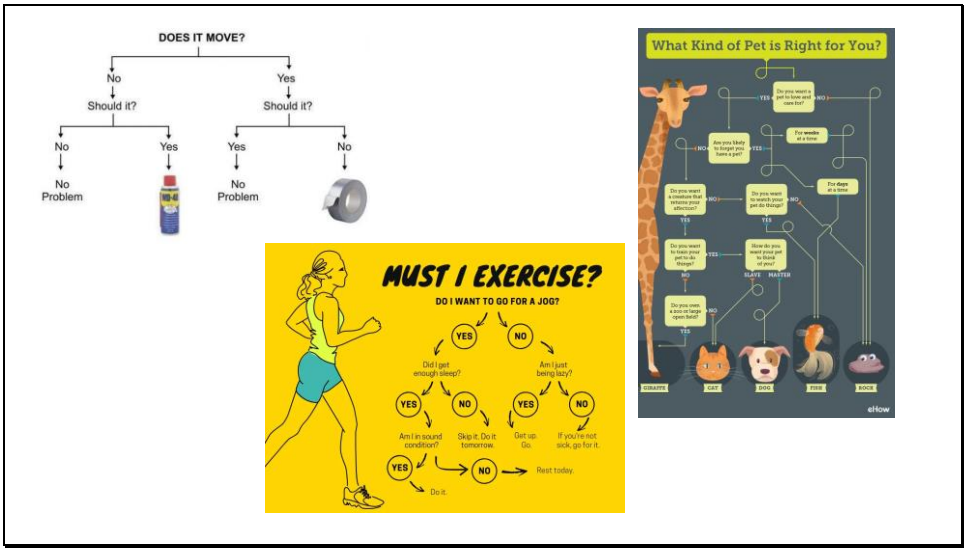
Avisos

- 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.

Slide 24

Obrigado!

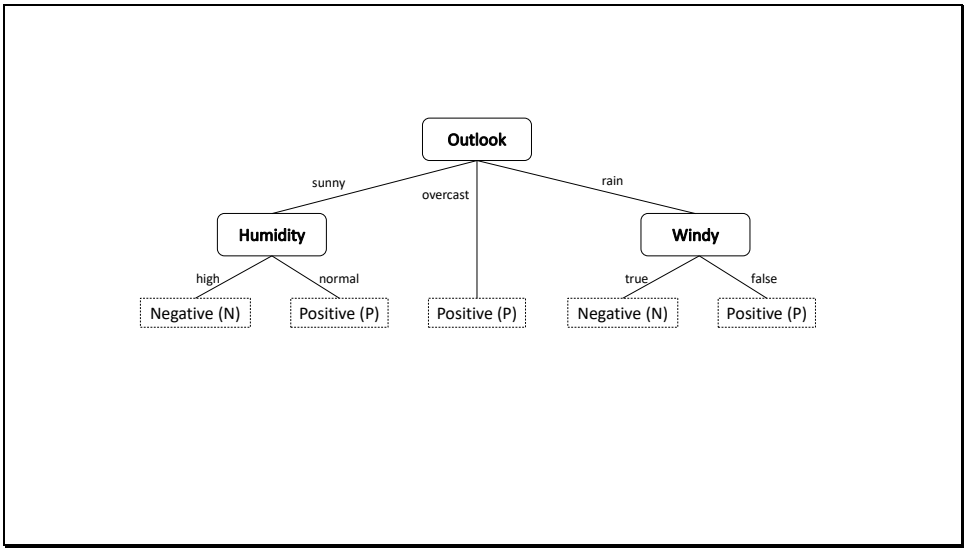
Slide 25


DOES IT MOVE?
No
Yes
Should it?
Should it?
No
Yes
Yes
No
No Problem
No Problem
MUST I EXERCISE?
DO I WANT TO GO FOR A JOG?
YES
NO
Did I get enough sleep?
Am I just being lazy?
YES
NO
YES
NO
Am I in sound condition?
Skip it. Do it tomorrow.
Get up. Go.
If you're not sick, go for it.
YES
NO
Rest today.
Do it.
What Kind of Pet is Right for You?
eHow

