APRENDIZAGEM DE MÁQUINA

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br
https://github.com/josenalde/machinelearning

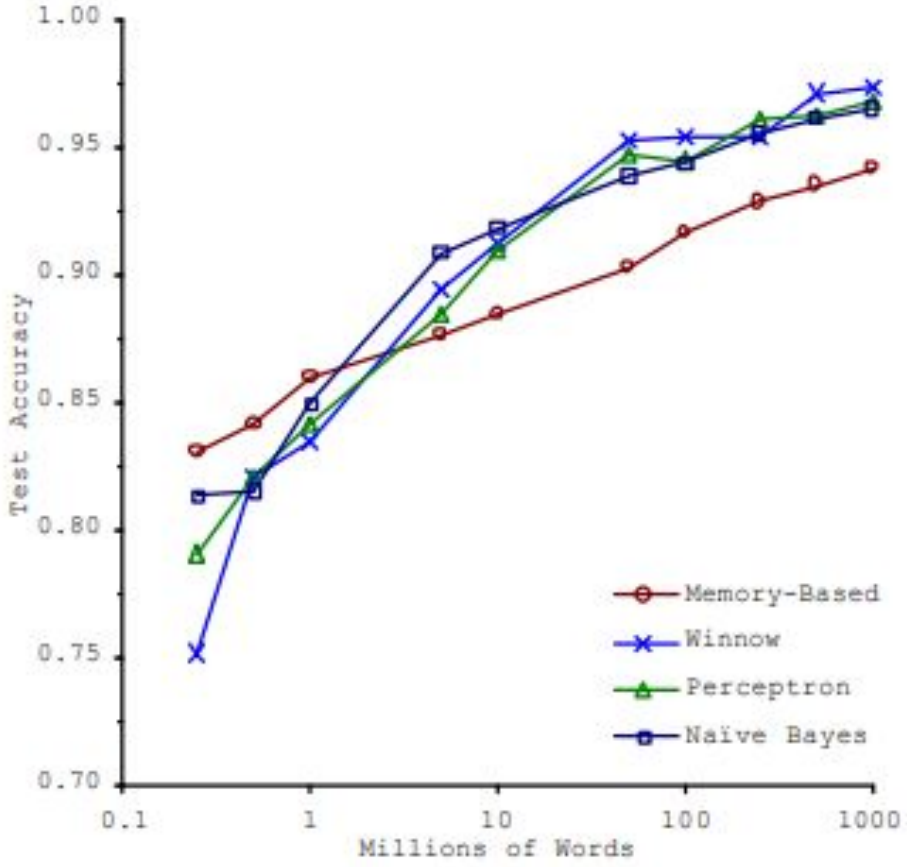
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com os Dados

[1] Quantidade insuficiente de dados de treinamento: (centenas, milhares, milhões...) com exceções para técnicas de **transfer learning e few-shot**

Em 2001, pesquisadores da Microsoft demonstraram que algoritmos de AM bastante distintos (e simples) tiveram desempenho quase idêntico em problema complexo de desambiguação de linguagem natural (diferenciar to, two, too dependendo do contexto) com alimentação de dados suficientes (*corpus*)



DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com os Dados

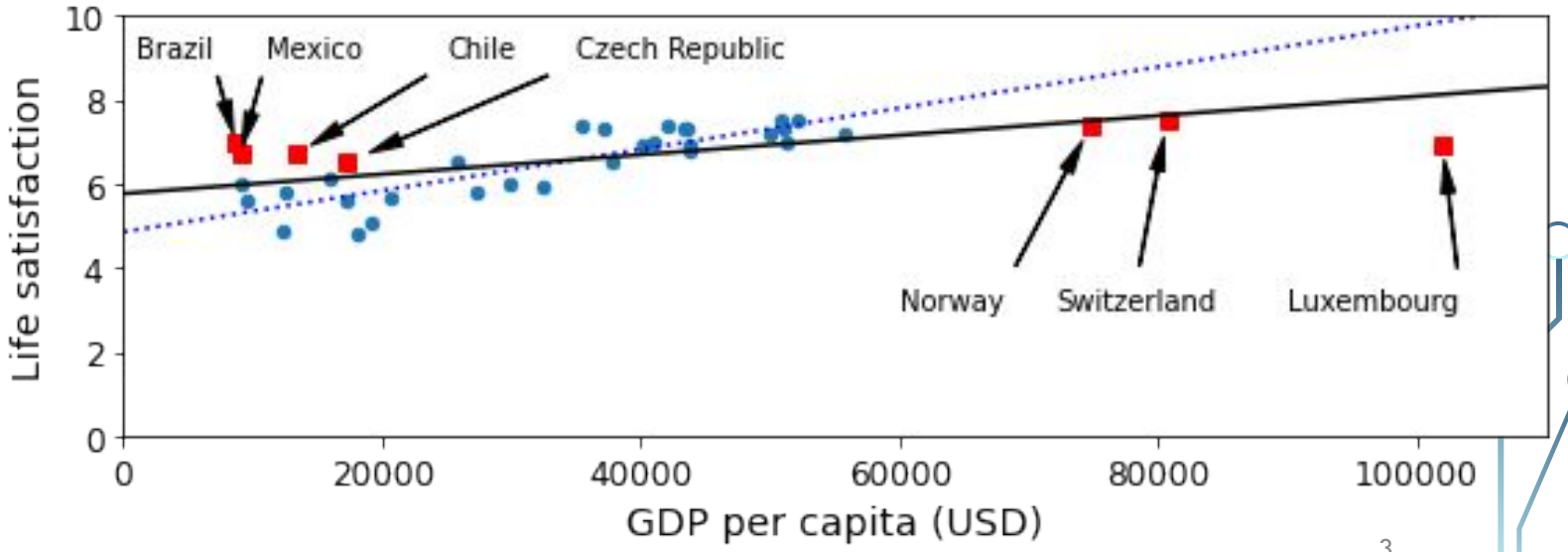
[2] Dados de treinamento não representativos, logo não se conseguirá boas generalizações. No treinamento é importante ter representações dos dados novos, ou seja, do que se espera que “apareça”.

Viés de amostragem: muito pequena: ruído de amostragem (dados não significativos como resultado do acaso)

muito grande mas não representativo, com método de amostragem falho

Linha sólida: modelo com dados novos

Linha pontilhada: modelo antigo



DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com os Dados

[3] Dados de baixa qualidade: garbage-in, garbage-out – dados com erros, outliers, ruídos

Gastar tempo com limpeza de dados, início correto do processo

Tratamento de outliers e dados faltantes (descartar colunas? Preencher valores ausentes? Como? Treinar um modelo separado sem esses dados?

[4] Dados irrelevantes

Treino precisa ser mais relevantes, poucas irrelevantes; criar um bom conjunto de características se chama engenharia de features (feature engineering), com etapas de seleção de características (feature selection), extração de características (feature extraction) – ou seja, juntar características, criação de novas features

DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com os Dados

Modelos são sintonizados por valores de seus hiperparâmetros
É um parâmetro de algoritmo de ML (não do modelo). É definido antes do treinamento

[5] Sobreajuste dos dados de treinamento (overfitting)

