

# INTELIGÊNCIA ARTIFICIAL

Dados &  
Experimentos em ML

**Everton Dias**

etgdb@cesar.school

**Material produzido por**

**JP Magalhaes**

jp@cesar.school



C E S A R  
school





# Dados

## Tratamento de Dados para Machine Learning



# Tratamento de Dados para ML



# Dados

Dados são observações documentadas ou o resultado de processos de medição, podendo ser obtidos de diversas formas.

- Não tem muito valor isoladamente ou sem o devido tratamento e análise
- Para um bom resultado, são necessários: quantidade, diversidade e qualidade
- Insumo essencial para a Ciência de Dados e o Aprendizado de Máquinas



# [WARNING] Privacidade e LGPD

Dados contém informações valiosas sobre o negócio de empresas e privadas a respeito de pessoas.

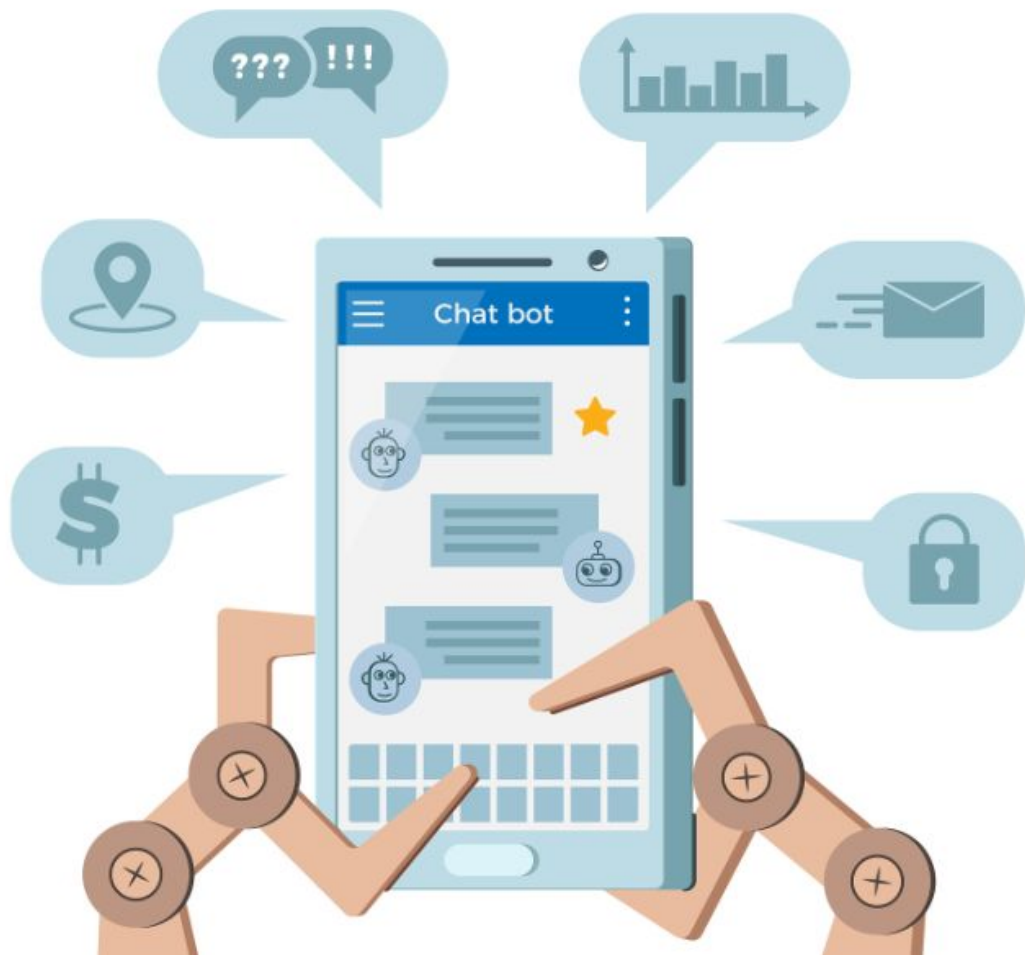
- Legislação brasileira regula o tratamento, armazenamento e compartilhamento de dados pessoais através da **Lei Geral de Proteção de Dados Pessoais- LGPD**
  - **Dados pessoais:** Nome, RG, CPF, e-mail - **qualquer conjunto de informações que identifique unicamente uma pessoa.**
  - **Dados pessoais sensíveis:** dado biométrico, genético ou discriminatório (minorias, etc)
  - **Anonimização:** impossibilita a associação dos dados a um indivíduo
    - dados anonimizados **não são** considerados dados pessoais
  - **Pseudoanonimização:** processos de dificultar a associação dos dados com uma pessoa.
    - dados pseudoanonimizados **são** considerados dados pessoais!



# Dados

Podem ser:

- Não estruturados
  - Imagens
  - Vídeos
  - Áudios
  - Textos "ricos"
  - Etc



# Dados

Podem ser:

- Estruturados
  - numéricos ou textuais
  - binários, contínuos ou discretos
  - categórico, ordinal, nominal

Nome	Idade	Camisa	Sexo	Pet	Gasto	Observação
Joao Paulo	38	G	M	Cao	50	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Michele	39	M	F	Gato	100	-
Miguel	5	P	M	Peixe	0	



# Dados - Libs Básicas

## Pandas

- Pandas é uma biblioteca open source escrita sobre Numpy;
- Permite rápida visualização e limpeza de dados;
- Semelhante ao Excel;
- Pode trabalhar com diversos tipos de dados;
- Possui método próprio de visualização de dados.





# Dados - Libs Básicas

## Pandas

- Series;
- DataFrames;
- Dados Ausentes;
- GroupBy;
- Concatenar, juntar e mesclar;
- Operações;
- Entrada e saída de dados.



# Dados - Libs Básicas

## Pandas - Series

- Series nada mais é que um array de 1 dimensão. Você pode considerar um Series também como uma coluna de uma tabela. Exemplo:
- `>>> s = pd.Series(data=[3, -5, 7, 4], index=['a', 'b', 'c', 'd'])`

**INDEX** —

A	3
B	-5
C	7
D	4



# Dados - Libs Básicas

## Pandas - DataFrame

- Um DataFrame é simplesmente um conjunto de Series. Trata-se de uma estrutura de dados de 2 dimensões – colunas e linhas – que transforma os dados em uma bela tabela.  
Exemplo:



# Dados - Libs Básicas

## Pandas - DataFrame

- Um DataFrame é simplesmente um conjunto de Series. Trata-se de uma estrutura de dados de 2 dimensões – colunas e linhas – que transforma os dados em uma bela tabela.  
Exemplo:

#Criando um dicionário onde cada chave será uma coluna do DataFrame:

```
data = {  
    'País': ['Bélgica', 'Índia', 'Brasil'],  
    'Capital': ['Bruxelas', 'Nova Delhi', 'Brasília'],  
    'População': [123465, 456789, 987654]}
```



# Dados - Libs Básicas

## Pandas - DataFrame

- Um DataFrame é simplesmente um conjunto de Series. Trata-se de uma estrutura de dados de 2 dimensões – colunas e linhas – que transforma os dados em uma bela tabela.  
Exemplo:

#Criando o DataFrame

```
df = pd.DataFrame(data, columns=['País','Capital','População'])
```

INDEX	País	Capital	População	
	1	Bélgica	Bruxelas	123465
	2	Índia	Nova Delhi	456789
	3	Brasil	Brasília	987654



# Carregamento dos dados

- Utiliza-se Pandas **DataFrame** para trabalhar com os dados
  - *'Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns).'*
- Normalmente as bases de dados estão em arquivos de texto
  - **CSV**

```
import pandas as pd
```

```
df = pd.read_csv('file_name.csv')
```

```
df
```

```
# Carrega um CSV que está no mesmo diretório
```

```
# retorna todas as linhas e colunas do dataframe
```



# Exploração dos Dados - Básico

`df.head()` # exibe as 5 primeiras linhas do dataframe

Out[5]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0



# Exploração dos Dados - Básico

`df.info()` # Exibe características de cada coluna

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude                20640 non-null float64
latitude                 20640 non-null float64
housing_median_age       20640 non-null float64
total_rooms              20640 non-null float64
total_bedrooms           20433 non-null float64
population               20640 non-null float64
households               20640 non-null float64
median_income            20640 non-null float64
median_house_value       20640 non-null float64
ocean_proximity          20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```



# Exploração dos Dados - Básico

`df.describe()`      # Exibe estatística básica de cada coluna

	PassengerId	Survived
count	891.000000	891.000000
mean	446.000000	0.383838
std	257.353842	0.486592
min	1.000000	0.000000
25%	223.500000	0.000000
50%	446.000000	0.000000
75%	668.500000	1.000000
max	891.000000	1.000000



