

# KNN

## k-nearest neighbors



# Distância Euclidiana

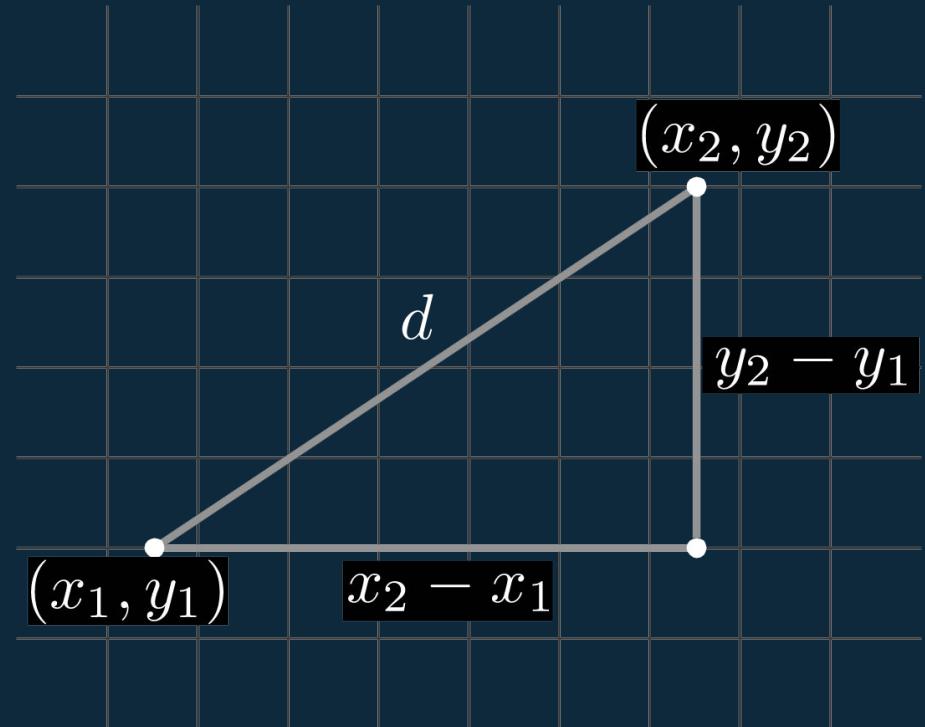
	Formato	Fruta cítrica	Cor	Rugosidade	Cheiro
Laranja	0	1	2	1	0
Maça	0	0	1	0	0
Tangerina	0	1	2	1	1

◊ A tangerina parece mais com a laranja ou com a maçã?



# Distância Euclidiana

$$\begin{aligned} d(p, q) &= d(q, p) \\ &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \end{aligned}$$





# Distância Euclidiana

- ◊ Dist. Euclidiana(Tangerina, Maçã) = 2
- ◊ Dist. Euclidiana(Tangerina, Laranja) = 1

A tangerina parece mais com uma laranja!!!



# Introdução

- ◊ O meu comportamento é influenciado (ou caracterizado) por todos esses fatores, logo olhar para os vizinhos mais próximos de mim nessa dimensão parece ser melhor que olhar para toda a população
- ◊ Essa é a idéia por trás do k-Vizinhos mais próximos (kNN)



# O Algoritmo KNN

- ◊ O kNN é um dos modelos de predição mais simples que existem. Ele utiliza apenas:
  - Alguma noção de distância (similaridade)
  - A suposição que pontos que estão próximos uns dos outros são semelhantes
- ◊ Para a predição de um novo ponto, observa-se apenas os pontos mais próximos