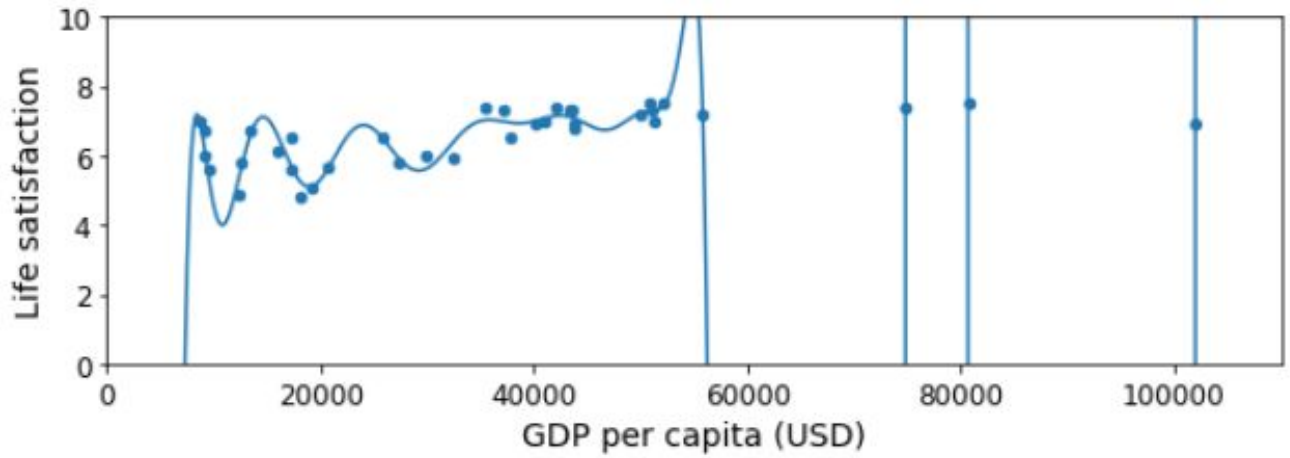
Ex: se o modelo é complexo e você inclui a feature nome do país, o modelo pode aprender que países com W tem safistação de vida maior, como New Zealand, Norway, Sweden, Switzerland, mas no caso generalizaria para Rwanda, Zimbabwe, ou seja, é um padrão captado por acaso, mas o modelo não sabe se o padrão é bom ou ruim (resultado do ruído)

Possíveis soluções:

a) simplificar o modelo com menos parâmetros (linear ao invés de polinomial), reduzir número de atributos, restringir modelo (regularização – ponderar features)

b) coletar mais dados de treinamento

c) reduzir o ruído nos dados de treinamento (melhorar limpeza)



DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com os Dados

[5] Subajuste dos dados de treinamento (underfitting)
Muito simples para detectar os padrões, causa imprecisão nas predições
Possíveis soluções:
a) selecionar modelo mais poderoso
b) melhorar a feature engineering
c) minimizar as restrições, por exemplo, reduzindo a regularização

