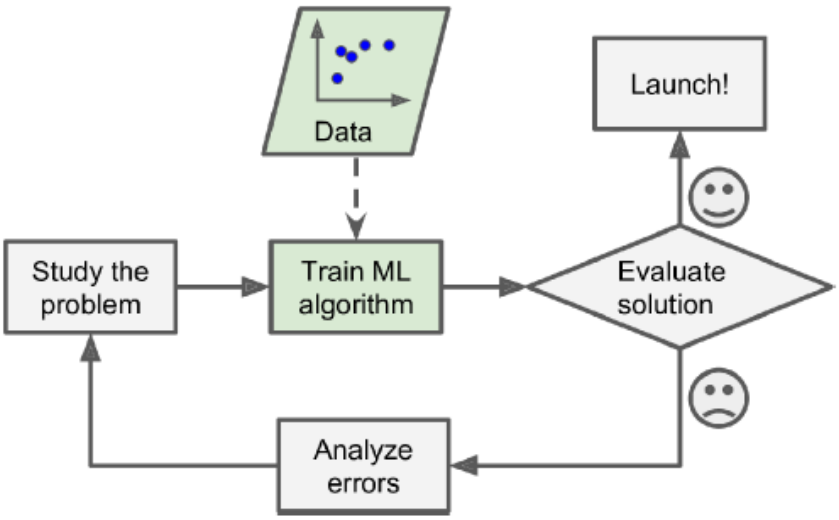
Prática em Pesquisa

Introdução

Machine Learning (aprendizado de máquinas): campo de estudo que dá aos computadores a habilidade de aprenderem sem terem que ser explicitamente programados. Ou seja, dá aos computadores a capacidade de aprender a partir da experiência.

- Conjunto de treinamento: amostra de dados utilizada o aprendizado do computador.
- Acurácia: medida de desempenho utilizada em tarefas de classificação.



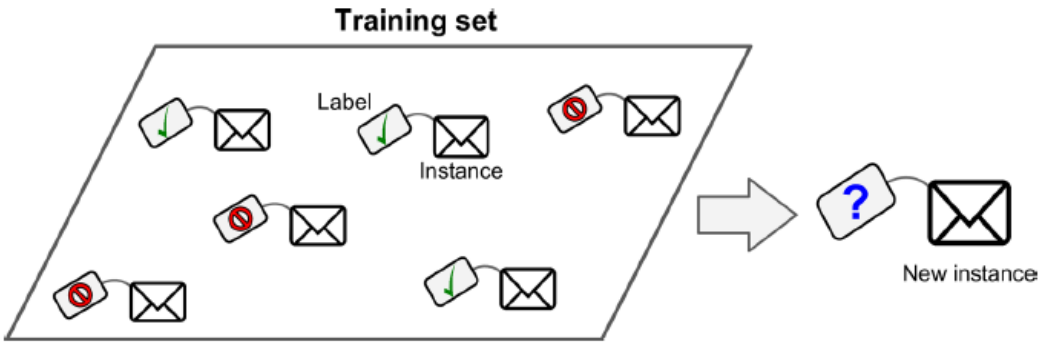
Tipos de sistemas de aprendizado de máquinas:

- Aprendizado supervisionado, não supervisionado, semi-supervisionada e por reforço: podem ou não ser treinados com supervisão humana.
- Aprendizado online *versus* em batch (offline): podem ou não aprender de forma incremental na hora (momento).
- Aprendizado baseado em instância *versus* baseado no modelo: trabalham comparando novos pontos de dados com pontos de dados conhecidos ou

detectam padrões nos dados de treinamento e constrõem um modelo preditivo.

Obs.: os 3 tipos acima não são excludentes, podendo ser combinados.

Aprendizado Supervisionado



- O conjunto de treinamento inclui as soluções desejadas (rótulos).
- Tarefa típica: classificação (previsão de classes)

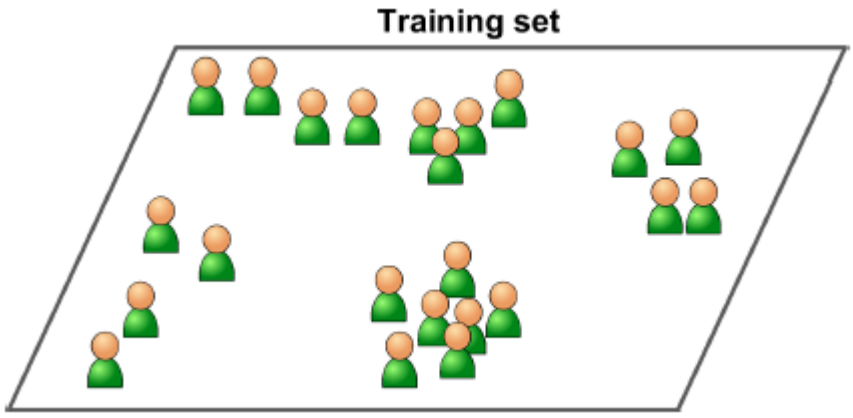
  Filtro de spam (email): é trienado com muitos emails de exemplo junto com suas classes (spam ou não spam). Com isso, o filtro deve aprender como classificar novos emails.