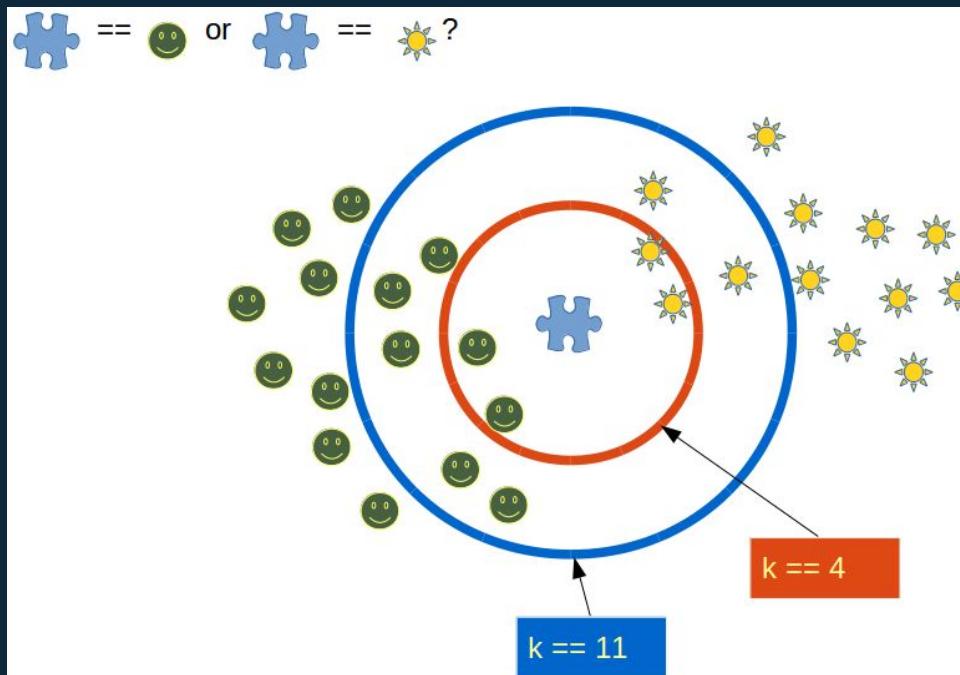
# O Algoritmo KNN

- ◊ Não ajuda a entender o fenômeno observado: Não há como recuperar os pesos das features, por exemplo
- ◊ Armazena-se uma base de exemplos (instances) que é usada para realizar a classificação de uma nova query (exemplo não visto);

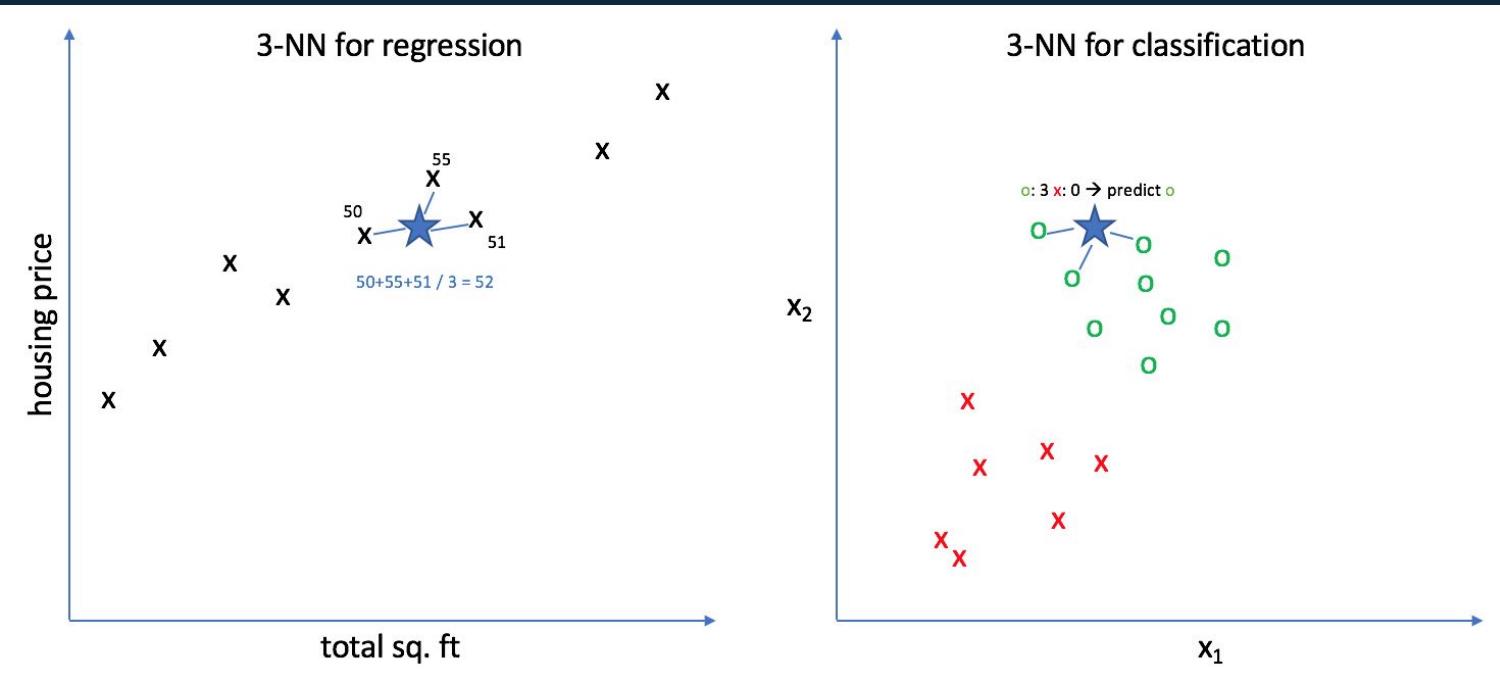
# O Algoritmo KNN

- ◊ Em geral temos os dados rotulados
- ◊ Para um ponto novo, procura-se os  $k$  pontos mais próximos a ele e olha-se os seus labels para decidir qual será o label do novo
- ◊ E se der empate? Temos várias opções:
  - Escolher um dos ganhadores aleatoriamente
  - Atribuir pesos aos votos baseado na distância
  - Reduzir  $k$  até encontrar um vencedor