# Dados incompletos

Algumas colunas podem **não** ter valores em todas as linhas

- Parte das vezes, conseguimos substituir os valores por algum que não atrapalhe o modelo
  - Normalmente faremos uma função específica para isto

```
# Aplica function passando 2 colunas ('Col1' e 'Col2') e atribui o retorno a coluna 1.  
df['Col1'] = df[['Col1','Col2']].apply(function,axis=1)
```



# Dados incompletos

- Outras vezes, não conseguimos/queremos completar estes valores ausentes
  - As linhas ou colunas correspondentes devem ser removidas durante o processamento

```
df.drop('Col',axis=1,inplace=True) # Remove a coluna com nome 'Col'
```

```
# axis : {0 or 'index', 1 or 'columns'}, default 0
```

```
# inplace : bool, default False - If True, do operation inplace and return None.
```

```
df.dropna(axis=0, how='any')
```

```
# Remove linhas (axis=0 ou 'index') ou colunas (axis=1 ou 'columns')
```

```
# que contenham algum (how='any') ou todos (how='all') os valores nulos
```



# Valores Literais

- São os valores não numéricos
  - Normalmente formatados como caracteres, palavra(s) ou textos (Strings)
  - **Algoritmos de Machine Learning, em geral, só aceitam dados numéricos como entrada**
- Precisamos transformar estes valores literais em numéricos para aproveitá-los como fonte de informação
- Podem ser de dois tipos
  - Categóricos (Binários, ordinais ou nominais)
  - Textuais
- Valores textuais 'abertos' são difíceis de serem tratados - NLP
  - Nome, endereço, observação, etc - não trataremos deles aqui



# Valores Categóricos

Nome	Sexo	Pet	Camisa
JP	M	Cao	G
Michele	F	Gato	M
Miguel	M	Peixe	P



# Valores Categóricos - Tratamento

- Queremos estabelecer uma relação de 1 para 1 entre categorias e índices
  - [**Nominal**] Sexo: Masculino, Feminino
  - [**Nominal**] Pet: Cão, Gato, Pássaro, Iguana, Cobra
  - [**Ordinal**] Tamanho: P, M, G
- Para isso, transformamos valores categóricos em numéricos



# Valores Categóricos - Ordinais - Tratamento

Quando estas categorias apresentarem **ordinalidade**, utilizamos **LabelEncoder**

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(["P", "M", "M", "G"])
list(le.classes_)                # ['P', 'M', 'G']
le.transform(["G", "G", "M"])    # array([2, 2, 1]...)
list(le.inverse_transform([2, 2, 1])) # ['G', 'G', 'M']
```

\* Atentar se o *encoding* estabelece as associações de escala e direção corretas - muitas vezes, uma função própria é necessária.



# Valores Categóricos - Ordinais - Tratamento

Nome	Camisa	<b>Camisa_N</b>
JP	G	2
Michele	M	1
Miguel	P	0





# Valores Categóricos - Nominais

- Alguns valores categóricos **não** apresentam relação de distância entre suas categorias
  - Maioria dos modelos utilizam espaços (matemáticos), o que acaba criando relações de distância
  - Se utilizássemos Pets, no exemplo anterior significaria que Pássaro está mais próximo de Cão do que Iguana
- **One-Hot-Encoding** trata estes casos
  - Relaciona os índices a um vetor de representação binário
  - Cria novas (**i - 1**) colunas, onde **i** é o número de índices

# Transforma variáveis categóricas em indicativas

```
pd.get_dummies(df[Pet], drop_first=True)
```

# drop\_first : se deve excluir uma das colunas

```
>>> s = pd.Series(list('abca'))
```

```
>>> pd.get_dummies(s)
```

	a	b	c
0	1	0	0
1	0	1	0
2	0	0	1
3	1	0	0



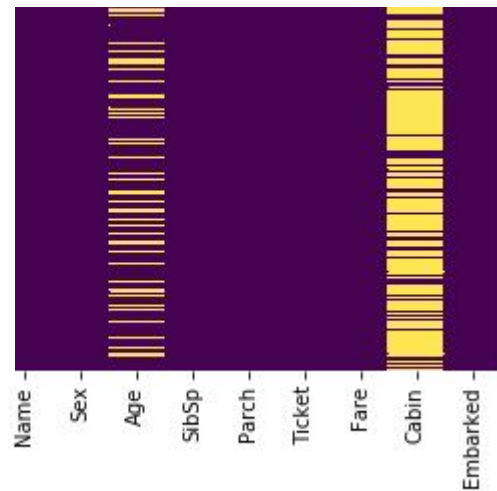
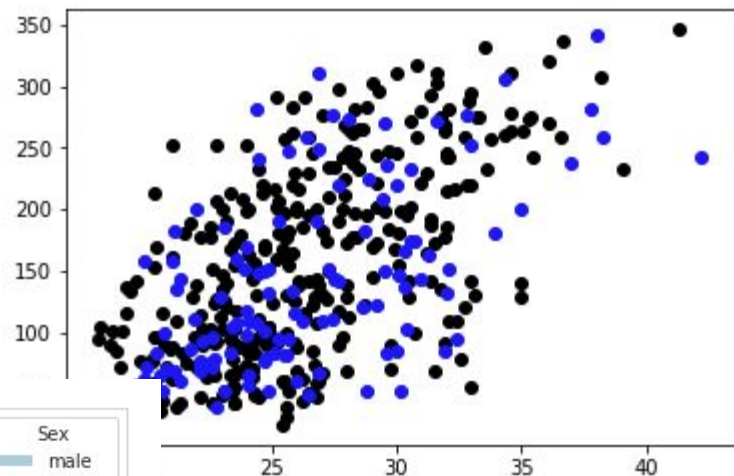
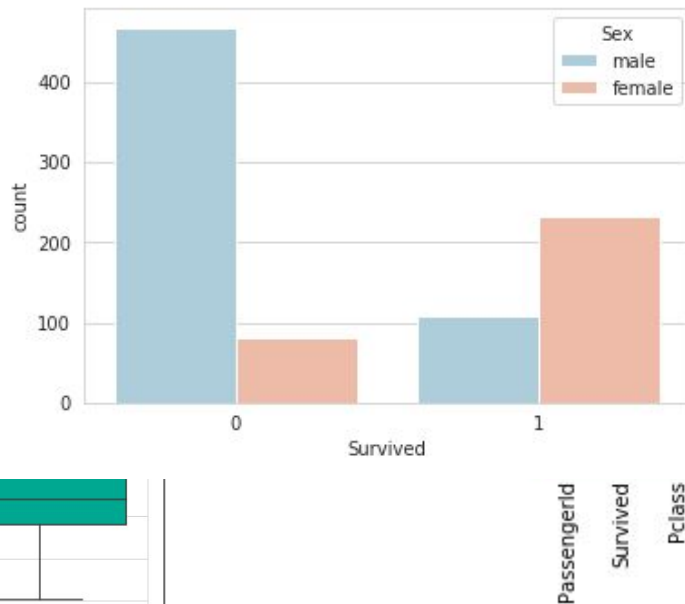
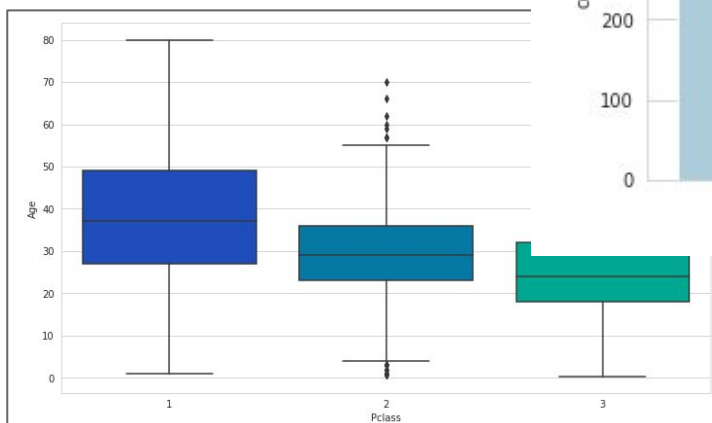
# Valores Categóricos - Nominais - Tratamento

Nome	Sexo	Sexo_M	Pet	Cão	Gato	Peixe
JP	M	1	Cao	1	0	0
Michele	F	0	Gato	0	1	0
Miguel	M	1	Peixe	0	0	1



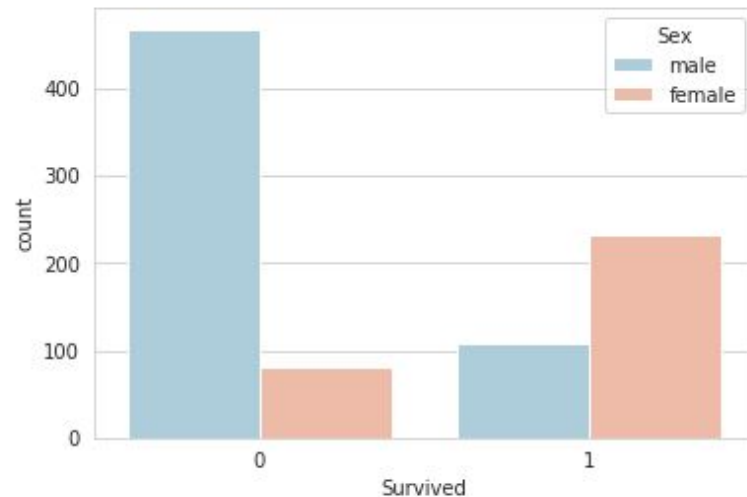
# Exploração dos Dados

- Outras bibliotecas agregam mais recursos com facilidade
  - **Pandas**
  - **Seaborn**
  - **Matplotlib**



# Exploração dos Dados

- Contagem



# Exibe um gráfico de contagem, configurando antes para exibir o grid

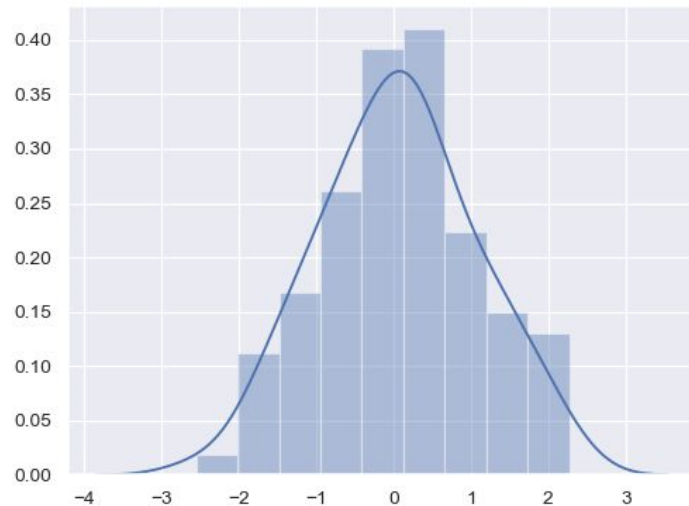
```
sns.set_style('whitegrid')
```

