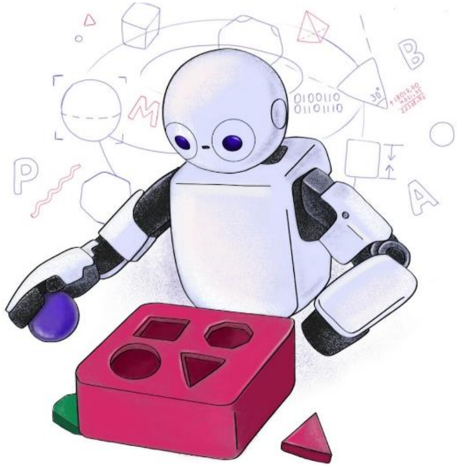
TP555 - Inteligência Artificial e Machine Learning: *Árvores de Decisão*



Inatel

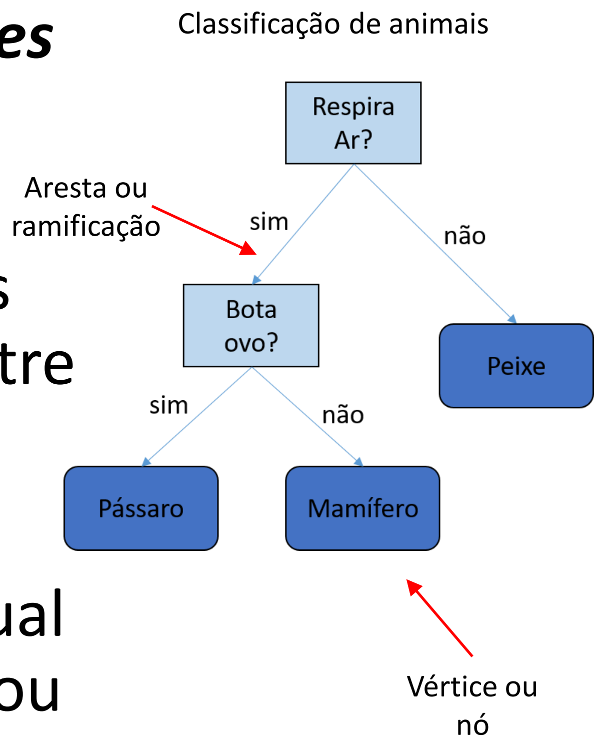
Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Árvores de decisão

- Assim como o 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***.
- ***Não-paramétrico***: o modelo não tem um número fixo de parâmetros, como por exemplo, modelos de regressão linear e logística.
  - O modelo é construído apenas com base nos dados observados e o número de parâmetros tende a crescer com ele (e.g., tamanho da árvore).
  - Ou seja, os ***parâmetros são criados durante o treinamento para explicar os dados***.
- Outra forma de definir ***não-paramétrico*** é perceber que o modelo não assume nenhuma distribuição dos dados e, portanto, não fixa o número de parâmetros.
- ***Não-linear***: as funções hipótese e discriminante são ***funções não-lineares***, i.e., os dados não são aproximados ou separados por um ***hiperplano***.

Árvores de decisão

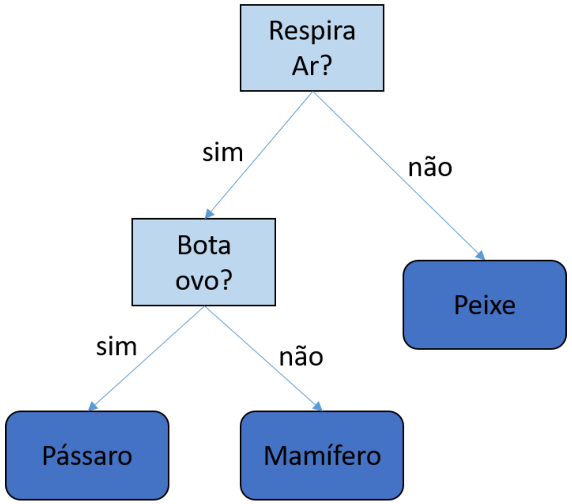
- O objetivo é criar um modelo que prediz o valor de uma variável de saída (i.e., uma classe ou valor), aprendendo ***regras simples de decisão*** inferidas a partir dos atributos do conjunto de treinamento.
- As ***árvores de decisão*** são os componentes fundamentais das ***florestas aleatórias*** (do inglês, ***random forests***) que estão entre os ***mais poderosos algoritmos de aprendizado de máquina*** disponíveis atualmente.
- 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].
  - Por exemplo, o vértice "Pássaro" só se conecta com o vértice "Mamífero" através do vértice "Bota ovo?"



Árvores de decisão

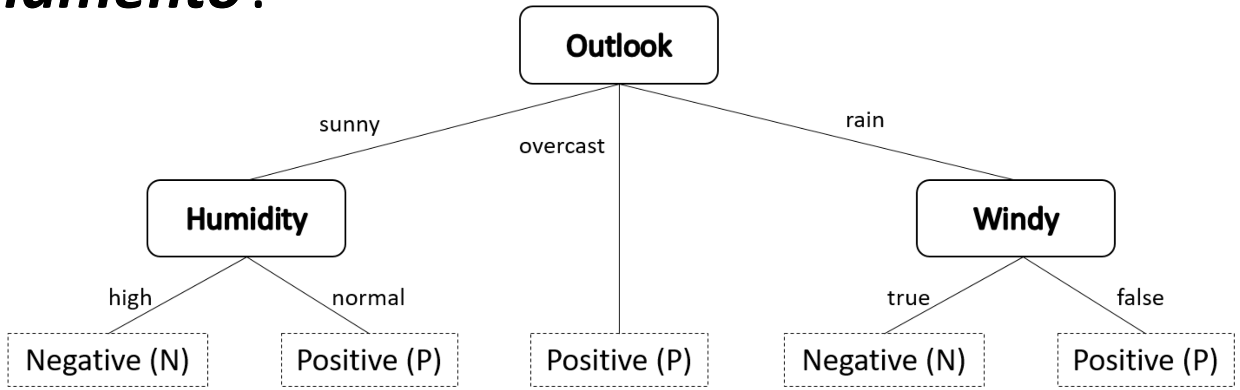
- A árvore possui um ***nó raiz***, do qual ***parte o processo de decisão***.
- Nesse processo, ***valores distintos de atributos geram arestas*** (i.e., ramificações) e, quando se chega a um ***nó folha***, ocorre uma atribuição de uma classe ou valor.
- ***Árvores de decisão*** são modelos de ***caixa 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).