# KNN - Exemplo



# KNN - Exemplo



# KNN - Regressão

Label	Distance(d)
50,0	0,7
55,0	0,3
51,0	2,1

Média ponderada

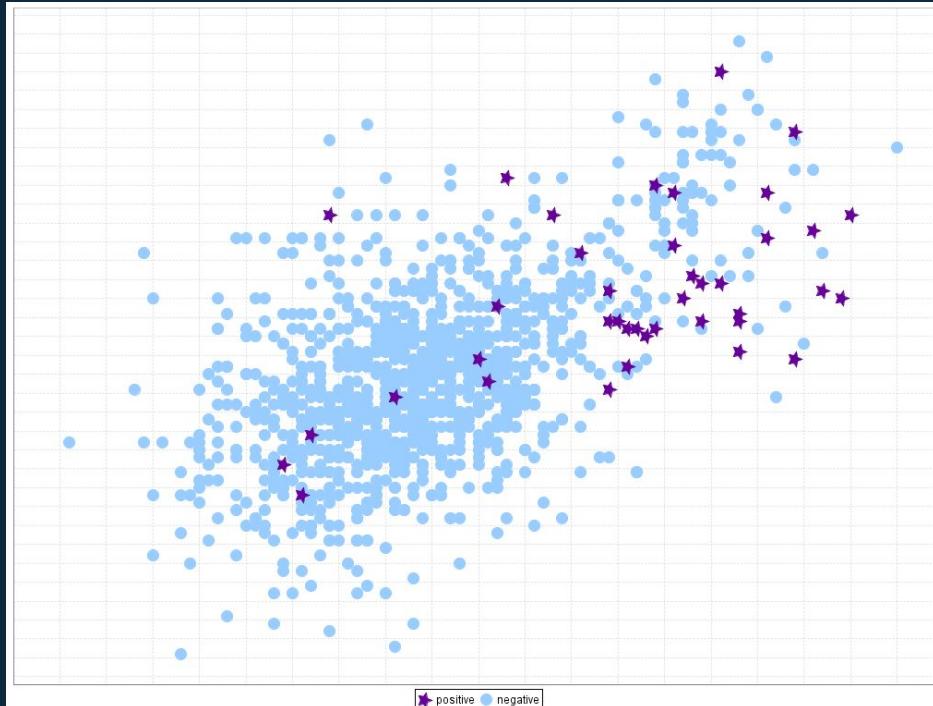
$$\frac{p_1 \cdot x_1 + p_2 \cdot x_2 + p_3 \cdot x_3 + \dots + p_n \cdot x_n}{p_1 + p_2 + p_3 + \dots + p_n} = \frac{\sum_{i=1}^n (p_i * x_i)}{\sum_{i=1}^n p_i}$$

Onde  $P_i = 1/d$

Média Ponderada ~ 53,3

Média = 52

# KNN - Classificação



1. Dist. ponderada =  $1/d$
2.  $f = \max(\sum \text{Dist.Pond})$



# Considerações

- ◊ Favorecemos as amostras mais similares
- ◊ Podemos aplicar uma média ponderada para considerar os vizinhos mais próximos
- ◊ kNN tem problemas com dimensões maiores, que se resume a idéia de que espaços de dimensão alta são vastos. Pontos nesses espaços tendem a não estar próximos.



# Considerações

- ◊ Cada dimensão é uma chance de aumentar a distância
- ◊ Normalmente usa-se a redução de dimensionalidade antes de usar o kNN
- ◊ A predição é sobre demanda, e isso pode ser muito custoso.