```
sns.countplot(x='Survived', hue='Sex', data=df, palette='RdBu_r')
```



# Histograma

- Representação aproximada da distribuição de dados numéricos ou categóricos

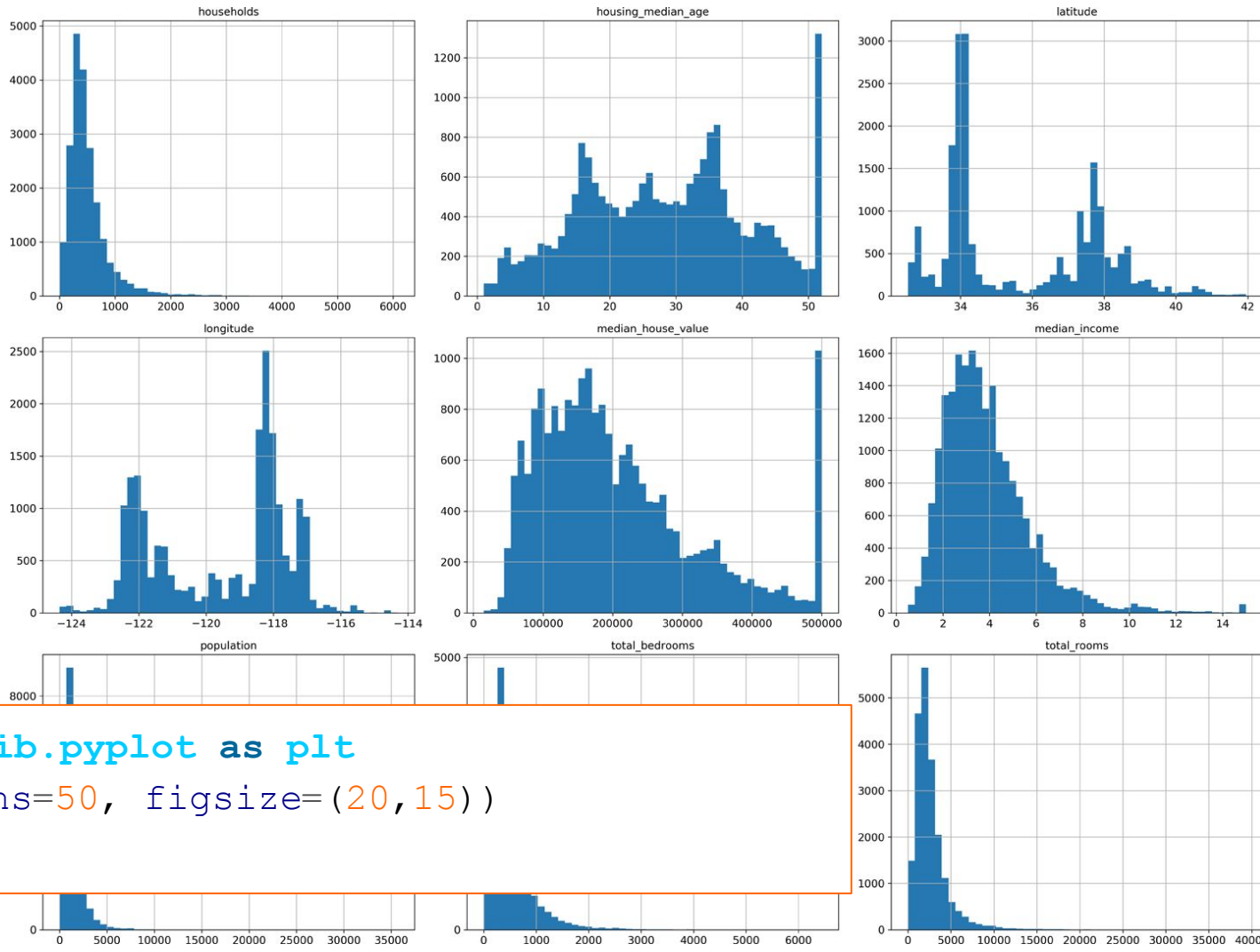


```
# Show a default plot with a kernel density estimate and  
# histogram with bin size determined automatically
```