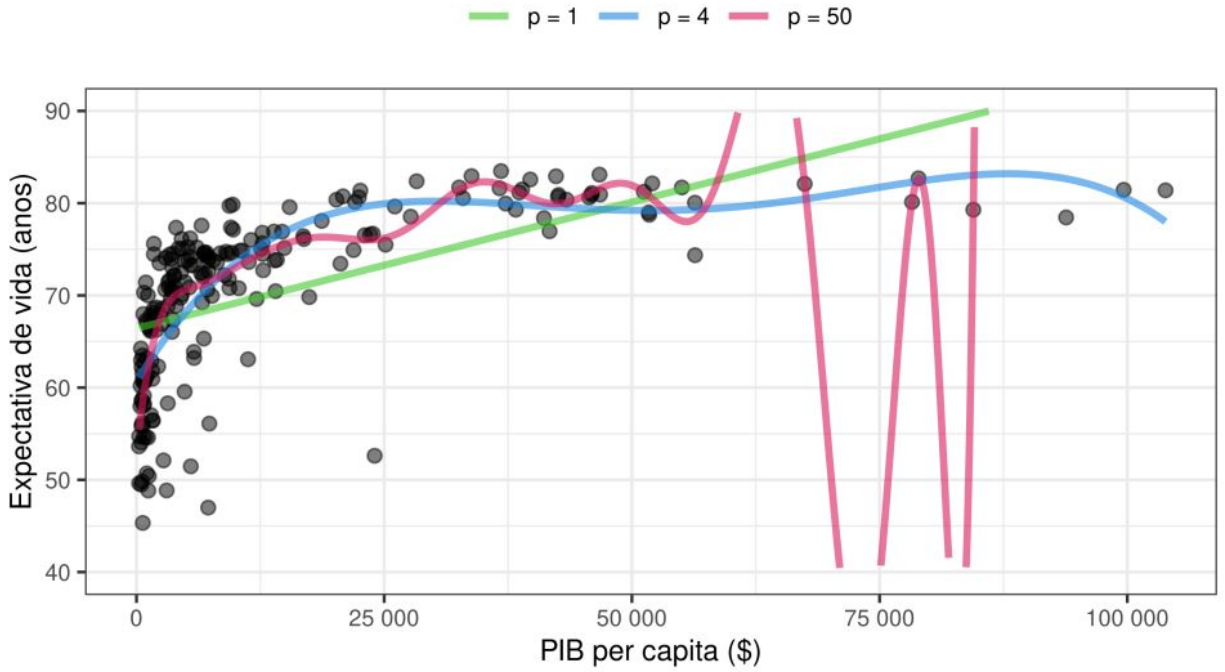


Figura 1.4: Comparação dos modelos $g(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x^i$, para $p \in \{1,4,50\}$ nos dados de esperança de vida (Y) versus PIB per Capita (X).

p=1 (linear, subajuste)

$$g(x) = \beta_0 + \beta_1 x_1$$

p=4 (polinomial)

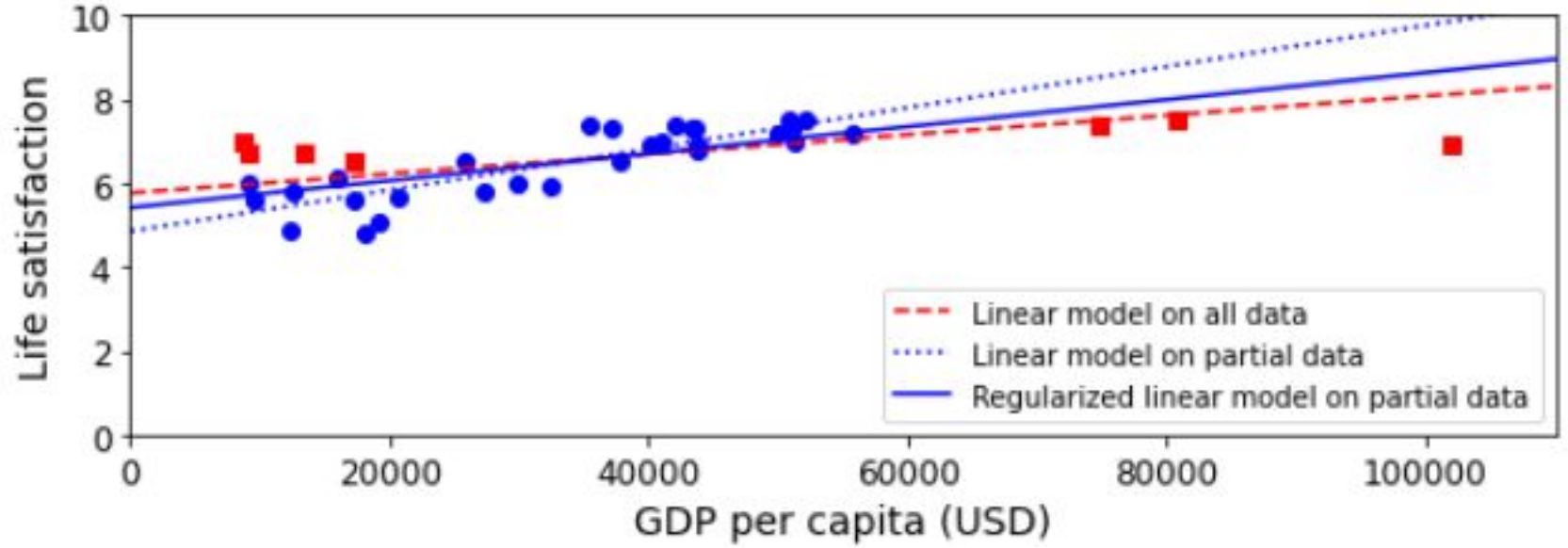
p=50 (polinomial, sobreajuste)

Fonte: IZBICKI, R.; SANTOS, T.M. **Aprendizagem de Máquina**: uma abordagem estatística, 2020.

DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com os Dados

A regularização é técnica muito utilizada para controle de over-under fitting
Em redes neurais, técnicas como **dropout** tem finalidade semelhante



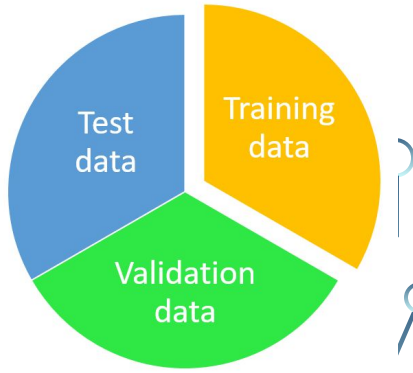
DESAFIOS AO APRENDIZADO DE MÁQUINA

Problemas com o Modelo

[6] Teste e validação
- colocar em produção e monitorar a qualidade pelo feedback?
- o mais simples é dividir em dois conjuntos (80/20, por exemplo), mas depende da quantidade de instâncias no dataset
- a taxa de erro nos casos novos se chama erro de generalização. Ao avaliar o modelo no conjunto de teste se tem uma estimativa deste erro
- se o erro de treinamento for baixo e o de generalização alto, o modelo está sobreajustado aos dados de treinamento.

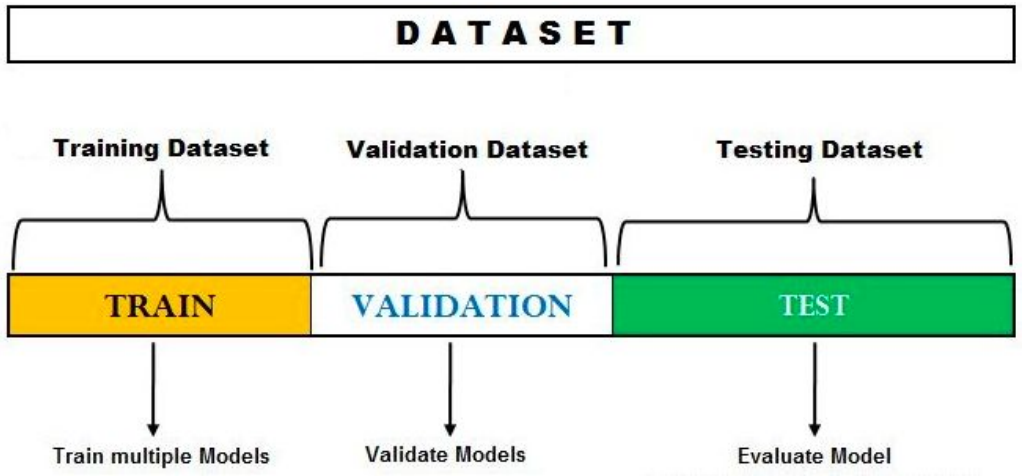
O problema é que o modelo pode ficar também sobreajustado aos dados de TESTE!

Modelos diferentes, como escolher? Imagine linear e polinomial e que o linear generalize melhor.
Opção: usando o conjunto de teste e comparar a generalização!
Para evitar sobreajuste você implementa regularização, mas como escolher o hiperparâmetro?
Treinar X modelos com X hiperparâmetros diferentes e ver o com menor erro, suponha 5%.
Mas em produção deu 15% de erro, pois foi ajustado para os dados de teste.
Vejamos possíveis soluções:



OPÇÕES DE VALIDAÇÃO CRUZADA

HOLDOUT



Por exemplo:
80/20, onde
Os 80% de treino pode ser
70% treino e 30% validação
Escolha aleatória!

