Modelos de Classificação

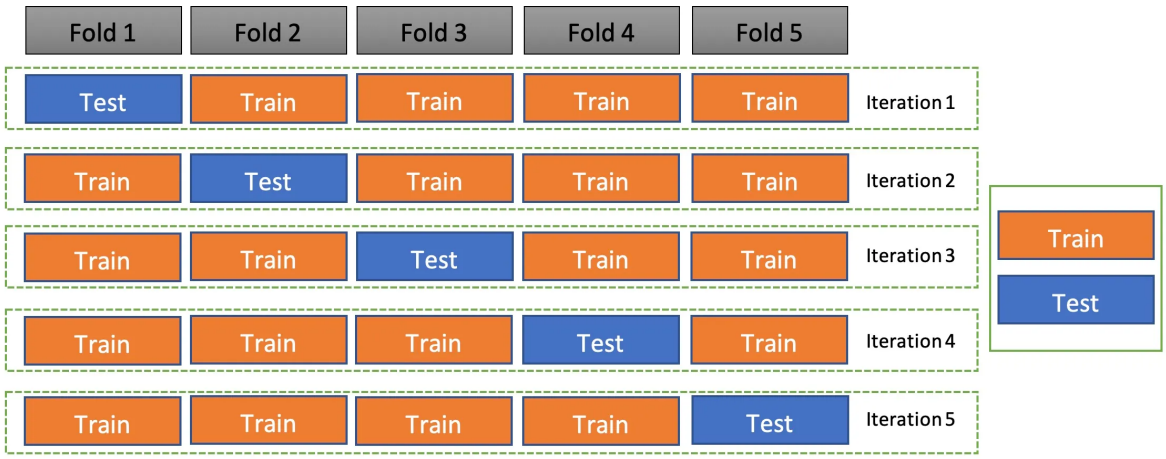
Ressampling e cross validation

Divisão entre dados de treino e teste

- Parte dos dados serão utilizados para treino e parte para teste
- Mais utilizados 70/30 ou 80/20
- Método mais veloz
- Possibilidade de alta variância
- .fit para treinar o modelo
- .score para testar o modelo