  Exemplo: Regressão Logística (utilizada para prever a probabilidade de que a variável alvo pertença a uma determinada classe).
- Tarefa típica: regressão (previsão de valores)

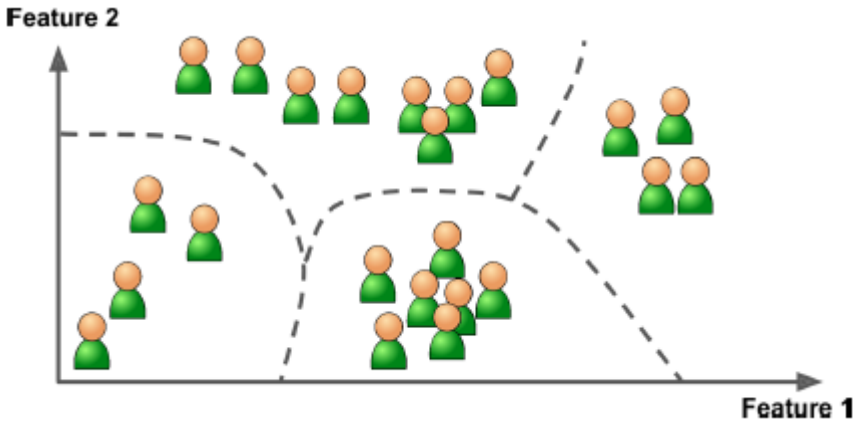
  Previsão de um valor numérico alvo: para o treinamento do sistema é preciso ter um conjunto de exemplos grande, incluindo as variáveis preditoras e os rótulos.

  Exemplo: Regressão Linear (utilizada para prever o valor dado um conjunto de features de input).
- Principais algoritmos: k-Nearest Neighbors, Regressão Linear, Regressão Logística, Support Vector Machine, Decision Tree, Random Forest, Neural Networks.

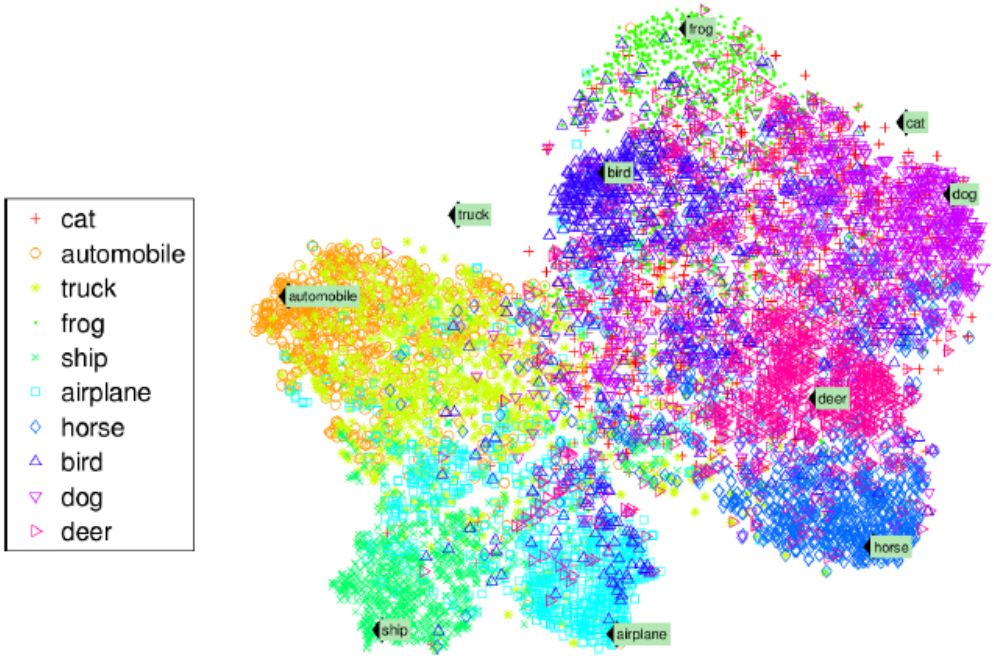
Aprendizado não supervisionado



- Os dados de treinamento não são rotulados.
- O sistema tenta aprender sem a intervenção humana.
- Principais tipos de algoritmos: Clusterização, Detecção de Anomalia ou Detecção de Novidade, Visualização e Redução de Dimensionalidade, Aprendizado de Regras de Associação.
- Clusterização:

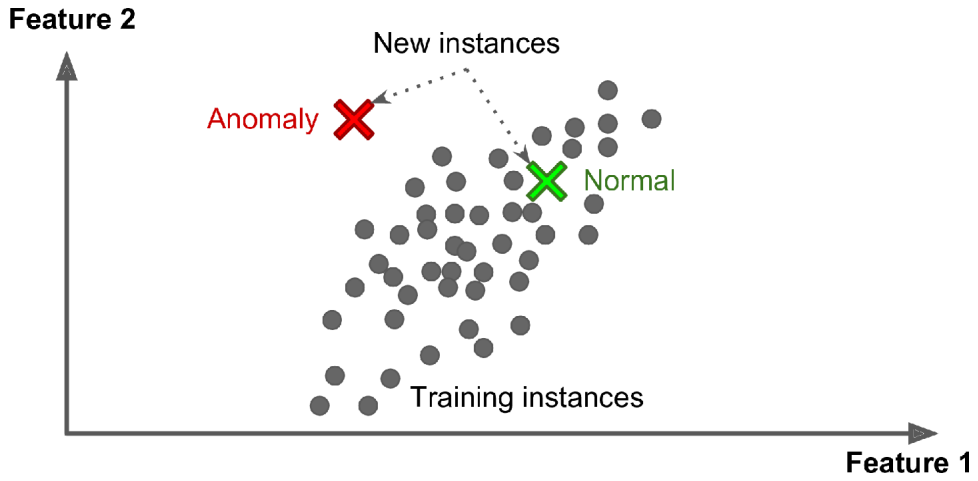


  - Exemplo de algoritmo: K-Means
  - É utilizado um algoritmo de clusterização para tentar detectar grupos de elementos similares.
  - É possível utilizar um algoritmo de hierarquização de clusterização para dividir cada grupo em grupos menores.
- Visualização:



- Exemplo de algoritmo: t-Distributed Stochastic Neighbor Embedding (t-SNE)
- A partir de um grande conjunto de dados complexos e não rotulados, o algoritmo gera uma representação em 2D ou 3D.
- Tenta preservar ao máximo a estrutura para a compreensão de como os dados estão organizados e, por conseguinte, identificar padrões não suspeitos.
- Tenta evitar que os clusters de separados no espaço se sobreponham na visualização.