# Análise do Overfitting e Underfitting

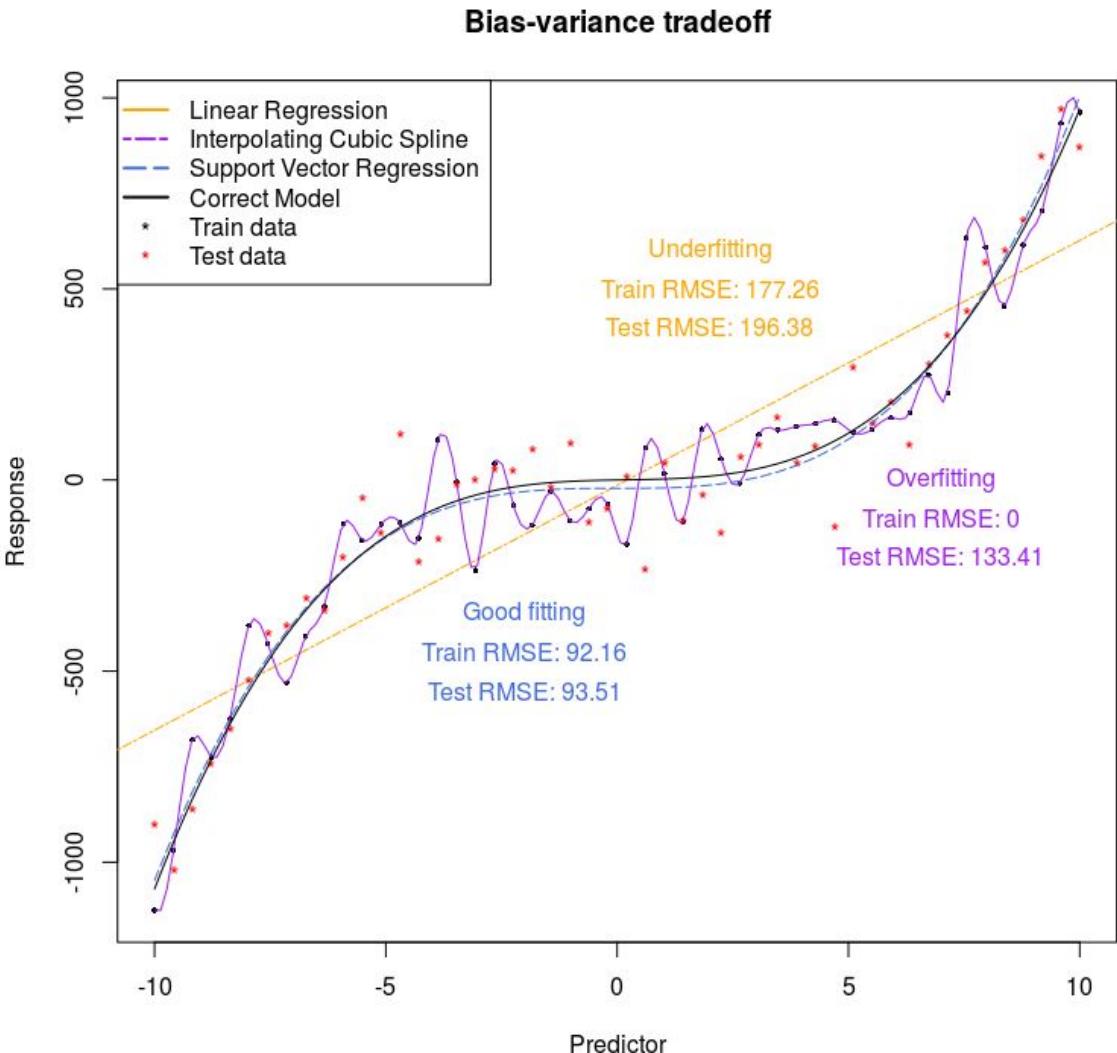


# Definições

- ◆ **Underfitting** - Utilizar um modelo simples que é bem generalizável, **mas não reduz consideravelmente** o erro de previsão no train set. Nesse caso estamos optando por um modelo com viés mais alto, mas variância baixa.
- ◆ **Overfitting** - Utilizar um modelo complexo que é capaz de **reduzir consideravelmente** o erro de previsão no train set, mas ao mesmo tempo **não é tão generalizável a ponto de apresentar um bom resultado no test set**. Nesse caso estamos optando por estimar um modelo com viés baixo e variância alta.