```
import seaborn as sns, numpy as np  
sns.histplot(x)
```

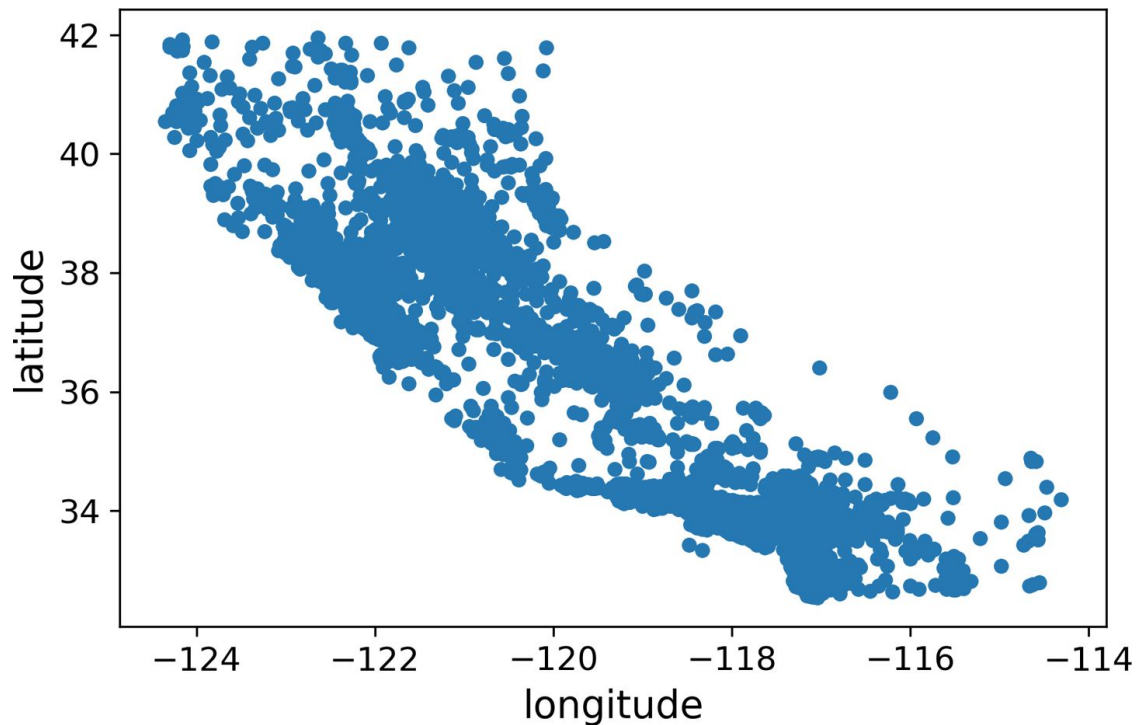


# Histograma



# Dispersão

- Plota um gráfico de dispersão

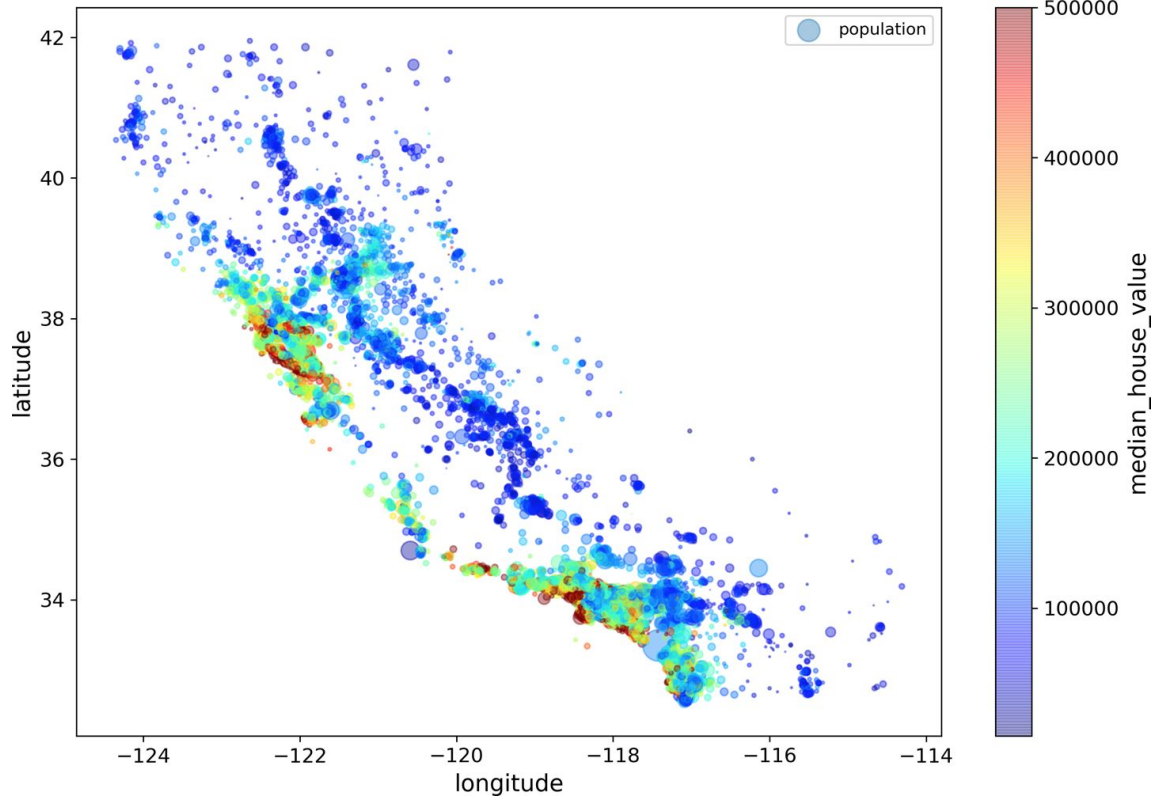


```
dataframe.plot(kind="scatter", x="longitude", y="latitude")
```



# Dispersão

- Plota um gráfico de dispersão



```
df.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,  
        s=housing["population"]/100, label="population", figsize=(10,7),  
        c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True)  
plt.legend()
```



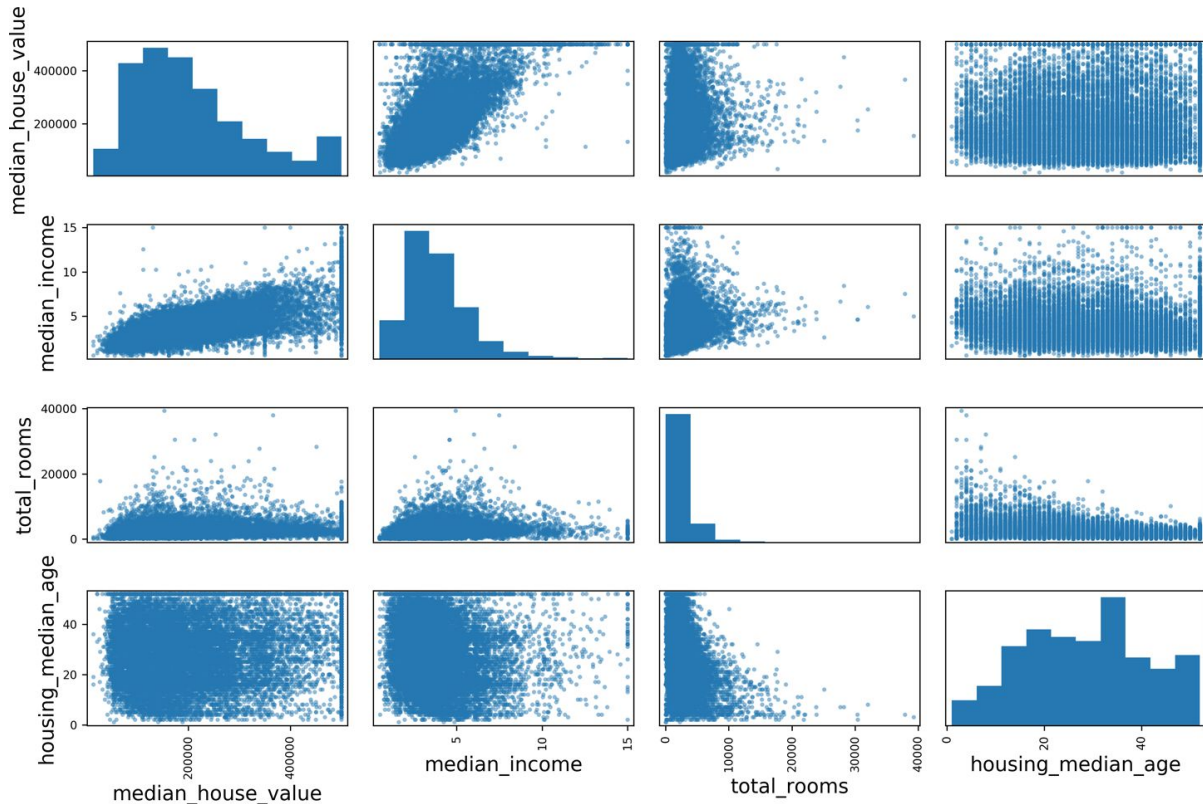


# Correlação

Qualquer relacionamento estatístico entre duas variáveis.

Varia de -1 a +1, com 0 significando a ausência de correlação linear e, os extremos, a presença de uma forte correlação positiva ou negativa.

- Não indica causalidade!
- Ver [Spurious correlations](#)