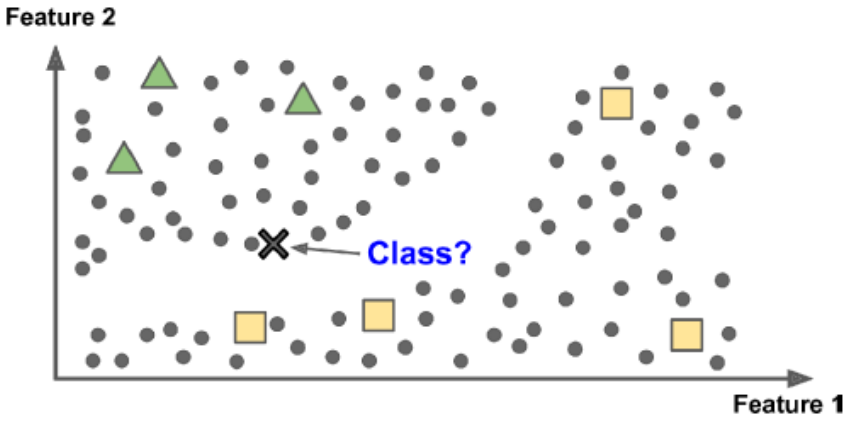
- Redução de Dimensionalidade
  - Exemplo de algoritmo: Principal Components Analysis (PCA).
  - O objetivo é simplificar os dados sem perda de informação.
- Detecção de Anomalias



- Ex.: detecção de transações não usuais no cartão de crédito para prevenção de fraudes.
- Durante a etapa de treinamento o sistema aprende e reconhece as instâncias mais normais. A partir daí, ao ver uma nova instância ele tenta dizer se se assemelha à instância normal ou se é uma anomalia.

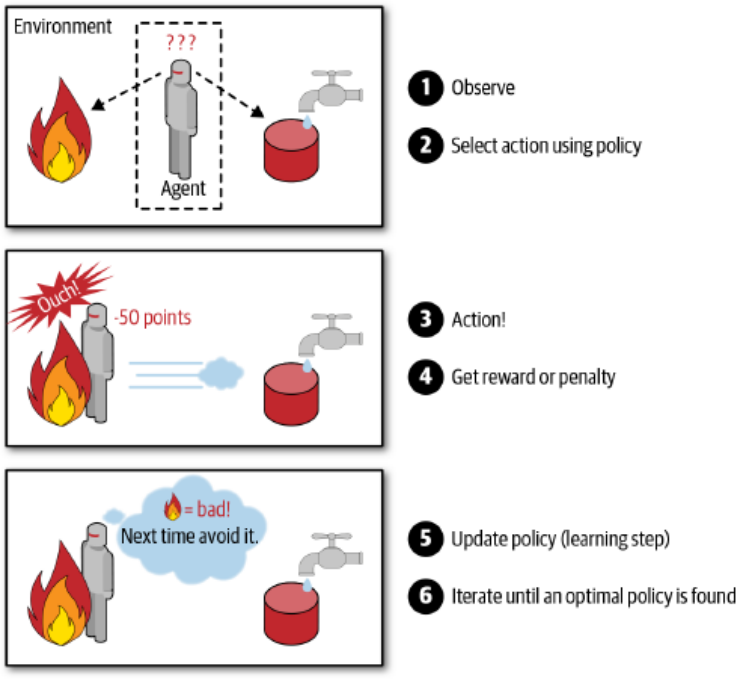
- Detecção de Novidade:
  - É utilizado para detectar instâncias novas que são diferentes das instâncias do conjunto de treinamento.
  - Necessita que o conjunto de treinamento esteja "limpo", desprovido de qualquer instância a ser detectada pelo algoritmo.
- Regra de Associação:
  - O objetivo é escavar grande quantidade de dados a fim de descobrir relações de interesse entre os atributos.
  - Ex.: detecção de relações entre itens comprados no supermercado.

Aprendizado semi-supervisionado



- O conjunto de treinamento possui tanto dados rotulados quanto dados não rotulados.
- Os exemplos não rotulados podem ajudar a classificar novas instâncias.

Aprendizado por reforço



- O sistema de aprendizado por observar o ambiente, selecionar e executar ações e obter recompensas positivas ou negativas.
- O sistema aprende por si mesmo qual é a melhor estratégia a fim de obter a melhor recompensa ao longo do tempo.

Aprendizado em Batch (ou offline)

- O sistema é incapaz de aprender incrementalmente.
- O sistema é treinado utilizando todos os dados disponíveis.
- Primeiro o sistema é treinado e depois é posto em produção, rodando sem qualquer aprendizado (aplica o que já foi aprendido).
- Se utilizar o sistema para aprender sobre novos dados será necessário treinar uma nova versão do sistema do início em relação ao dataset completo e, posteriormente, substituir o sistema anterior pelo novo.

Aprendizado Online

- O sistema é treinado incrementalmente, alimentado com instâncias de dados sequencialmente ou individualmente, ou em pequenos grupos (mini-batches).
- É ideal para sistemas que recebem dados em um fluxo contínuo e necessitam se adaptar a mudanças de forma rápida ou autônoma.
- Taxa de aprendizado: parâmetro utilizado nos sistemas de aprendizado online para analisar quão rápido eles devem se adaptar à mudança nos dados. Assim, uma taxa de aprendizado alta indica que o sistema irá se adaptar rapidamente ao novo dado, mas também tenderá esquecer rapidamente o dado antigo.