Cross validation

- Divide os dados em k folds

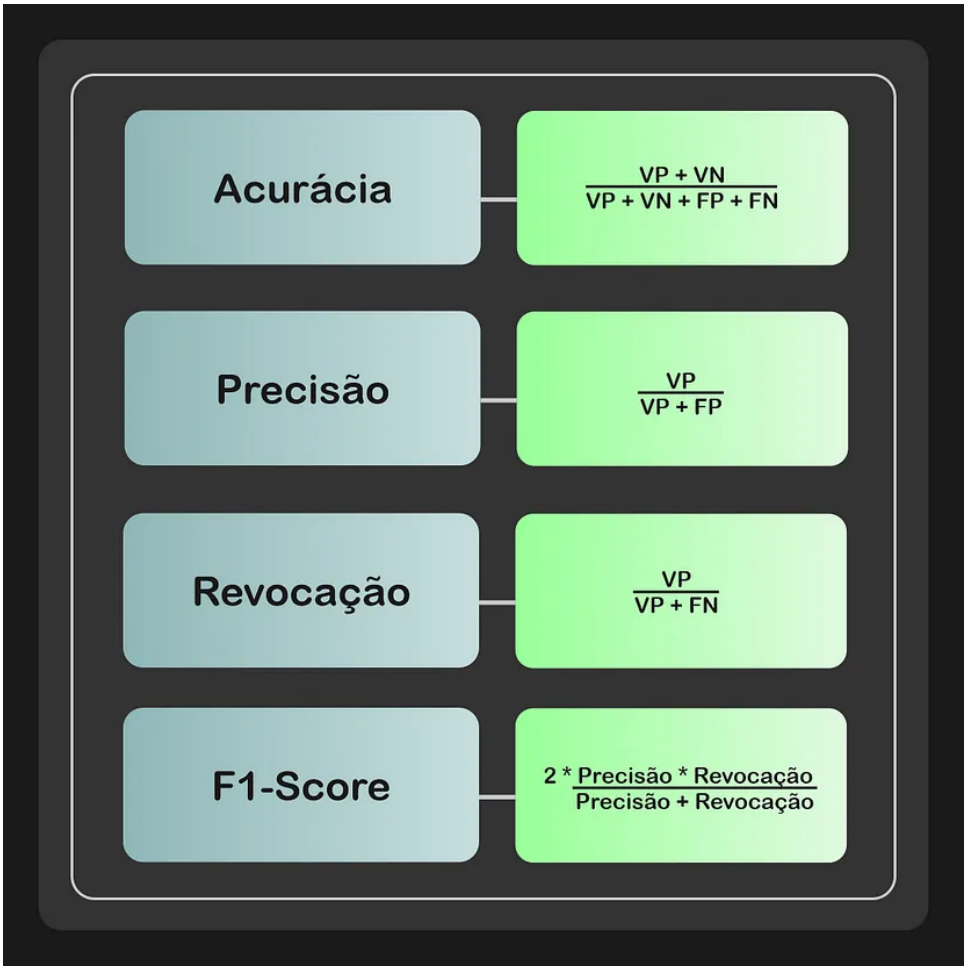


- cross_validation.KFold(n = num_instancias # 'len(X)', n_folds #escolhe)
- cross_validation.cross_val_score(modelo, x, y, cv=kfold)

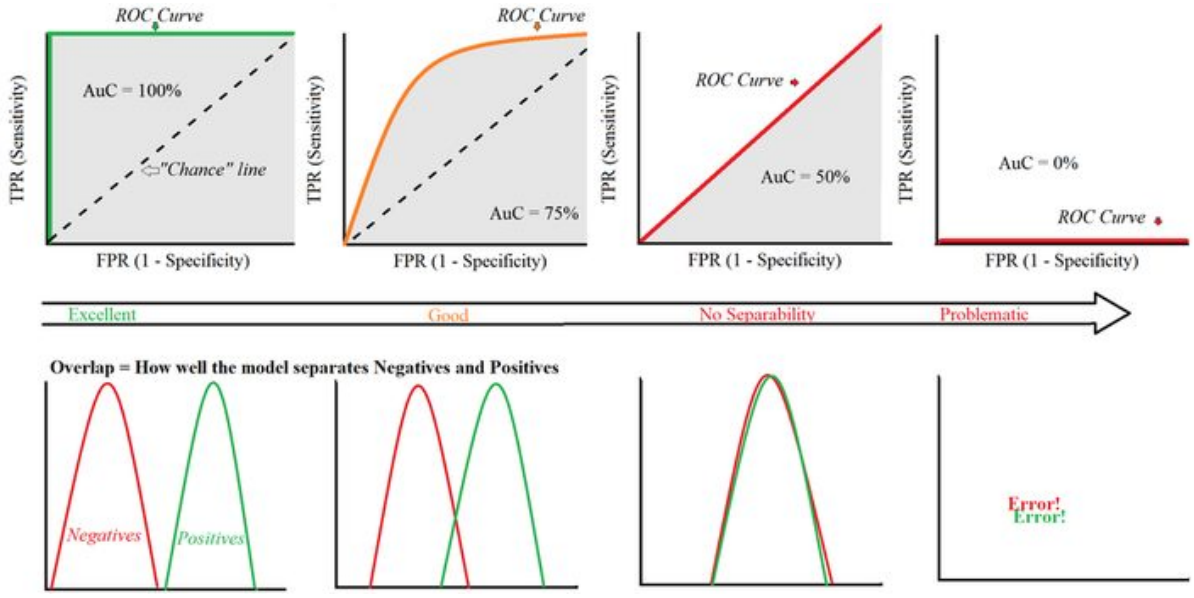
Métricas de avaliação de classificação

- Acurácia: indica uma performance geral do modelo. Dentre **todas** as classificações, quantas o modelo classificou corretamente
- Precisão: dentre todas as classificações de classe Positivo **que o modelo fez**, quantas estão corretas
- Recall: dentre todas as situações de classe Positivo **como valor esperado**, quantas estão corretas
- • **F1-Score:** média harmônica entre precisão e recall
- classification_report: mostra uma matriz com as 4 métricas

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)



