Fundamentos de Data Science, Data Mining e Análise Preditiva

Especialização em Ciência de Dados com Big Data, BI e Data Analytics

Prof. Dr. Carlos Barros

Princípios data mining

- Introdução a Data Mining e Ciência dos Dados. Obtendo informações a partir dos dados. Principais Paradigmas e Modelos para mineração de dados. Dados incertos, com ruídos/outliers e confiança nos dados. Análise de dados exploratória. Introdução ao uso de modelos de predição. Escolha de modelos para mineração de dados. Redução de dimensionalidade e engenharia de dados. Minerando dados complexos. Trabalhando e limpando os Dados.

Pré-processamento de dados e análise exploratória de dados

Conteúdo

Base de dados

- Dados de crédito
- Censo

- Carregamento de bases de dados
- Valores inconsistente
- Valores faltantes
- Escalonamento de atributos
- Transformação de variáveis categóricas
- Introdução a avaliação de algoritmos
- Base de treinamento e base de teste
- Vários recursos do pandas (localizar, remover linhas e colunas, alterar valores)

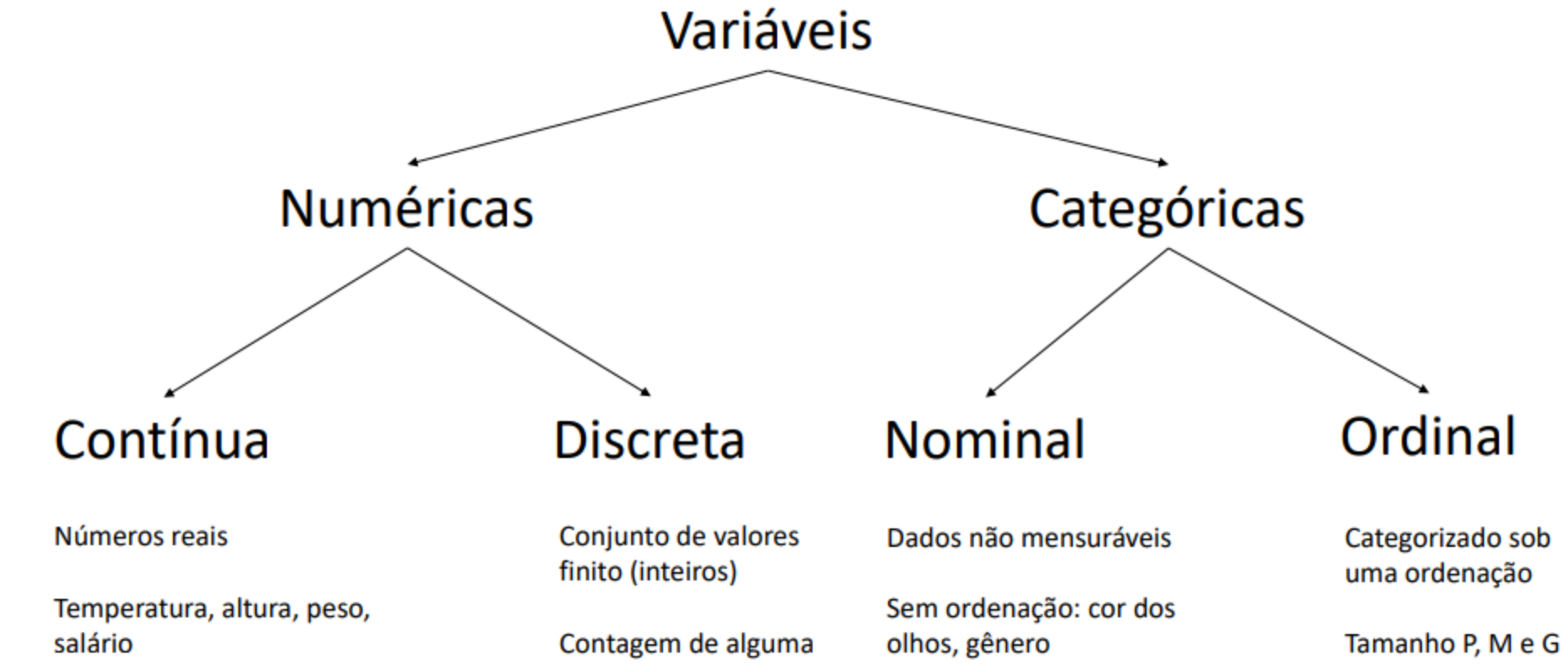
Tarefas

Classificação

Regressão

Agrupamentos

Regras de Associação


Variáveis
Numéricas
Categóricas
Contínua
Discreta
Nominal
Ordinal
Números reais
Temperatura, altura, peso, salário
Conjunto de valores finito (inteiros)
Contagem de alguma coisa
Dados não mensuráveis
Sem ordenação: cor dos olhos, gênero
Categorizado sob uma ordenação
Tamanho P, M e G