Classificação de animais



Árvores de decisão

- A figura abaixo mostra uma árvore de decisão criada a partir de informações se jogadores jogarão tênis ou não dependendo 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.
- Na figura, cada atributo leva a uma ramificação até que se atinge um nó folha, onde decide-de 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 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*** da classificação ou regressão ***e selecionar a mais simples*** (menos complexa) utilizando a regra da ***Navalha de Occam***.
- Entretanto, essa abordagem, pode ser computacionalmente muito custosa.
- Encontrar uma árvore de decisão ótima é um problema NP-completo.
  - Non-Deterministic Polynomial time, ou seja, é um problema de "Tempo polinomial não-determinístico".
- Vários métodos utilizam ***abordagens heurísticas*** para induzir árvores de decisão.
- Dentre os mais conhecidos podemos citar: **ID3**, C4.5, C5.0 e CART.

O processo de indução de uma árvore de decisão

- Um dos métodos heurísticos de indução de árvores de decisão mais utilizados é o ***método ID3*** (do inglês, ***Iterative Dichotomiser 3***).
  - ***Dicotomização iterativa*** significa dividir-se repetidamente os atributos em duas ou mais classes a cada iteração (ou passo).
- O ***método ID3*** é uma abordagem que ***não garante*** a obtenção da ***menor árvore possível***, mas busca obter ***árvores apropriadas num intervalo de tempo relativamente curto***.
- Ele usa uma ***estratégia gananciosa top-down***, 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?
- A metodologia do ***método ID3*** se baseia na ***teoria da informação*** para selecionar o melhor 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 exatamente.
- Um ***atributo perfeito*** (ou seja, excelente) divide perfeitamente todos os exemplos em grupos.
  - Ou seja, é o atributo que consegue explicar perfeitamente o conjunto de dados.
- 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***.
- A seguir, veremos alguns conceitos úteis para que possamos usar o ***método ID3*** para inferir árvores de decisão.

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, a cada momento, 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, um ***ganho de informação*** corresponde a uma redução na ***entropia***.
  - O quanto conseguimos explicar do conjunto com aquele atributo.
- ***Exemplo***: 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 e, portanto, sua ***entropia*** é igual a ***zero***. Isso significa que ***não se ganha nenhuma informação nova ao se observar o valor***.

Ganho de informação e entropia

- 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

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

Aprendizado 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

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

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

$$B(q) = -\big(q\log_2(q) + (1-q)\log_2(1-q)\big).$$

- 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 $1-(p/(p+n))$.

Aprendizado de uma árvore de decisão

- Um teste em um único atributo, x_k, pode nos dar apenas parte da ***entropia do objetivo***, $H(Goal)$.
- 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***: $E_1, \ldots, E_d$. Cada subconjunto E_i possui p_i exemplos da classe positiva, *P*, e n_i exemplos da classe negativa, *N*.
- Um exemplo escolhido aleatoriamente do conjunto de treinamento tem o *i*-ésimo valor para o atributo x_k com probabilidade $(p_i + n_i)/(p + n)$. Assim, a ***entropia*** restante esperada após o teste do atributo x_k é dada por

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

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

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

Método ID3

- A ideia por trás do ***método ID3*** é maximizar o ***ganho de informação*** e, então, usar o procedimento ***recursivamente*** para cada um dos subconjuntos $E_1, \ldots, E_d$.
- Ou seja, escolhe-se o atributo que gera a primeira ramificação e, então, se repete o processo para construir o restante da árvore.
- O ***método ID3*** pode ser resumido através da seguinte sequência de passos:
  a) Cálculo da ***entropia do objetivo*** para o (sub)conjunto de treinamento ***corrente***.
     - **OBS**.: o (sub)conjunto é alterado a cada nova iteração do método de acordo com o(s) atributo(s) sendo testado(s).

  b) Cálculo do ***ganho de informação*** de cada atributo $x_k, k = 1, \ldots, K$ do conjunto de treinamento E.

  c) Criação de um nó da árvore de decisão contendo o atributo que maximizou o ***ganho de informação***.

  d) Particionamento do (sub)conjunto em subconjuntos usando o atributo x_k para o qual o ***ganho de informação*** resultante após a divisão é maximizado.

  e) Repetir os itens a) até d) em ***subconjuntos de treinamento*** usando os atributos restantes. Esse ***processo continua até que a árvore classifique perfeitamente todos 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 a seguir.

Características do método ID3

- Por usar uma abordagem gananciosa (do inglês, *greedy*), o ID3 não garante uma solução ideal, podendo ficar preso em mínimos locais.
- Pode sobreajustar aos dados de treinamento
  - Para evitar o 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.
- É mais difícil de usar com dados contínuos. Se os valores de um atributo forem contínuos, haverá muito mais lugares para dividir os dados nesse 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***.
- 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. 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 atributos, porém classes diferentes***, precisará ser reconsiderado, pois 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***.

Exemplo de Árvore de Decisão com método ID3

- 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 correspondente ao vetor de atributos.
- Nesse conjunto, cada entrada diz respeito à condição meteorológica de um dia e se os jogadores jogaram ou não.
- Existem ***quatro*** atributos 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 binário.
- Um exemplo de condição meteorológica de um determinado dia poderia ser descrito por: {nublado, 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 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

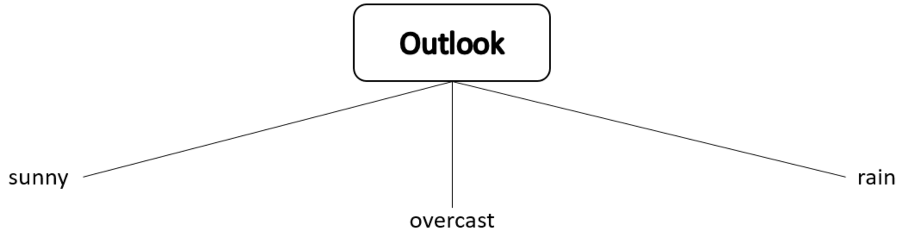
$$Gain(\textbf{outlook}) = 0.9403 - \left[\frac{5}{14}H\left(\frac{2}{5}\right) + \frac{4}{14}H(1) + \frac{5}{14}H\left(\frac{3}{5}\right)\right] = \boxed{0.247}$$