```
from pandas.plotting import scatter_matrix
```

```
attributes = ["median_house_value", "median_income", "total_rooms",  
             "housing_median_age"]
```

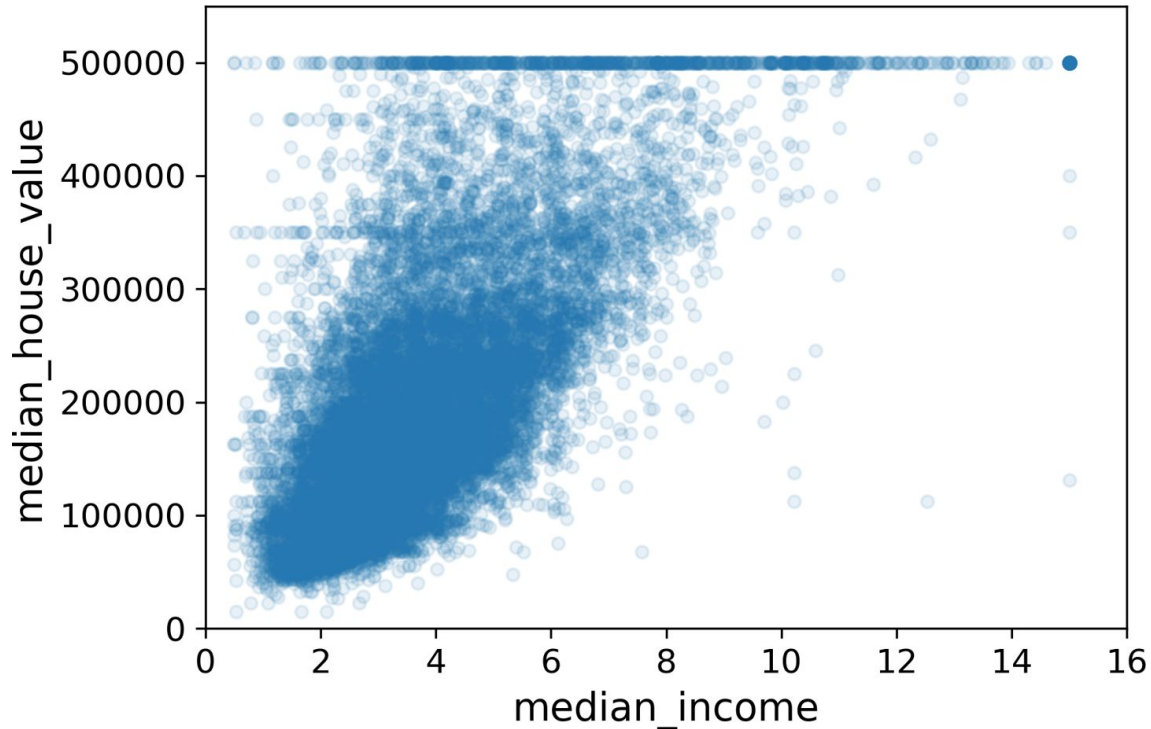
```
scatter_matrix(df[attributes], figsize=(12, 8))
```



# Correlação

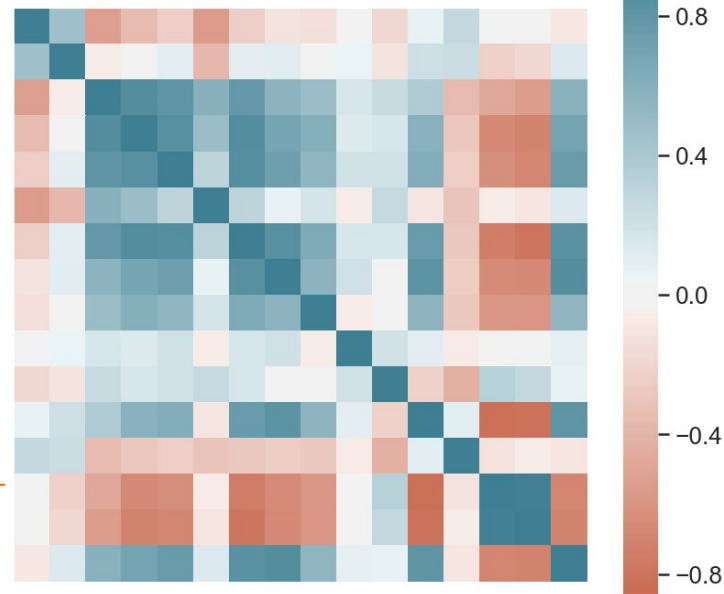
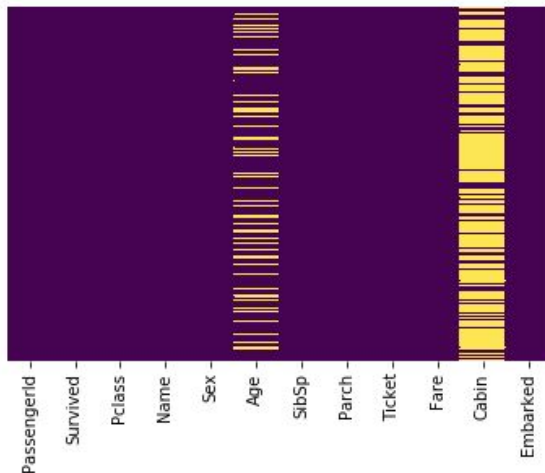
Exemplo:

- Existe correlação?
- Qual a tendência?
- Quais outros detalhes podem ser observados?
- Algum detalhe indica um possível problema nos dados?



# Exploração dos Dados

- Heatmap



```
import seaborn as sns
```

```
# Exibe um heatmap
```

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
# yticklabels : “auto” - if True, plot the column names of the dataframe
```

```
# cbar : boolean, optional - Whether to draw a colorbar
```

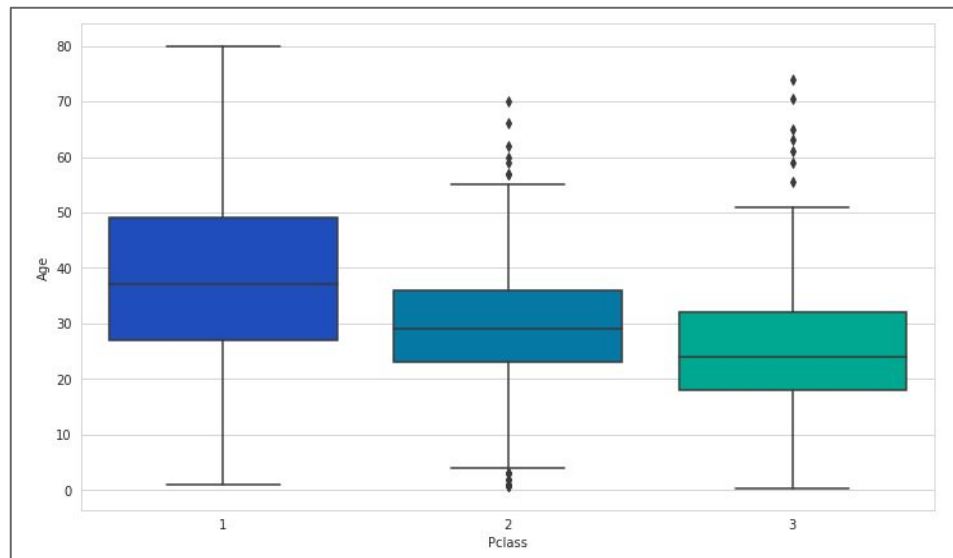
```
# cmap : The mapping from data values to color space (YlGnBu)
```

[Better Heatmaps and Correlation Matrix Plots in Python](#)



# Exploração dos Dados

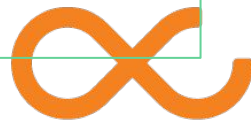
- Boxplot  
Exibe a distribuição com respeito  
as categorias



```
import seaborn as sns
```

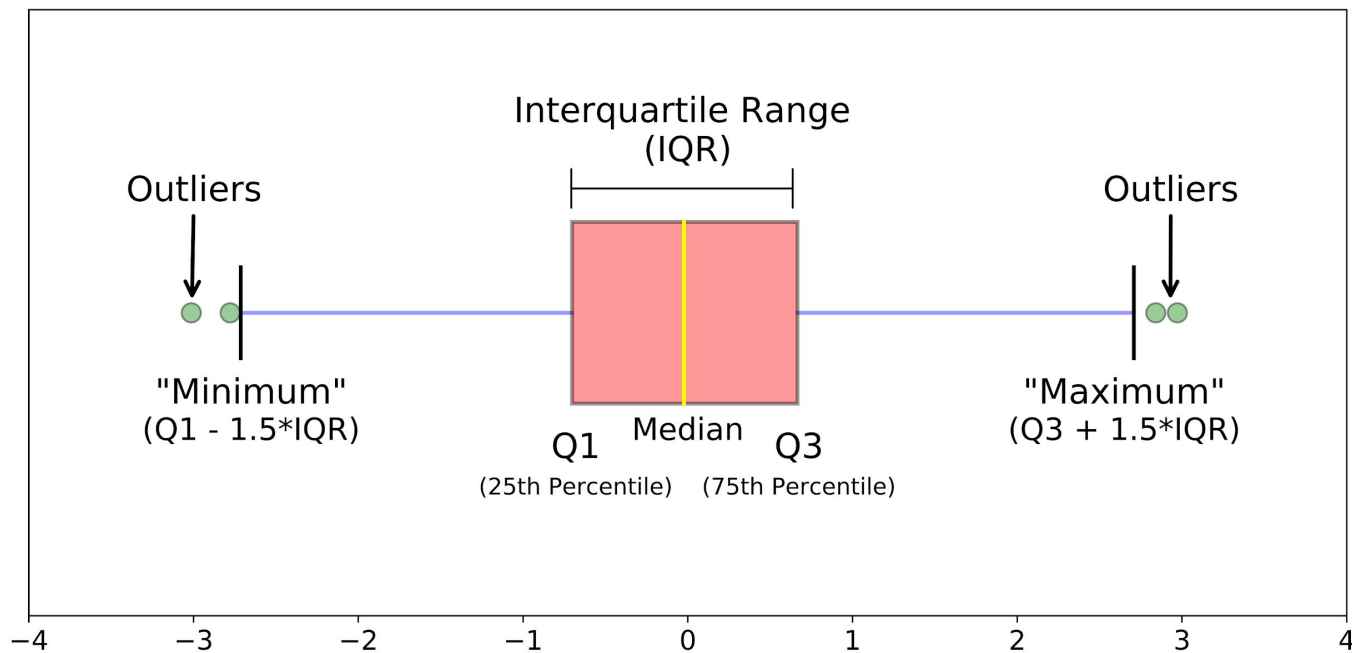
```
sns.set(style="whitegrid")
```

```
ax = sns.boxplot(x='Pclass', y='Age', data=df, palette='winter')
```



