Hands-On Data Science

Uma introdução a conceitos e técnicas de Ciência de Dados

Por quê estudar ciência de dados?

Nasser Santiago Boan

> arquivologia - UNB

> tecnologia do Petróleo e Gás - UFRJ

> MBA Processos de Negócio - IBMEC

> IBM Data Science Professional Certified

> líder de Ciência de Dados - Certsys

> docente Pós-Graduação Ciência de Dados

Ementa

1. Metodologia na Ciência de Dados
2. Estatística descritiva básica
3. Data Viz
4. Aprendizagem Supervisionada - Regressão
5. Projeto -> Valores de Casa
6. Aprendizagem Supervisionada - Classificação
7. Projeto -> Titanic

ALIBABA TECH HEALTH CARE CORONAVIRUS CHINA DIAGNOSIS

Alibaba's new AI system can detect coronavirus in seconds with 96% accuracy

AAAS Become a Member

Science Contents News C

SHARE

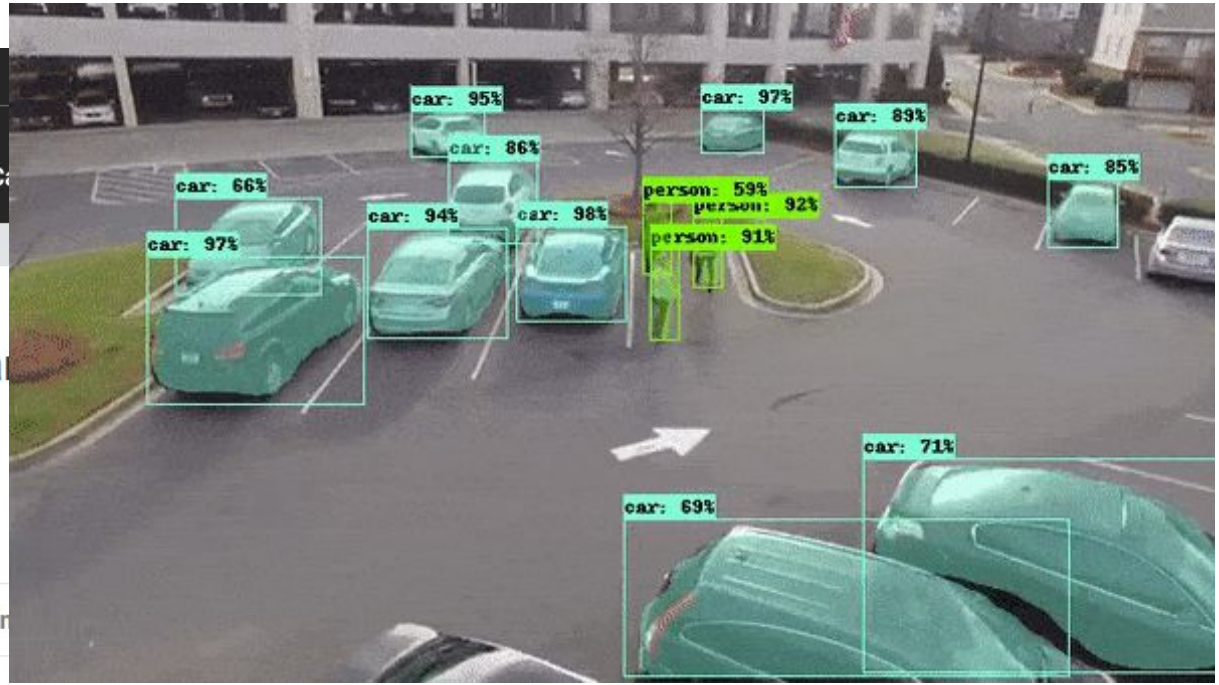
PERSPECTIVE | MEDICINE

Artificial intelligence in ca

Dean Ho
+ See all authors and affiliations

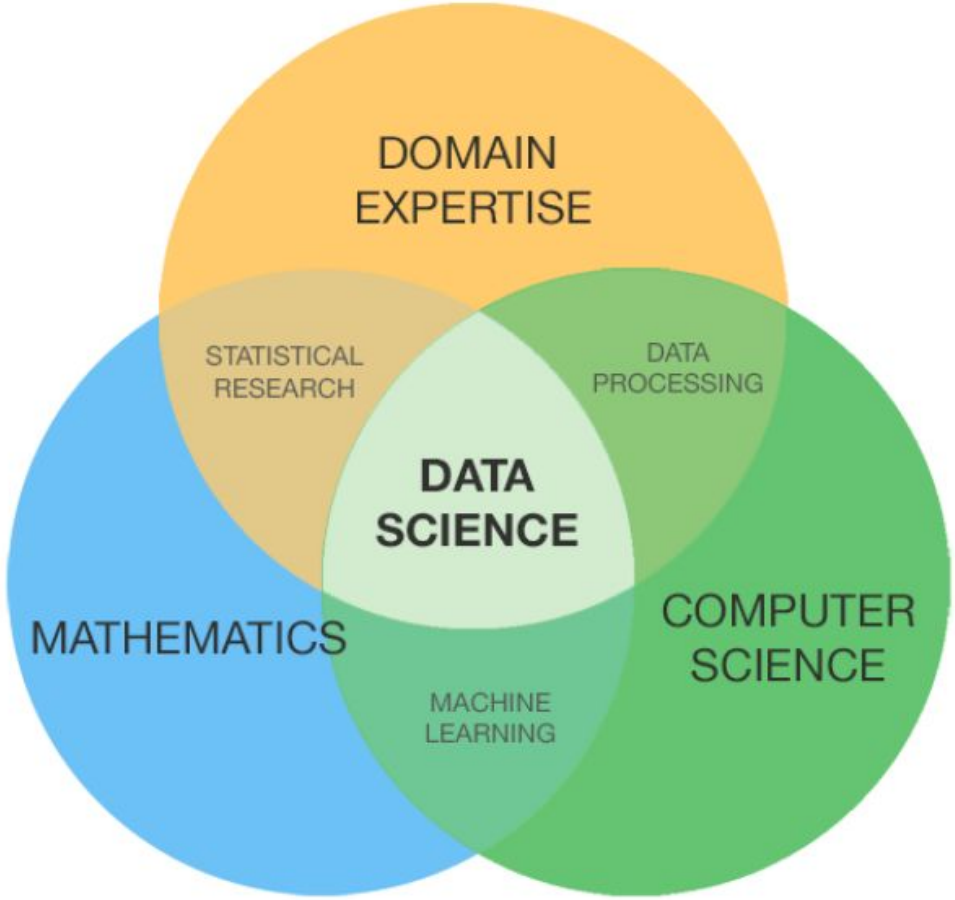
Science 28 Feb 2020:
