Validação Cruzada

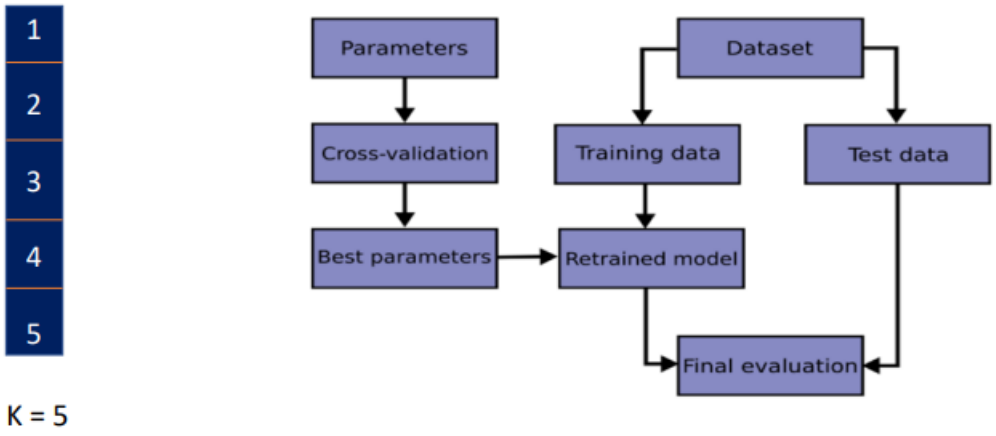
November 18, 2025

0.1 Validação Cruzada

Em termos simples, a Validação Cruzada é uma técnica usada para avaliar o desempenho de nossos modelos de aprendizado de máquina em dados não vistos. A validação cruzada é um procedimento de reamostragem usado para avaliar modelos de aprendizado de máquina em uma amostra de dados limitada.

[2]:
```
from IPython.display import Image
Image(filename='1_1.png')
```

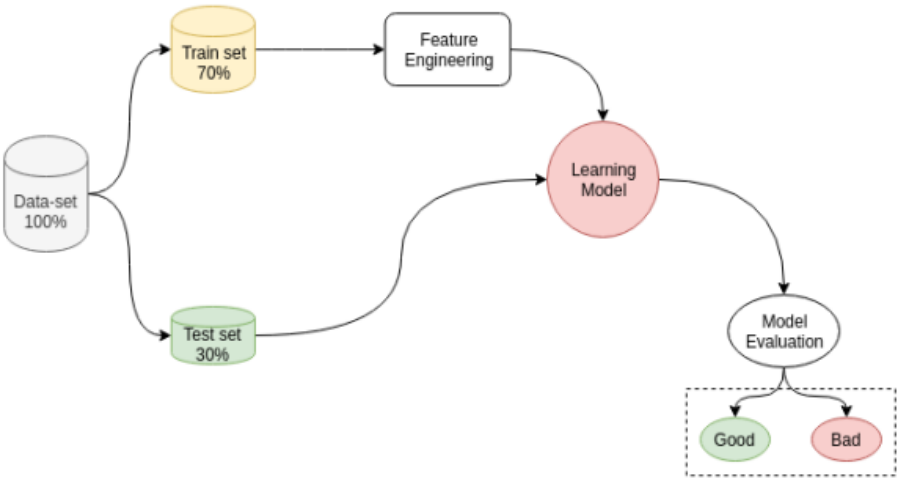
[2]:



Sempre que construímos um modelo de aprendizado de máquina, alimentamos o modelo com dados iniciais para treiná-lo. Em seguida, fornecemos alguns dados desconhecidos (dados de teste) para entender como o modelo se comporta e se generaliza em dados não vistos. Se o modelo se sair bem nos dados não vistos, ele é consistente e capaz de prever com boa precisão em uma ampla gama de dados de entrada; então, esse modelo é estável.

[3]:
```
Image(filename='2_1.png')
```

[3]:



Mas isso nem sempre é o caso!

Os modelos de aprendizado de máquina nem sempre são estáveis e precisamos avaliar a estabilidade do modelo. É aí que a Validação Cruzada entra em cena.

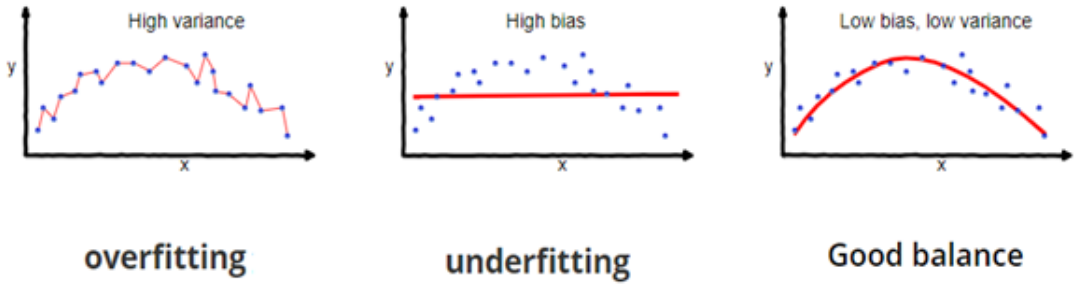
Suponha que construímos um modelo de aprendizado de máquina para resolver um problema e treinamos o modelo em um conjunto de dados. Quando verificamos a precisão do modelo nos dados de treinamento, ela é próxima de 95%. Isso significa que nosso modelo foi bem treinado e é o melhor modelo por causa da alta precisão?