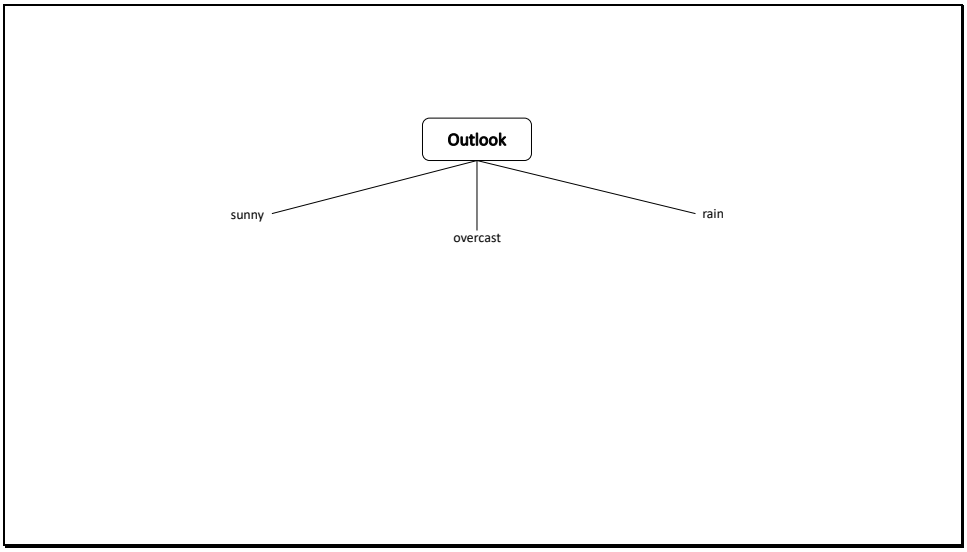
Slide 26

Figuras

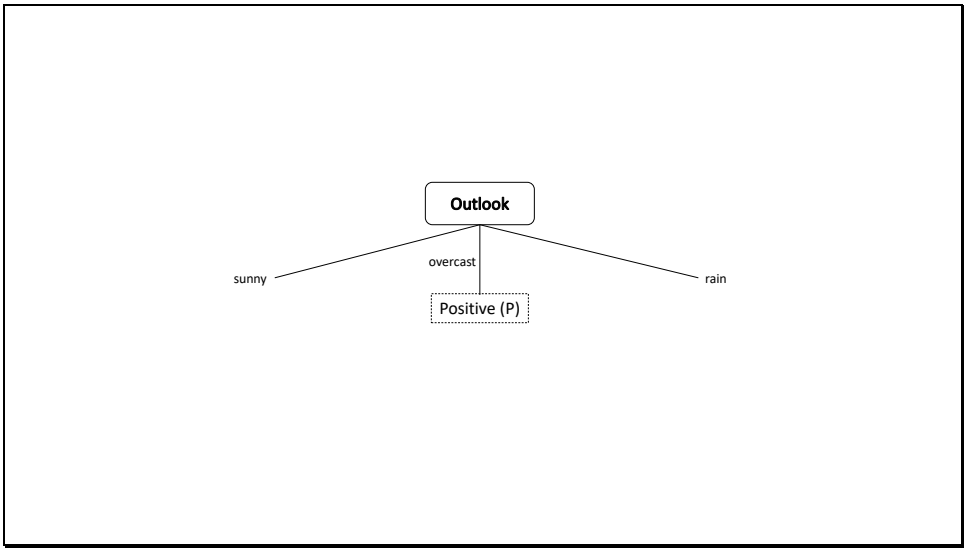
Slide 27


Outlook
sunny
overcast
rain
Humidity
Windy
high
normal
true
false
Negative (N)
Positive (P)
Positive (P)
Negative (N)
Positive (P)