Vol. 367, Issue 6481, pp. 982-983
DOI: 10.1126/science.aaz3023

Article Figures & Data



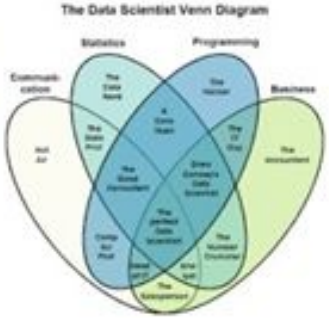
21st

O que é a ciência de dados?



Source: Palmer, Shelly. Data Science for the C-Suite. New York: Digital Living Press, 2015. Print.


Computer Science
Machine Learning
Maths & Statistics
Data Science
Traditional Software
Data Analysis
Business / Domain Expertise

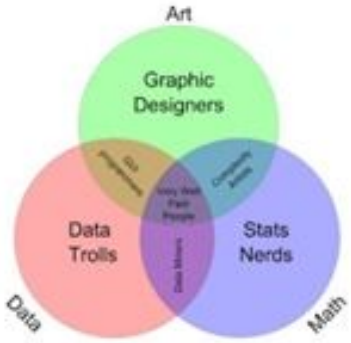

The Data Scientist Venn Diagram
Statistics
Programming
Communication
Business
Data


Coding
Stats
Data Science
Domain


Social Sciences
Hacking Skills
Traditional Research
Statistics
Danger Zone
Traditional Research
Domain Expertise

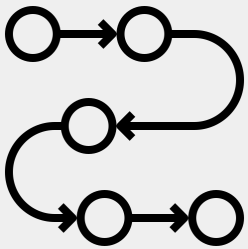

Hacking Skills
Math & Statistics Knowledge
Data Science
Substantive Expertise


Data Science


Art
Graphic Designers
Data Trolls
Stats Nerds
Data
Math

Você está usando dados? Você está tomando decisões baseadas neles? Está usando um processo pra isso?