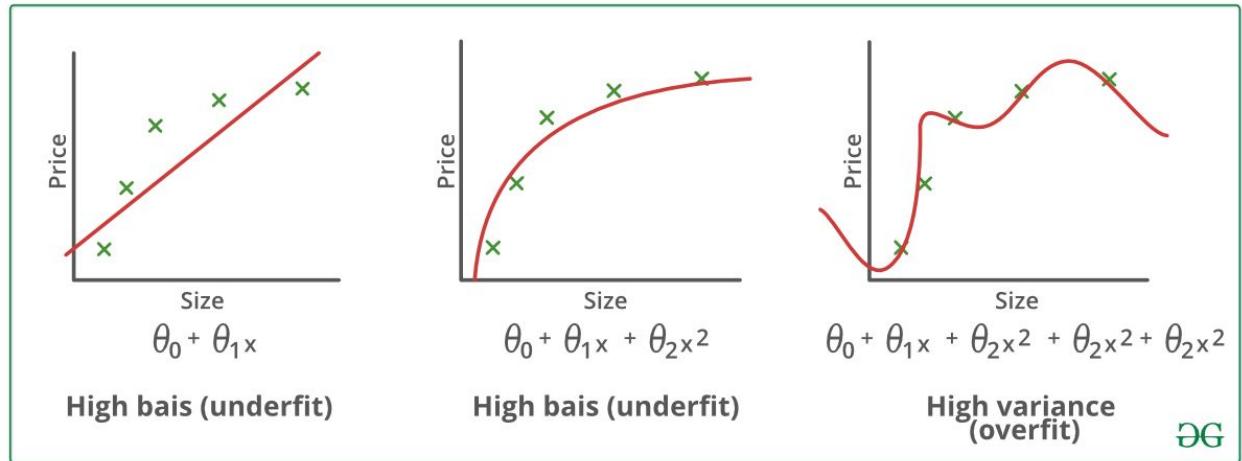


# Bias/ Variance tradeoff





# Bias/ Variance tradeoff



# Bias vs Variance

$$E[(y - \hat{f}(x))^2] = Bias[\hat{f}(x)]^2 + Var[\hat{f}(x)] + \sigma^2$$

$$Bias[\hat{f}(x)] = E[\hat{f}(x) - f(x)],$$

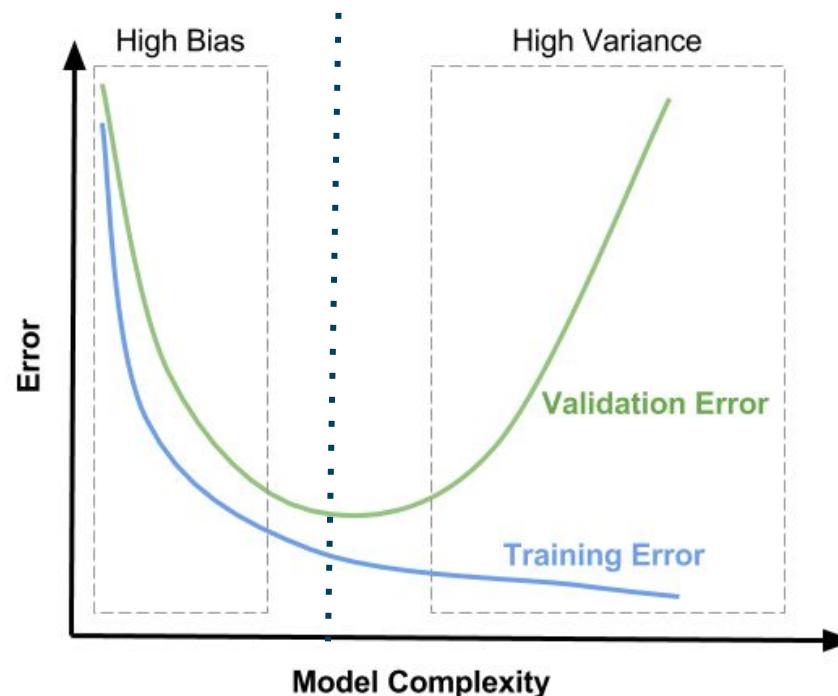
$$Var[\hat{f}(x)] = E[\hat{f}(x)^2] - E[\hat{f}(x)]^2.$$

# Bias vs Variance

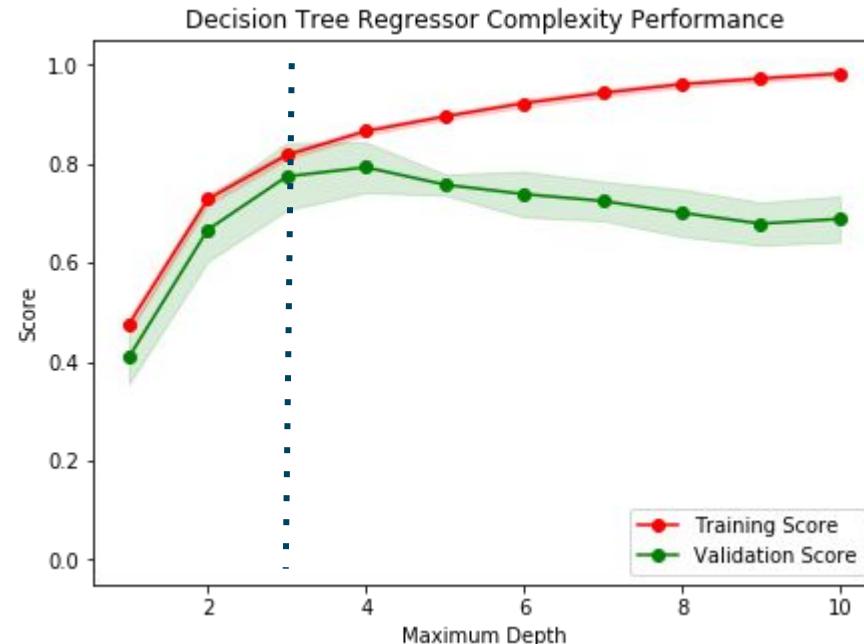
- ◊ O objetivo é escolher uma  $f(x)$ , um modelo, próxima do ideal, visto que tanto o viés quanto a variância aumentam o erro de previsão.
- ◊ Obviamente a escolha entre bias e variance é um tradeoff, e o ideal é permanecermos em um meio termo entre um modelo complexo e um bem generalizável.