Base de Dados de credito

```
import pandas as pd
base = pd.read_csv('credit_data.csv')
base.describe()
base.loc[base['age'] < 0]
# apagar a coluna
base.drop('age', 1, inplace=True)
# apagar somente os registros com problema
base.drop(base[base.age < 0].index, inplace=True)
# preencher os valores manualmente
# preencher os valores com a média
base.mean()
base['age'].mean()
base['age'][base.age > 0].mean()
base.loc[base.age < 0, 'age'] = 40.92
```

Base de Dados de credito

- Tratamento de Valores faltantes

```
base.loc[base['age'] < 0]
# apagar a coluna
base.drop('age', 1, inplace=True)
# apagar somente os registros com problema
base.drop(base[base.age < 0].index, inplace=True)
# preencher os valores manualmente
# preencher os valores com a média
base.mean()
base['age'].mean()
base['age'][base.age > 0].mean()
base.loc[base.age < 0, 'age'] = 40.92

pd.isnull(base['age'])
base.loc[pd.isnull(base['age'])]

previsores = base.iloc[:, 1:4].values
classe = base.iloc[:, 4].values

from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)
imputer = imputer.fit(previsores[:, 0:3])
previsores[:,0:3] = imputer.transform(previsores[:,0:3])
```

Base de Dados de credito

- Escalonamento de atributos

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values='NaN', strategy='mean', axis=0)
imputer = imputer.fit(previsores[:, 0:3])
previsores[:,0:3] = imputer.transform(previsores[:,0:3])

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
previsores = scaler.fit_transform(previsores)
```

Padronização (Standardisation)

$$x = \frac{x - média(x)}{desvio\ padrão(x)}$$

Normalização (Normalization)

$$x = \frac{x - mínimo(x)}{máximo(x) - mínimo(x)}$$

Base de Dados do Censo

- Transformação de variáveis categóricas

```
import pandas as pd

base = pd.read_csv('census.csv')

previsores = base.iloc[:, 0:14].values
classe = base.iloc[:, 14].values

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_previsores = LabelEncoder()
#labels = labelencoder_previsores.fit_transform(previsores[:, 1])
previsores[:, 1] = labelencoder_previsores.fit_transform(previsores[:, 1])
previsores[:, 3] = labelencoder_previsores.fit_transform(previsores[:, 3])
previsores[:, 5] = labelencoder_previsores.fit_transform(previsores[:, 5])
previsores[:, 6] = labelencoder_previsores.fit_transform(previsores[:, 6])
previsores[:, 7] = labelencoder_previsores.fit_transform(previsores[:, 7])
previsores[:, 8] = labelencoder_previsores.fit_transform(previsores[:, 8])
previsores[:, 9] = labelencoder_previsores.fit_transform(previsores[:, 9])
previsores[:, 13] = labelencoder_previsores.fit_transform(previsores[:, 13])

onehotencoder = OneHotEncoder(categorical_features = [1,3,5,6,7,8,9,13])
previsores = onehotencoder.fit_transform(previsores).toarray()

labelencoder_classe = LabelEncoder()
classe = labelencoder_classe.fit_transform(classe)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
previsores = scaler.fit_transform(previsores)
```

Variáveis "Dummy"

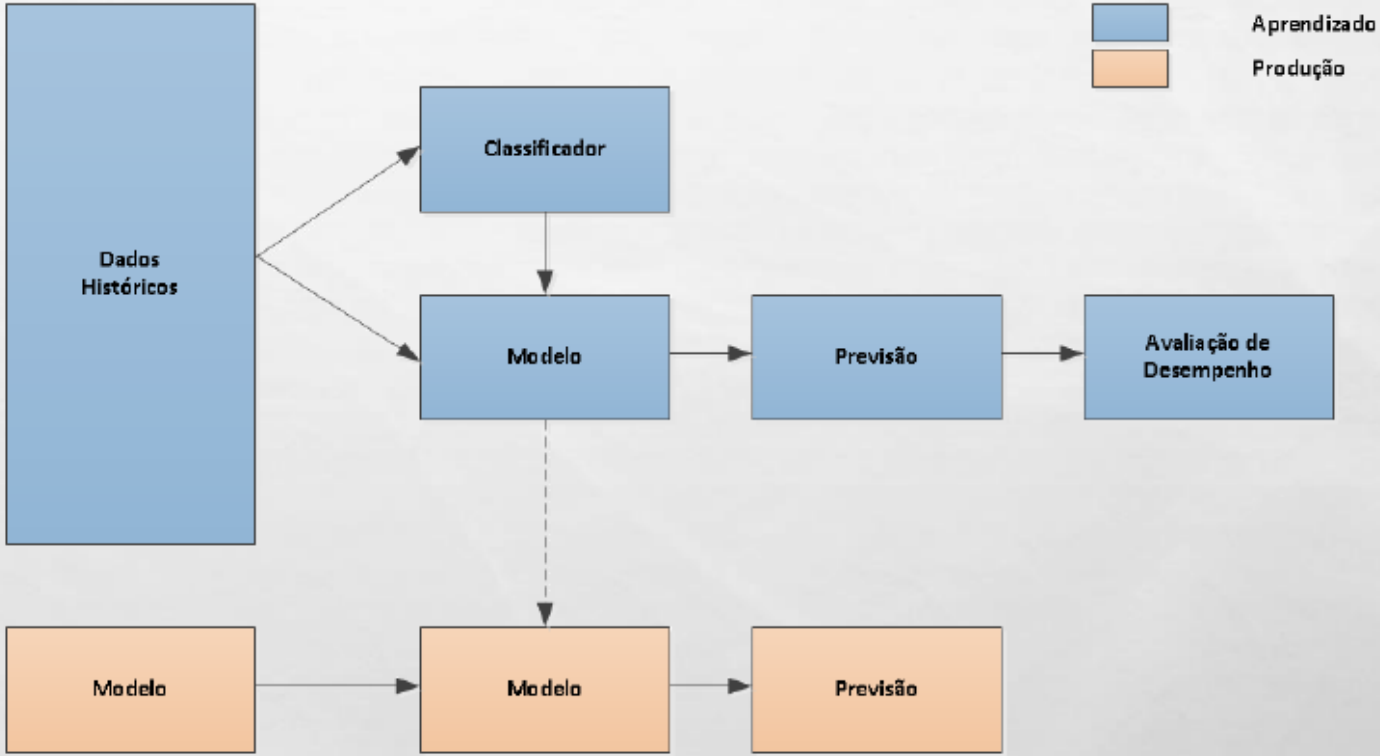
Race	White	Black	Asian-Pac-Islander	Amer-Indian-Eskimo
White	1	0	0	0
Black	0	1	0	0
Asian-Pac-Islander	0	0	1	0
Amer-Indian-Eskimo	0	0	0	1
Black	0	1	0	0
Black	0	1	0	0

Avaliação de Algoritmos –

-> Matriz de Confusão
-> Validação Cruzada
-> Holdout
-> GridSearch
-> Outliers
-> Overfitting e underfitting

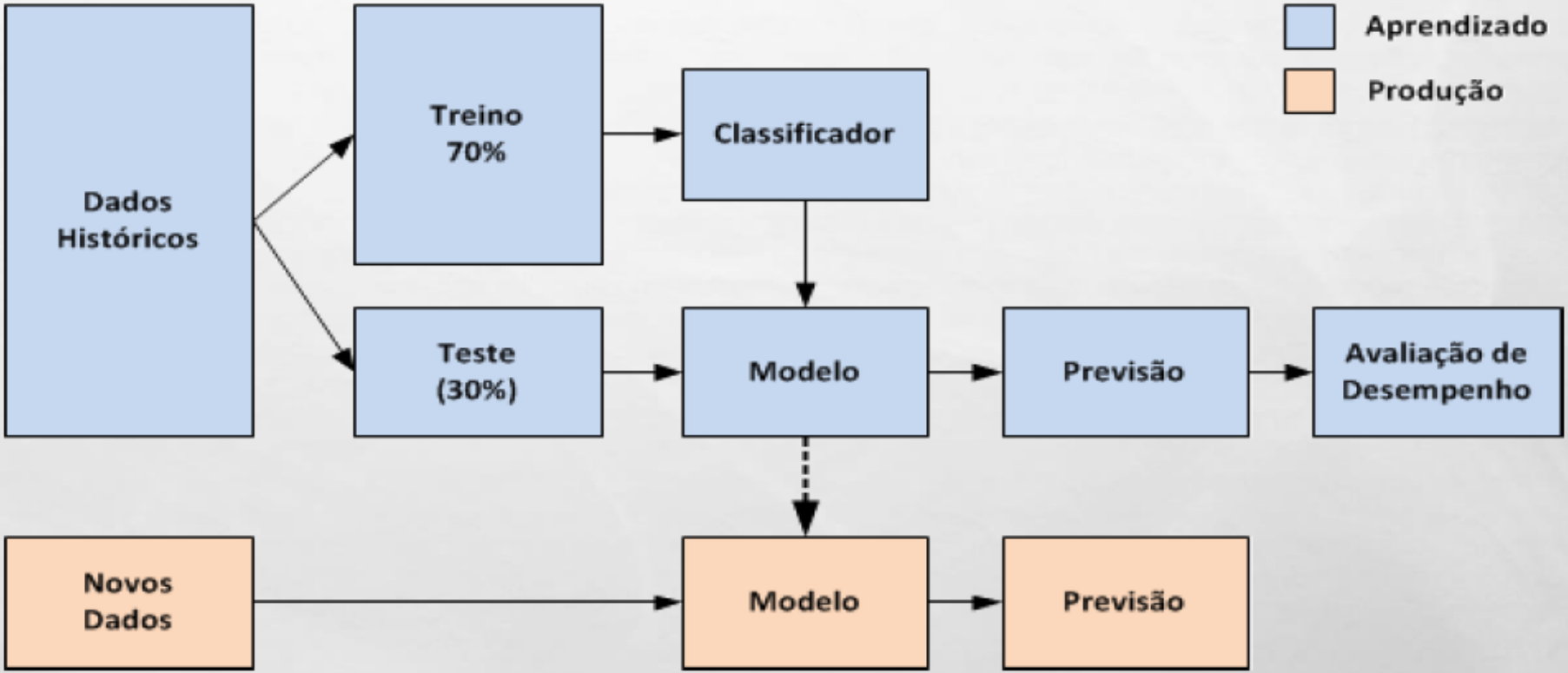
AVALIANDO O QUE FOI APRENDIDO

- **1 - USANDO MESMO CONJUNTO DE DADOS**



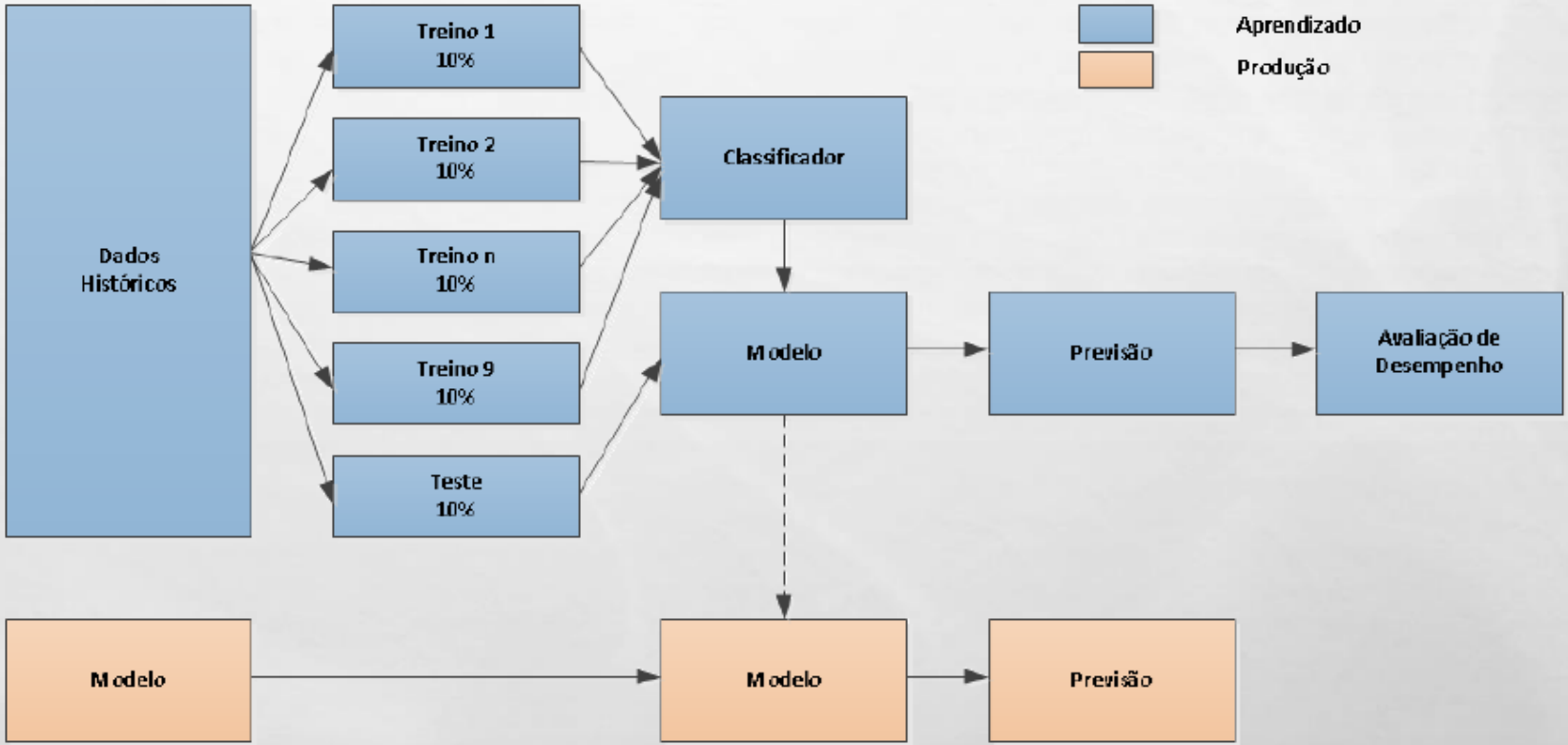
AVALIANDO O QUE FOI APRENDIDO

- **2 – HOLD OUT**



AVALIANDO O QUE FOI APRENDIDO

- **3 – VALIDAÇÃO CRUZADA**

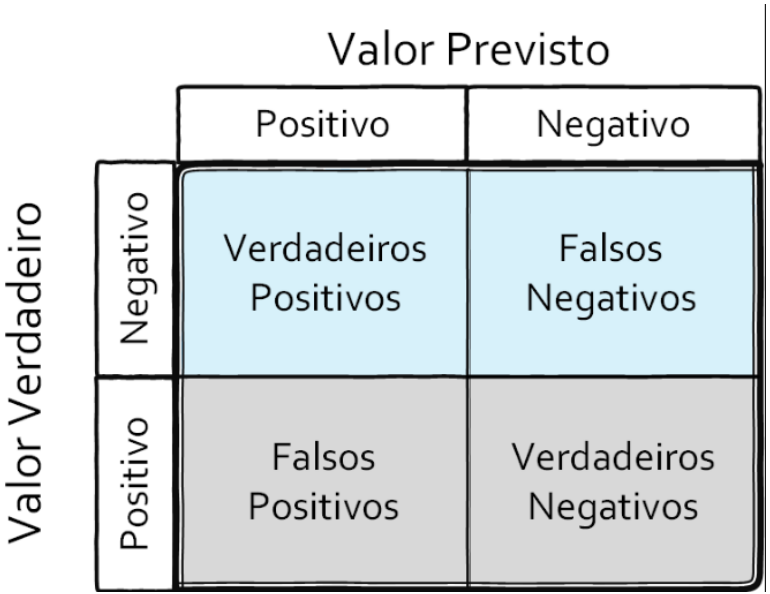


COMO “MELHORAR” UM MODELO?

- TESTANDO DIFERENTES ALGORITMOS
- PARAMETRIZANDO ALGORITMOS
- SELECIONANDO E TRATANDO DADOS
- SELEÇÃO DE ATRIBUTOS

Avaliação de Algoritmos –

-> Matriz de Confusão



MÉTRICAS

		Treino	
		Aprovado	Reprovado
Originais	Aprovado	1004	208
	Reprovado	202	294

Métrica	Fórmula	Descrição	Cálculo
Acertos	(VP+VN)/Total	Total de Acertos	75,99
Erros	(FP+FN)/Total	Total de Erros	24,00
Precisão	VP/(VP+FP)	Quantos registros de fato são positivos	83,25
Lembrança ou Positivos Verdadeiros	VP/(VP+FN)	Positivos corretamente previstos	82,83
Negativos Verdadeiros	VN/(VN+FP)	Total de Negativos Verdadeiros	59,27
Positivos Falsos	FP/(VN+FP)	Total de Positivos Falsos	40,72
Negativos Falsos	FN/(VP+FN)	Total de Negativos Falsos	17,16

Métricas de Erros

Previsão de valores numéricos (reais, inteiros)

Métricas diferentes da previsão de categorias

Uso:

- Regressão clássica
- Regressão ML
- Series Temporais
- Etc.

Mean Erro (ME)

Dependente de Escala

A média da diferença entre realizado e previsto

Previsto	Realizado	Dif.
3,34	3,00	-0,34
4,18	4,00	-0,18
3,00	3,00	0
2,99	3,00	0,01
4,51	4,50	-0,01
5,18	4,00	-1,18
8,18	4,50	-3,68

$$MAE = \sum_{I=1}^{N} \frac{p_i - t_i}{n}$$

$$ME = \frac{-5,38}{7} = -0,76$$

Mean Absolute Erros (MAE)

Dependente de Escala

A média da diferença absoluta entre o realizado e o previsto

Previsto	Realizado	Dif. Absoluta
3,34	3,00	|0,34|
4,18	4,00	|0,18|
3,00	3,00	|0|
2,99	3,00	|0,01|
4,51	4,50	|0,01|
5,18	4,00	|1,18|
8,18	4,50	|3,68|
		5,4

$$MAE = \sum_{I=1}^{N} \frac{|p_i - t_i|}{n}$$

$$MAE = \frac{5,4}{7} = 0,77$$

Root Mean Squared Error (RMSE)

Independente de Escala

O desvio padrão da amostra da diferença entre o previsto e o teste

Previsto	Realizado	Dif. ao Quad.
3,34	3,00	0,1156
4,18	4,00	0,0324
3,00	3,00	0
2,99	3,00	1E-04
4,51	4,50	1E-04
5,18	4,00	1,3924
8,18	4,50	13,5424

$$RMSE = \sqrt{\frac{\sum_{I=1}^{N}(p_i - t_i)^2}{N}}$$

$$RMSE = \sqrt{\frac{15{,}083}{7}}$$

$$RMSE = 1{,}46$$

Mean Percentage Error (MPE)

Independente de Escala (%)

Diferença percentual de erro

Previsto	Realizado	Erro %
3,34	3,00	-11,3333
4,18	4,00	-4,5
3,00	3,00	0
2,99	3,00	0,333333
4,51	4,50	-0,22222
5,18	4,00	-29,5
8,18	4,50	-81,7778

$$\text{MPE} = \frac{\sum_{I=1}^{N} \frac{(t_i - p_i)}{t_i - 100}}{N}$$

$$\text{MPE} = \frac{-127}{7}$$

$$\text{MPE} = -18{,}14$$

Mean Absolute Percentage Error (MAPE)

Independente de Escala (%)

Diferença absoluta percentual de erro

Previsto	Realizado	Erro abs.	Erro % abs.
3,34	3,00	0,1156	0,1133333
4,18	4,00	0,0324	0,045
3,00	3,00	0	0
2,99	3,00	1E-04	0,0033333
4,51	4,50	1E-04	0,0022222
5,18	4,00	1,3924	0,295
8,18	4,50	13,5424	0,8177778

%

$$MAPE = \frac{\sum_{I=1}^{N} \frac{|p_i - t_i|}{|t_i|}}{N}$$

$$MAPE = \frac{1{,}2766667}{7}$$

$$MAPE = 0{,}18$$

Previsto	Realizado	Diferença	Dif. Abs.	Dif. Quad.	Erro %	Erro % abs
3,34	3	-0,34	0,34	0,1156	-11,3333	11,33333
4,18	4	-0,18	0,18	0,0324	-4,5	4,5
3	3	0	0	0	0	0
2,99	3	0,01	0,01	1E-04	0,33333	0,333333
4,51	4,5	-0,01	0,01	1E-04	-0,22222	0,222222
5,18	4	-1,18	1,18	1,3924	-29,5	29,5
8,18	4,5	-3,68	3,68	13,5424	-81,7778	81,77778

ME	-0,76857
MAE	0,77143
RMSE	1,46789
MPE	-18,1429
MAPE	18,2381

Outliers

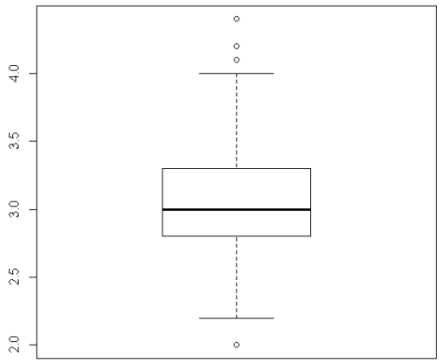
-> Valores fora do padrão: Afastado dos demais elementos- média
-> Várias formas de encontrá-los

Valores Anormais podem ser:

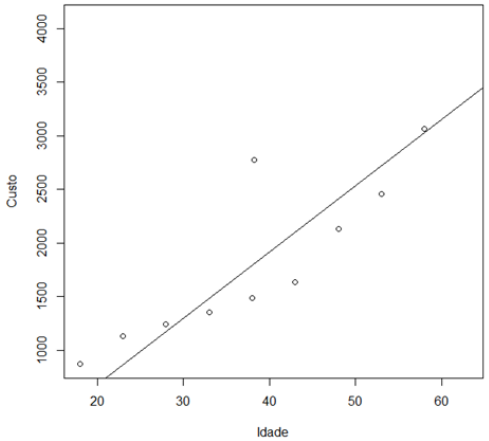
- Natural
- Erros
- Ruídos
- Exceções
- Fraudes

Outliers

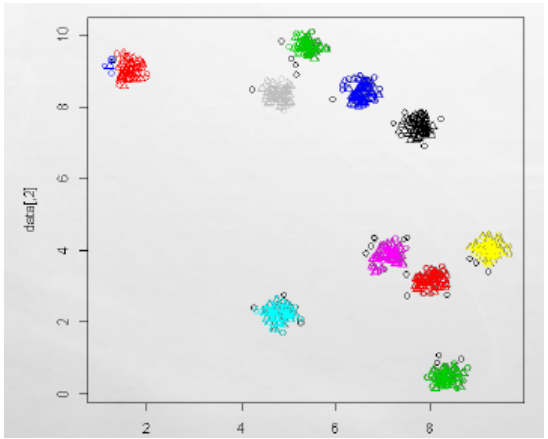
- Única variável



- Duas variáveis



- N variáveis



- Variáveis categóricas

Outliers – O que fazer?

- • Remover o registro
- • Não fazer nada
- • Substituir

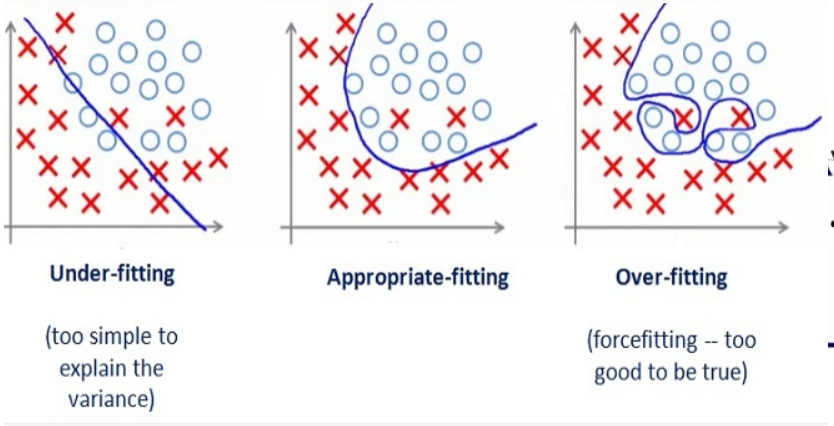
```python
import matplotlib.pyplot as plt
import pandas as pd
from pyod.models.knn import KNN

iris = pd.read_csv('iris.csv')

plt.boxplot(iris.iloc[:,1], showfliers = True)
outliers = iris[(iris['sepal width'] > 4.0) | (iris['sepal width'] < 2.1)]

sepal_width = iris.iloc[:,1]
sepal_width = sepal_width.reshape(-1,1)
detector = KNN()
detector.fit(sepal_width)

previsoes = detector.labels_
```

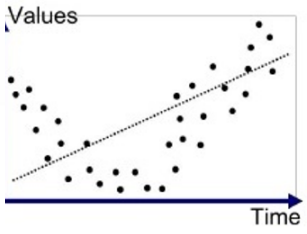
```python
import pandas as pd

base = pd.read_csv('credit_data.csv')
base = base.dropna()

# outliers idade
import matplotlib.pyplot as plt
plt.boxplot(base.iloc[:,2], showfliers = True)
outliers_age = base[(base.age < -20)]

# outliers loan
plt.boxplot(base.iloc[:,3])
outliers_loan = base[(base.loan > 13400)]
```

```python
import pandas as pd

base = pd.read_csv('credit_data.csv')
base = base.dropna()
base.loc[base.age < 0, 'age'] = 40.92

# income x age
import matplotlib.pyplot as plt
plt.scatter(base.iloc[:,1], base.iloc[:,2])

# income x loan
plt.scatter(base.iloc[:,1], base.iloc[:,3])

# age x loan
plt.scatter(base.iloc[:,2], base.iloc[:,3])

base_census = pd.read_csv('census.csv')

# age x final weight
plt.scatter(base_census.iloc[:, 0], base_census.iloc[:,2])
```

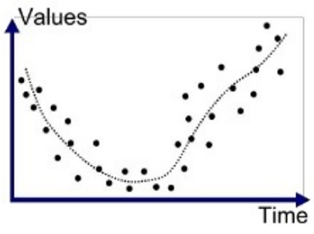
```python
import pandas as pd

base = pd.read_csv('credit_data.csv')
base = base.dropna()

from pyod.models.knn import KNN
detector = KNN()
detector.fit(base.iloc[:,1:4])

previsoes = detector.labels_
confianca_previsoes = detector.decision_scores_

outliers = []
for i in range(len(previsoes)):
    #print(previsoes[i])
    if previsoes[i] == 1:
        outliers.append(i)

lista_outliers = base.iloc[outliers, :]
```

Overfitting-e-underfitting

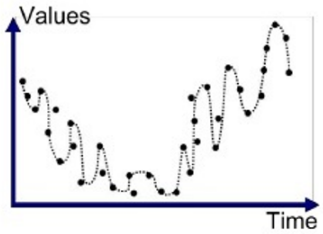




Underfitted



Good Fit/Robust



Overfitted

Overfitting-e-underfitting

Underfitting
- Resultados ruins na base de treinamento

Overfitting
- Resultados bons na base de treinamento
- Resultados ruins na base de teste
- Muito específico
- Memorização
- Erros na variação de novas instâncias

Avaliação de Algoritmos

```
import pandas as pd

base = pd.read_csv('credit_data.csv')
base.loc[base.age < 0, 'age'] = 40.92

previsores = base.iloc[:, 1:4].values
classe = base.iloc[:, 4].values

from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(previsores[:, 1:4])
previsores[:, 1:4] = imputer.transform(previsores[:, 1:4])

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
previsores = scaler.fit_transform(previsores)

from sklearn.cross_validation import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento, classe_teste = train_test_split(previsores, classe, test_size=0.25, random_state=0)

# importação da biblioteca
# criação do classificador
classificador.fit(previsores_treinamento, classe_treinamento)
previsoes = classificador.predict(previsores_teste)

from sklearn.metrics import confusion_matrix, accuracy_score
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)
```

Avaliação de Algoritmos

```
import pandas as pd

base = pd.read_csv('census.csv')

previsores = base.iloc[:, 0:14].values
classe = base.iloc[:, 14].values

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_previsores = LabelEncoder()
previsores[:, 1] = labelencoder_previsores.fit_transform(previsores[:, 1])
previsores[:, 3] = labelencoder_previsores.fit_transform(previsores[:, 3])
previsores[:, 5] = labelencoder_previsores.fit_transform(previsores[:, 5])
previsores[:, 6] = labelencoder_previsores.fit_transform(previsores[:, 6])
previsores[:, 7] = labelencoder_previsores.fit_transform(previsores[:, 7])
previsores[:, 8] = labelencoder_previsores.fit_transform(previsores[:, 8])
previsores[:, 9] = labelencoder_previsores.fit_transform(previsores[:, 9])
previsores[:, 13] = labelencoder_previsores.fit_transform(previsores[:, 13])

onehotencoder = OneHotEncoder(categorical_features = [1,3,5,6,7,8,9,13])
previsores = onehotencoder.fit_transform(previsores).toarray()

labelencoder_classe = LabelEncoder()
classe = labelencoder_classe.fit_transform(classe)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
previsores = scaler.fit_transform(previsores)

from sklearn.cross_validation import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento, classe_teste = train_test_split(previsores, classe, test_size=0.15, random_state=0)

# importação da biblioteca
# criação do classificador
classificador.fit(previsores_treinamento, classe_treinamento)
previsoes = classificador.predict(previsores_teste)

from sklearn.metrics import confusion_matrix, accuracy_score
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)
```

EDA- Análise Exploratória de Dados

Análise Exploratória de Dados – AED
John Wilder Tukey em 1977- EDA

Busca Obter:

1. Variação
2. Anomalias
3. Distribuições
4. Tendências
5. Padrões
6. Relações

Iniciar uma analise de dados pela EDA,
Só então decidir como buscar a solução para o problema

EDA vs Gráficos

- Não são a mesma coisa, porém eda é altamente baseado na produção de gráficos
- Gráficos de dispersão, box plots, histogramas etc