Você está fazendo Data Science!



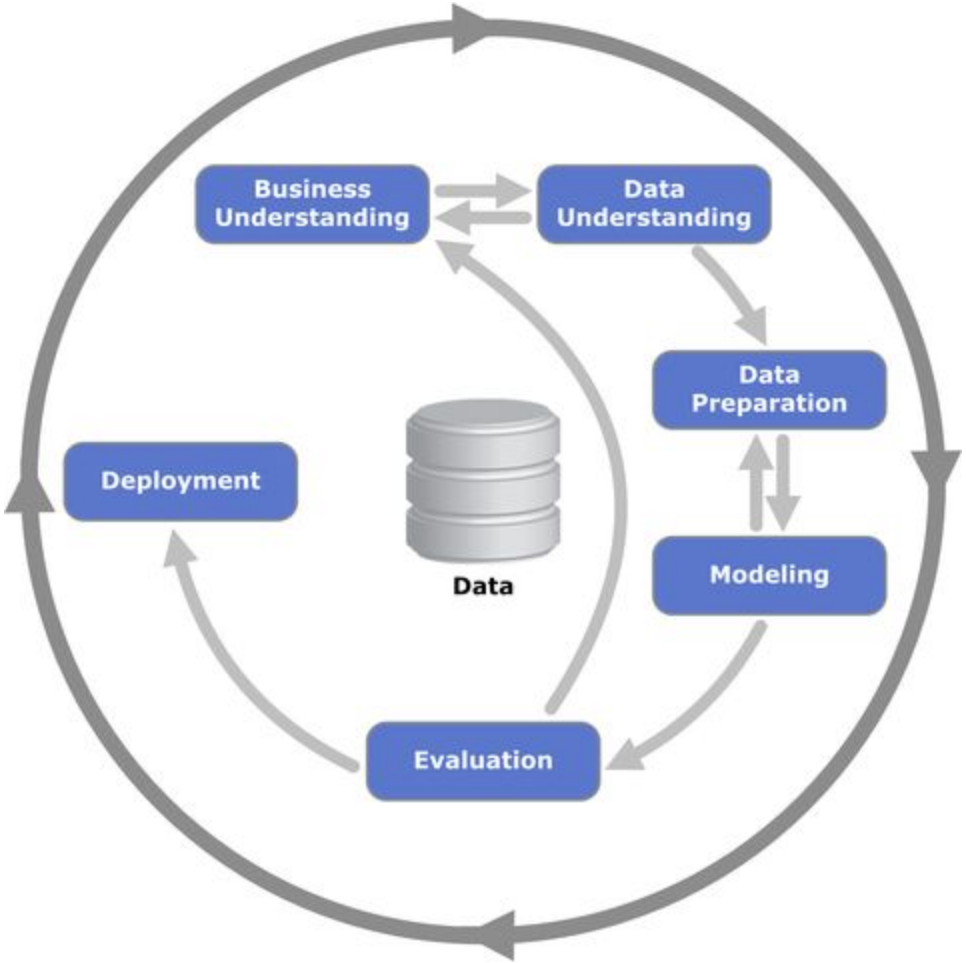
Metodologia na Ciência de Dados

Como fazer

ciência de dados?

CRoss-Industry Standard Process for Data Mining

… CRISP-DM


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

> Entender a situação atual!

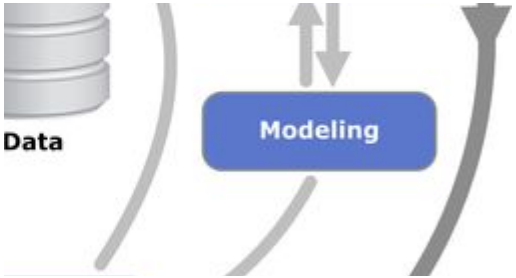
- Preciso desse trabalho porque ...
- O resultado desse trabalho é melhor do que ...
- Para entregar esse trabalho eu preciso ...
- Vou acompanhar meu sucesso usando ...
- Vou precisar investir ...