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

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

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

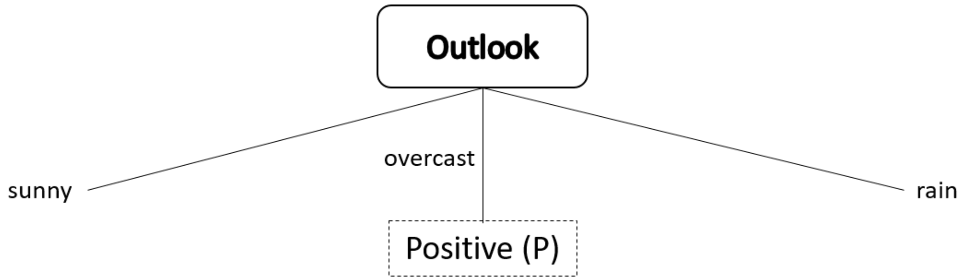
O ***ganho de informação*** é maximizado com o atributo ***outlook***, que é, portanto, escolhido como o nó raíz da árvore.



Exemplo de Árvore de Decisão com método ID3

- Agora, criamos subconjuntos para cada um dos valores do atributo ***Outlook*** (i.e., overcast, sunny e rain) 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*), 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



Exemplo de Árvore de Decisão com método ID3

- Quando ***Outlook = rain***

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

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

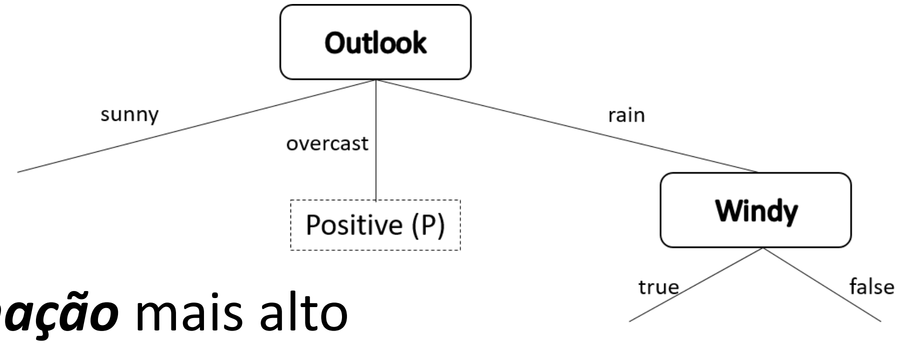
Outlook = rain		Jogar? P	Jogar? N	
Windy	true	0	2	2
	false	3	0	3
				5

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

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

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

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



- 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**.

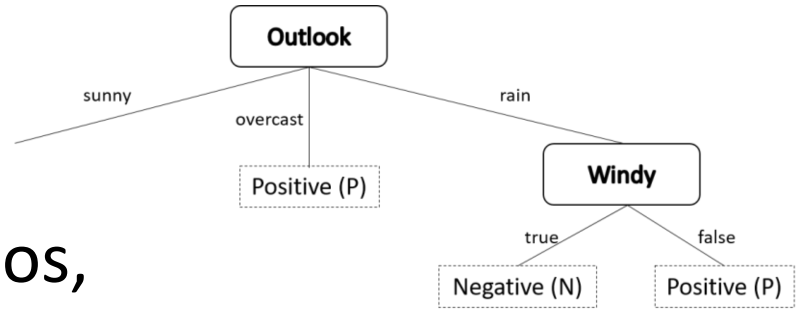
Exemplo de Árvore de Decisão com método ID3

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

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

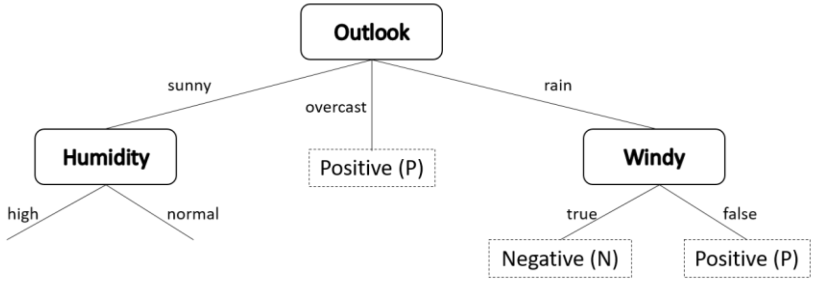
Outlook = sunny		**Jogar?** P	N	
Windy	true	1	1	2
	false	1	2	3
				5

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

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

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

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



- 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***.

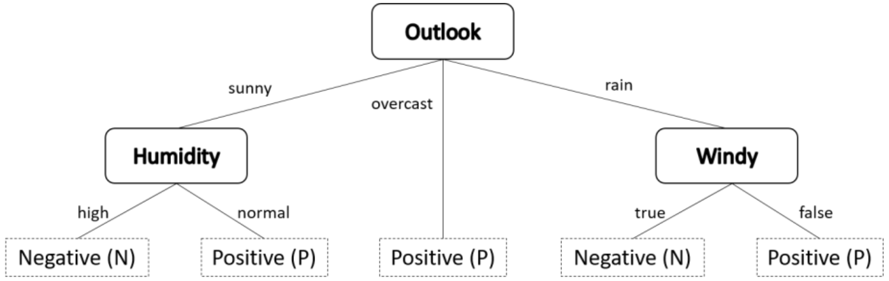
Exemplo de Árvore de Decisão com método ID3

- Analisando o subconjunto onde ***Outlook = sunny*** e ***humidity = normal***, percebemos que a decisão será sempre pela classe ***Positiva*** (*P*). A entropia H(y|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(y 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.
- Algum atributo ficou de fora? Sim, **Temperature**, indicando que ele não traz informação.

Exercício

- Lista #7:
  - Exercício #1 (Árvores de Decisão)

Considerações sobre as Árvores de Decisão

- Uma ***árvore de decisão*** infere 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 precisam ser tratados de alguma forma.