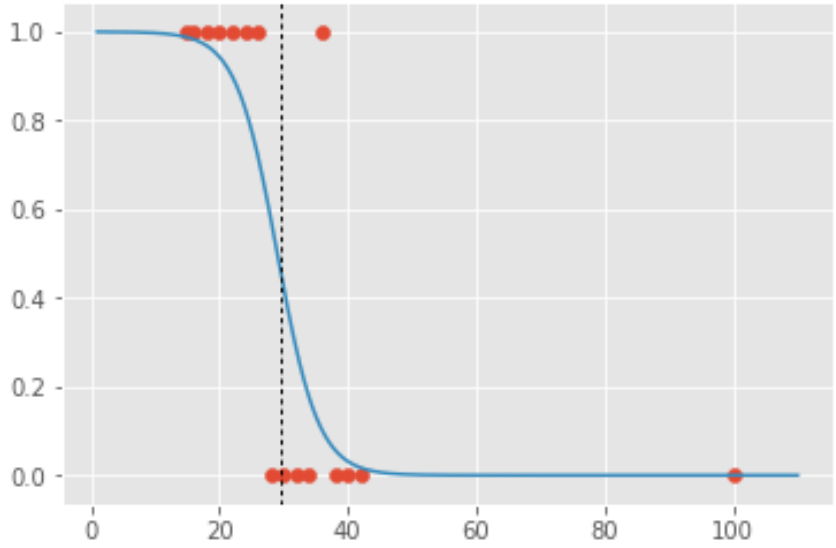
- Curva ROC:



Classificação

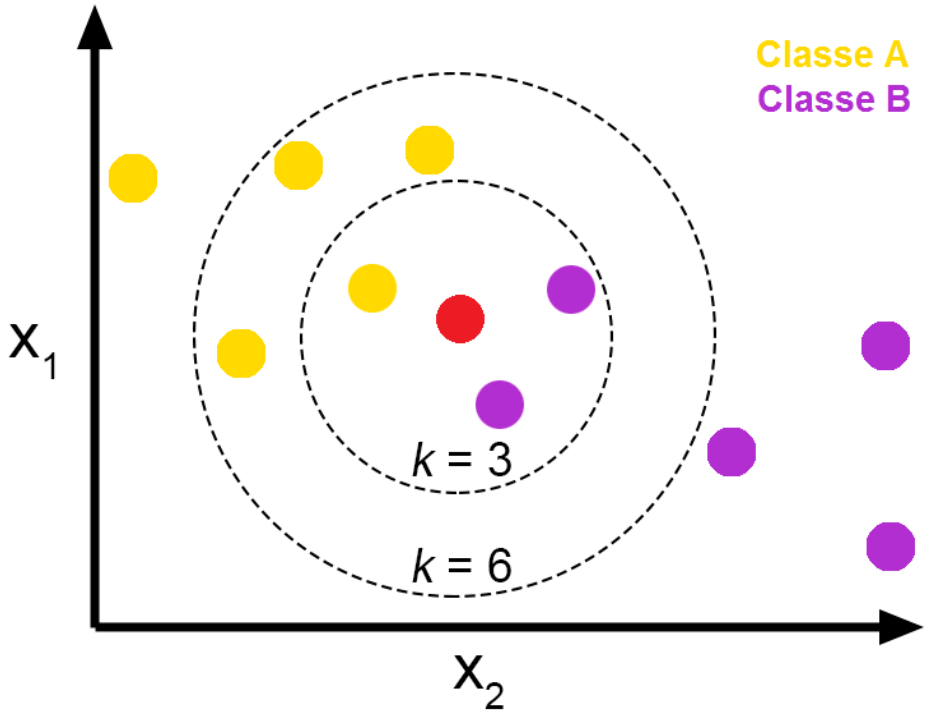
Regressão Logística

- Assume que os dados estão em uma distribuição normal
- Binária



KNN

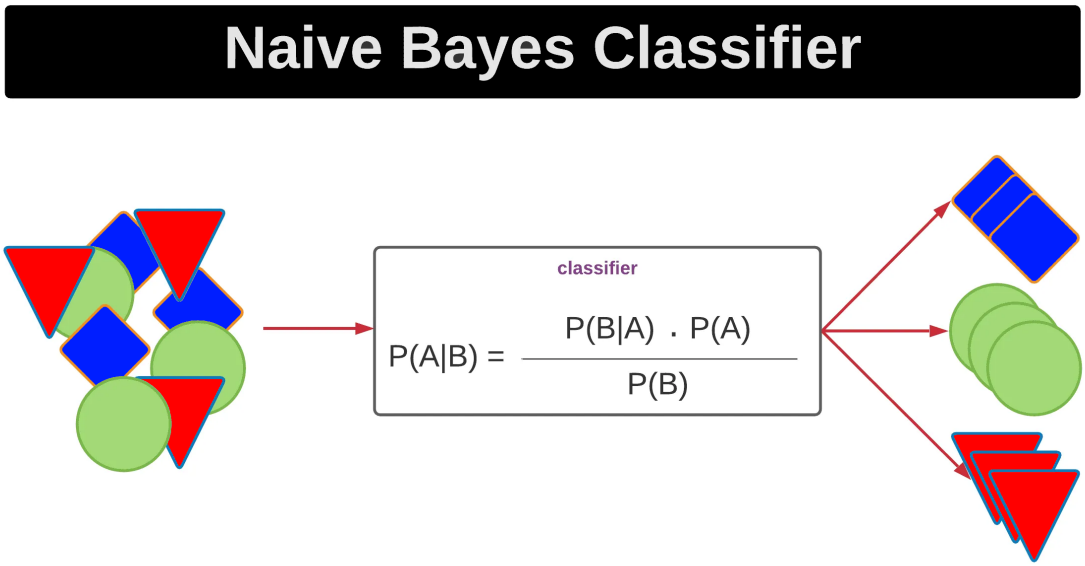
- Utiliza a menor distância para um k número de vizinhos para classificar