> Entender os dados

- Coletar, se necessário
- Entender as fontes
- Descrever os dados
- Explorar os dados
- Avaliar qualidade dos dados

> Preparar os dados

- Limpar os dados
- Feature Engineering
- Integrar outras bases
- Formatar os dados (tokenização, vetorização, split etc)



> Modelar

- Criar um experimento
- Construir e treinar modelo baseline
- Construir e treinar modelos experimentais

```
[41]  1  dummy = DummyRegressor()
      2  dummy.fit(x,y)
```

> Avaliar

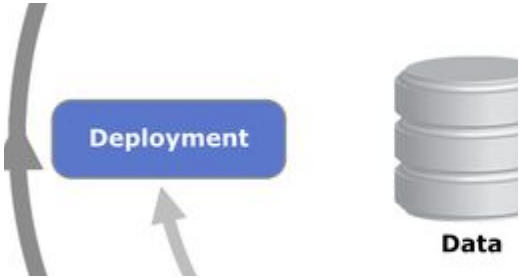
- Avaliar resultados
- Revisar processo
- Determinar próximos passos

```
[41]  1  dummy = DummyRegressor()
      2  dummy.fit(x,y)
      3  np.sqrt(mean_squared_error(y,dummy.predict(x)))

   677.0409870632897
```

> Avaliar

- Avaliar resultados
- Revisar processo
- Determinar próximos passos

```
[41]  1  dummy = DummyRegressor()
      2  dummy.fit(x,y)
      3  np.sqrt(mean_squared_error(y,dummy.predict(x)))

   677.0409870632897
```