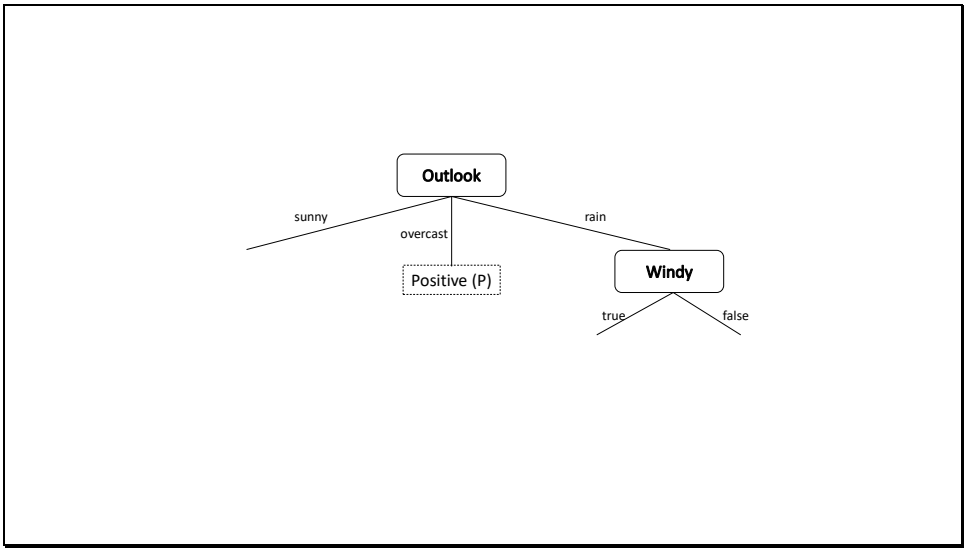
Slide 28



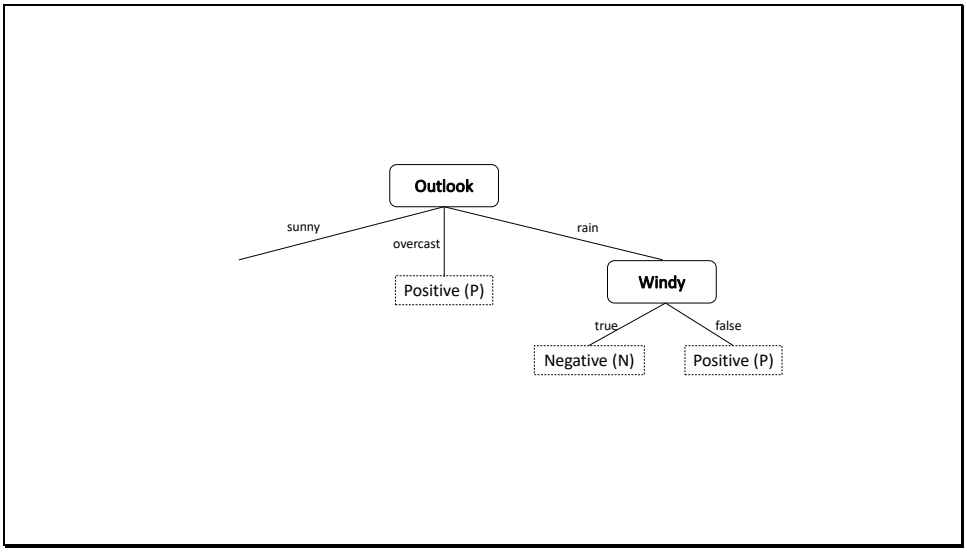
Slide 29



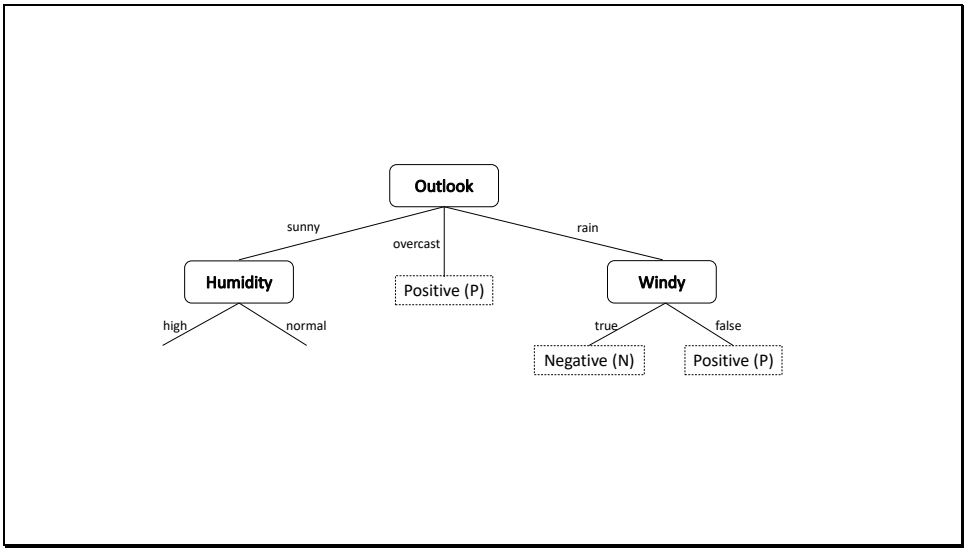
Slide 30



Slide 31


Outlook
sunny
overcast
rain
Positive (P)
Windy
true
false
Negative (N)
Positive (P)

Slide 32


Outlook
sunny
overcast
rain
Humidity
Positive (P)
Windy
high
normal
true
false
Negative (N)
Positive (P)