Naive Bayes

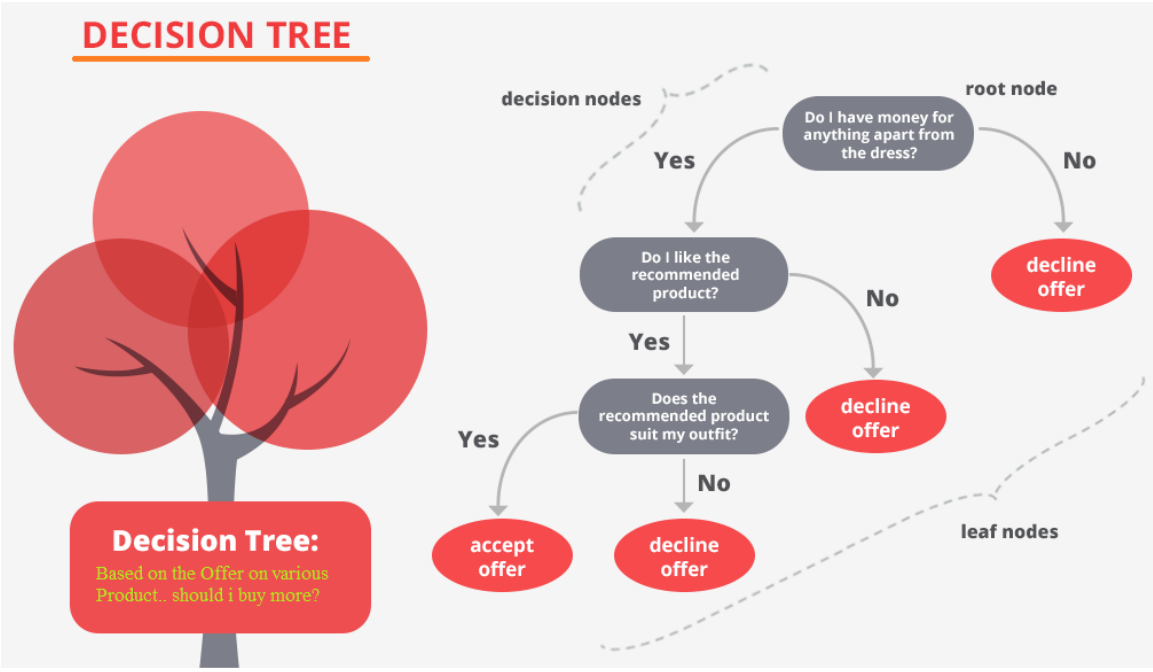
- Assume que as variáveis são independentes
- **Sistemas de recomendação:** Considerando os critérios e comportamento de usuários, o algoritmo Naive Bayes também poderia fazer recomendações considerando um ou mais critérios. Se um cliente visitou páginas de

determinados produtos, pode se interessar por outros. Essa classificação é facilmente gerenciada com os dados de classificação gerados pelo algoritmo.