- Dividir parte do conjunto de treinamento a fim de avaliar modelos concorrentes e selecionar o melhor
- Conjunto de validação (ou *dev set*): treina-se no conjunto de treinamento limitado com vários hiperparâmetros e seleciona modelo com melhor desempenho (métrica escolhida) no conjunto de validação
- Após este processo, treina o melhor modelo em todo o conjunto de treinamento (incluindo o de validação), o que fornece o modelo final
- Este então é avaliado no conjunto de testes
- Escolha do tamanho dos conjuntos! Validação pequeno (avaliação imprecisa); Validação grande (conjunto de treino limitado pequeno, bem menor que o conjunto de treino completo)
- Custo computacional menor, eficiência em conjunto de dados grandes, para conjunto pequeno ineficiente
- Fortemente dependente da "qualidade" das divisões – ou seja, da aleatoriedade

OPÇÕES DE VALIDAÇÃO CRUZADA

HOLDOUT

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score

# Carregar o conjunto de dados
data = load_iris()
X, y = data.data, data.target
# Dividir o conjunto de dados em treinamento (70%) e teste (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)
X_train_limited, X_validation, y_train_limited, y_validation = train_test_split(X_train,
y_train, random_state=10)
#
```

OPÇÕES DE VALIDAÇÃO CRUZADA

HOLDOUT

- Altere o código a seguir para incluir o holdout para testar o KNN com k=3,4,5,6,7
- O com maior acurácia, treinar (fit) no conjunto completo de treino (X_train,y_train)
- Aplicar no conjunto de TESTE e calcular métricas

```
Definir os modelos
model_k_3 = KNeighborsClassifier(k=3)
model_k_4 = KNeighborsClassifier(k=4)
model_k_5 = KNeighborsClassifier(k=5)
model_k_6 = KNeighborsClassifier(k=6)
Model_k_7 = KNeighborsClassifier(k=7)