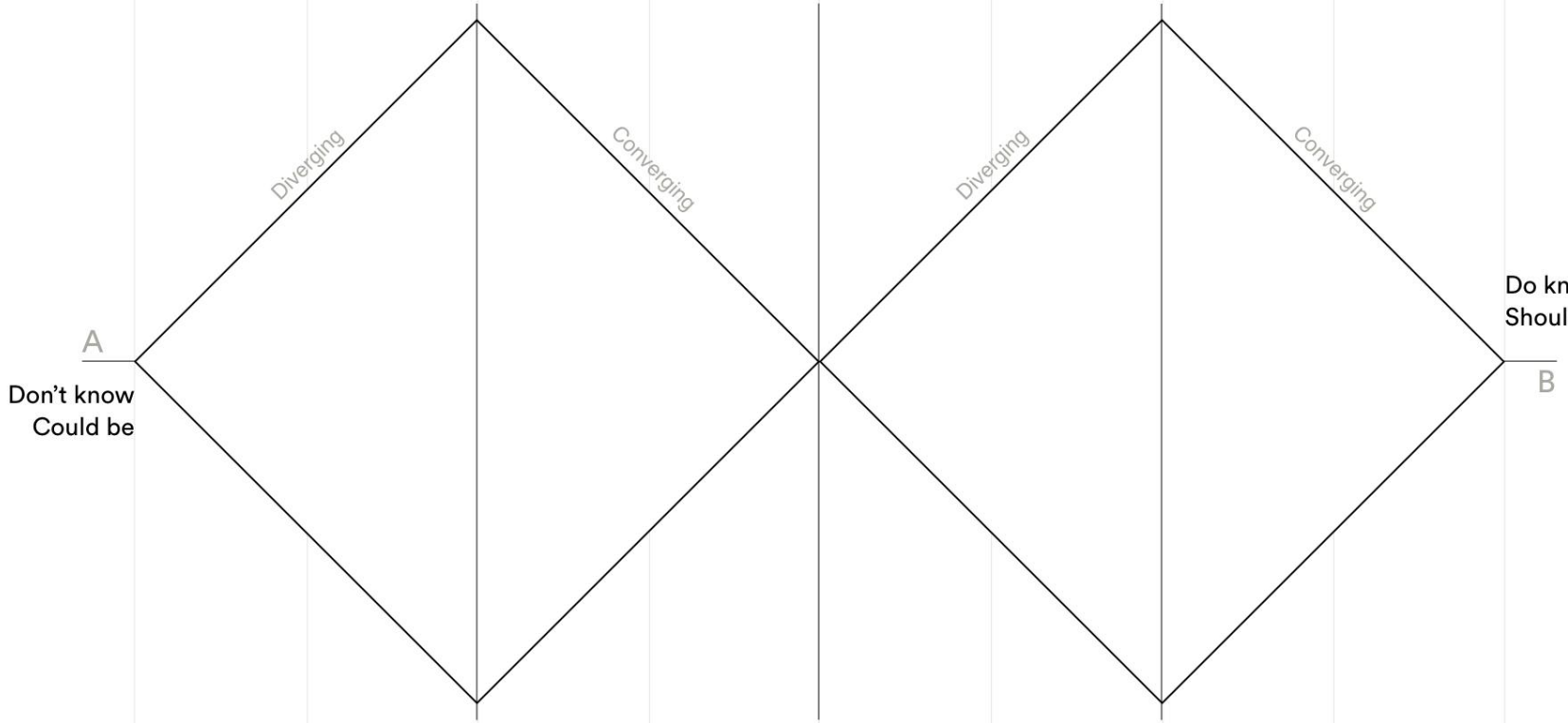


> Deployment

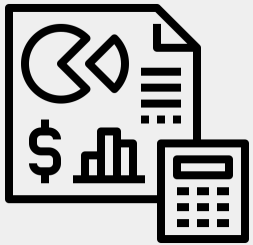
- Planejar deployment
- Planejar monitoramento e manutenção
- Produzir produto final
- Revisar projeto

Revamped Double Diamond

Four steps



Hinderling Volkart, Dan Nessler, January 2018
hinderlingvolkart.com, @dnessler.com



Python & Estatística

conda

conda

environment.yml

```yaml
name: stats
dependencies:
  - jupyterlab
  - pandas
  - numpy
  - scikit-learn
```

```
(base) nzboan@ubutop:~$ conda env create -f environment.yml
```

```
(base) nzboan@ubutop:~$ conda env create -f environment.yml
Collecting package metadata (repodata.json): done
Solving environment: done

Downloading and Extracting Packages
pandas-1.0.0         | 8.8 MB    | ##################################### | 100%
jupyterlab-1.2.5     | 2.8 MB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate stats
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) nzboan@ubutop:~$
```

Jupyter

Installing the Jupyter Software

Get up and running with the JupyterLab or the classic Jupyter Notebook on your computer within minutes!

Getting started with JupyterLab

Installation

JupyterLab can be installed using `conda` or `pip`. For more detailed instructions, consult the installation guide.

conda

If you use `conda`, you can install it with:

```
conda install -c conda-forge jupyterlab
```