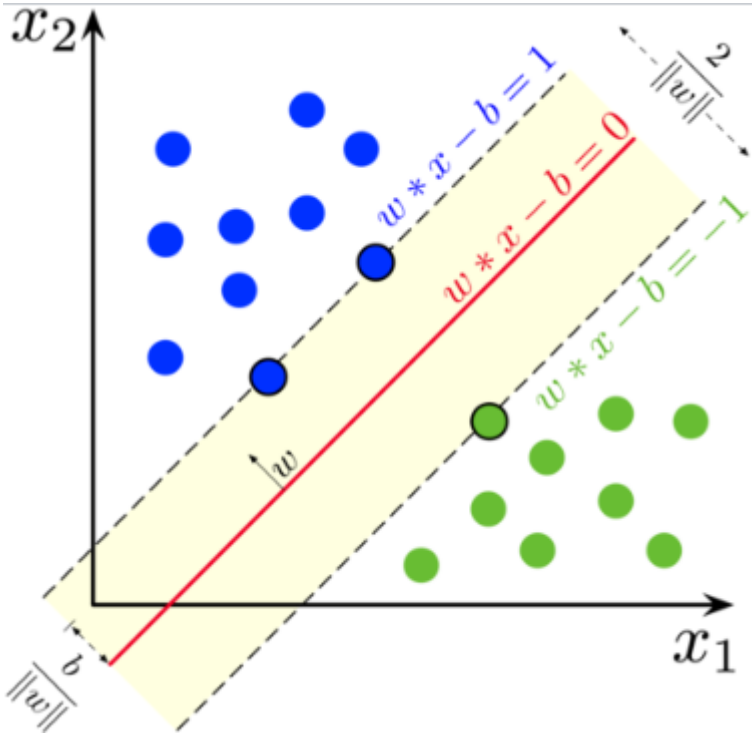
Árvores de Decisão

- O funcionamento da Árvore de Decisão visa formar “caminhos” que vão dividindo os dados em pequenos grupos. Essa divisão acontece com base nas características dos dados para que, no final, possamos entrar com um novo registro e o modelo nos dizer em qual classe esse dado se encaixa melhor.



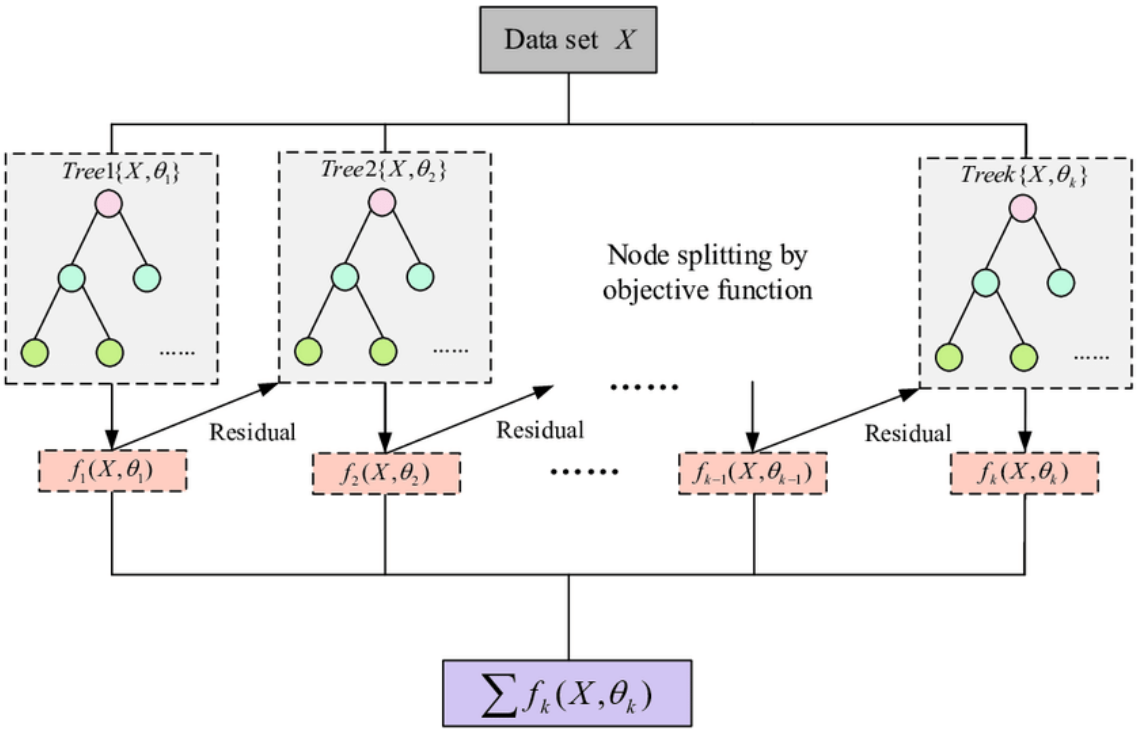
SVM

- O **SVM** é um algoritmo que busca uma linha de separação entre duas classes distintas analisando os dois pontos, um de cada grupo, mais próximos da outra classe. Isto é, o **SVM** escolhe a reta — também chamada de **hiperplano** em maiores dimensões— entre dois grupos que se distancia mais de cada um.