Considerações sobre as Árvores de Decisão

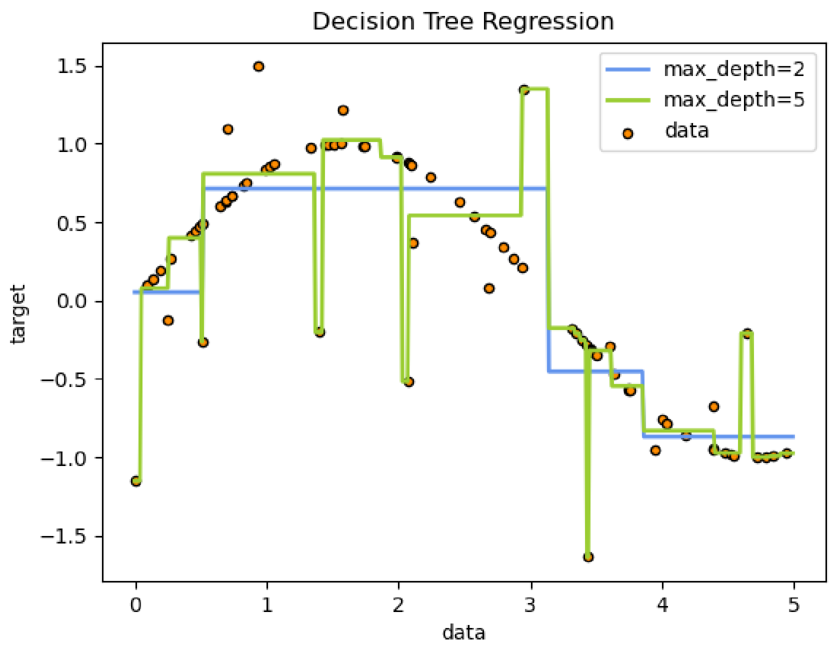
- O principal problema das ***árvores de decisão*** é que elas são ***muito sensíveis a pequenas variações nos dados de treinamento*** (e.g., rotação).
- ***Árvores de decisão*** criam ***fronteiras de decisão ortogonais*** (i.e., todas as ***fronteiras de decisão*** são perpendiculares aos eixos), o 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., CART seleciona aleatoriamente o conjunto de recursos para avaliar em cada nó) e, portanto, podem gerar ***árvores completamente diferentes*** com o mesmo modelo e conjunto de dados, mas sementes 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

```python
# 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.

```python
# 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)
```

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

```python
# 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)
```

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

```python
# Instantiate classifier.
clf = DecisionTreeClassifier(criterion='entropy')
```

Instancia classificador. Não suporta atributos categóricos, apenas contínuos

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

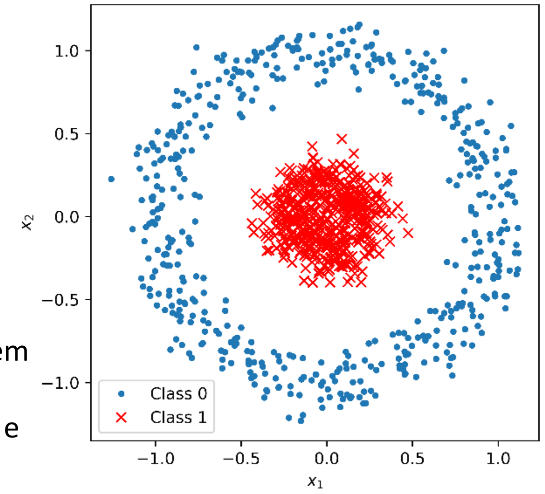
Treina o classificador.

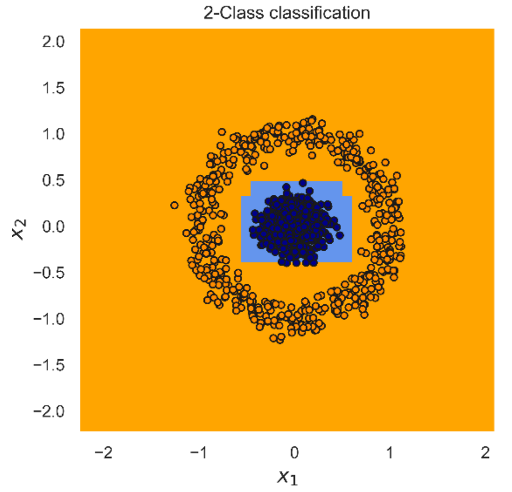
```python
# Use the trained classifier to predict labels for the test features.
y_pred = clf.predict(x_test)
```

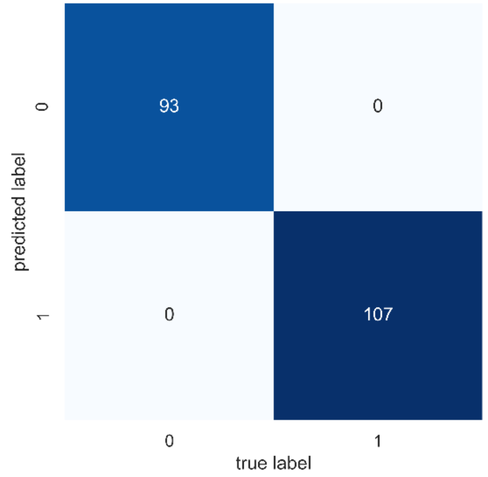
Realiza predição com conjunto de testes.

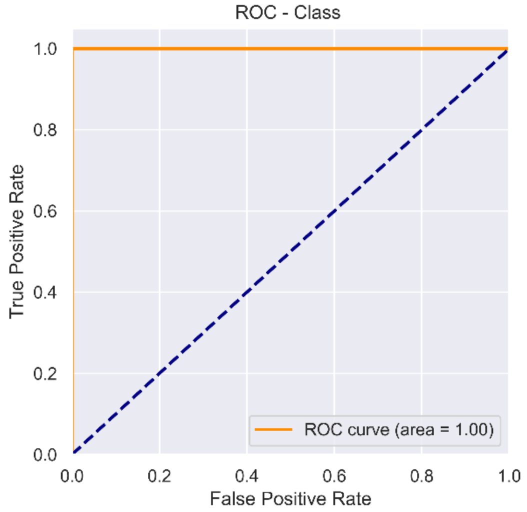
```python
# Calculate and return the accuracy on the test data
accuracy = accuracy_score(y_test, y_pred)
```

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

```python
# 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%).