  Por outro lado, taxa de aprendizado baixo mostra que o sistema terá maior inércia (irá aprender mais devagar), mas será menos sensível a ruídos ou outliers.

Aprendizado baseado em instâncias

- O sistema aprende com os exemplos e, com isso, procura generalizar os novos casos utilizando uma medida de similaridade para comparar estes novos casos com os exemplos aprendidos.

Aprendizado baseado em modelo

- O sistema generaliza o conjunto de exemplo por meio da construção de um modelo, utilizando-o para fazer previsões.

Separação das amostras

- A forma de ver o quão bem um modelo irá generalizar para novos casos é verificar seu desempenho para novos casos (outro conjunto de dados).
- Recomenda-se separar a amostra de dados em conjunto de treino (in-sample) e de teste (out-of-sample).

  Conjunto de treino: contém dados utilizados para treinar o modelo (geralmente, 70% dos dataframe).

  Conjunto de teste: contém dados utilizados para testar a capacidade de generalização do modelo (geralmente, 30% do dataframe).

Observação: tanto no R quanto no Python, toda vez que o programa é executado novamente é gerado um conjunto de teste diferente. Uma opção é construir um gerador de números aleatórios por meio do comando `set.seed( )` (R). Isso garante que sempre serão gerados os mesmos números aleatórios (se a análise for repetida no futuro um resultado idêntico seja obtido).

Não há uma regra para o valor ⇒ há trabalhos que utilizam `set.seed(42)`, `set.seed(123)`, `set.seed(12345)`, `set.seed(2345)`, `set.seed(300)`, `set.seed(333)`, `set.seed(1303)` etc.

Segundo James et al. (2013), o argumento entre parênteses é um argumento inteiro arbitrário.

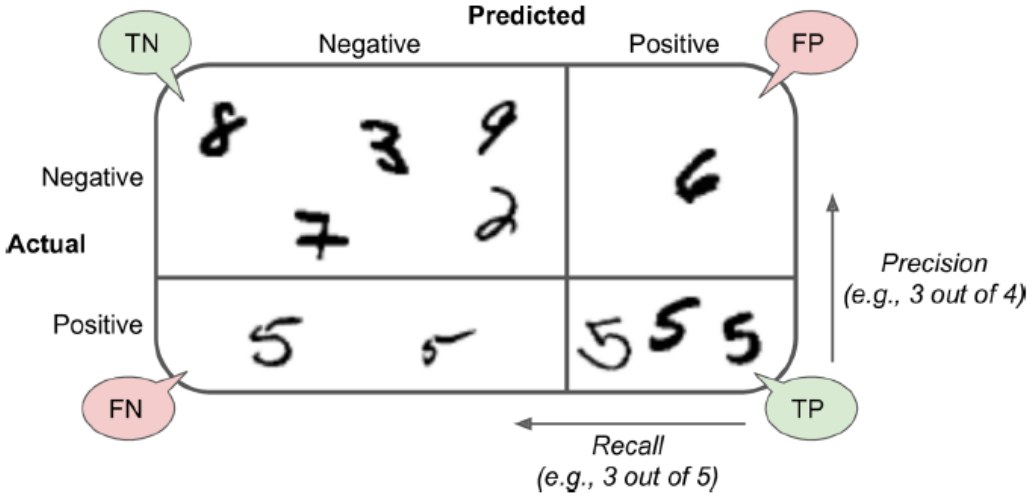
Segundo Géron (2019), geralmente as pessoas utilizam `set.seed(42)`.

Medidas de performance dos modelos de aprendizado de máquinas

As medidas utilizadas para avaliar um classificador diferem das utilizadas para avaliar um regressor.

1. Matriz de Confusão

```
bvsppred_directionin    0    1
                    0 1388  137
                    1  134 1822
```



- Cada linha representa uma classe real e cada coluna representa uma classe predita.

  0: classe negativa

  1: classe positiva

- Diagonal principal: células $a_{1,1}$ representa verdadeiro negativo e $a_{2,2}$ representa verdadeiro positivo ⇒ precisão.
- Diagonal não principal: células $a_{1,2}$ representa falso positivo e $a_{2,1}$ representa falso negativo.
- Um classificador perfeito deveria ter somente verdadeiros positivos e verdadeiros negativos.

Precisão do classificador (*precision*): medida de acurácia das predições positivas.

$$precision = \frac{TP}{TP + FP}$$

onde TP é o número de verdadeiros positivos e FP é o número de falsos positivos.

Recall (ou *sensitivy* ou *taxa de verdadeiro positivo*): razão entre instâncias positivas que foram corretamente detectadas pelo classificador

$$recall = \frac{TP}{TP + FN}$$

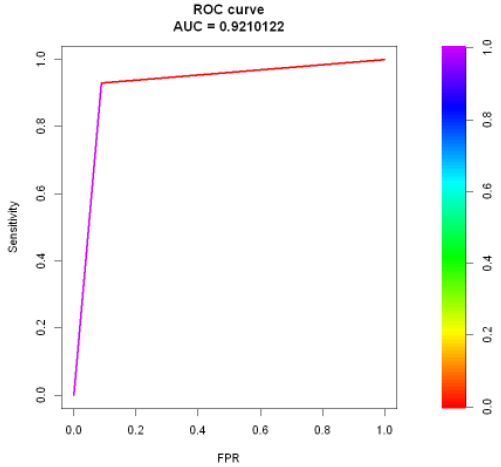
onde FN é o número de falsos negativos.

F1 score: média harmônica das medidas precision e recall.