# Exploração dos Dados

- Boxplot





## Experimentos em ML



# Dados de Entrada - $X$

Conjunto de dados presentes na origem da utilização do modelo, com os quais se deseja prever uma saída

- Normalmente disponíveis como uma tabela de múltiplas colunas (Características) e linhas (exemplos) ou uma matriz:

$$X = \begin{bmatrix} [x_{11}, x_{12}, x_{13}, \dots, x_{1j}], \\ [x_{21}, x_{22}, x_{23}, \dots, x_{2j}] \end{bmatrix}$$

- Conhecidos como **Features/Características, atributos, covariáveis** ou **variáveis independentes**
  - **Atributos:** nome, ID, idade, cargo
  - **Características:** nome = 'Joao', ID = 2490, idade = 36, cargo = 'Consultor'
- Cada uma das característica é normalmente chamada de dimensão
  - Dimensionalidade = Número de características
  - Ex.: 5 características - problema 5-dimensional



# Conjunto de Saída - **y**

- É o resultado esperado a ser alcançado pelo algoritmo de aprendizagem
- Chamado também de **variável dependente**, **predição** ou **label**, quando valores são discretos
- Normalmente 'disponível' como uma coluna em uma tabela ou um vetor com valor único correspondente a cada conjunto de características de entrada  $X_i$

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

- Pode ter valor contínuo ou categórico
  - Valor médio gasto em um site
  - Bom ou mau pagador
  - Variação do preço de uma ação
  - Score, normalmente ordinal ou entre 0 e 1 - utilizado para Ranking





# Normalização

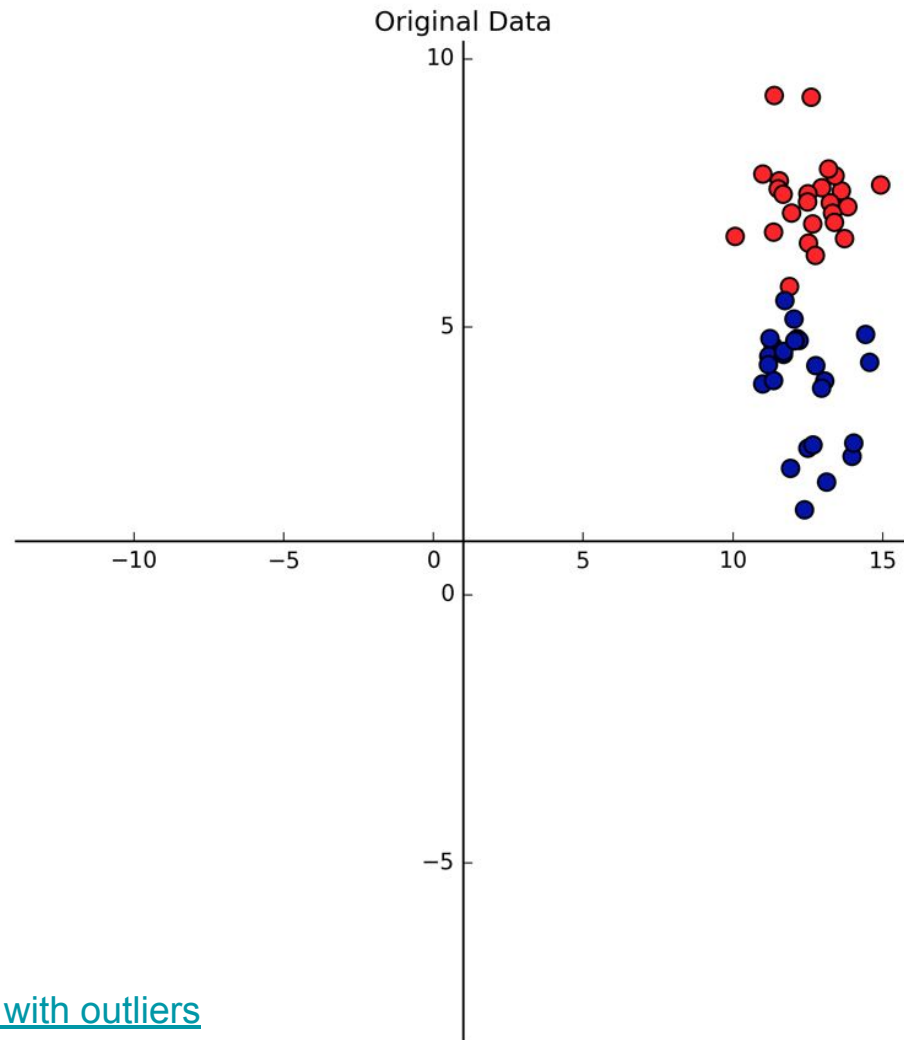
Alguns algoritmos são sensíveis à dimensão dos dados numéricos

Solução é ajustar estas características numéricas para que elas não atrapalhem o modelo escolhido

scikit

- StandardScaler
- RobustScaler
- MinMaxScaler
- Normalizer

Ver: [Compare the effect of different scalers on data with outliers](#)



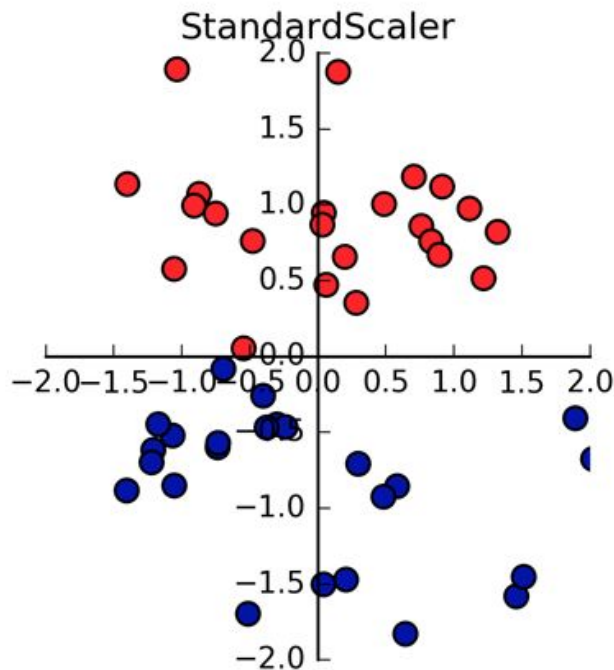
# Standard Scaler

$$x' = \frac{x - \bar{x}}{\sigma}$$

Transforma os valores de forma que a média seja igual a zero e a variância seja unitária

- Assume que as features são normalmente distribuídas
- Características tratadas com esta transformação tem a mesma magnitude
- Não garante que as features terão valores mínimos e máximos específicos
- Impactada por Outliers

```
data = [[0, 0], [0, 0], [1, 1], [1, 1]]  
scaler = StandardScaler()  
data_scaled = scaler.fit(data).transform(data)
```



Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

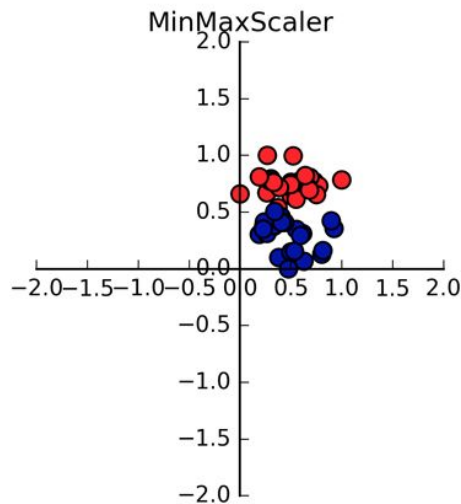
# MinMaxScaler

Transforma os dados - com relação a escala - para um intervalo pré-definido, de forma proporcional. Tipicamente é escolhido o intervalo [0, 1] (equação ao lado).

- Não altera a distribuição dos dados, não sendo a melhor opção para dados com outliers ou distorções severas.

```
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()  
scaler.fit(data)  
scaler.transform(data))
```

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$



# Combinação de atributos

Algumas vezes as features podem ser combinadas de forma a gerar outras mais significativas.

Exemplo: a partir da geolocalização temporal de um carro, quais características podem ser geradas?

$$X = [x, y, t]$$

$$X' = ?$$



# Combinação de atributos

Algumas vezes as features podem ser combinadas de forma a gerar outras mais significativas.

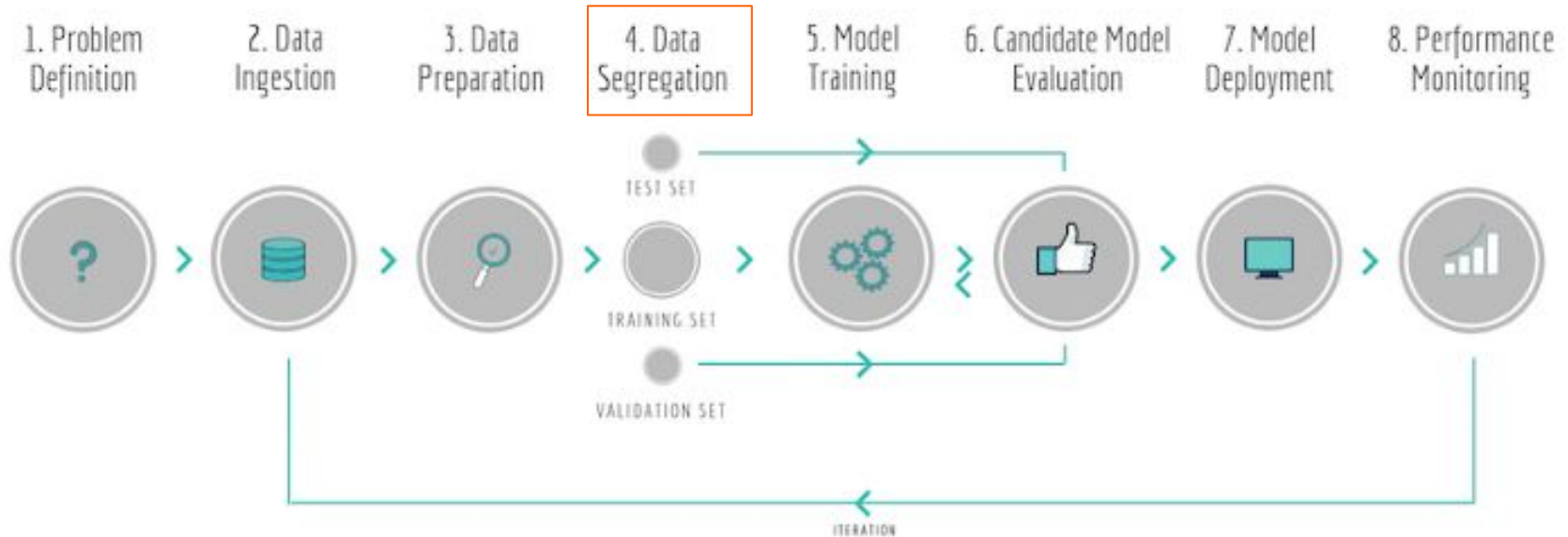
Exemplo: a partir da geolocalização temporal de um carro, quais características podem ser geradas?

$$X = [x, y, t]$$

$$X' = [x, y, t, v, a, \dots]$$



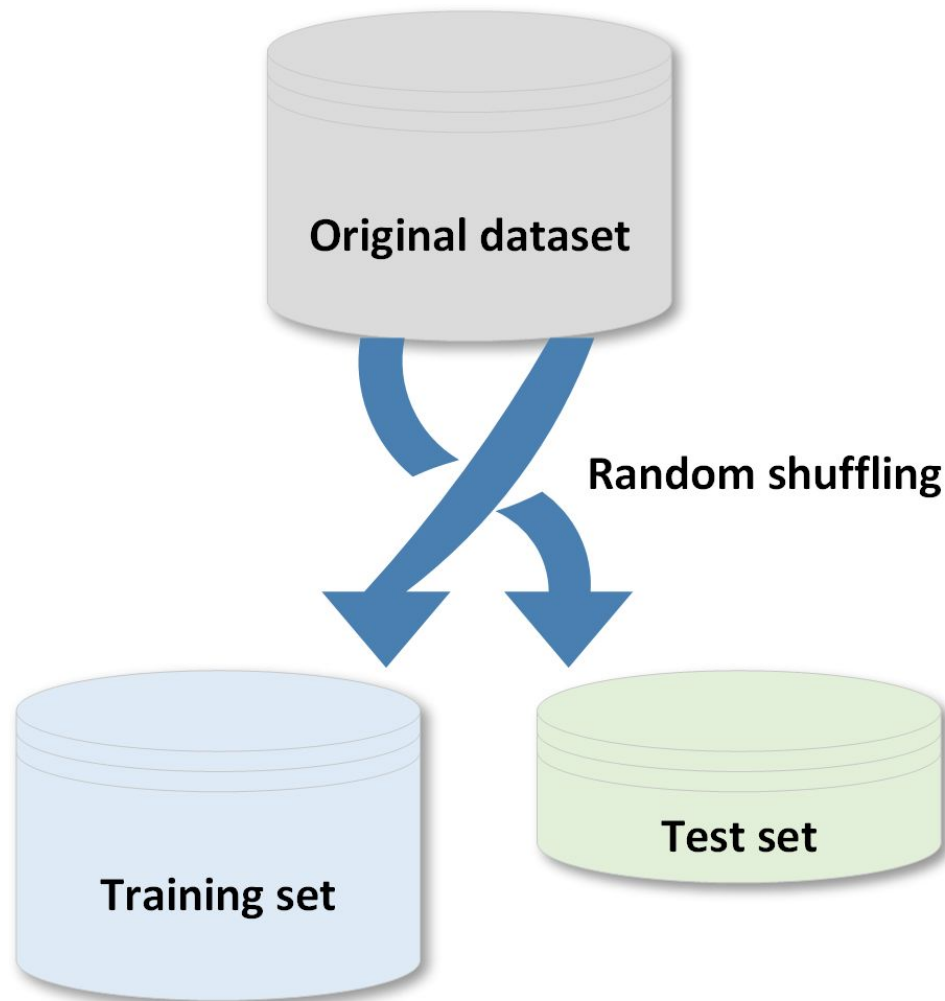
# Machine Learning Pipeline



# Holdout Method

O mais básico tipo de divisão de dados se dá em dois conjuntos:

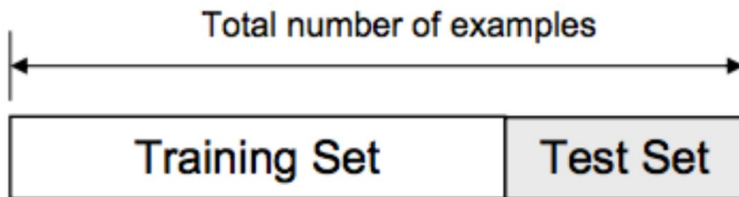
- **Base de Treinamento**  
Utilizado para 'alimentar' o algoritmo de ML
- **Base de Testes**  
Utilizado para avaliar o aprendizado adquirido pelo algoritmo de treinamento



# Divisão da base de dados - Holdout Method

O objetivo é dividir a base em **subconjuntos para treinamento e testes** de forma que o modelo possa ser avaliado de forma independente do treinamento

- **Forma de inferir como o modelo reage a dados não vistos**
- Premissas
  - Dados precisam representar o problema por completo
  - Amostras devem ser uniformemente distribuídas entre todos os subconjuntos