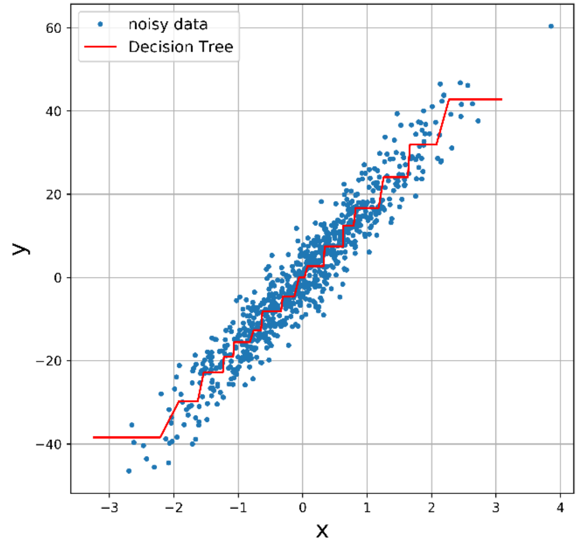
Lista de valores a serem testados.

Executa o grid search.

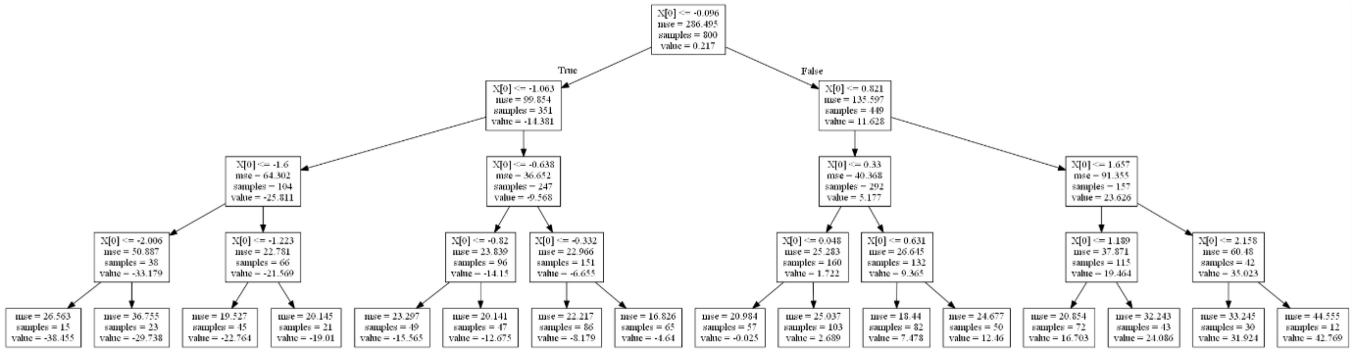
Imprime os valores ótimos dos hiperparâmetros.

Realiza predição com conjunto de testes.

Calcula a performance com conjunto de teste.



Exemplo: DTMakeRegression.ipynb

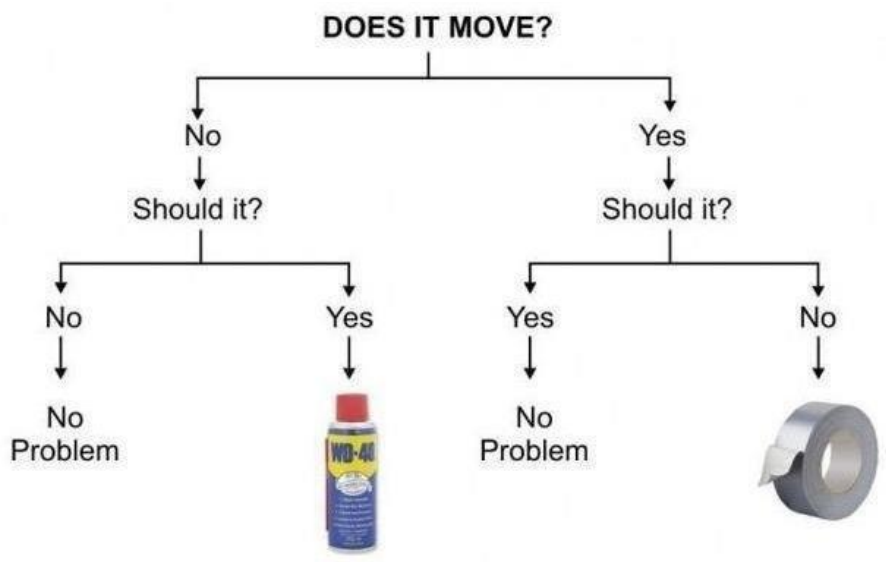


Exemplo de ***regressão*** utilizando ***GridSearch*** para encontrar os valores ótimos para os hiperparâmetros 'max_depth' e 'min_samples_leaf'. As figuras acima mostram os dados ruidosos, a curva de regressão e a árvore de decisão do regressor.