Não, não é! Porque nosso modelo foi treinado nos dados fornecidos, ele conhece bem os dados, capturou até mesmo as menores variações (ruído) e se generalizou muito bem nos dados fornecidos. Se expusermos o modelo a dados completamente novos e não vistos, ele pode não prever com a mesma precisão e pode falhar em se generalizar nos novos dados. Esse problema é chamado de overfitting.

Às vezes, o modelo não treina bem no conjunto de treinamento, pois não consegue encontrar padrões. Nesse caso, ele também não terá um bom desempenho no conjunto de teste. Esse problema é chamado de underfitting.

[4]:
```
Image(filename='3_1.png')
```

[4]:



Para superar problemas de overfitting, usamos uma técnica chamada Validação Cruzada.

A Validação Cruzada é uma técnica de reamostragem com a ideia fundamental de dividir o conjunto de dados em duas partes: dados de treinamento e dados de teste. Os dados de treinamento são usados para treinar o modelo e os dados de teste não vistos são usados para previsão. Se o modelo se sair bem nos dados de teste e fornecer boa precisão, isso significa que o modelo não sofreu overfitting nos dados de treinamento e pode ser usado para previsão.

0.2 Tipos de Validação Cruzada

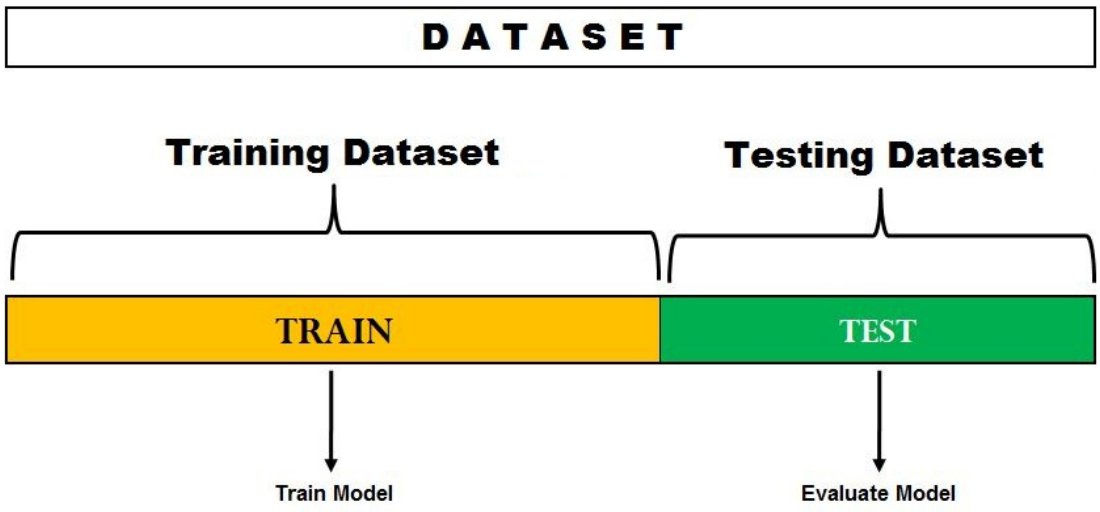
- Hold Out Cross Validation
- K-Fold Cross Validation
- Leave One-Out Cross Validation (LOOCV)
- Stratified K-Fold Cross Validation

0.2.1 Hold Out Cross Validation

Esta é uma abordagem bastante básica e simples na qual dividimos todo o nosso conjunto de dados em duas partes: dados de treinamento e dados de teste. Como o nome sugere, treinamos o modelo nos dados de treinamento e depois avaliamos no conjunto de teste. Normalmente, o tamanho dos dados de treinamento é mais do que o dobro dos dados de teste, então os dados são divididos na proporção de 70:30 ou 80:20.

[5]:
```
Image(filename='4_1.jpeg')
```

[5]:



Nesta abordagem, os dados são embaralhados aleatoriamente antes da divisão. Como o modelo é treinado em uma combinação diferente de pontos de dados, ele pode fornecer resultados diferentes a cada vez que o treinamos, e isso pode ser uma causa de instabilidade. Além disso, nunca podemos garantir que o conjunto de treinamento escolhido seja representativo de todo o conjunto de dados.