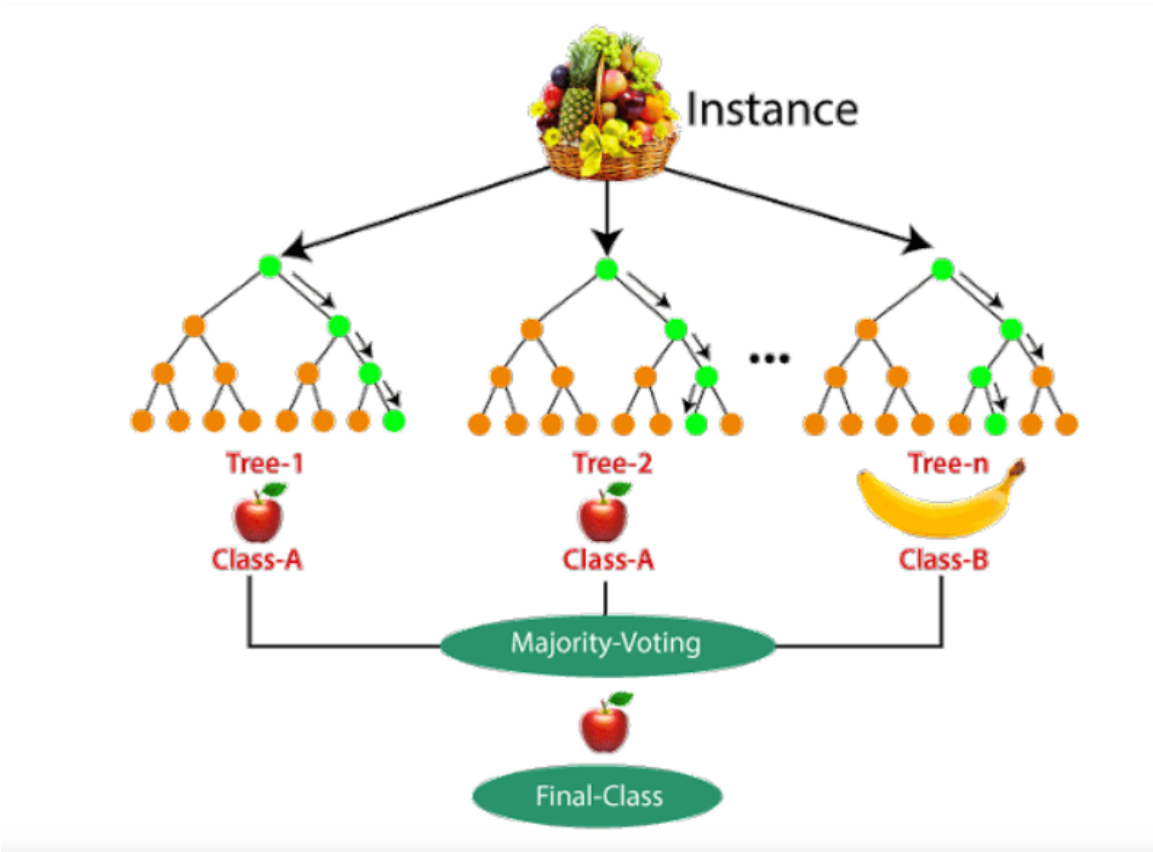
XGBoost

- O XGBoost usa árvores de decisão para seu modelo de conjunto. Cada árvore é um aprendiz fraco. O algoritmo segue construindo sequencialmente mais árvores de decisão, cada uma corrigindo o erro da árvore anterior até que uma condição de parada seja alcançada. Inclui diferentes penalidades de regularização para evitar *overfitting* onde essas regularizações de penalidade produzem treinamento bem-sucedido para que o modelo possa generalizar adequadamente.