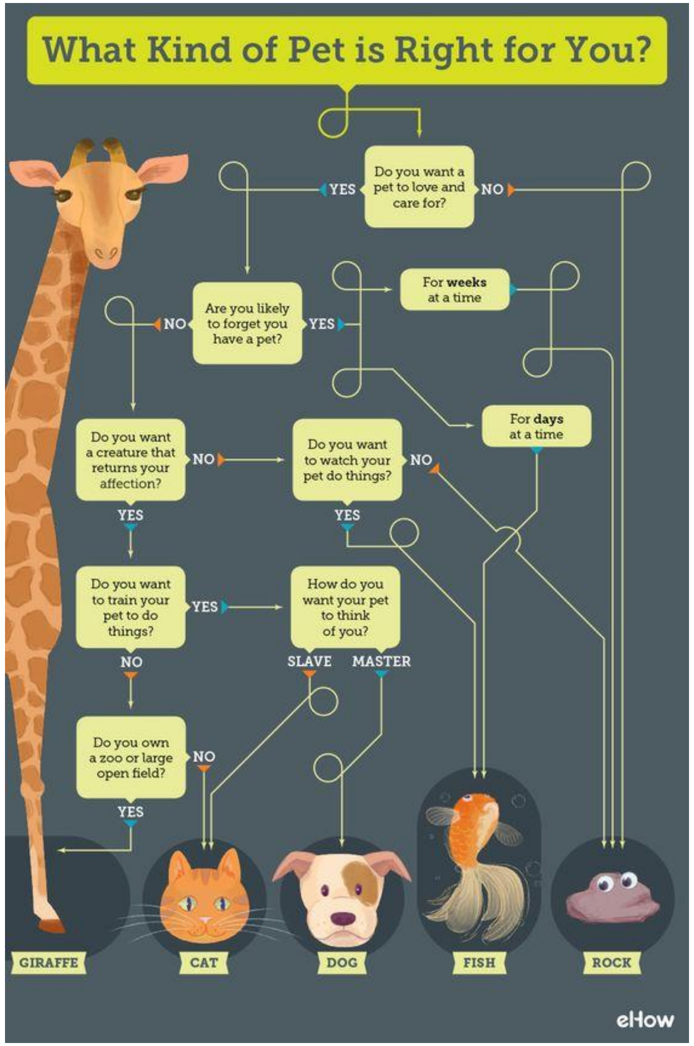
Avisos

- Estudo dirigido sobre ***florestas aleatórias*** (Lista #8) e ***k-Means*** (Lista #9) na próxima semana.
- Vídeo sobre introdução às redes neurais.
  - Pasta “Recordings” do Teams.

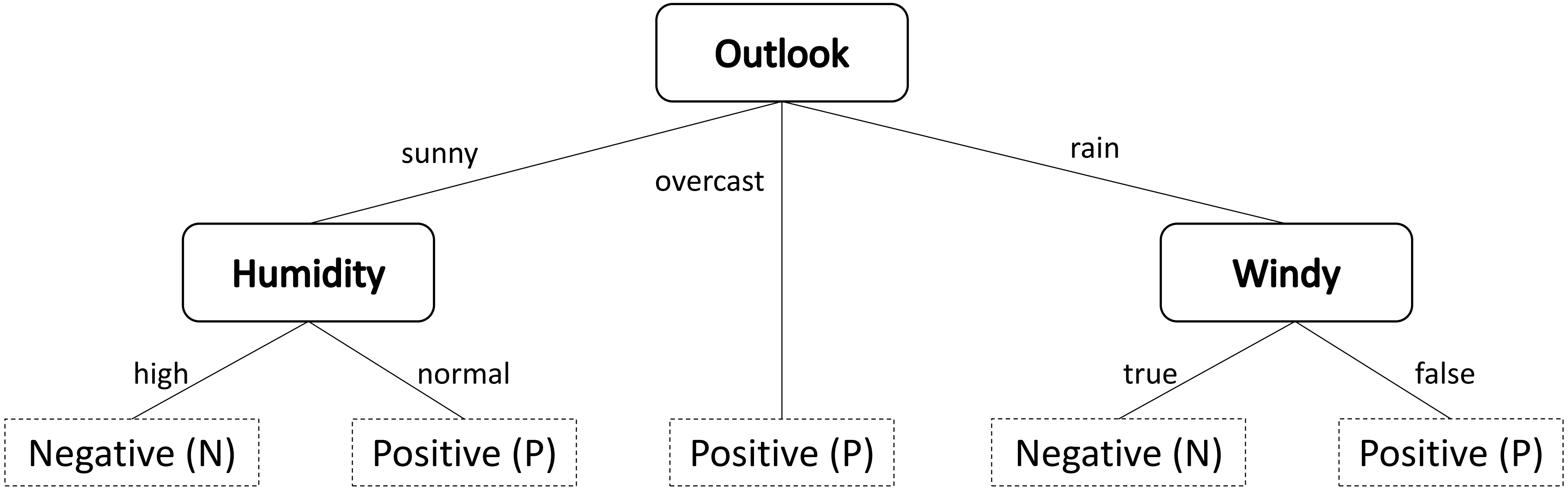
Obrigado!

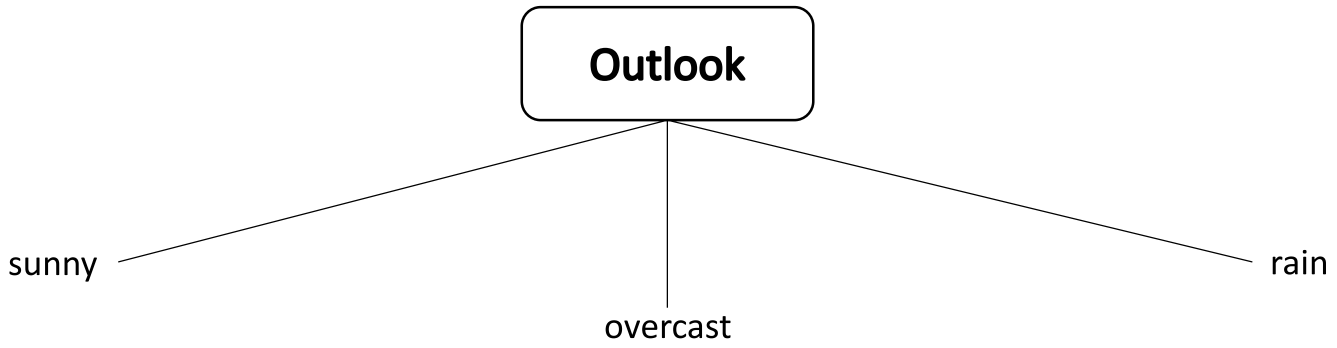

DOES IT MOVE?
No
Yes
Should it?
Should it?
No
Yes
Yes
No
No Problem
No Problem

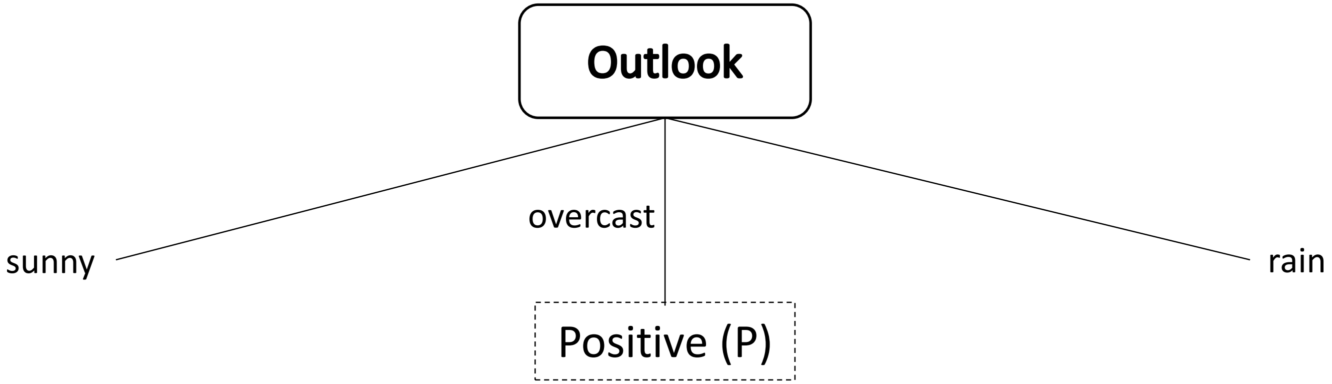

MUST I EXERCISE?
DO I WANT TO GO FOR A JOG?
YES
NO
Did I get enough sleep?
Am I just being lazy?
YES
NO
YES
NO
Am I in sound condition?
Skip it. Do it tomorrow.
Get up. Go.
If you're not sick, go for it.
YES
NO
Rest today.
Do it.