pip

If you use `pip`, you can install it with:

```
pip install jupyterlab
```

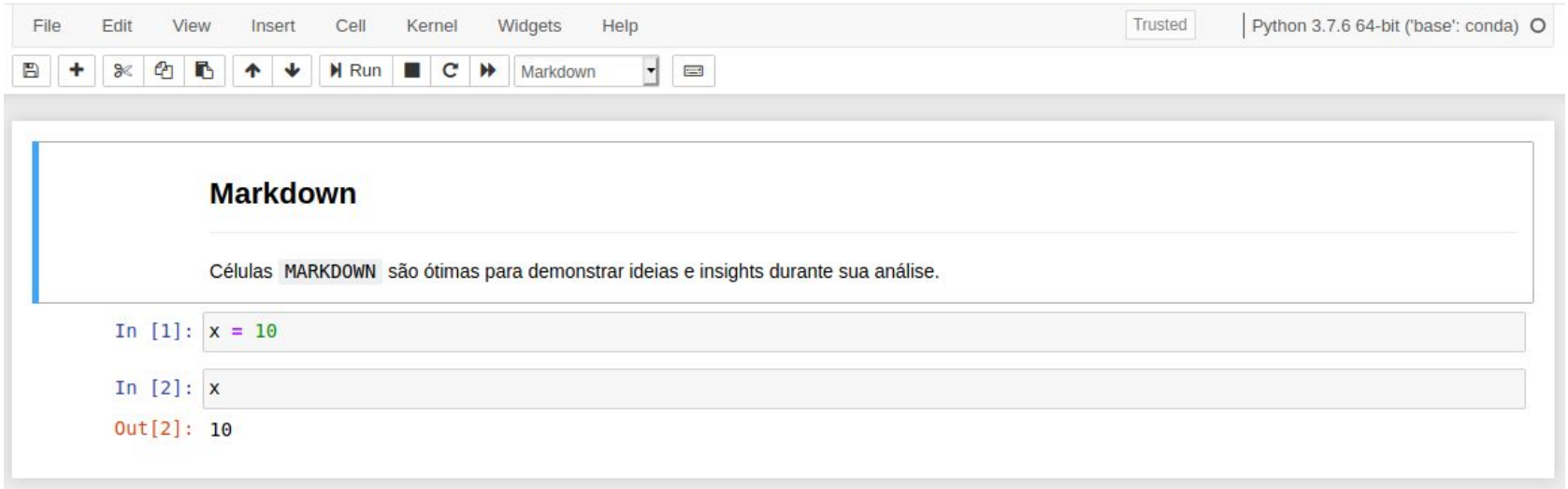
If installing using `pip install --user`, you must add the user-level `bin` directory to your `PATH` environment variable in order to launch `jupyter lab`.

https://jupyter.org/install

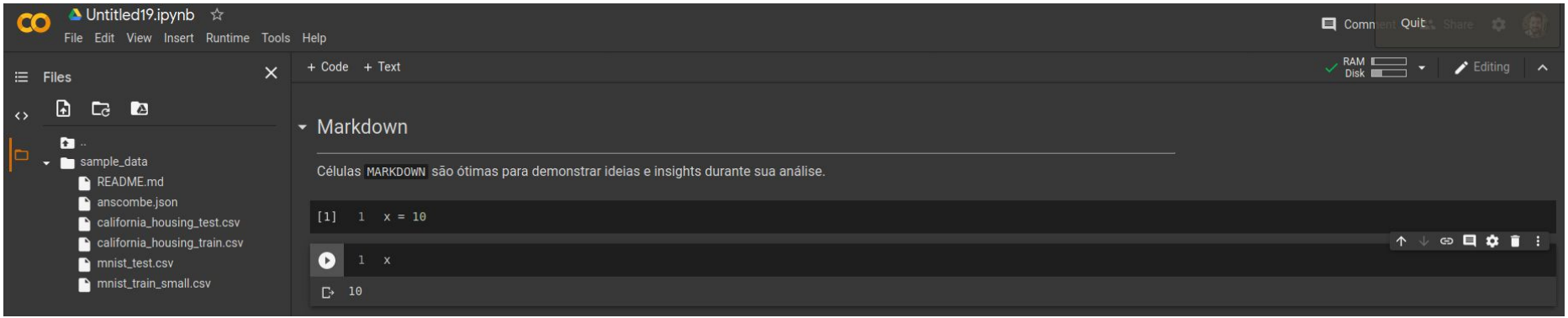
Jupyter


jupyter
Quit
Logout
Files
Running
Clusters
Conda
Select items to perform actions on them.
Upload
New
0
/
Name
Last Modified
File size
content
17 days ago
mlruns
11 days ago
style
17 days ago
mlflow.ipynb
11 days ago
4.72 kB
Untitled.ipynb
3 months ago
493 kB
Untitled1.ipynb
3 months ago
806 B
Untitled2.ipynb
17 days ago
591 kB
hello.py
a month ago
323 B
model.png
3 months ago
36.2 kB
network.gv
3 months ago
3.31 kB
network.gv.pdf
3 months ago
20.3 kB