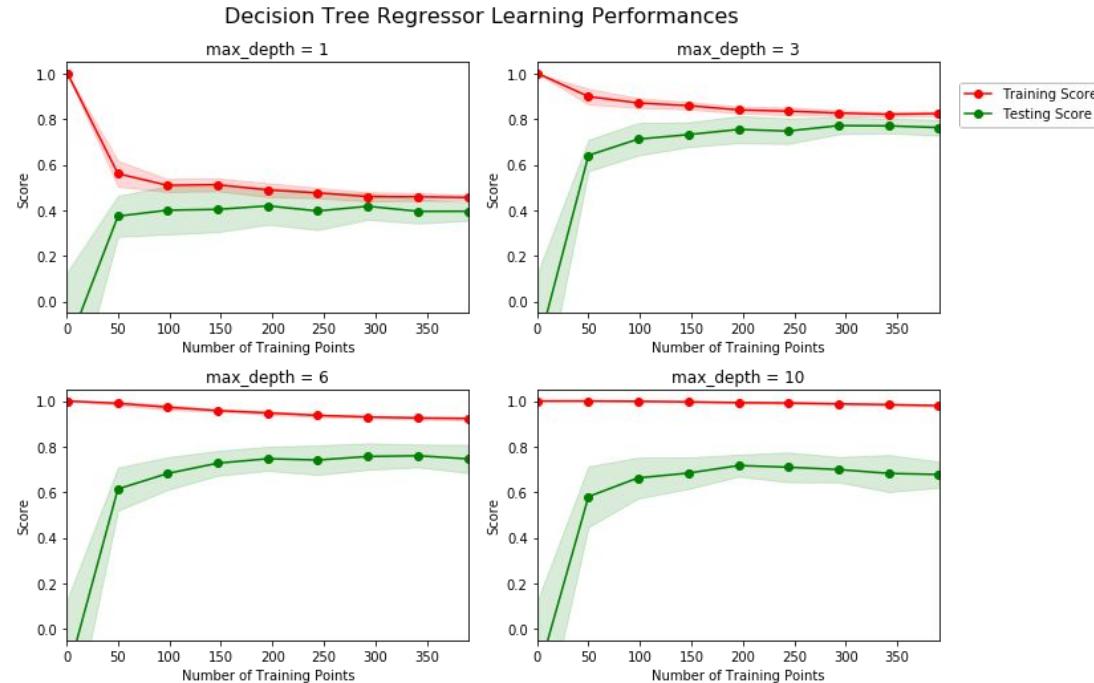
# Complexidade do modelo



# Overfitting vs Underfitting



# Overfitting vs Underfitting



# Grid Search

- ◆ Tem por objetivo identificar qual o melhor hiperparâmetro do algoritmo de aprendizagem de máquina que pode ser utilizado durante a construção do modelo, considerando a métrica de avaliação utilizada para validar o quanto bom um modelo é.
  
- ◆ Para ser utilizada basta que você especifique quais valores, ou range, de hiperparâmetros que você deseja testar.

# Grid Search

- ◊ Sempre que formos aplicar esta técnica devemos ter em mente que este processo é **custoso**, pois **quanto mais valores a serem analisados mais demorado será**.

## MODEL SELECTION

Finding the machine learning algorithm and its hyperparameter values that produce the best model.