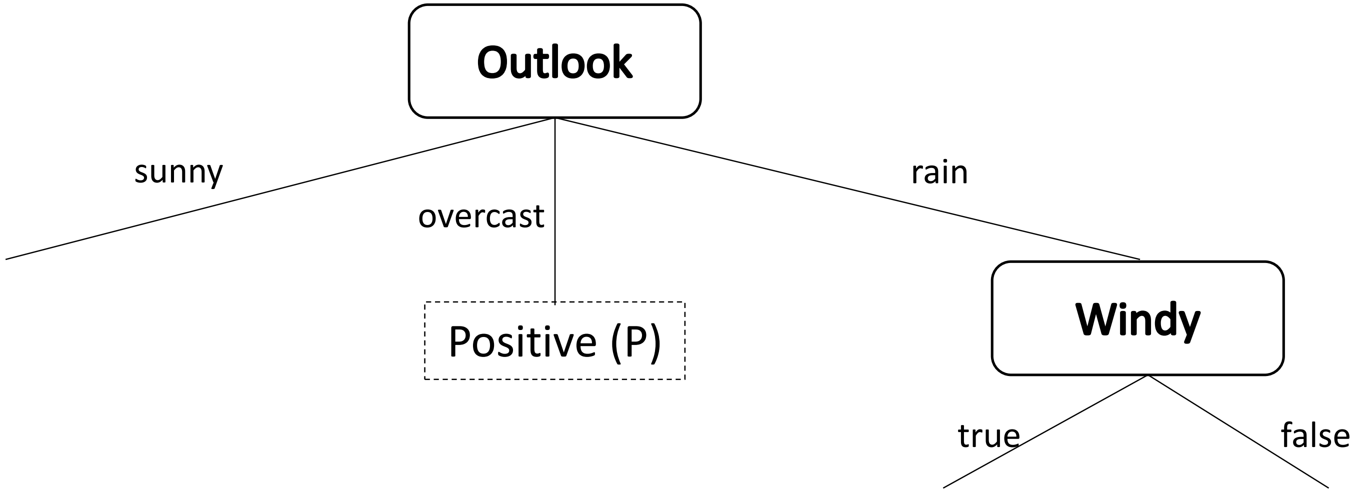

What Kind of Pet is Right for You?
Do you want a pet to love and care for?
YES
NO
Are you likely to forget you have a pet?
NO
YES
For weeks at a time
For days at a time
Do you want a creature that returns your affection?
NO
YES
Do you want to watch your pet do things?
NO
YES
Do you want to train your pet to do things?
YES
NO
How do you want your pet to think of you?
SLAVE
MASTER
Do you own a zoo or large open field?
NO
YES
GIRAFFE
CAT
DOG
FISH
ROCK
eHow

Figuras

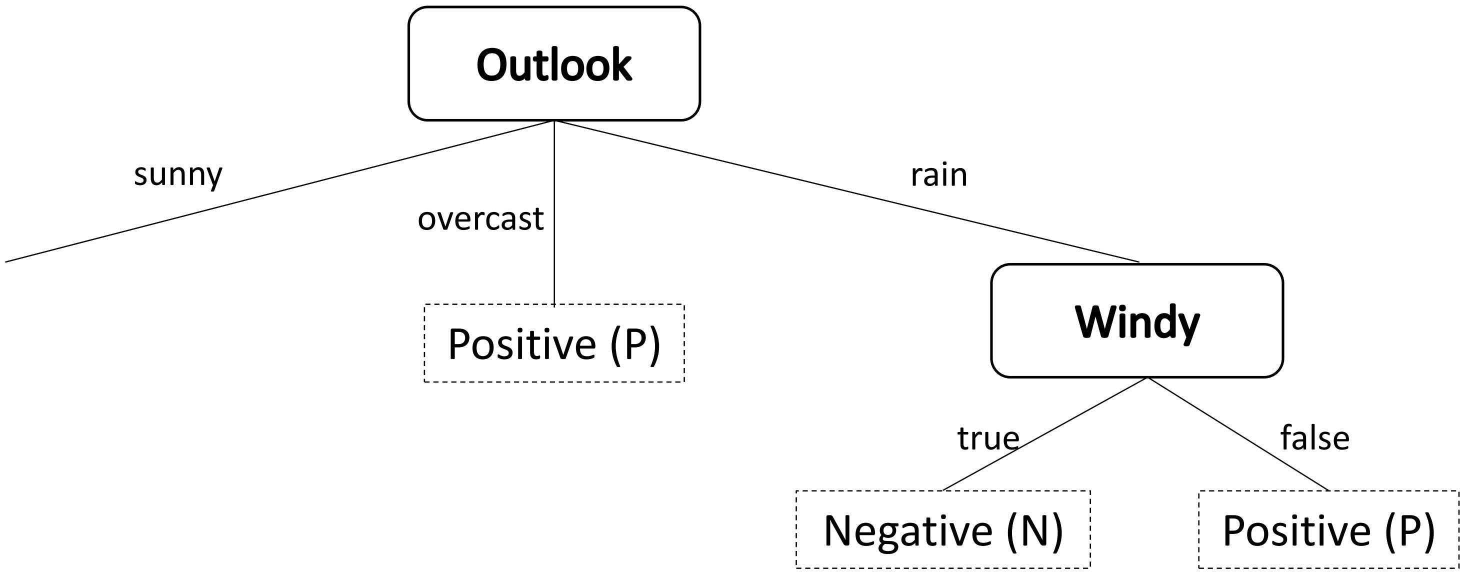

Outlook
sunny
overcast
rain
Humidity
Windy
high
normal
true
false
Negative (N)
Positive (P)
Positive (P)
Negative (N)
Positive (P)