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Obs.: o classificador somente terá um F1 alto se *precision* e *recall* forem altos (ambos).

2. Curva ROC



- Gráfico da taxa de verdadeiro positivo (*recall*) contra a taxa de verdadeiro negativo (*specificity*).

  Specificity: razão de instâncias negativa que foram corretamente classificadas como negativas.

- A linha em 45º (diagonal) representa a curva ROC de um classificador puramente aleatório. Um bom classificador se distancia o máximo possível desta linha (em direção ao canto superior esquerdo).

- AUC (área sob a curva): medida para comparar classificadores. Assim, um classificador perfeito possui $AUC = 1$, enquanto que um classificador puramente aleatório possui $AUC = 0.5$.

Modelos de treinamento

1. Regressão Logística

Apresentação:

- É um classificador binário.
- É comumente utilizada para estimar a probabilidade de que uma instância pertença a uma classe particular. Assim, se a probabilidade estimada for superior a 0.5, então o modelo prevê que a instância pertence a esta classe (classe positiva ou "1"), caso contrário prevê que não pertence (classe negativa ou "0").

Funcionamento:

- A regressão logística calcula a soma ponderada das features de entrada, tendo como output uma função logística.

$$\hat{p} = h_\theta\left(\mathrm{X}\right) = g\left(\theta^T \mathrm{X}\right)$$

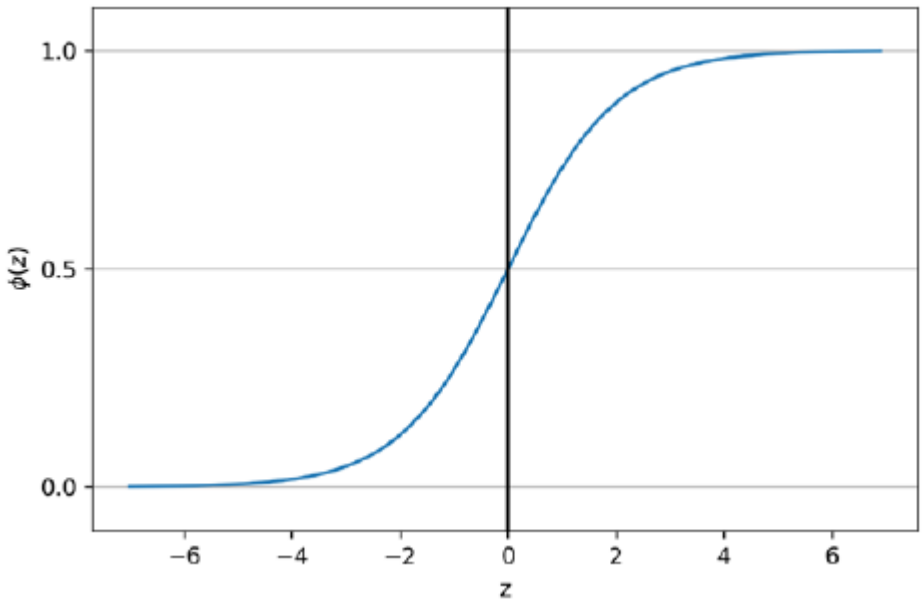
onde $\hat{p} = h_\theta\left(\mathrm{X}\right)$ é a probabilidade estimada de que uma instância pertença a uma determinada classe e h_θ é a função de hipótese usando os parâmetros do modelo θ. X é o vetor de features da instância, contendo x_0 até x_n, com $x_0 = 1$. $\theta^T = [\theta_0, \theta_1, ..., \theta_n]^T$ vetor de parâmetros (pesos das features), contendo o termo de viés θ_0 .

Obs.: $\theta^T \mathrm{X}$ equivale a $\theta_0 + \theta_1 x_1 + ... + \theta_n x_n$.

- A função logística $g(.)$ é uma função sigmoide que transforma um número real em um valor entre $[0, 1]$.

$$g(z) = \frac{1}{1 + \exp\left(-z\right)}$$

onde $z = \theta^T \mathrm{X}$



- Uma vez que o modelo de regressão logística estima a probabilidade $\hat{p} = h_\theta\left(\mathrm{X}\right)$ de que a uma instância pertença à classe positiva, então a predição do modelo pode ser representada pela equação abaixo:

$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases}$$

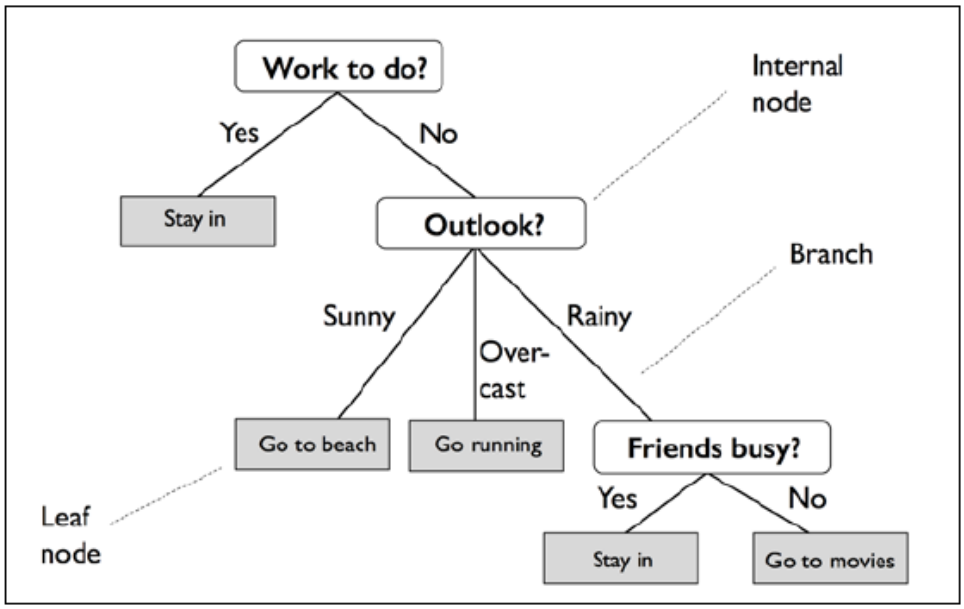
- Além disso, $g(z) < 0.5$ quando $z < 0$ e $g(z) \geq 0$ quando $z \geq 0$. Assim, o modelo de regressão logística prevê 1 se $\theta^T \mathrm{X}$ é positivo e 0 se $\theta^T \mathrm{X}$ é negativo.