Chris Albon



# Exemplo

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
# Create a based model
rf = RandomForestRegressor()
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid, scoring='accuracy')
```

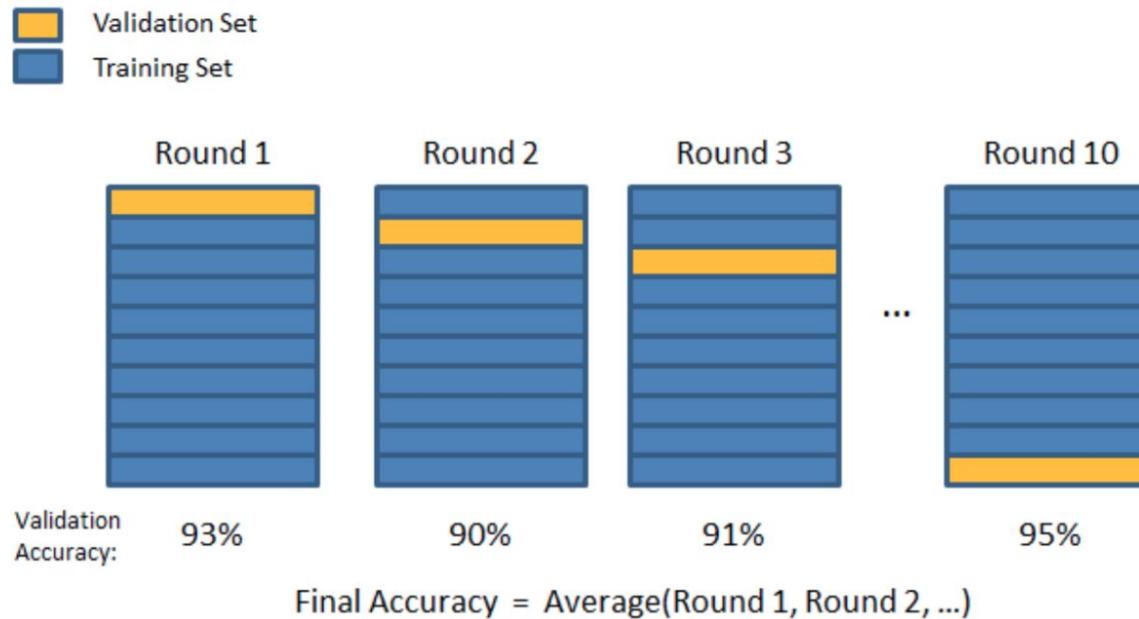
# Cross Validation

- ◆ Uma das formas mais simples e primordiais para validar um modelo é criar, no mínimo, um subconjunto de testes para avaliá-lo.
- ◆ A técnica de validação-cruzada de treinamento é um método capaz de avaliar o quanto eficaz o modelo é para um conjunto de novas amostras, conjunto de teste.
- ◆ Seu principal objetivo é evitar que o problema do overfitting ocorra, validando assim o quanto genérico é um modelo. "Esta é uma técnica bastante empregada para modelos de predição".

# Cross Validation

- ◆ A aplicação da técnica consiste em dividir o conjunto de treinamento em K-folds, ou seja, dividir o conjunto de treinamento em K subconjuntos disjuntos, onde  $(k-1)$  subconjuntos de dados fará parte do conjunto de treinamento e 1 subconjunto fará parte da validação.
  
- ◆ Após a criação do modelo e validação, um novo subconjunto dentre os k subconjuntos é selecionado e o mesmo processo se repete, até que todos os subconjuntos tenha sido processados, ou seja, o processo é executado K-vezes. O modelo que apresentar o melhor resultado considerando a métrica de avaliação é selecionado.