Outlook
sunny
overcast
rain

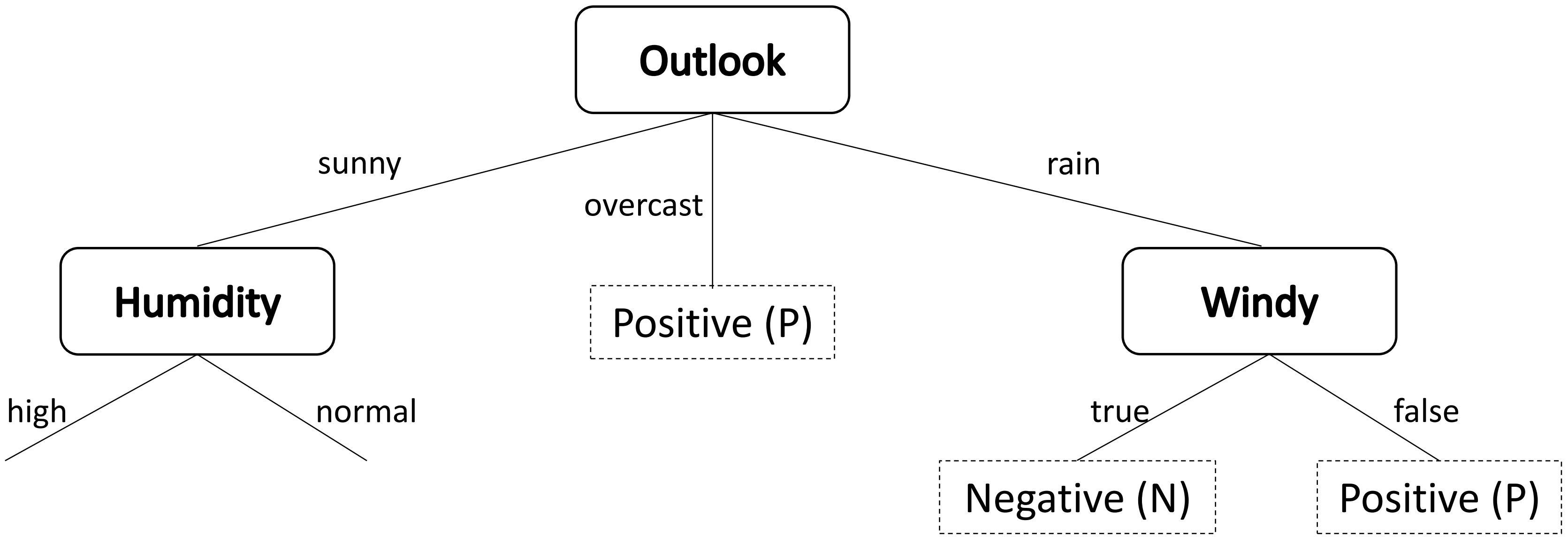

Outlook
overcast
sunny
rain
Positive (P)

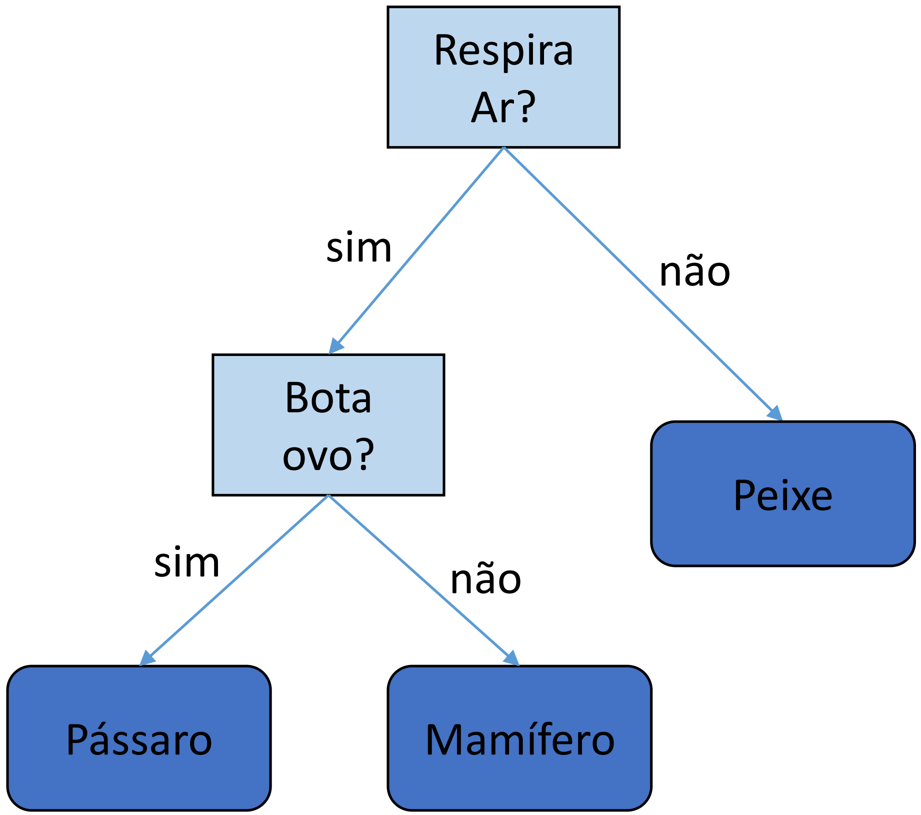


Outlook
sunny
overcast
rain
Positive (P)
Windy
true
false



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


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


Respira Ar?
sim
não
Bota ovo?
Peixe
sim
não
Pássaro
Mamífero

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

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

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

- 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

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

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

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

Cálculo da entropia para mais de duas classes

Obs.:

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

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

$$H(x1 = 0) = -\sum_{i=0}^{2} p(y = i|x1 = 0)\log_2\big(p(y = i|x1 = 0)\big)$$

$$= -\big[p(y = 0|x1 = 0)\log_2\big(p(y = 0|x1 = 0)\big) + p(y = 1|x1 = 0)\log_2\big(p(y = 1|x1 = 0)\big) +$$

		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