# Holdout Method

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

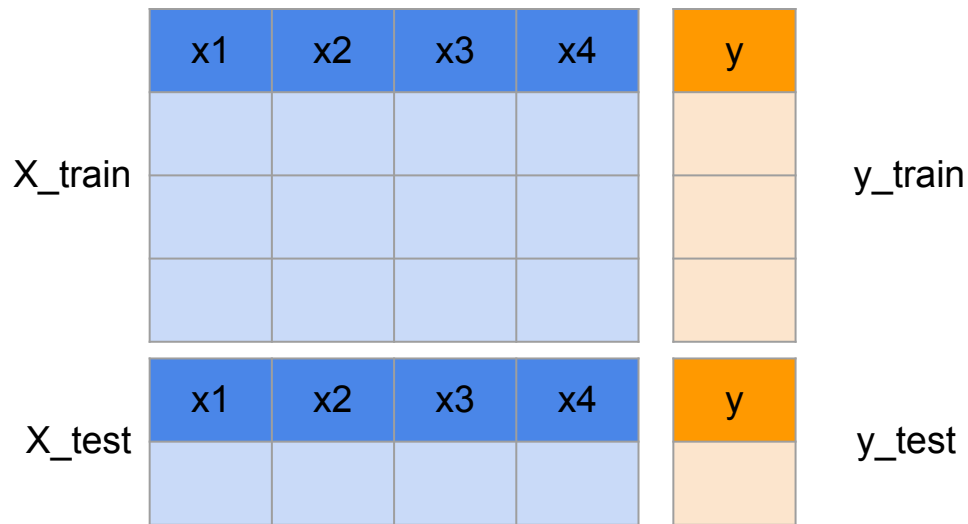
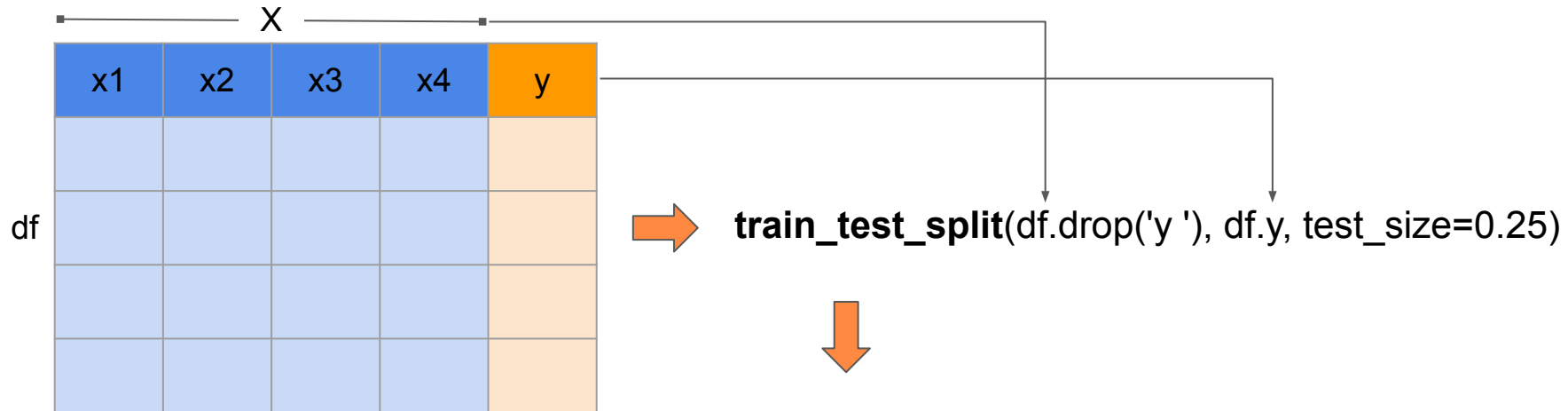
```
# X - dataframe com as características de treinamento
```

```
# y - dataframe com a saída (target ou label) equivalente a cada exemplo de entrada
```

```
# test_size - percentual dos dados utilizado para testes
```

```
# random_state : If int, random_state is the seed used by the random number generator
```





## [Warning] Holdout Method

- Todas as etapas do pré-processamento e treinamento devem ser feitas considerando-se que os dados de testes não estão disponíveis durante o treinamento
  - Objetivo é, de fato, avaliar como o modelo se comportaria em um ambiente real
  - Etapas de processamento de dados que calculam parâmetros a partir dos dados devem considerar apenas os dados de treinamento
  - Exemplo:

```
from sklearn.preprocessing import StandardScaler  
  
scaler = StandardScaler()  
scaler.fit(X_train)  
X_train_scaled = scaler.transform(X_train)  
X_test_scaled = scaler.transform(X_test)
```





# Hands On!

Tratamento e  
Divisão da Base de Dados



# Setup

Pra começar, vamos precisar do seguinte item:

- **Google Colab**
- Computação científica
  - [NumPy](#)
- Análise de Dados
  - [pandas](#)
- Machine Learning
  - [SciKit Learn](#)
- Visualização
  - [Matplotlib](#)
  - [Seaborn](#)



# Preparação de Dados - Titanic

Vamos utilizar uma conhecida base pública para praticar os conceitos vistos

Carregue a base de dados [Titanic do Kaggle](#) (train) no Jupyter em um DataFrame

- Faça o tratamento dos dados desta base de acordo com o que foi visto nesta aula, deixando-a preparada para aplicação de modelos de Machine Learning. Sempre que possível, gere visualizações que levem a insights sobre o problema sendo analisado.



# Preparação de Dados - Titanic

Vamos utilizar uma conhecida base pública para praticar os conceitos vistos

1. Carregue a base de dados Titanic no Colab em um DataFrame Pandas
  - `df = pd.read_csv('titanic_train.csv')`
2. Explore e exiba algumas informações úteis sobre os dados
  - Sugestão: `df.head` / `describe` / `df.info()`, `sns.countplot` / `histplot` / `boxplot` / `heatmap`...
3. Remova as colunas que não agregam valor e justifique
  - `df.drop()`
4. Trate valores nulos por coluna - Ex.: Como preencher os valores nulos da coluna 'Age', baseado em 'Pclass' e 'Sex'?



# Preparação de Dados - Titanic

5. Remova linhas que ainda contenham valores nulos
  - `df.dropna()`
6. Substitua colunas literais por valores numéricos
  - `get_dummies`, `concat`
7. Remova as colunas literais correspondentes
  - `df.drop()`
8. Divida a base de dados para ser treinada e testada
  - `train_test_split`
9. Normalize os valores numéricos que não são binários
  - `StandardScaler().fit(data).transform(data)`





# Onde encontrar dados reais?

- [UC Irvine Machine Learning Repository](#)
- [Amazon's AWS datasets](#)
- [Data Portals](#)
- [OpenDataMonitor](#)
- [Quandl](#)

Outros:

- [Wikipedia](#)
- [Quora.com](#)
- [reddit](#)



# Fontes de Dados - Governo

- [dados.gov.br](https://dados.gov.br)
- [dadosabertos.camara.leg.br](https://dadosabertos.camara.leg.br)
- [senado.leg.br/dados-abertos](https://senado.leg.br/dados-abertos)
- [justica.gov.br/dados-abertos](https://justica.gov.br/dados-abertos)
- [dados.pe.gov.br](https://dados.pe.gov.br)
- [dados.recife.pe.gov.br](https://dados.recife.pe.gov.br)
- [patiodigital.prefeitura.sp.gov.br](https://patiodigital.prefeitura.sp.gov.br)
- [Brasil.io](https://brasil.io)



# Fontes de Dados - Finanças

- [dadosabertos.bcb.gov.br](https://dadosabertos.bcb.gov.br)
- [B3 - Ações - Cotações históricas](#)
- [CVM - Séries históricas](#)



# Fontes de Dados - kaggle

≡ kaggle

+ Create

🏠 Home

🏆 Competitions

📁 Datasets

🔗 Code

💬 Discussions

🎓 Courses

⌵ More

🔍 Search

Sign In

Register

## Datasets

Explore, analyze, and share quality data. [Learn more](#) about data types, creating, and collaborating.

+ New Dataset

🔍 Search datasets

⌵ Filters

Datasets

Tasks

Computer Science

Education

Classification

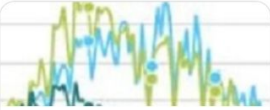
Computer Vision

NLP

Data Visualization

📈 Trending Datasets


See All



### Israel Coronavirus DataSet

yvtan levy · Updated 7 hours ago  
Usability 8.8 · 30 kB  
2 Files (CSV)


6



### Farsnews-1399

Arman Malekzadeh Lashkaryani · Update...  
Usability 10.0 · 439 MB


7



### Military Equipment for Local Law Enforcement

JohnM · Updated a day ago  
Usability 9.4 · 7 MB  
2 Files (CSV, other)

3



### Taylor Swift Song Lyrics (All Albums)

Jan Llenzi Dagohoy · Updated a day ago  
Usability 10.0 · 127 kB  
9 Files (CSV)

10

# Para continuar...

- Katti Faceli et al. Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina. LTC, 2019.
  - **Capítulo 2. Análise de dados**
  - **Capítulo 3. Pré-processamento de dados**
- Sarah Guido; Andreas Müller. Introduction to Machine Learning with Python - A Guide for Data Scientists. O'Reilly Media, 2016.
  - **Chapter 1. Introduction**
- Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 2a Edição: O'Reilly Media, 2019.
  - **Chapter 2. End-to-End Machine Learning Project**



## Outros:

- [LGPD](#)
- [O que você deve saber sobre a lei de proteção de dados pessoais do Brasil](#)
- [Scikit Learn - Preprocessing data](#)





C . E . S . A . R

sch∞l