O objetivo de treinar o modelo de regressão logística é obter o vetor de parâmetros θ de modo a obter probabilidades altas para instâncias positivas ($y = 1$) e probabilidades baixas para instâncias negativas ($y = 0$). Em outras palavras, busca minimizar

2. Decision Tree

Apresentação:

- Algoritmo de aprendizado supervisionado que constrói uma árvore de classificação ou de regressão.
- O modelo pode ser compreendido como divisão dos dados para tomada de decisões com base em uma série de perguntas.
- É um conjunto de regras que envolvem estratificação ou segmentação do espaço de predição em regiões mais simples.
- Com base nas *features* no conjunto de treinamento, o modelo aprende uma série de questões para inferir os rótulos das classes das amostras.



Funcionamento:

- É uma estrutura formada por um conjunto de nós de decisão (perguntas) que permitem a classificação de cada caso. Consiste em uma hierarquia de testes a algumas das variáveis envolvidas no problema de decisão.
- O processo tem como ponto de partida o nó raiz da árvore, onde é realizada uma pergunta acerca do atributo.
- Cada folha da árvore representa uma estimativa de atributos e cada nó separa os dados de acordo com uma condição das features.
- Há 2 tipos de nós:

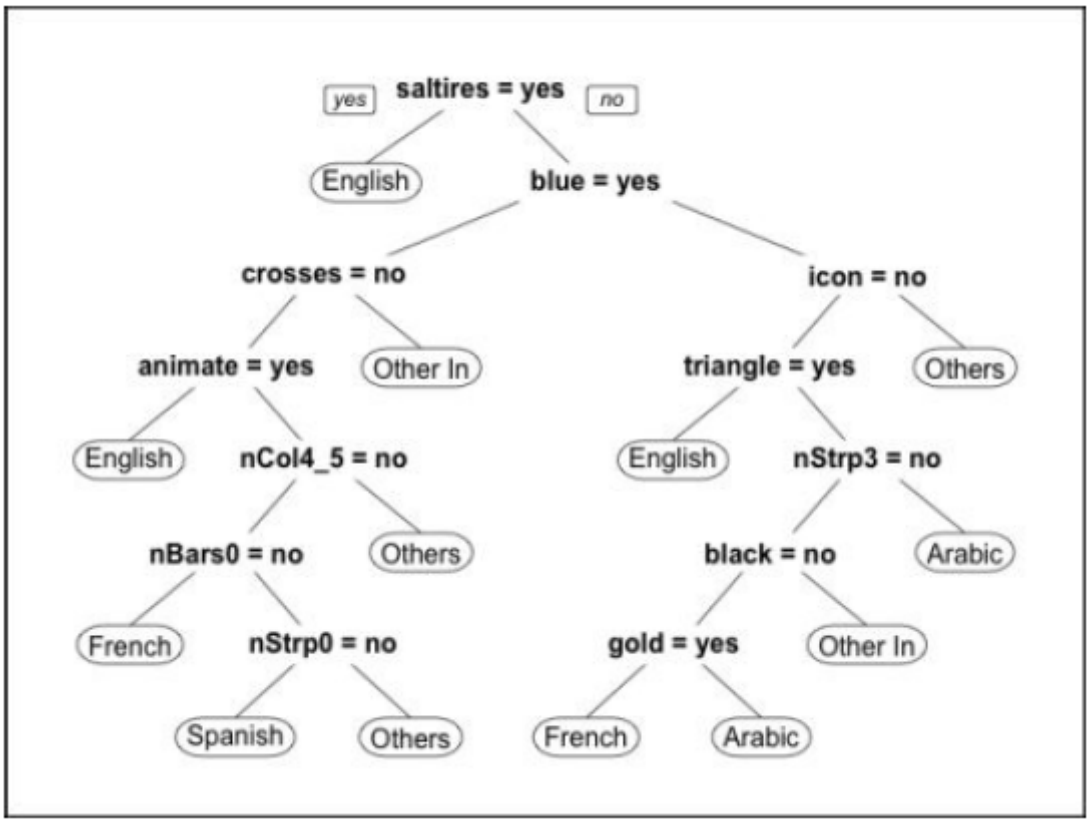
  Nós de decisão: possuem sucessores e são utilizados para classificar um novo registro.

Nós terminais: não possuem sucessores.

- O objetivo do algoritmo é definir a feature mais relevante e dividir o conjunto em 2 grupos de acordo com esta feature. Portanto, para cada grupo, o algoritmo identifica a feature mais relevante e divide os objetos dos grupos em 2 partes.

  A separação dos nós com os atributos mais informativos é realizada visando a maximização do ganho de informação de cada divisão.

  Este procedimento se repete até a identificação da folha como pequenos grupos de objetos (ou seja, até as folhas serem puras).



O algoritmo pode lidar com features categóricas e/ou numéricas.

- Regression Tree: utilizado para prever uma resposta quantitativa.