# Treinar os modelos no conjunto de treinamento limitado
model_k_3.fit(X_train_limited, y_train_limited)...

# Prever no conjunto de validação
y_pred_model_k_3 = model_k_3.predict(X_validation)...
# Avaliar a acurácia
accuracy = accuracy_score(y_validation, y_pred_model_k_3)
print(f'Acurácia: {accuracy}')
```

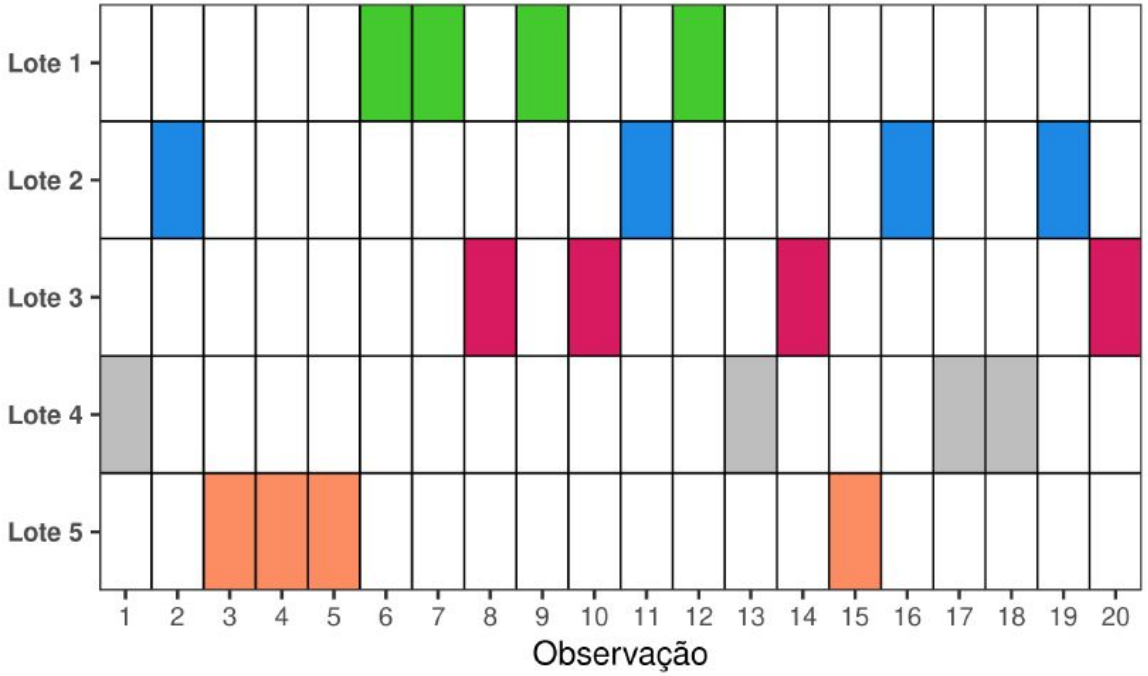
OPÇÕES DE VALIDAÇÃO CRUZADA

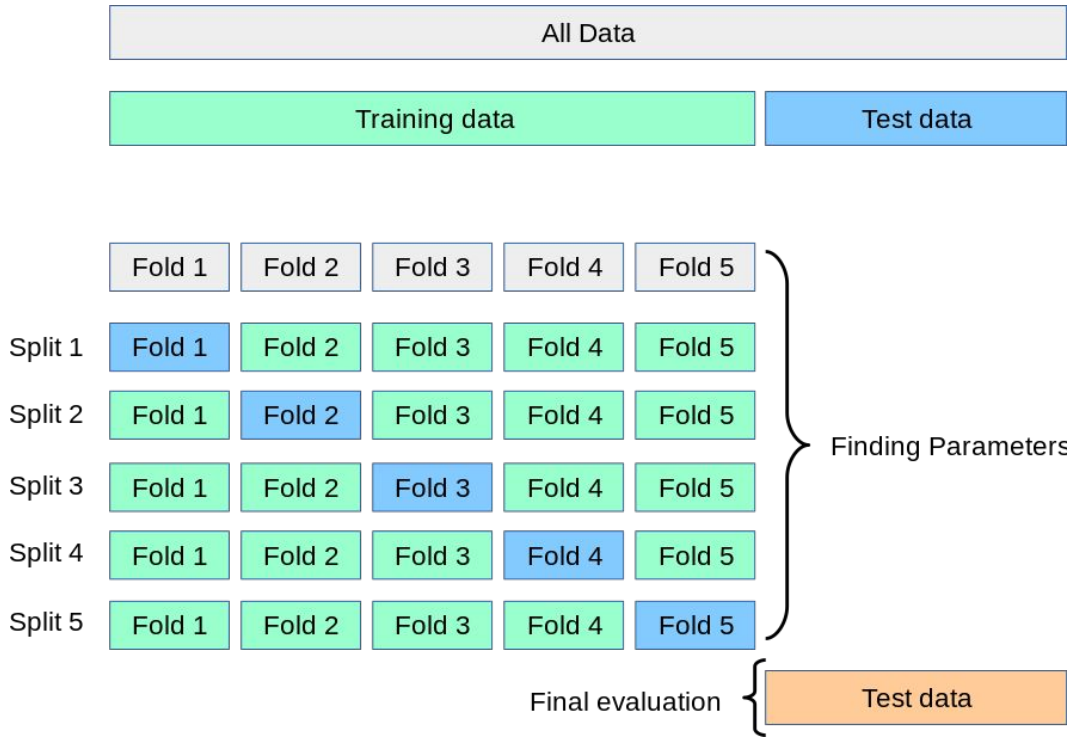
K-FOLD

- A ideia é usar vários conjuntos de validação pequenos, onde cada modelo é avaliado uma vez por conjunto de validação após ser treinado no restante dos dados. Tem-se então a média de todas as avaliações, mais preciso, embora mais custoso

- Permite também avaliar a variância da avaliação
- Modelo é treinado em k-1 folds e avaliado no que sobrou. Cada modelo temporário é descartado

5-fold: https://scikit-learn.org/stable/modules/cross_validation.html





K-FOLD: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html#sklearn.model_selection.cross_val_score

Exemplo com base didática Titanic

https://www.kaggle.com/competitions/titanic/data

A validação cruzada é para avaliar o modelo em dados não vistos e não para gerar um modelo final
Uma vez avaliado a métrica média, treina-se no conjunto completo e aplica-se ao conjunto de teste
Dica de Leitura: Como treinar o modelo final de machine learning?

https://github.com/josenalde/machinelearning/blob/main/src/cv_titanic_knn_cv.ipynb