Jupyter

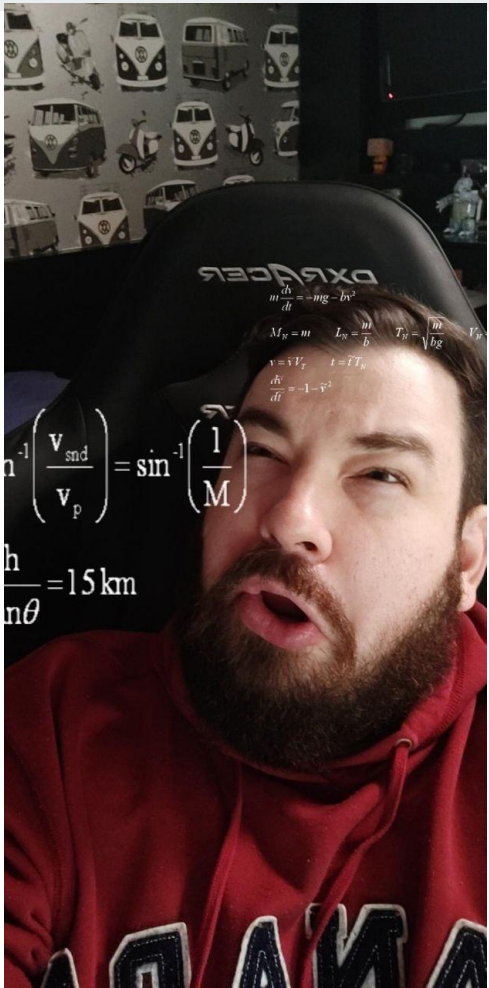

File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3.7.6 64-bit ('base': conda)
Run
Markdown
Markdown
Células MARKDOWN são ótimas para demonstrar ideias e insights durante sua análise.
In [1]:
x = 10
In [2]:
x
Out[2]:
10

Colab


Untitled19.ipynb
File Edit View Insert Runtime Tools Help
Files
+ Code + Text
RAM
Disk
Editing
Markdown
Células MARKDOWN são ótimas para demonstrar ideias e insights durante sua análise.
[1] 1 x = 10
1 x
10
sample_data
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv

Estatística

1. Média
2. Moda
3. Mediana
4. Percentis
5. Skew
6. Correlação
7. Análise gráfica



Média

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

In [36]:
```
import numpy as np

ls = [1,15,88,4,44,22,34,5,6,7,55,66,77,92,588,144,157]

np.mean(ls)
```

Out[36]: 82.6470588235294

In [37]:
```
sum(ls)/len(ls)
```

Out[37]: 82.6470588235294

In [39]:
```
## media dos preços de venda das casas

np.mean(data.SalePrice)
```

Out[39]: 180921.19589041095

In [43]:
```
sum(data.SalePrice)/len(data.SalePrice)
```

Out[43]: 180921.19589041095

Moda

In [15]:
```
## encontrando a moda

import statistics
statistics.mode(data.SaleCondition)
```

Out[15]:
```
'Normal'
```

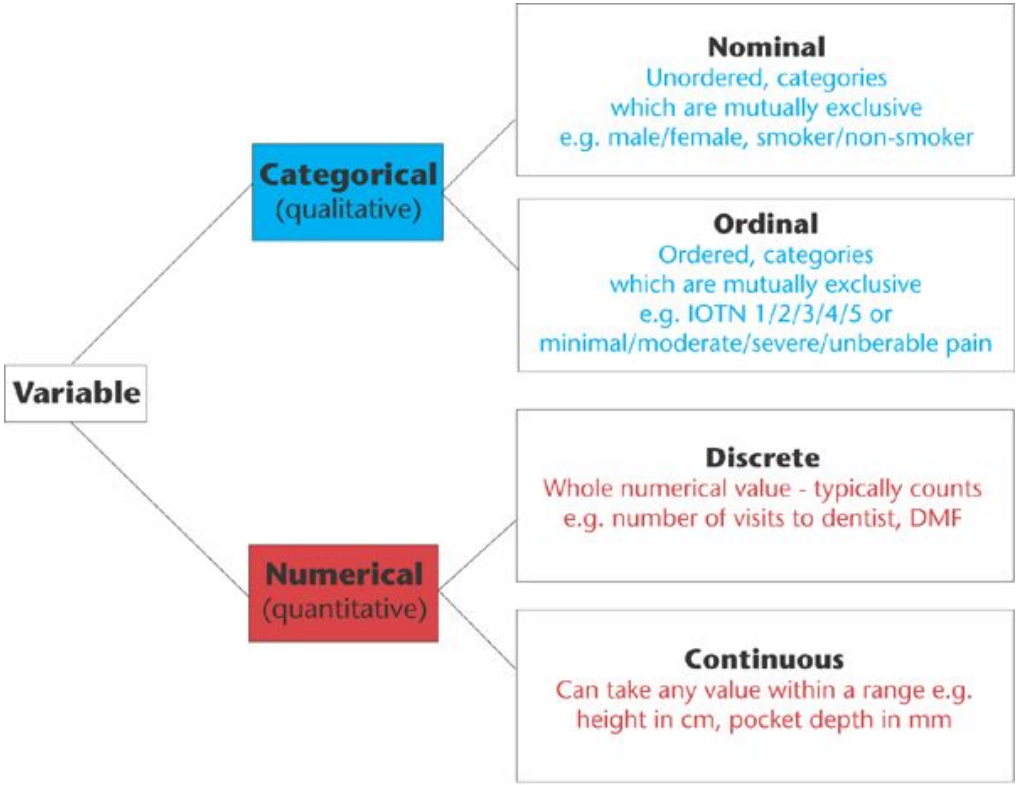
In [16]:
```
## usando o pandas

data.SaleCondition.value_counts()
```