Random Forest

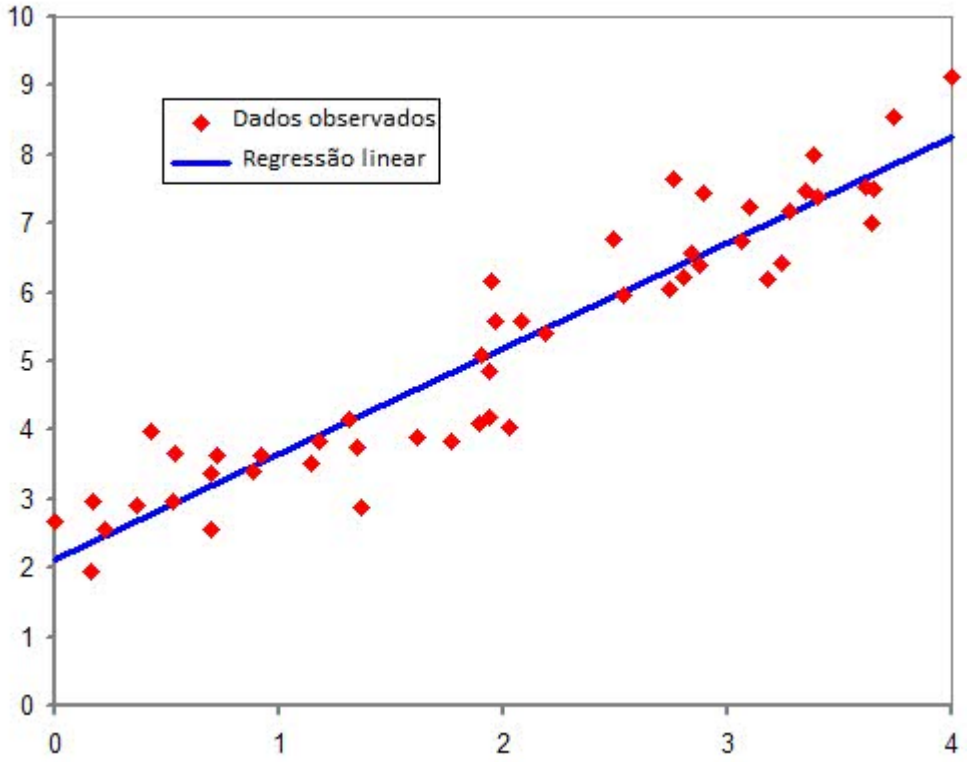
- Random Forest (Floresta Aleatória) é um método de aprendizado conjunto. A ideia, porém, é treinar várias árvores de decisão (descorrelacionadas), obtidas a partir de amostras do dataset, e fazer predições utilizando os resultados que mais aparecem em caso de um problema de classificação, ou a média dos valores obtidos em caso de regressão.



Regressão

Regressão linear

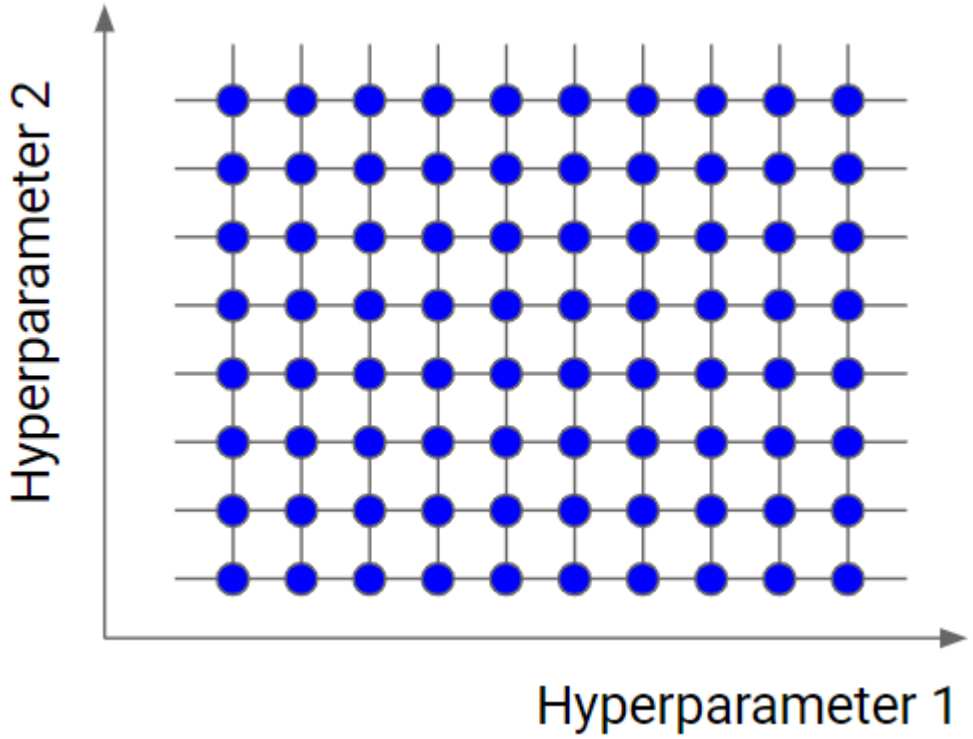
- Regressão linear é um algoritmo supervisionado de machine learning usado para estimar o valor de algo baseado em uma série de outros dados históricos, portanto olhando para o passado você pode “prever” o futuro. Em sua essência, uma técnica de regressão linear simples tenta traçar um gráfico de linhas entre duas variáveis de dados, x e y. Como variável independente, x é plotada ao longo do eixo horizontal. Variáveis independentes também são chamadas de variáveis explicativas ou variáveis preditoras. A variável dependente, y, é plotada no eixo vertical. Você também pode fazer referência aos valores de y como variáveis de resposta ou variáveis previstas.