Quando nosso conjunto de dados não é muito grande, há uma alta possibilidade de que os dados de teste possam conter algumas informações importantes que perdemos, pois não treinamos o modelo no conjunto de teste.

O método hold-out é bom para usar quando você tem um conjunto de dados muito grande, está com pouco tempo ou está começando a construir um modelo inicial em seu projeto de ciência de dados.

0.2.2 K-Fold Cross Validation

A abordagem de validação cruzada K-Fold divide o conjunto de dados de entrada em K grupos de amostras de tamanhos iguais. Essas amostras são chamadas de folds. Para cada conjunto de aprendizado, a função de previsão usa k-1 folds, e o restante dos folds é usado para o conjunto de teste. Esta abordagem é muito popular porque é fácil de entender e o resultado é menos tendencioso do que outros métodos.

[6]:
```
Image(filename='5_1.png')
```

[6]:

Vamos generalizar o valor de K. Se K=5, significa que no conjunto de dados fornecido, estamos dividindo em 5 folds e executando o Treinamento e Teste. Durante cada execução, um fold é considerado para teste e o restante será para treinamento. A representação acima daria uma ideia do fluxo do tamanho definido pelo fold.

Em cada iteração, cada ponto de dados é usado uma vez no conjunto de teste e K-1 vezes no treinamento. Durante a iteração completa, pelo menos uma vez, um fold será usado para teste e o restante para treinamento.

No conjunto acima, 5 folds são usados para teste e 20 para treinamento. Em cada iteração, obteremos uma pontuação de precisão, somaremos todas e encontraremos a média. Aqui podemos entender como os dados estão distribuídos de forma consistente e concluir se o modelo está pronto para produção ou não.

0.2.3 Regras Básicas Associadas ao K-Fold

Agora, vamos discutir algumas regras básicas ao usar o K-Fold:

- K deve ser sempre >= 2 e igual ao número de registros (LOOCV).
  - Se K=2, teremos apenas 2 iterações.
  - Se K for igual ao número de registros no conjunto de dados, teremos 1 para teste e n-1 para treinamento.
- O valor otimizado para K é 10 e é usado com dados de bom tamanho (comumente usado).
- Se o valor de K for muito grande, isso levará a menor variância no conjunto de treinamento e limitará a diferença de desempenho do modelo nas iterações.
- O número de folds é inversamente proporcional ao tamanho do conjunto de dados, ou seja, se o conjunto de dados for muito pequeno, o número de folds pode aumentar.
- Valores maiores de K aumentam o tempo de execução do processo de validação cruzada.

0.2.4 Validação Cruzada K-Fold para os seguintes propósitos no aprendizado de máquina:

- Seleção de modelos
- Ajuste de parâmetros
- Seleção de características

[7]:
```
# Importando bibliotecas necessárias
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# Carregando o conjunto de dados de dígitos manuscritos
digits = load_digits()

# Dividindo o conjunto de dados em treinamento e teste (70% treinamento, 30%␣
 ↪teste)
X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target,␣
 ↪test_size=0.3)

# Regressão Logística
# Criando o modelo de Regressão Logística com solver 'liblinear' e multi_class␣
 ↪'ovr'
lr = LogisticRegression(solver='liblinear', multi_class='ovr')
# Treinando o modelo com os dados de treinamento
lr.fit(X_train, y_train)
# Avaliando o modelo com os dados de teste
print("Logistic Regression Score:", lr.score(X_test, y_test))

# Random Forest
# Criando o modelo de Random Forest com 40 estimadores
rf = RandomForestClassifier(n_estimators=40)
# Treinando o modelo com rf.fit(X_train, y_train)os dados de treinamento
rf.fit(X_train, y_train)
# Avaliando o modelo com os dados de teste
print("Random Forest Score:", rf.score(X_test, y_test))

# Usando a função cross_val_score para obter as pontuações, passando diferentes␣
 ↪algoritmos com o conjunto de dados e cv

# Ajuste de Parâmetros Usando K-Fold
from sklearn.model_selection import cross_val_score
```

```
# Usando CV=10 (comumente usado)
# Avaliando o modelo de Random Forest com 5 estimadores e CV=10
scores1 = cross_val_score(RandomForestClassifier(n_estimators=5), digits.data,␣
 ↪digits.target, cv=10)
print('Avg Score for Estimators=5 and CV=10:', np.average(scores1))

# Avaliando o modelo de Random Forest com 20 estimadores e CV=10
scores2 = cross_val_score(RandomForestClassifier(n_estimators=20), digits.data,␣
 ↪digits.target, cv=10)
print('Avg Score for Estimators=20 and CV=10:', np.average(scores2))
```

```
Logistic Regression Score: 0.9574074074074074
Random Forest Score: 0.975925925925926
Avg Score for Estimators=5 and CV=10: 0.8725822470515208
Avg Score for Estimators=20 and CV=10: 0.9371229050279328
```

[ ]: