Pré-processamento de dados

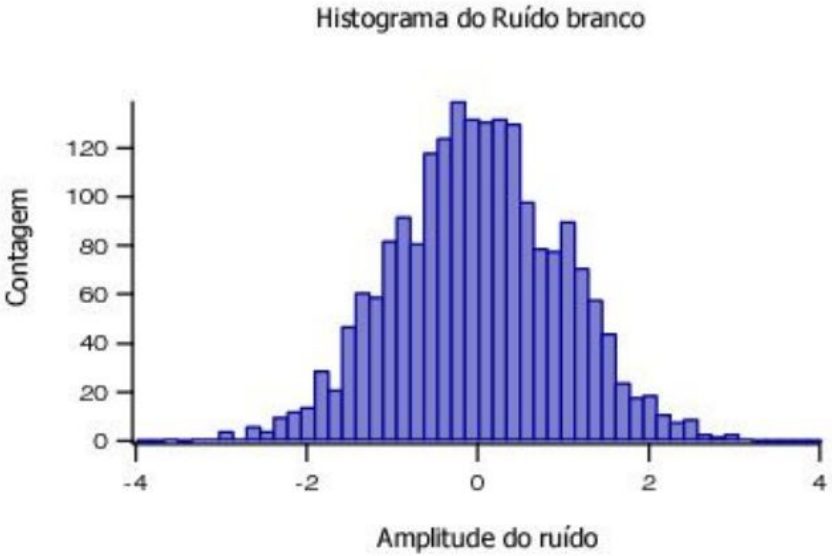
Jacqueline Midlej

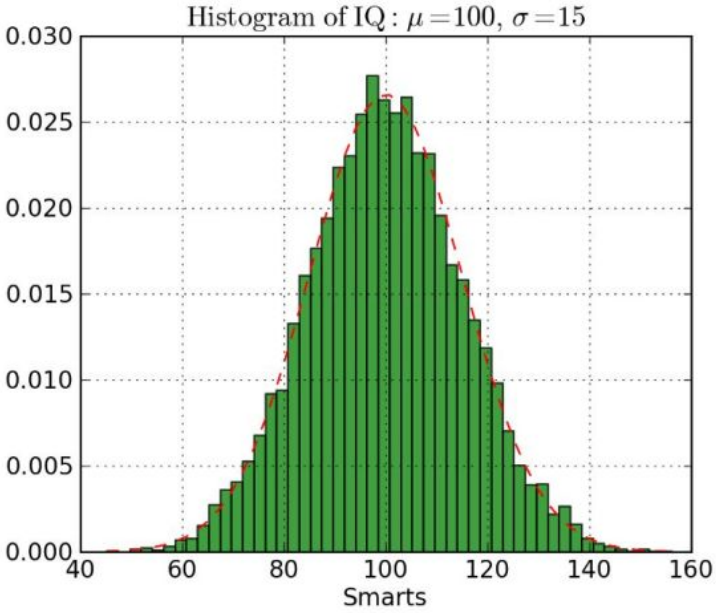
Tópicos

- Data Cleansing
- Data encoding
- Scaling and Normalization
- Feature selection

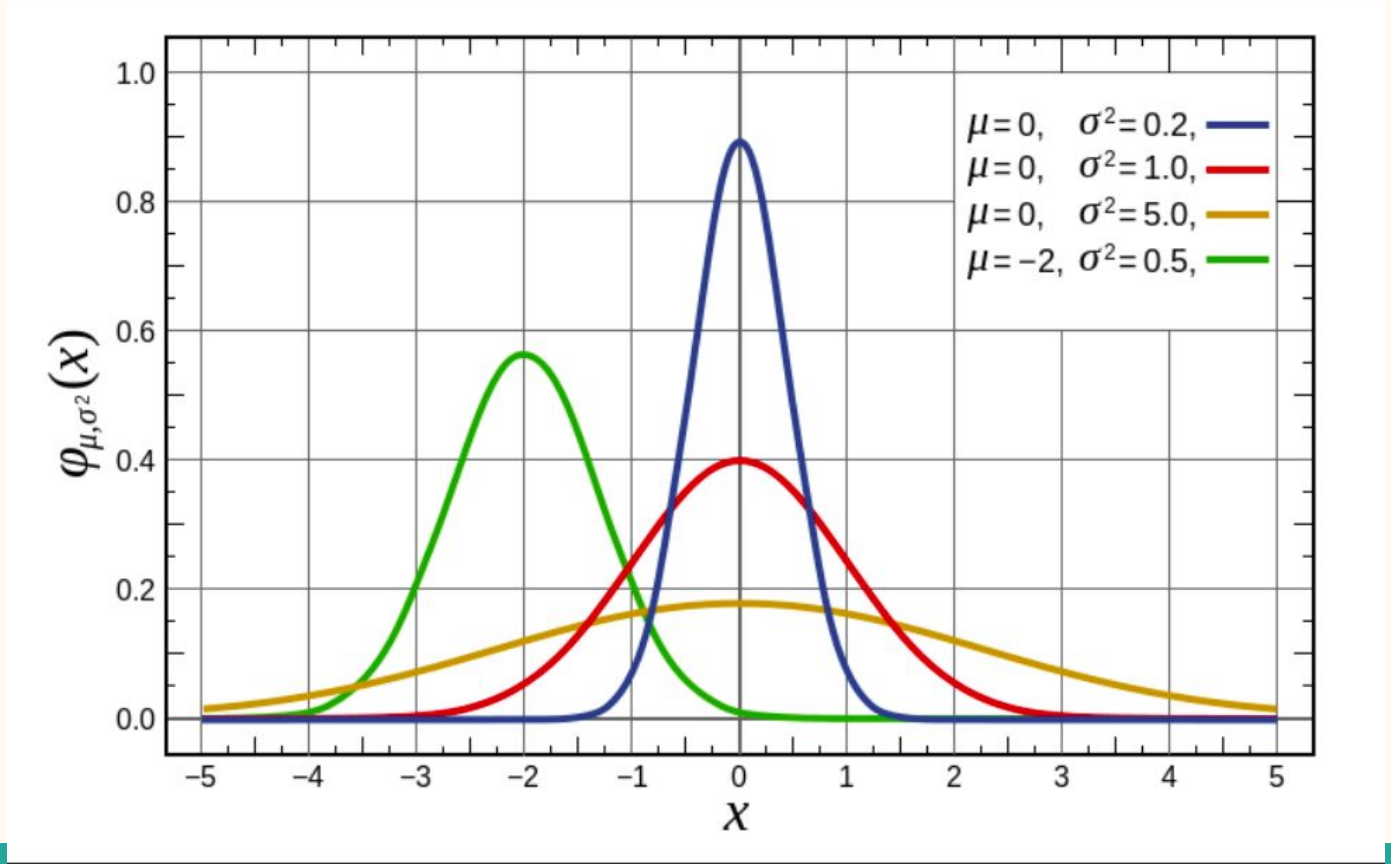
Conceitos estatísticos básicos

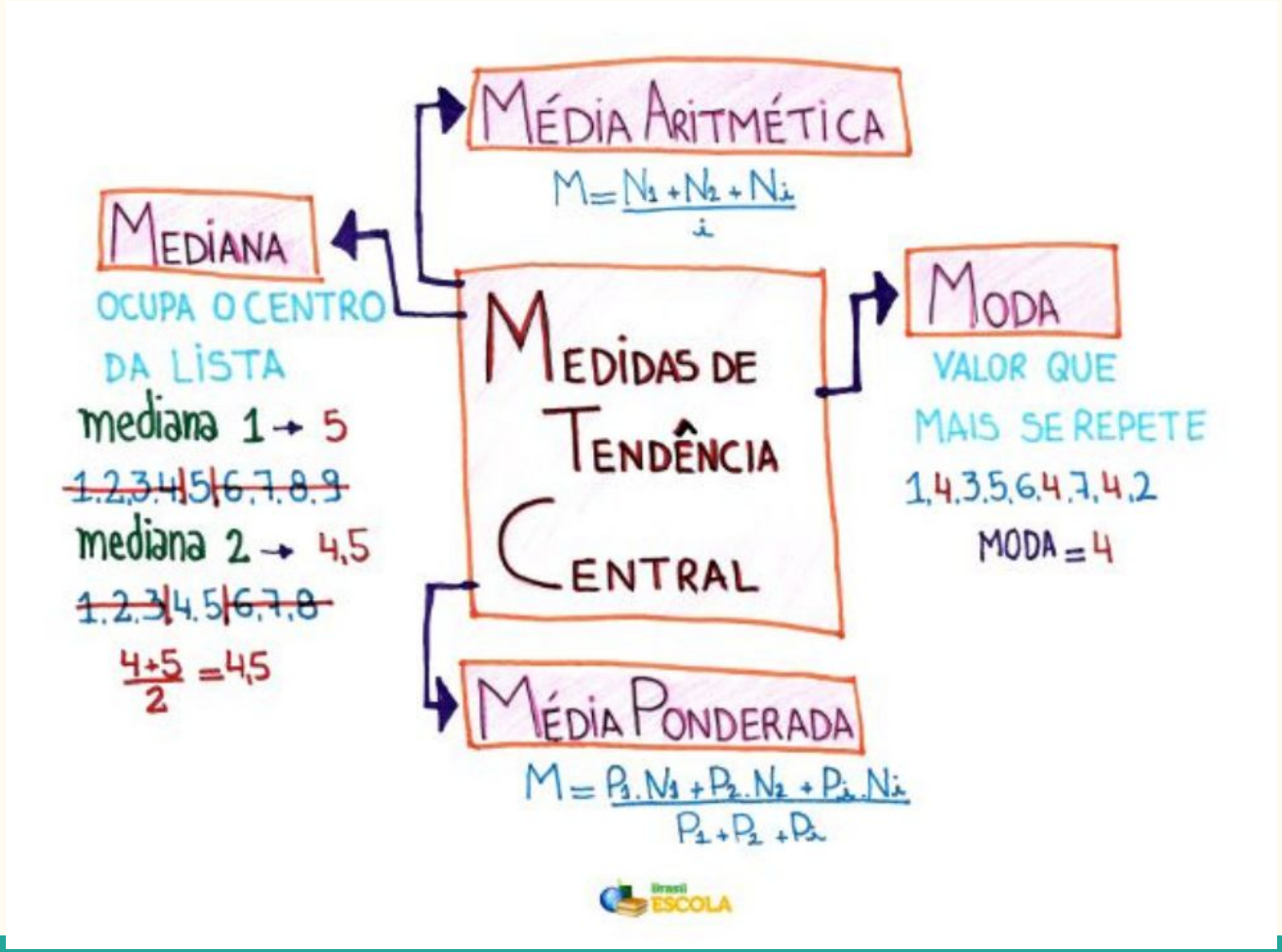
Histograma - Frequência absoluta ou relativa





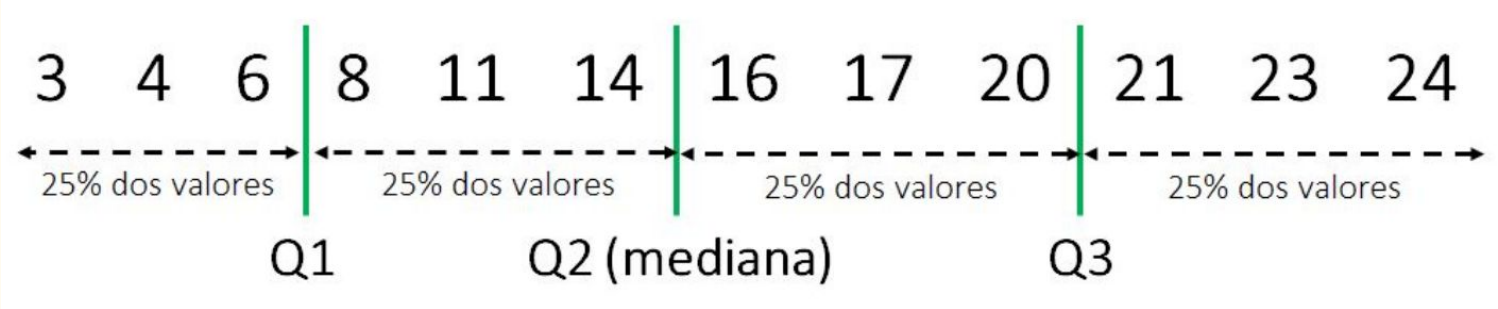
Distribuição normal ou gaussiana




MÉDIA ARITMÉTICA
$M = \frac{N_1 + N_2 + N_i}{i}$
MEDIANA
OCUPA O CENTRO DA LISTA
mediana 1 → 5
1.2.3.4.5.6.7.8.9
mediana 2 → 4,5
1.2.3.4.5.6.7.8
$\frac{4+5}{2} = 4,5$
MEDIDAS DE TENDÊNCIA CENTRAL
MODA
VALOR QUE MAIS SE REPETE
1,4,3,5,6,4,7,4,2
MODA = 4
MÉDIA PONDERADA
$M = \frac{P_1.N_1 + P_2.N_2 + P_i.N_i}{P_1 + P_2 + P_i}$
Brasil ESCOLA

Quartis

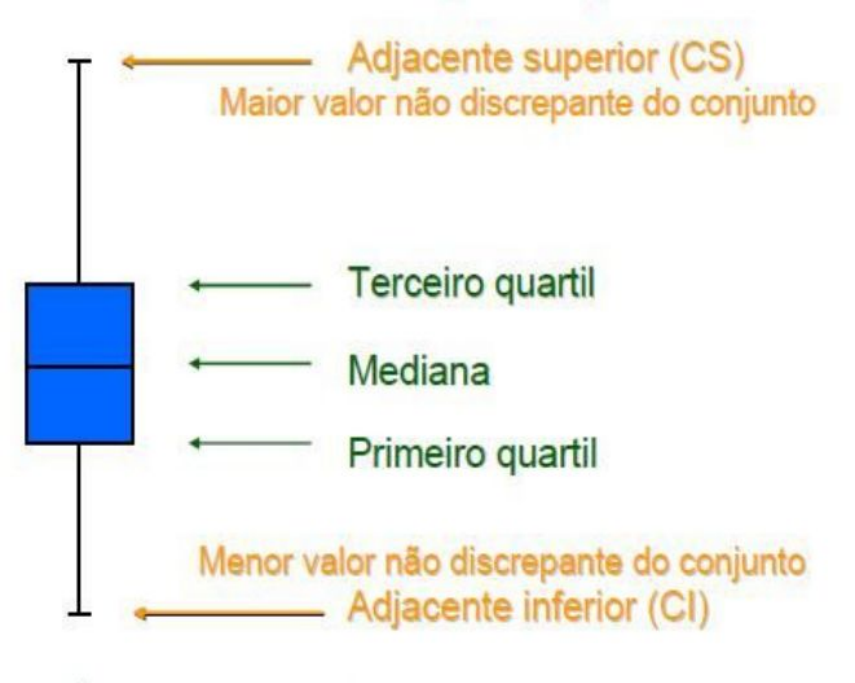
- medidas separatrizes do conjunto de dados de 25 em 25%



- Percentil: medida separatriz em qualquer valor de percentual, ex 5% e 95%

Boxplot

- Destaca outliers
- Evidencia distribuição (assim como histograma)
- Mais fácil para comparar quando temos diferentes variáveis

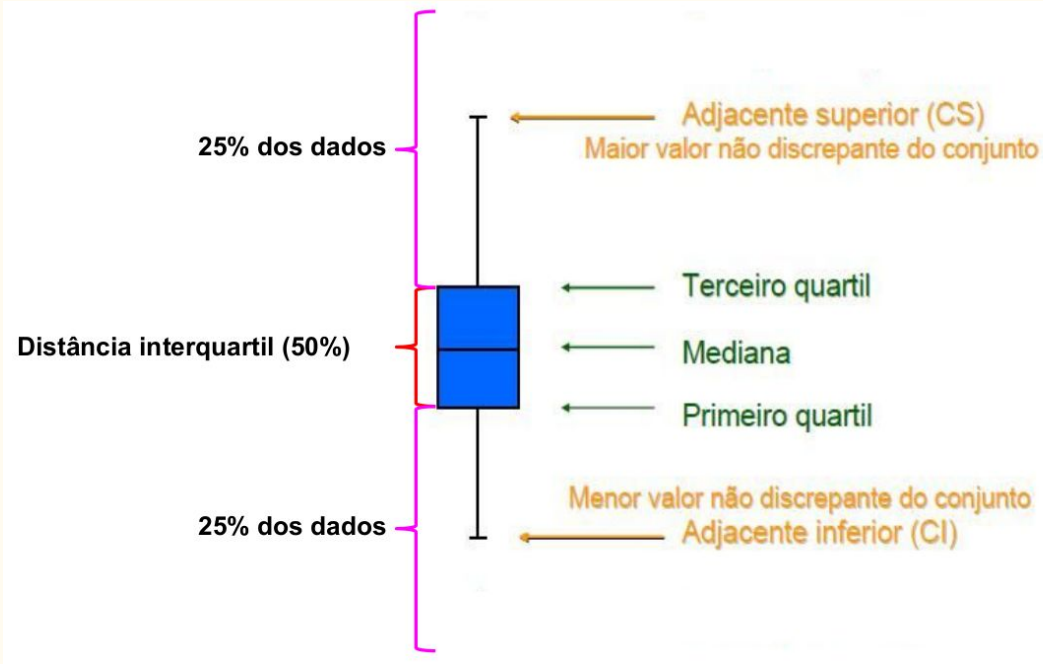


Boxplot

CS e C1 são baseados na Distância Interquartil (IQR)

Padrão: 1.5*IQR

Dados acima ou abaixo, são considerados outliers



Data Cleansing

Data Cleaning

- **Objetivo:** Corrigir e/ou remover dados inconsistentes

- Remover duplicatas:
  - Pode ocorrer quando concatena dados de diferentes fontes
  - Quando combina dados em diferentes escalas
    - Medidas capturadas por dia x por semana

data	temperatura
01/01	10ºC
02/01	12ºC
03/01	14ºC

data	mortes
01/01	2
01/01	3
04/01	6

data	temperatura	mortes
01/01	10ºC	2
01/01	10ºC	3
02/01	12ºC	
03/01	14ºC	
04/01		6

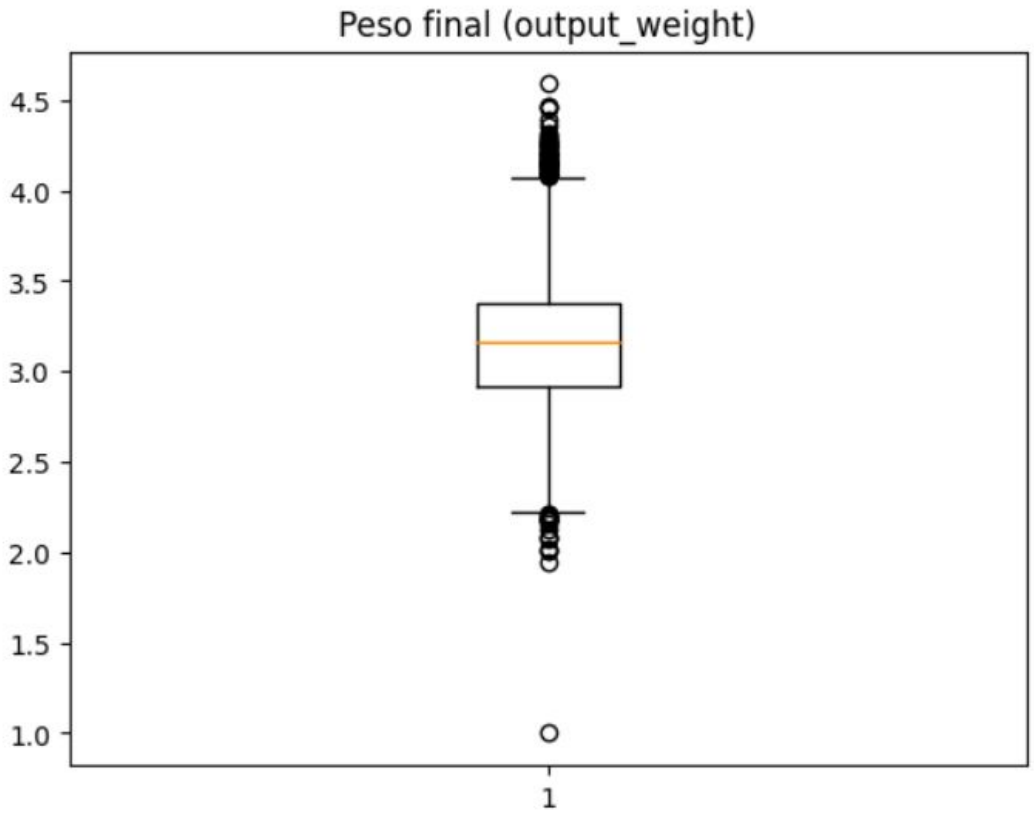
Data Cleaning

- Remover colunas irrelevantes (nem todos os dados são relevantes para o modelo)
- Correção de erros estruturais:
  - Mesma categoria/valor escrito de formas diferentes: “N/A” e “Not Applicable” significam mesma coisa.
  - Formatação de datas distintas: 00-00-9999 x 00/00/9999
  - Ignorar maiúsculas e minúsculas
- Filtrar outliers
  - Limites de corte estabelecido pela aplicação
  - Limites de corte baseado na distribuição dos dados (boxplot)
  - Observar percentual cortado, podemos ter contexto com abundância de dados, outras não
- Lidar com dados faltantes
  - Remover dados
  - Preencher
- Transformação de dados
  - Agrupamento em outras escalas

Filtrar outliers

- Boxplot
- 1.5*IQR

- Limites são 2.22 e 4.07




```
Range do boxplot : 2.22 - 4.07
Mediana : 3.17
Quartis : 2.22 - 2.91 - 3.38 - 4.07
```

Filtrar outliers

```
import matplotlib.pyplot as plt

box = plt.boxplot(df_filtered['output_weight'])
plt.title('Peso final (output_weight)')
plt.show()

# extraindo valores do gráfico de boxplot
v_min = [item.get_ydata()[0] for item in box['caps']][::2][0]
v_max = [item.get_ydata()[0] for item in box['caps']][1::2][0]
median = [item.get_ydata()[0] for item in box['medians']][0]
box_min = [item.get_ydata()[0] for item in box['whiskers']][::2][0]
box_max = [item.get_ydata()[0] for item in box['whiskers']][1::2][0]

print(f'Range do boxplot : {round(v_min, 2)} - {round(v_max, 2)}')
print(f'Mediana : {round(median, 2)}')
print(f'Quartis : {round(v_min,2)} - {round(box_min,2)} - {round(box_max,2)} -
{round(v_max,2)}')
```

Filtrar outliers

```
# corte

df_filtered=df_filtered[(df_filtered['output_weight']>=v_min) &

                    (df_filtered['output_weight']<=v_max)

                ]



# mostrando volume restante e percentual de corte

print('Volume de lotes: ', len(df_filtered), ' -- Corte de',
round((len(df)-len(df_filtered))/len(df)*100, 2)
```

_____

Volume de lotes: 15509 -- Corte de 4.09

Filtrar outliers

Outra maneira de calcular o valor máximo e mínimo, usando mesma estratégia, porém sem mostrar gráfico:

```
import numpy as np

q75,q25 = np.percentile(df_filtered['output_weight'],[75,25])
intr_qr = q75-q25

v_max = q75+(1.5*intr_qr)
v_min = q25-(1.5*intr_qr)


# corte
df_filtered=df_filtered[(df_filtered['output_weight']>=v_min) &
                        (df_filtered['output_weight']<=v_max)
                    ]
```

Filtrar outliers

- Outras maneiras:
  - Flexibilizar ou estreitar métrica do boxplot
    - Escolher fator que multiplica pelo IQR (ex de 1.5 para 2.5)
  - Z-score:
    - `z_score = (data_point -mean) / std. deviation`
    - também é uma métrica de distribuição
    - Geralmente usa-se valor 3.0 para corte. Isso significa cortar dados que são distantes da média em pelo menos 3*desvio padrão.
    - Abordagens mais flexíveis usam 5

Dados faltantes

- Removendo
  - Remove linha que tenha algum valor NaN em qualquer coluna: `df = df.dropna()`

Drop colunas nas quais **todos** os elementos sejam `nan`:

```
df.dropna(axis=1, how='all')
    A    B  D
0  NaN  2.0  0
1  3.0  4.0  1
2  NaN  NaN  5
```

Drop linhas nas quais **todos** os elementos sejam `nan` (nesse caso, não temos nenhuma):

```
df.dropna(axis=0, how='all')
     A    B   C  D
0  NaN  2.0 NaN  0
1  3.0  4.0 NaN  1
2  NaN  NaN NaN  5
```

Dados faltantes

- Removendo

No contexto abaixo, se existe mortalidade = 0 na base, os produtores não fizeram o lançamento, pois tal cenário é impossível. Portanto: dado faltante!

```
for var in ['mortality', 'output_age', 'density_stocking', 'weight_avg_by_age']:

    df_filtered.dropna(subset=[var], inplace=True)

    df_filtered=df_filtered[df_filtered[var]>0]

    print('Número de lotes com',var,'> 0:', len(df_filtered), ' -- Corte de ',
round((len(df)-len(df_filtered))/len(df)*100, 2))
```

Número de lotes com mortality > 0: 15741 -- Corte de 2.65

Dados faltantes

- Preenchendo
  - Mean/Median Imputation:

```
# Mean imputation
df3 = df2.fillna(df2.Age.mean())
```

    - Mediana é mais robusto, pois menos sensível a outliers
  - Randomicamente:
    - Se distribuição normal, preencher com números randômicos gerados ao redor da média

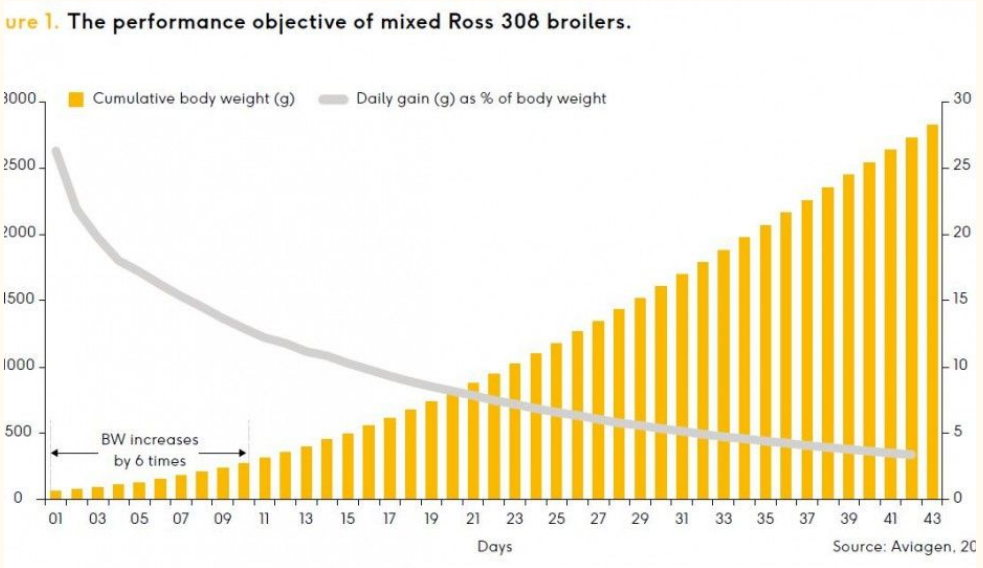
```
rand = np.random.randint(average_age - 2*std_age, average_age + 2*std_age, size = count_nan_age)
dataframe["age"][np.isnan(dataframe["age"])] = rand
```

Dados faltantes

- Preenchendo
  - Linear Regression:
    - Se existe correlação entre a variável com dados faltantes e a variável predita
  - Hot-deck:
    - Cópia de valores de outras amostras similares
      - Dividir em 2 grupos, exemplo, animais macho e fêmea. Preencher com mediana/média ou randômico dentro de cada classe.
      - ***k* nearest** neighbour imputation – encontra as k amostras mais perto para imputar o dado
      - Média/Mediana móvel em séries temporais - se bem comportada
  - Flag
    - Tratar com NaNs que são ignorados em quaisquer análises estatística
    - Preencher com 0

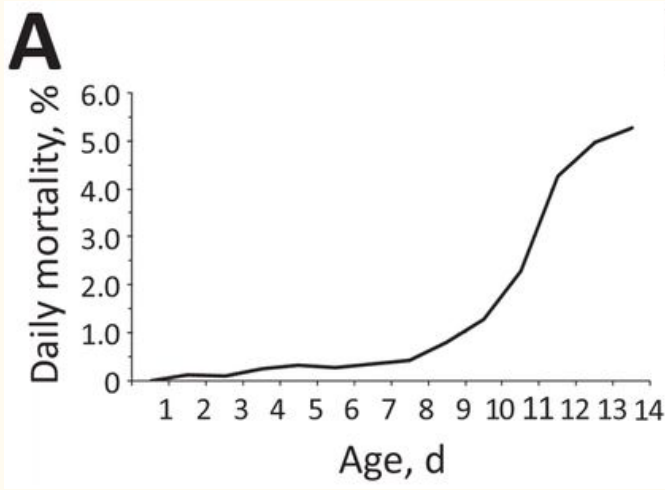
Exemplos

- Série temporal de ganho de peso de animal
  - Se tem dados faltantes poderia ser preenchido com média dos +-1 ou 2 dias
  - Se tem curvas para diferentes raças (aqui, só Ross), faltantes poderia ser preenchido com média dos +-1 ou 2 dias de curva agrupada por raça
  - Poderia usar regressão linear



Exemplos

- A coleta de dados para chegar nesse gráfico pode ser uma planilha de lançamentos de nº mortes.
- Nem todos os dias há lançamento. Preencher com 0, nesse caso.



data	temperatura
01/01	10ºC
02/01	12ºC
03/01	14ºC

data	mortes
01/01	2
01/01	3
04/01	6

merge

data	temperatura	mortes
01/01	10ºC	2
01/01	10ºC	3
02/01	12ºC	
03/01	14ºC	
04/01		6

Resumo

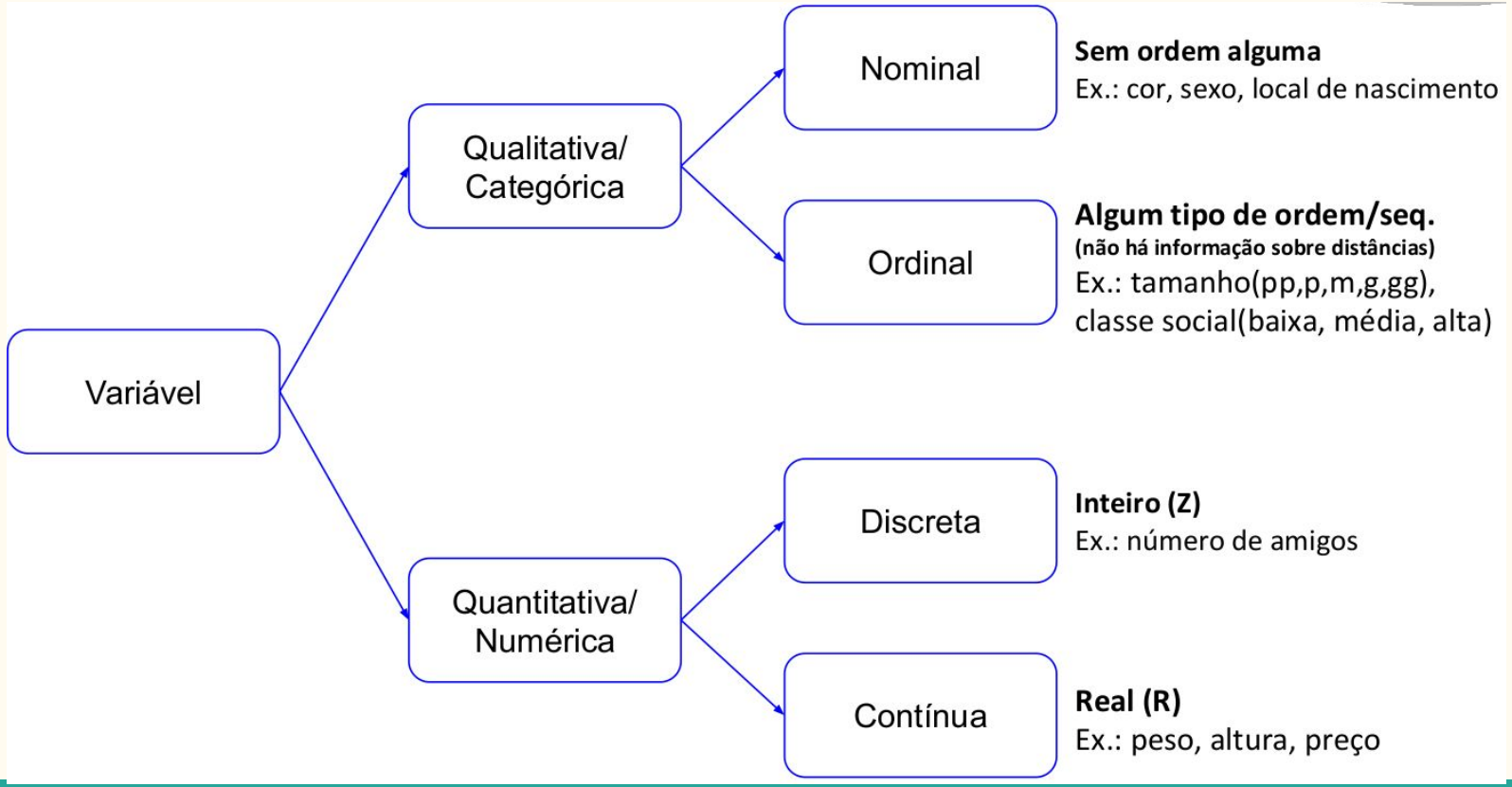
No geral, a melhor decisão depende do contexto da base:

- O que as características significam
- Quais seus limiares máximos e mínimos aceitáveis
- Como os dados são coletadas e preenchidos (com que frequência, por quem, via sistema, via papel...)
- Quem preenche os faz da forma correta?

Nem sempre temos essas informações, então usamos a estatística, que acertará em grande parte das vezes.

Data encoding

Tipos de variáveis



Data encoding

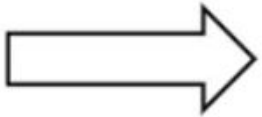
- Como codificar variáveis categóricas?
  - One-Hot Encoding
  - Dummy Encoding
  - Ordinal Encoding
  - Binary Encoding
  - Count Encoding
  - Target Encoding

One-Hot Encoding

- Uma coluna binária para cada categoria

One-Hot Encoding

Places
New York
Boston
Chicago
California
New Jersey



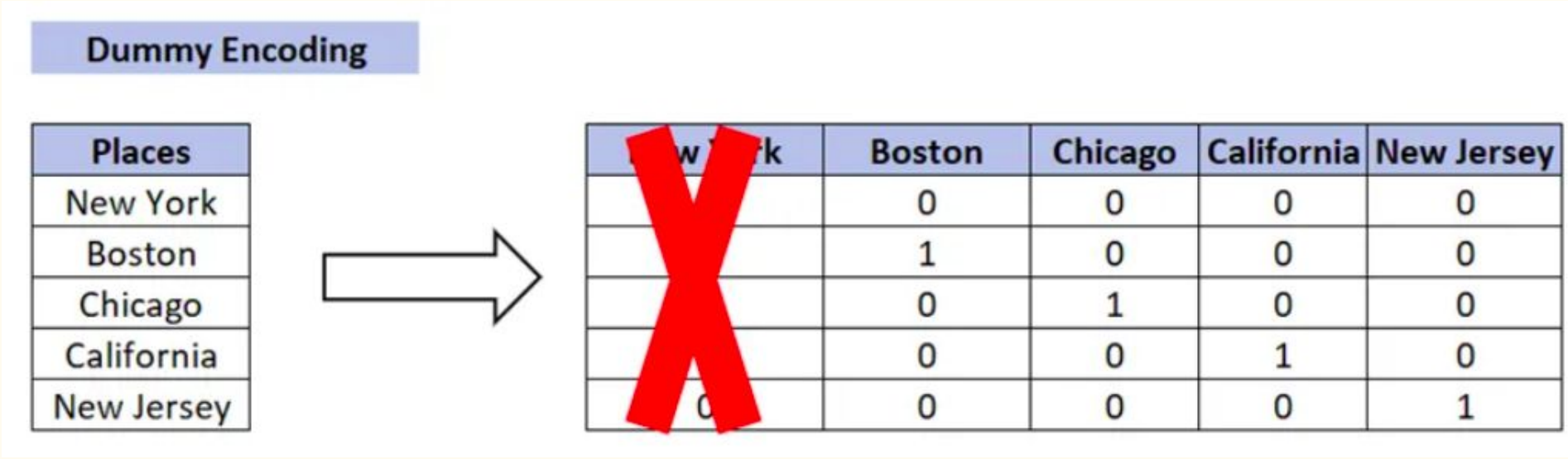
New York	Boston	Chicago	California	New Jersey
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

One-Hot Encoding

```
import pandas as pd

# One-Hot Encoding:

# create a sample dataframe with a categorical variable

df = pd.DataFrame({'color': ['red', 'green', 'blue', 'red'],

                   'fruits': ['morango', 'limão', 'blueberry', 'tomate']})


# perform one-hot encoding on the 'color' column

one_hot = pd.get_dummies(df['color'])


# concatenate the one-hot encoding with the original dataframe

df1 = pd.concat([df, one_hot], axis=1)


# drop the original 'color' column

df1 = df1.drop('color', axis=1)
```

Dummy Encoding

- N-1 colunas binárias para cada categoria.

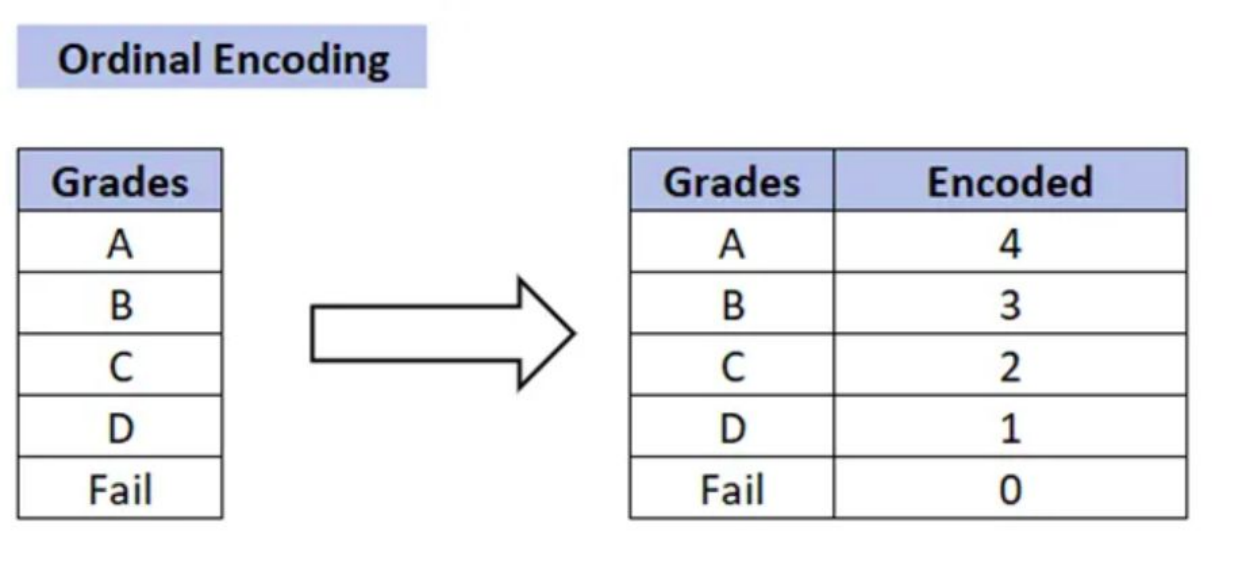


Dummy Encoding

```
import pandas as pd


# create a sample dataframe with a categorical variable

df = pd.DataFrame({'color': ['red', 'green', 'blue', 'red'],

                   'fruits': ['morango', 'limão', 'blueberry', 'tomate']})


# perform dummy encoding on the 'color' column

dummy_df = pd.get_dummies(df['color'], drop_first=True, prefix='Color')


# concatenate the one-hot encoding with the original dataframe

df1 = pd.concat([df, dummy_df], axis=1)


# drop the original 'color' column

df1 = df1.drop('color', axis=1)
```

Ordinal Encoding or label encoding

- Usadas para variáveis categóricas com ordem.

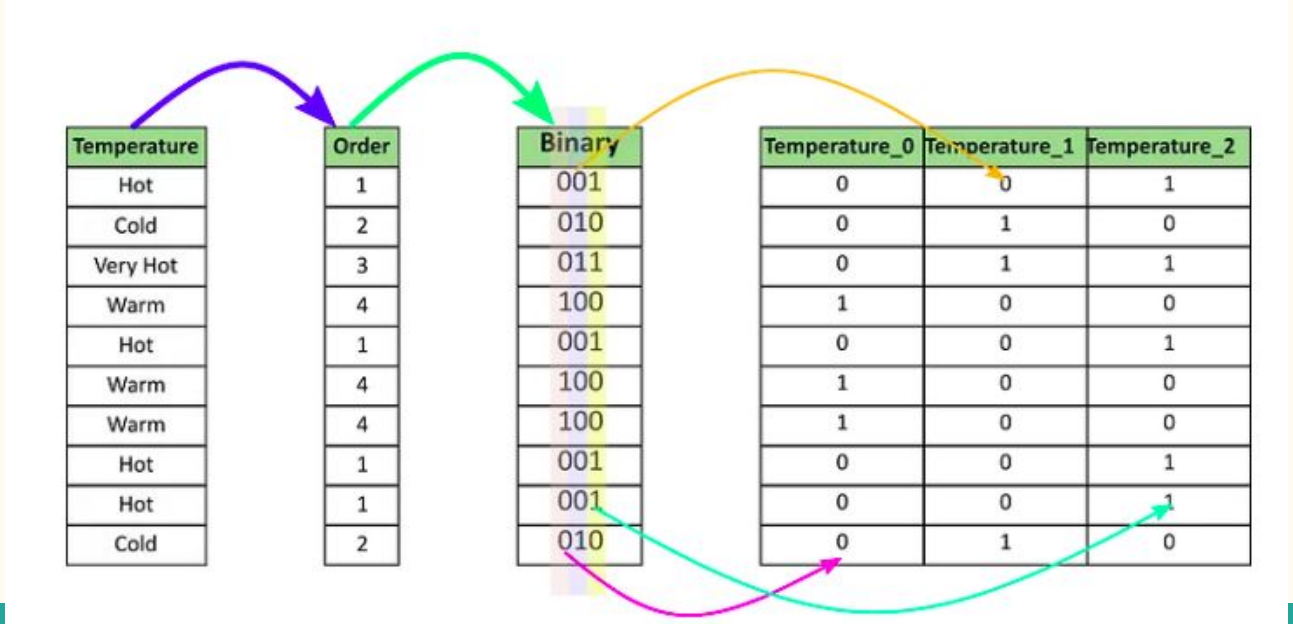


Ordinal Encoding or label encoding

```
# Ordinal Encoding:

# create a sample dataframe with a categorical variable

df = pd.DataFrame({'quality': ['low', 'medium', 'high', 'medium']})


# specify the order of the categories

quality_map = {'low': 0, 'medium': 1, 'high': 2}


# perform ordinal encoding on the 'quality' column

df['quality_map'] = df['quality'].map(quality_map)
```

Binary encoding

- Um código binário para cada categoria
- Usa menos quantidade de colunas que one hot encoding
- Para 100 categorias, precisaríamos apenas de 7 colunas/bits



Temperature
Hot
Cold
Very Hot
Warm
Hot
Warm
Warm
Hot
Hot
Cold

Order
1
2
3
4
1
4
4
1
1
2

Binary
001
010
011
100
001
100
100
001
001
010

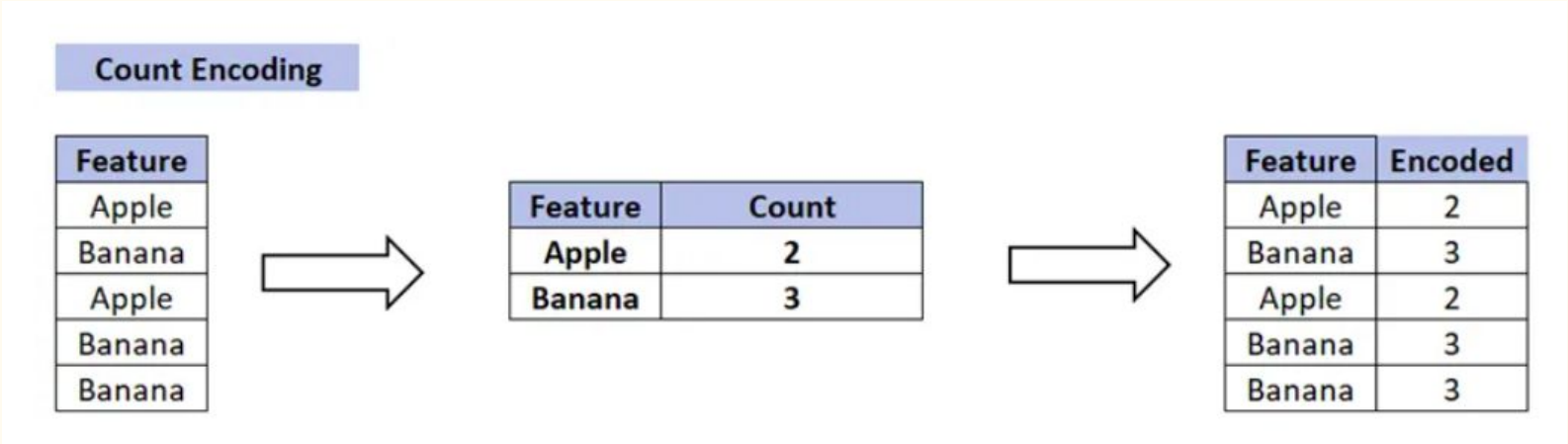
Temperature_0	Temperature_1	Temperature_2
0	0	1
0	1	0
0	1	1
1	0	0
0	0	1
1	0	0
1	0	0
0	0	1
0	0	1
0	1	0

Binary encoding

```
import pandas as pd
import category_encoders as ce

# create a sample dataframe with a categorical variable
df = pd.DataFrame({'color': ['red', 'green', 'blue', 'red'],
                   'fruits': ['morango', 'limão', 'blueberry', 'tomate']})

# perform binary encoding on the 'color' column
encoder = ce.BinaryEncoder(cols=['color'])
df_bin = encoder.fit_transform(df['color'])

# concatenate the one-hot encoding with the original dataframe
df1 = pd.concat([df, df_bin], axis=1)

# drop the original 'color' column
df1 = df1.drop('color', axis=1)
```

Count/frequency encoding

- Faz sentido nos casos em que a frequência está relacionada de alguma forma à variável predita/classificada. Ajuda o modelo ao evidenciar pesos.

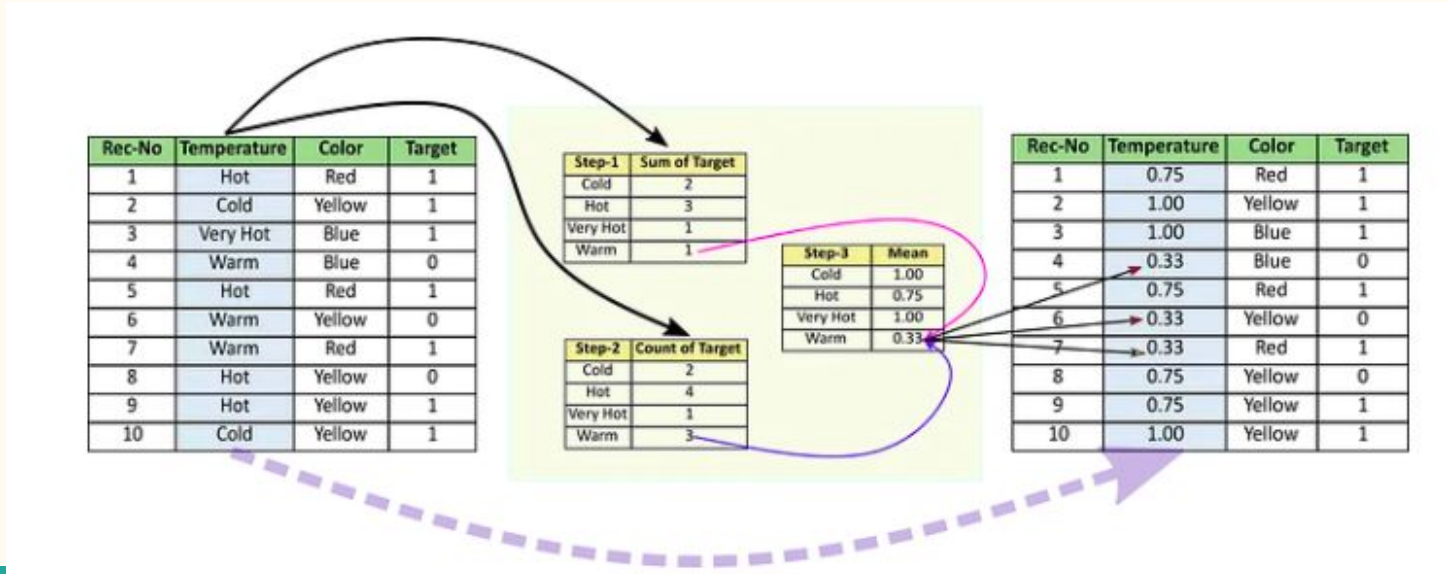


Count/frequency encoding

```
# Count Encoding:

# create a sample dataframe with a categorical variable

df = pd.DataFrame({'fruit': ['apple', 'banana', 'apple', 'banana']})

print(f"Before Encoding the Data:\n\n{df}\n")


# perform count encoding on the 'fruit' column

counts = df['fruit'].value_counts()                    or    counts = df['fruit'].value_counts()/len(df)

df['fruit'] = df['fruit'].map(counts)


# print the resulting dataframe

print(f"After Encoding the Data:\n\n{df}\n")
```

Target encoding or Mean encoding

- Usada para casos em que a variável categórica tem muitas classes.
- Cria uma única coluna com a média do valor da classe predita na base de treino, para cada categoria.
- Evidencia relação entre a variável categórica e a predita, ajudando o método de aprendizado.



Target encoding or Mean encoding

```
mean_encode = df.groupby('Temperature')['Target'].mean()
print(mean_encode)
df.loc[:, 'Temperature_mean_enc'] = df['Temperature'].map(mean_encode)
df
```

```
Temperature
Cold        1.000000
Hot         0.750000
Very Hot    1.000000
Warm        0.333333
Name: Target, dtype: float64
```

	Temperature	Color	Target	Temperature_mean_enc
0	Hot	Red	1	0.750000
1	Cold	Yellow	1	1.000000
2	Very Hot	Blue	1	1.000000
3	Warm	Blue	0	0.333333
4	Hot	Red	1	0.750000
5	Warm	Yellow	0	0.333333
6	Warm	Red	1	0.333333
7	Hot	Yellow	0	0.750000
8	Hot	Yellow	1	0.750000
9	Cold	Yellow	1	1.000000

Outras codificações

- Weight of Evidence Encoding
- Probability Ratio Encoding
- Hashing Encoding
- Backward Difference Encoding
- Leave One Out Encoding
- James-Stein Encoding
- M-estimator Encoding
- Thermometer Encoder

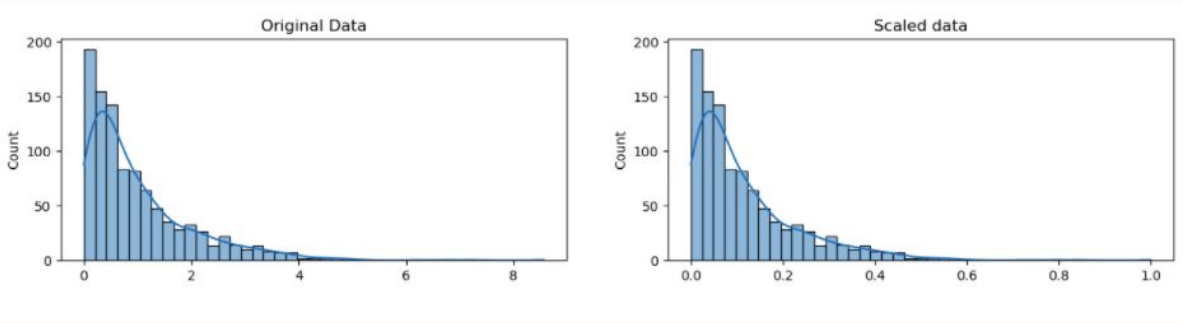
Scaling

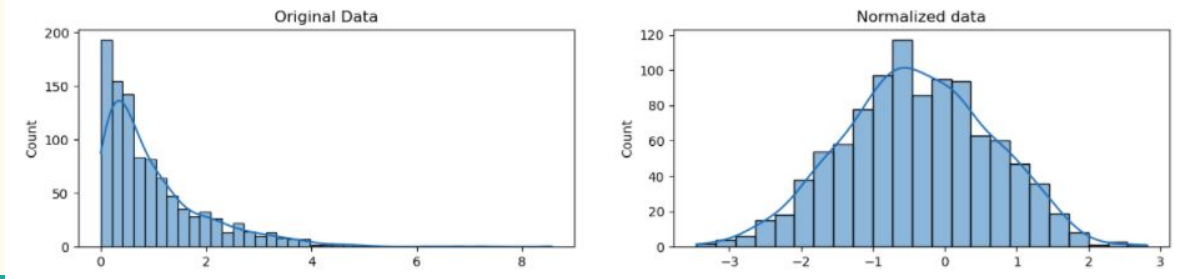
Scaling and Normalization

- Porque?
  - Conversão mais rápida do gradiente descendente
  - Reduz impacto de outliers
  - Mais fácil de comparação
  - Maioria dos algoritmos de machine learning lidam com diferença de 1 unidade de forma igual, independente da escala. Valores maiores têm mais peso no aprendizado e exercem mais impacto no modelo.

Scaling x normalization

- Scaling: muda a escala do dado
- Normalization: muda a distribuição do dado





Métodos para padronização dos dados

1) Min Max Scaler

2) Standard Scaler

3) Max Abs Scaler

4) Robust Scaler

5) Quantile Transformer Scaler

MinMaxScaler

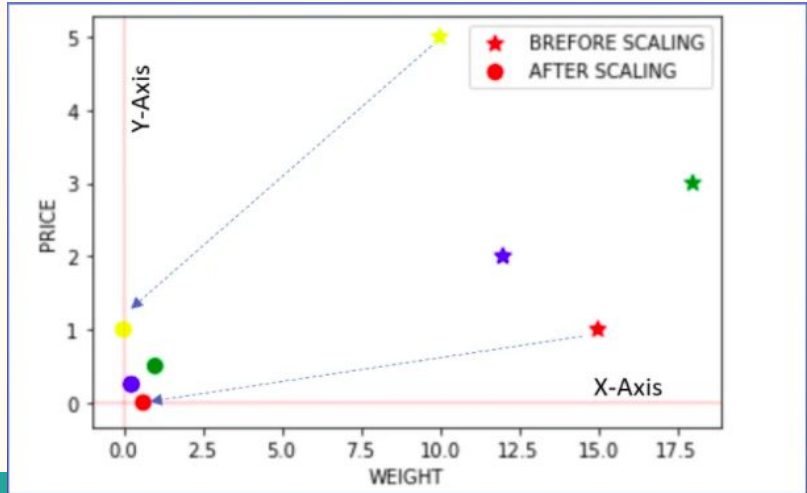
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- x_min e x_max deve ser extraído da base de treino
- É sensível a outliers
- Funciona melhor quando o desvio padrão é pequeno
- Dados não precisam ter distribuição normal

MinMaxScaler

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.DataFrame({'WEIGHT': [15, 18, 12,10],
                                    'PRICE': [1,3,2,5]},
                                     index = ['Orange','Apple','Banana','Grape'])

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df1 = pd.DataFrame(scaler.fit_transform(df),
                                    columns=['WEIGHT','PRICE'],
                                    index = ['Orange','Apple',
                                             'Banana','Grape'])
```

	WEIGHT	PRICE
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5



StandardScaler (Normalization)

$$x_{new} = \frac{x - \mu}{\sigma}$$

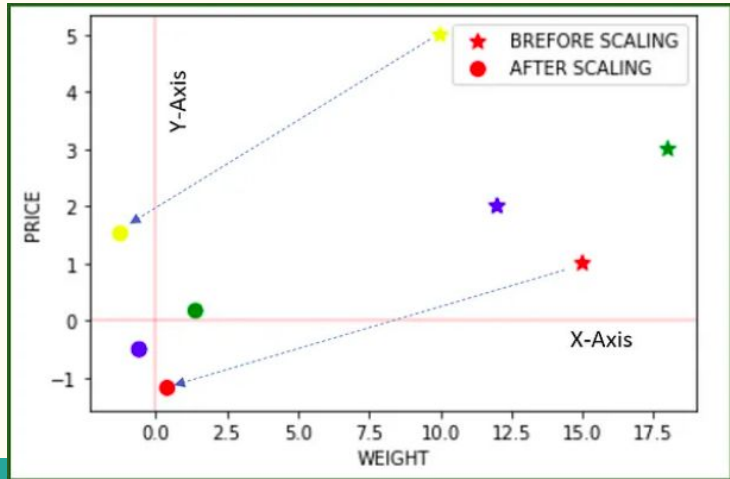
- Onde: **μ** média e σ desvio padrão
- Assume que os dados têm distribuição normal
- Modifica os dados para que a distribuição fique centralizada em 0 e com desvio de 1 unidade

MinMaxScaler

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.DataFrame({'WEIGHT': [15, 18, 12,10],
                                  'PRICE': [1,3,2,5]},
                                   index = ['Orange','Apple','Banana','Grape'])
```

	WEIGHT	PRICE
Orange	15	1
Apple	18	3
Banana	12	2
Grape	10	5

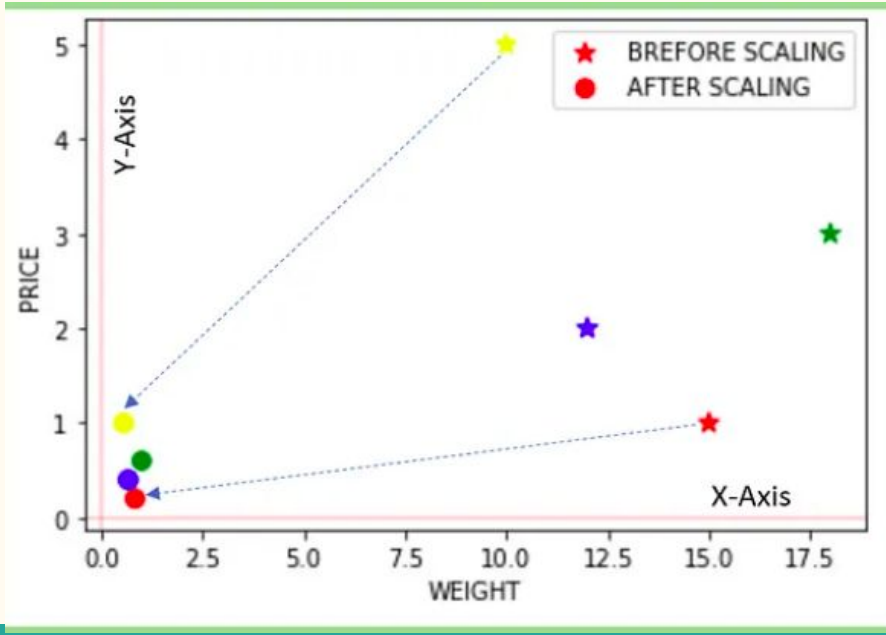
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df2 = pd.DataFrame(scaler.fit_transform(df),
                   columns=['WEIGHT','PRICE'],
                   index = ['Orange','Apple','Banana','Grape'])
```



Max Abs Scaler

- O valor máximo da base de treino é transformado para 1.0
- Similar ao MinMaxScaler, sensível a outlier

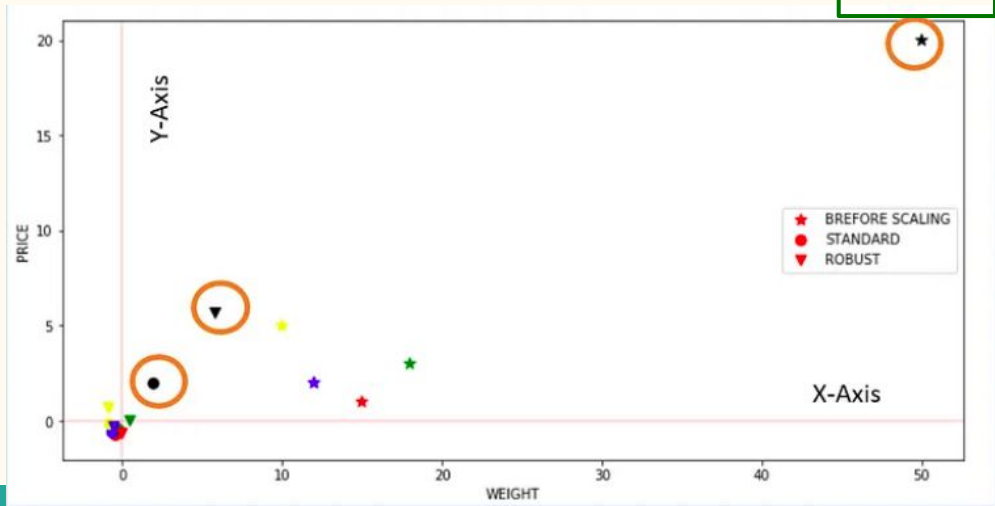
```
from sklearn.preprocessing import MaxAbsScaler
```



Robust Scaler

- Não é sensível a outliers
- Escala o dado de acordo com distância interquartil

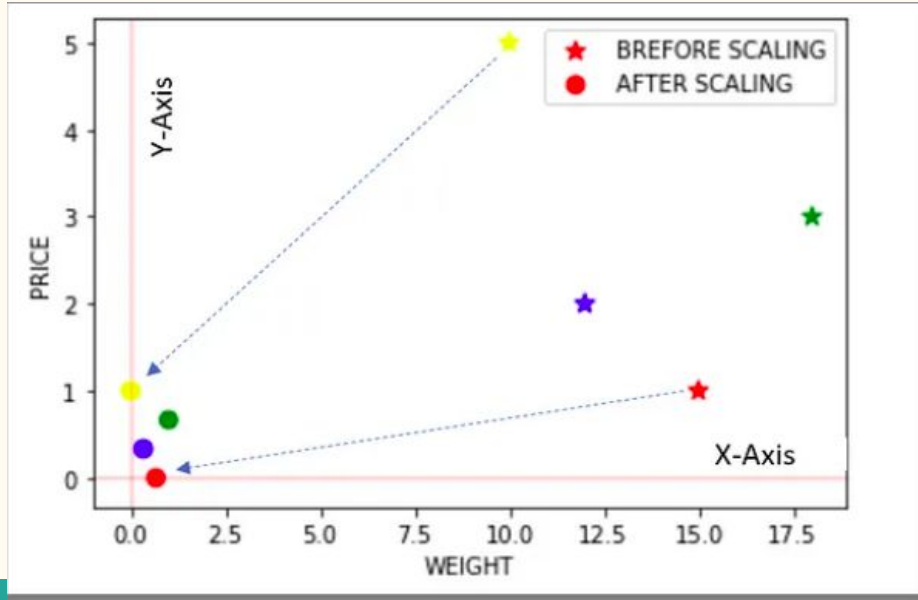
```
from sklearn.preprocessing import RobustScaler
```



Quantile Transformer Scaler

- Tendência a espalhar os dados de valores mais frequentes
- Não sensível a outlier

```
from sklearn.preprocessing import QuantileTransformer
```



Outros

6) Power Transformer Scaler

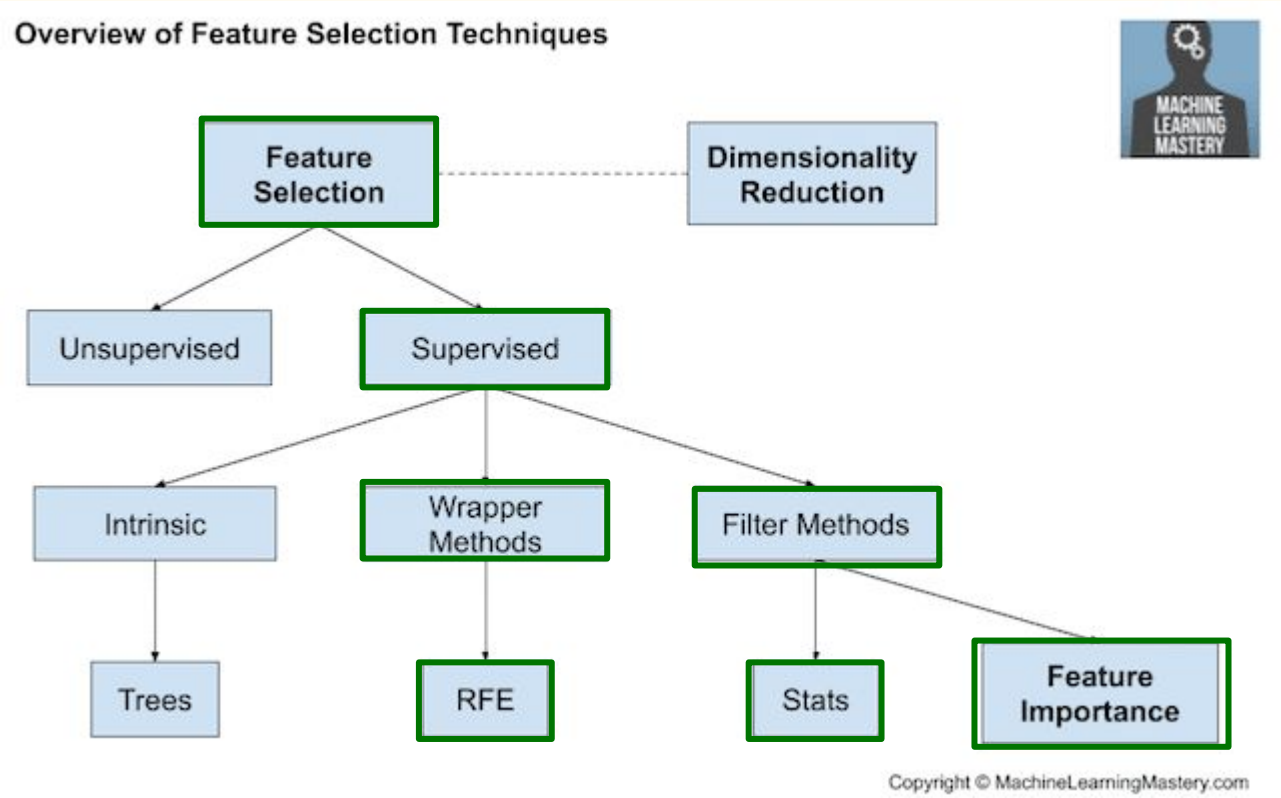
7) Unit Vector Scaler

Feature selection

Seleção de características

- **O quê:** o processo de reduzir o número de variáveis no desenvolvimento de um modelo preditivo
- **Por quê:**
  - Nem todas as variáveis podem ser importantes para o modelo
  - Quanto mais variáveis, mais complexo o modelo, mais demorado para aprender
  - Variáveis redundantes atrapalham o aprendizado de alguns modelos

Técnicas


Overview of Feature Selection Techniques
MACHINE LEARNING MASTERY
Feature Selection
Dimensionality Reduction
Unsupervised
Supervised
Intrinsic
Wrapper Methods
Filter Methods
Trees
RFE
Stats
Feature Importance
Copyright © MachineLearningMastery.com

Filter methods

- Baseada em uma métrica estatística para análise univariada
- Selecionar as características que encontramos correlação/importância maior em relação a variável predita
- Particularmente, usado para filtragem inicial. Isto é, deve ser combinada com outros métodos de seleção de características
- Possível problema: variáveis interdependentes não são analisadas. Uma vez que a análise é univariada, podemos ter 2 variáveis com correlação alta com a variável predita, mas em contrapartida, elas podem ser dependentes entre si, nesse caso, o ganho de uma delas é suficiente para o modelo.

Métricas estatísticas

Implementadas no sklearn.feature_selection:

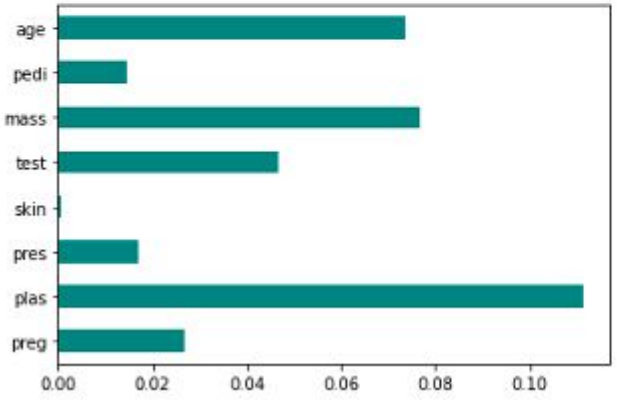
- Variance Threshold: VarianceThreshold()
- Pearson's Correlation Coefficient: r_regression()
- ANOVA: f_classif()
- Chi-Squared: chi2()
- Mutual Information: mutual_info_classif() and mutual_info_regression()

Implementadas no SciPy:

- Kendall's tau (kendalltau)
- Spearman's rank correlation (spearmanr).

Exemplos

```
from sklearn.feature_selection import mutual_info_classif
import matplotlib.pyplot as plt
%matplotlib inline

importances = mutual_info_classif(X, Y)
feat_importances = pd.Series(importances, dataframe.columns[0:len(dataframe.columns)-1])
feat_importances.plot(kind='barh', color = 'teal')
plt.show()
```



Exemplos

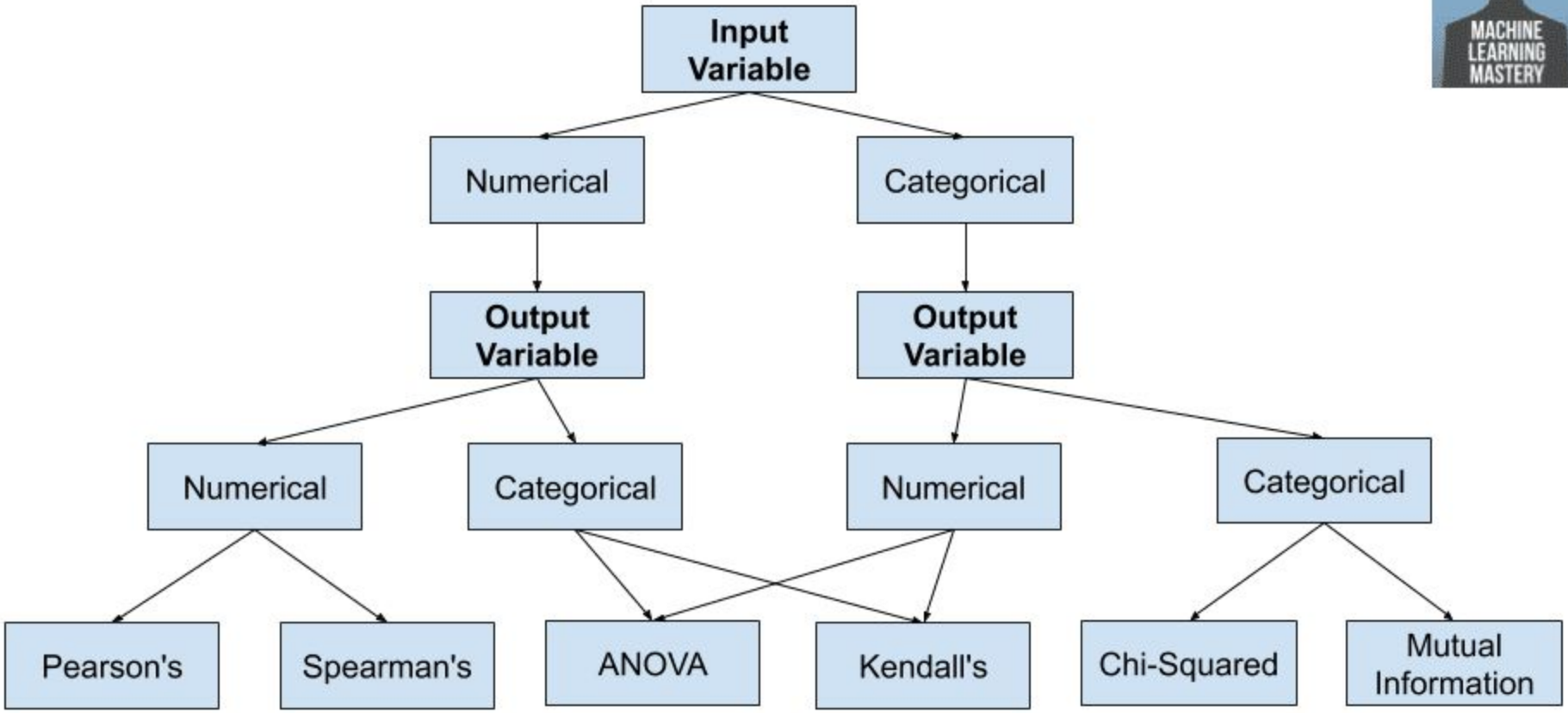
- Pode-se adotar um limiar de corte para variáveis

- Pode-se observar se 2 ou mais das pré-selecionadas são correlacionadas entre si (manter uma delas, apenas)

```
1  import seaborn as sns
2  import matplotlib.pyplot as plt
3  %matplotlib inline
4
5  # Correlation matrix
6  cor = dataframe.corr()
7
8  # Plotting Heatmap
9  plt.figure(figsize = (10,6))
10 sns.heatmap(cor, annot = True)
```

```
<AxesSubplot:>
```

How to Choose a Feature Selection Method



Wrapper methods

- Teste e validação de subconjuntos de características em um modelo de machine learning.
- Avalia todas as possibilidades
- Mais lento, melhor resultado que os Filter methods

Forward Feature Selection

- Inicia com o modelo com apenas uma característica de entrada (por exemplo, pode ser a característica mais correlacionada, usando métricas estatísticas).
- Adiciona ao conjunto de entrada, a segunda característica que obteve melhor ganho ao modelo.
- Prossegue até atingir algum critério de parada

```
# Forward Feature Selection
from mlxtend.feature_selection import SequentialFeatureSelector
ffs = SequentialFeatureSelector(lr, k_features='best', forward = True, n_jobs=-1)
ffs.fit(X, Y)
features = list(ffs.k_feature_names_)
features = list(map(int, features))
lr.fit(x_train[features], y_train)
y_pred = lr.predict(x_train[features])
```

Backward Feature Elimination

- Oposto ao anterior, inicia com todas as características e vai removendo

```
# Backward Feature Selection
from sklearn.linear_model import LogisticRegression
from mlxtend.feature_selection import SequentialFeatureSelector
lr = LogisticRegression(class_weight = 'balanced', solver = 'lbfgs', random_state=42, n_jobs=-1, max_iter=500)
lr.fit(X, Y)
bfs = SequentialFeatureSelector(lr, k_features='best', forward = False, n_jobs=-1)
bfs.fit(X, Y)
features = list(bfs.k_feature_names_)
features = list(map(int, features))
lr.fit(x_train[features], y_train)
y_pred = lr.predict(x_train[features])
```

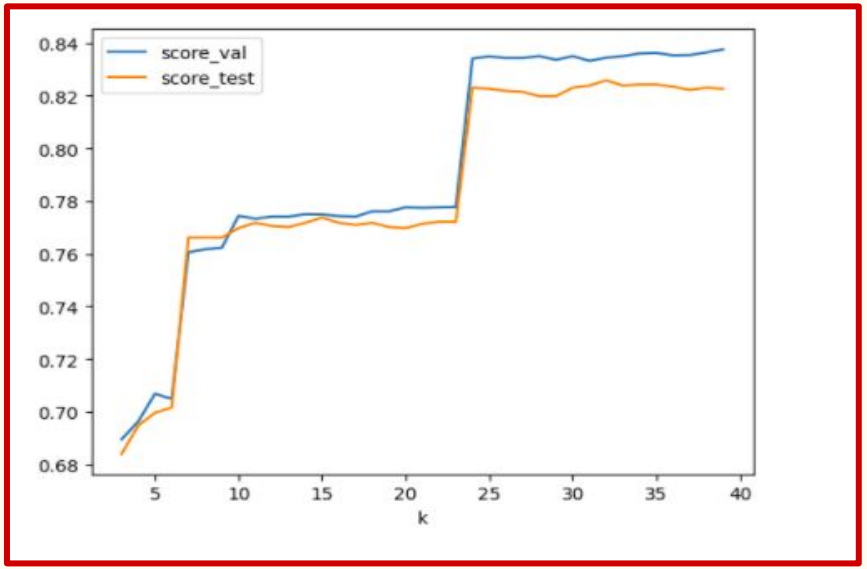
Outros

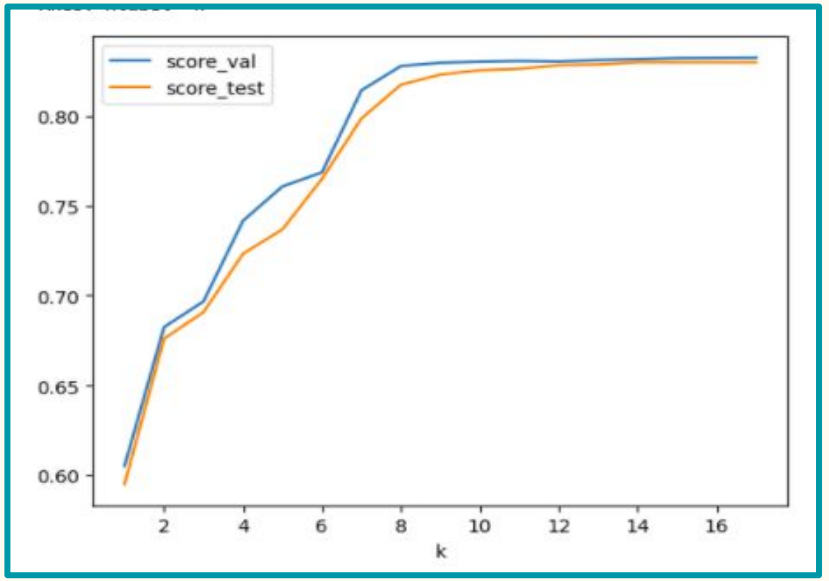
- Exhaustive Feature Selection
  - Força Bruta, testa todas as combinações de características

- Recursive Feature Elimination
  - semelhante ao backward, porém utiliza uma métrica para estimar importância de cada variável no modelo, e retira a de menor importância.

Exemplo:

Acurácia do modelo usando adição de características apenas por meio do método filter x método forward





Pipeline

Pipeline

Cleansing

Scaling

Feature selection

Model selection

Tunning - Hyperparameter

Melhor modelo: têm maior acurácia no treino e teste.

Lembrando que se apenas boa acurácia no treino, e baixa no teste, temos overfitting

Referências usadas

https://towardsai.net/p/l/encoding-categorical-data-the-right-way#:~:text=Encoding%20categorical%20data%20is%20a,can%20work%20only%20on%20numbers.

https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02

https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35

https://towardsdatascience.com/the-ultimate-guide-to-data-cleaning-3969843991d4