Ajuste de Parâmetros

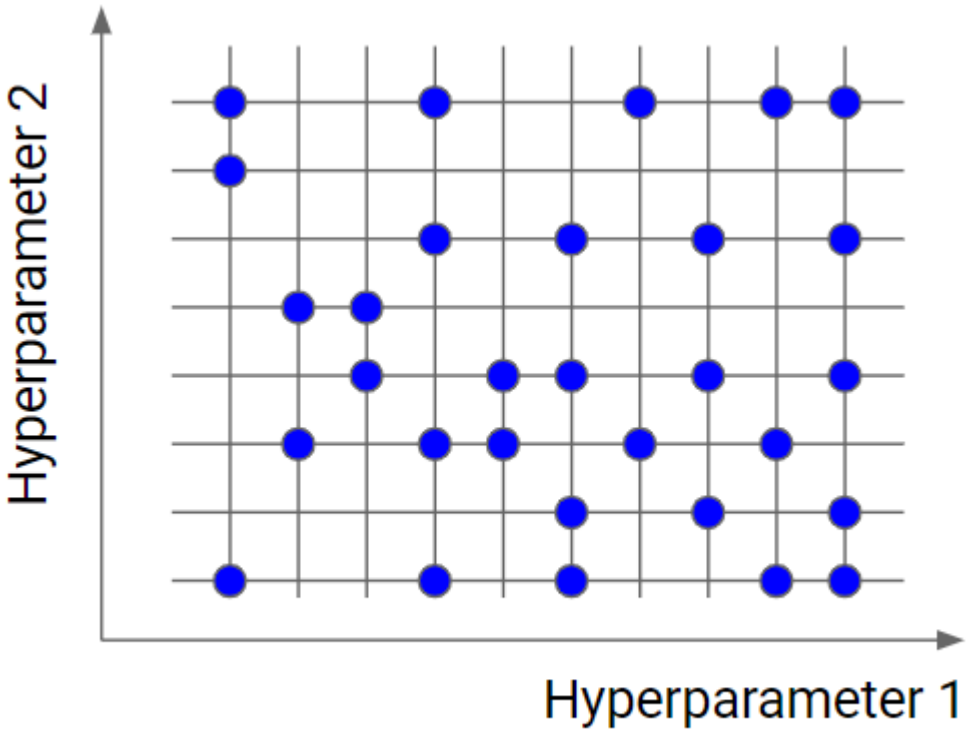
Grid Search Parameter Tuning

- Grid Search é o algoritmo mais simples para ajuste de hiperparâmetros. Basicamente, dividimos o domínio dos hiperparâmetros em um grid discreto. Em seguida, tentamos todas as combinações de valores desse grid, calculando algumas métricas de desempenho usando validação cruzada. O ponto do grid que maximiza o valor médio na validação cruzada é a combinação ótima de valores para os hiperparâmetros.



Random Search

- O random search é semelhante ao grid search, mas em vez de usar todos os pontos do grid, ele testa apenas um subconjunto selecionado aleatoriamente desses pontos. Quanto menor esse subconjunto, mais rápida, mas menos precisa, a otimização. Quanto maior esse conjunto de dados, mais precisa a otimização, mas mais próxima de um grid search.



Métodos Ensemble

- O conceito de **Ensemble Learning**, também chamado de aprendizado por agrupamento, se baseia na ideia de combinar diversos modelos de predição mais simples (**weak learner**), treiná-los para uma mesma tarefa, e produzir a partir desses um modelo agrupado mais complexo (**strong learner**) que é a soma de suas partes.
- **Bagging:** geralmente é feito com preditores homogêneos, cada um de forma independente em relação ao outro, de forma paralela. O algoritmo final é então feito a partir de algum tipo de resultado médio do que foi obtido a partir dos modelos bases.
- **Boosting:** geralmente é feito com preditores homogêneos, que são aplicados de forma sequencial (o posterior depende do antecessor) e depois combinados no modelo final.
- **Stacking:** geralmente é feito com preditores heterogêneos, treinando-os em paralelo. É então aplicado um modelo no *output* dos *weak learners* (podendo

incluir ou não as *features* utilizadas para treiná-los).