  A resposta predita para uma dada observação é dada pela resposta média das observações de treinamento, que pertencem ao mesmo nó terminal.

- Classification Tree: utilizado para prever uma resposta qualitativa.

É prevista que cada observação pertença à classe da observação de treinamento que ocorre com mais frequência na região a que pertence.

Na interpretação dos resultados, geralmente o interesse reside não somente na predição da classe correspondente a uma região do nó terminal, mas também nas proporções de classes entre as observações de treinamento que caem nesta região.

Crescimento da árvore de decisão:

- Há similaridade entre a classificação e a regressão.

  A tarefa de crescer a árvore de decisão para classificação é similar à tarefa para regressão.

- A separação dos nós ocorre via divisão binária recursiva.

  Contudo, na classificação o critério para divisão binária é diferente. Uma alternativa natural é a taxa de erro de classificação.

  Uma vez que o objetivo é avaliar uma observação em uma dada região para a classe que ocorre com mais frequência das observações de treinamento, a taxa de erro de classificação é a fração das observações de treinamento que não pertencem à classe mais frequente.

$$E = 1 - \max_k(\hat{p}_{mk})$$

  onde $\hat{p}_{mk}$ é a proporção das observações de treinamento na m-ésima região pertencente à k-ésima classe.

  Contudo, a taxa de erro de classificação não é sensível, o suficiente, para o crescimento de árvores.

- Quando uma classificação é construída as medidas utilizadas para avaliar a qualidade de uma divisão é possível utilizar uma das seguintes medidas (que são mais sensíveis à pureza do nó do que a taxa de erro de classificação).

  Índice de Gini: medida da variância total entre as K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

O índice de Gini toma valores pequenos se todos os $\hat{p}_{mk}$ estão próximos de zero ou de 1.

O índice de Gini se refere a uma medida de pureza, onde um valor pequeno indica que um nó contém predominantemente observações de uma única classe.

Cross-entropy: medida da variância total entre as K classes.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

Uma vez que $0 \leq \hat{p}_{mk} \leq 1$, $0 \leq -\hat{p}mk \log(\hat{p}_{mk})$

O *cross-entropy* toma valores pequenos (próximos de zero) se todos os $\hat{p}_{mk}$ estão próximos de zero ou de 1.

Similar ao índice de Gini, *cross-entropy* tomará um valor pequeno de o m-ésimo nó é puro.

3. Random Forest

Apresentação:

- É um monte de árvores de decisão (uma maneira de melhorar o desempenho das árvores de decisão).
- Desfruta das mesmas vantagens de uma árvore de decisão.
- Propõe criar várias pequenas árvores de decisão.

  Apesar de individualmente as árvores de decisão apresentarem um desempenho inferior, no conjunto apresentam um desempenho robusto.
- A previsão final é um resumo desses palpites, geralmente a média entre elas.
- Trabalha agregando as predições realizadas pelas diversas árvores de decisão (de profundidade variável)
- A ideia é treinar várias árvores de decisão (descorrelacionadas), obtidas a partir de amostras do dataset, e fazer predições utilizando os resultados

que mais aparecem em caso de um problema de classificação, ou a média dos valores obtidos em caso de regressão.

Funcionamento

- O algoritmo começa construindo árvores semelhantes à maneira como um algoritmo de árvore de decisão normal funciona.
- Toda vez que uma divisão tem que ser feita, ela usa apenas um pequeno subconjunto aleatório de recursos para fazer a divisão em vez do conjunto completo de recursos.

  O algoritmo introduz aleatoriedade extra durante o crescimento da árvore. Ao invés de buscar pela melhor *feature* quando ocorre a divisão de um nó, ele procura a melhor *feature* entre um subconjunto aleatório de *features*.

  O resultado é uma árvore com mais diversidade, que troca um viés elevado por uma menor variância, gerando um modelo geral melhor.
- Ele constrói várias árvores usando o mesmo processo, e então leva a média de todas as árvores para chegar ao modelo final. Isso funciona reduzindo a quantidade de correlação entre as árvores e, assim, ajudando a reduzir a variância da árvore final.

Bagging

- Método que permite obtenção de um conjunto diversos de classificadores.
- Utiliza o mesmo algoritmo de treinamento para cada preditor e treina-o sobre diferentes subconjuntos aleatórios do conjunto de treinamento.
- Ocorre quando a amostragem é realizada com substituição.
- Permite que as instâncias sejam treinadas diversas vezes para o mesmo preditor.

A porção de amostras que foram deixadas durante a construção de cada árvore de decisão é referenciada como dataset *out-of-bag* (OOB). O modelo avalia automaticamente o seu desempenho rodando cada uma das amostras no dataset OOB.

Bootstrapping:

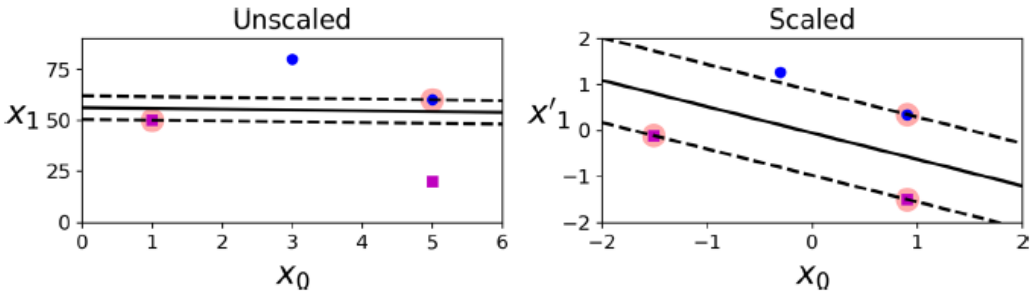
- Se várias árvores forem criadas e treinadas no mesmo dataset, suas predições serão idênticas. A alternativa mais imediata seria dividir o dataset em várias partes, uma para cada árvore.
- **Bootstrapping:** Em estatística, a partir dessa população reconstruída (amostra), é possível criar novas amostras e, dessa forma, estudar as propriedades das amostras em relação à população. As novas amostras são chamadas de bootstrap samples (amostras de bootstrap).
- O processo de criar novas amostras a partir da população reconstruída é equivalente a pegar elementos aleatórios da amostra inicial com reposição. Ou seja, escolhemos um elemento da amostra inicial, anotamos ele e colocamos ele de volta na amostra inicial antes de escolher o próximo elemento.