# Cross Validation

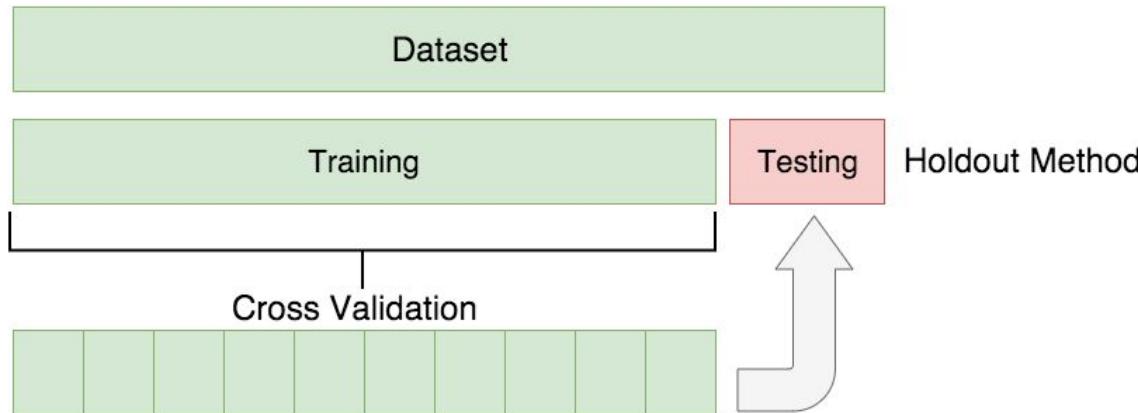


# Grid Search & Cross Validation

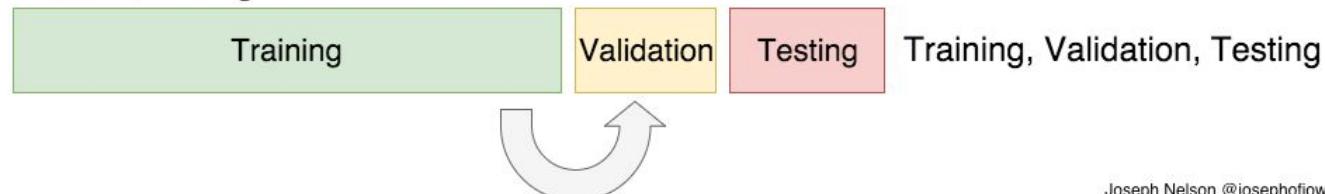
- ◊ O benefício de utilizar Cross Validation no Grid Search é permitir que o modelo seja avaliado ainda em sua fase de construção, treinamento, podendo posteriormente ser avaliado com uma nova amostra de dados, por exemplo com um conjunto de testes. Logo, esta prática já fornece um modelo minimamente avaliado com amostras que não fizeram parte do seu treinamento, tornando mais válida sua forma de avaliar o modelo considerando a métrica que é analisada no processo do Grid Search.
- ◊ **Dica:** Você pode treinar o modelo com os hiperparâmetros selecionados sobre todo o dataset e então disponibilizar este modelo em produção



# Cross Validation

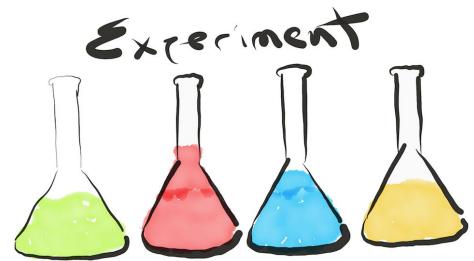


Data Permitting:





# Experimentos ...



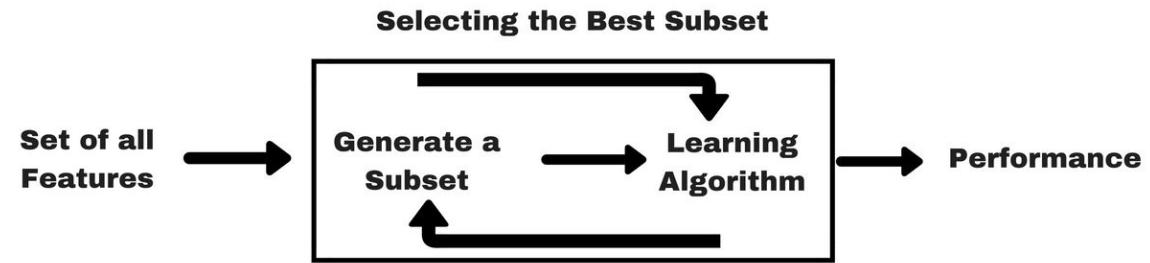


## Feature selection

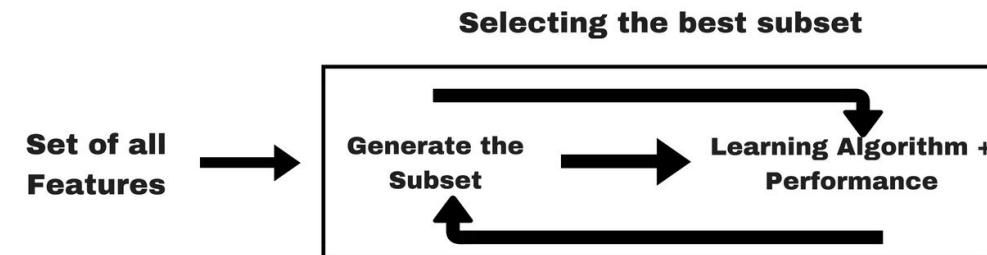
### Filter



### Wrapper



### Embedded





# Tabela de experimentos

Modelos	#Samples		
	5000	10000	15000
KNN	0,85	0,80	0,90
DT	0,90	0,90	0,88
RF	0,90	0,95	0,96

Tabela 1 - Valores de F1 score dos experimentos realizados com CV = 10



[http://dontpad.com/kdd\\_uni7](http://dontpad.com/kdd_uni7)

# Hands On