Support Vector Machine

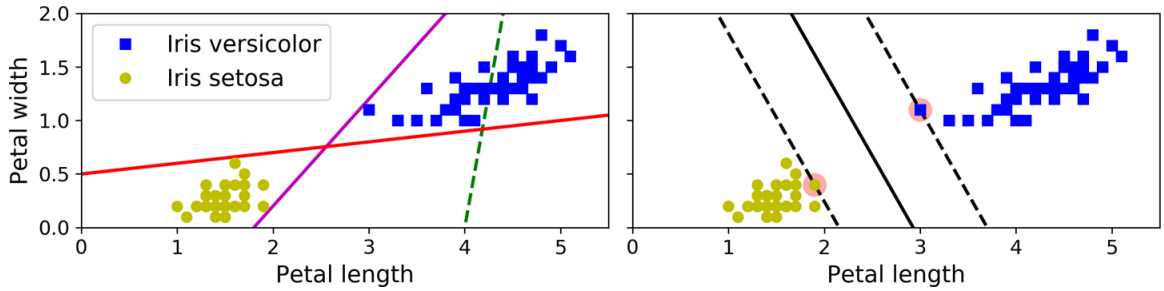
Introdução

- Algoritmo que pode ser utilizado tanto para tarefas de classificação como de regressão.
- É robusto para outliers.
- É sensível à escala das *features*.

  Obs.: no gráfico à esquerda as variáveis não estão padronizadas e no gráfico à esquerda estão.



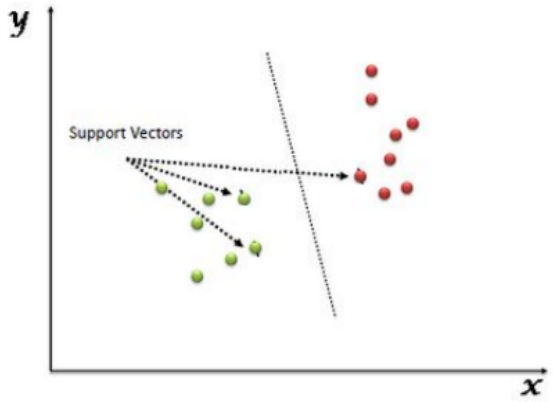
- Ideia: o limite de decisão do classificador, além de separar as classes, também fica o mais distante possível das instâncias de treinamento mais próximas.



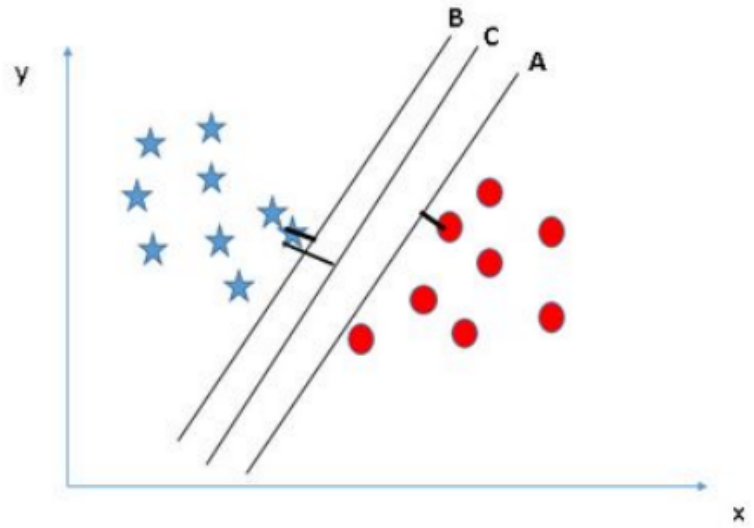
- Se as classes não forem linearmente separáveis é possível utilizar uma abordagem não linear (por meio de *features polinominais*).

Funcionamento

- Cada item de dados é plotado como um ponto no espaço n-dimensional (n = features), com o valor de cada feature sendo o valor de uma coordenada.
- A classificação é realizada encontrando o hiperplano que melhor diferencia as 2 classes (classes lienarmente separáveis).



- Constrói hiperplanos em um espaço multidimensional que separa os casos de diferentes rótulos e classifica os dados encontrando o melhor hiperplanos que separa todos os pontos de dados de uma classe daqueles de outra classe (separa as observações de acordo com seus rótulos de classe.
- O melhor hiperplano é aquele com a maior margem entre as 2 classes.
- Vetores de suporte: são as coordenadas da observação individual.
- Support Vector Machine: fronteira que melhor segrega as classes.
- Margem: distância entre o ponto de dados mais próximo (de qualquer classe) e o hiperplano.



- O SVM seleciona o hiperplano que classifica as classes com precisão antes de maximizar a margem.

Observação: o SVM possui um recurso para ignorar valores discrepantes e encontrar o hiperplano que tem margem máxima ⇒ é robusto para outliers.

Créditos

Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, 2019.

Sebastian Raschka, Python Machine Learning, 3rd ed, Packt Publishing, 2019.