Out[16]:
```
Normal     1198
Partial     125
Abnorml     101
Family       20
Alloca       12
AdjLand       4
Name: SaleCondition, dtype: int64
```

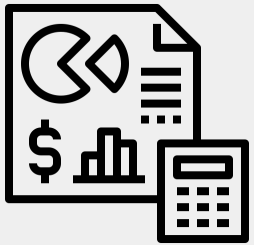
In [17]:
```
## usando o pandas

data.SaleCondition.value_counts(normalize=True)
```

Out[17]:
```
Normal     0.820548
Partial    0.085616
Abnorml    0.069178
Family     0.013699
Alloca     0.008219
AdjLand    0.002740
Name: SaleCondition, dtype: float64
```

Tipos de Dados



Mediana e Separatrizes
Skew
Correlação



Data-Viz

Storytelling

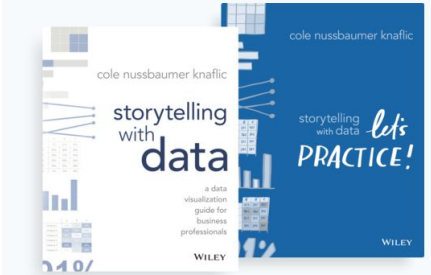
understand the
context

choose an
effective visual

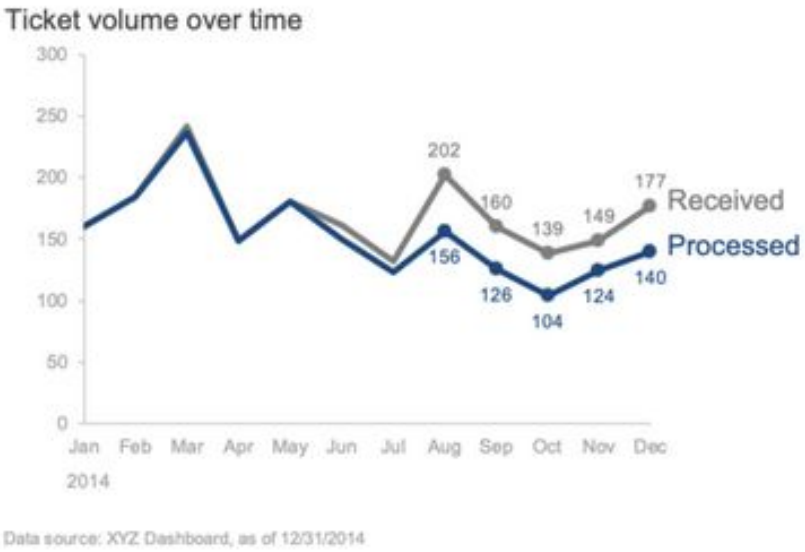
eliminate
clutter

focus
attention

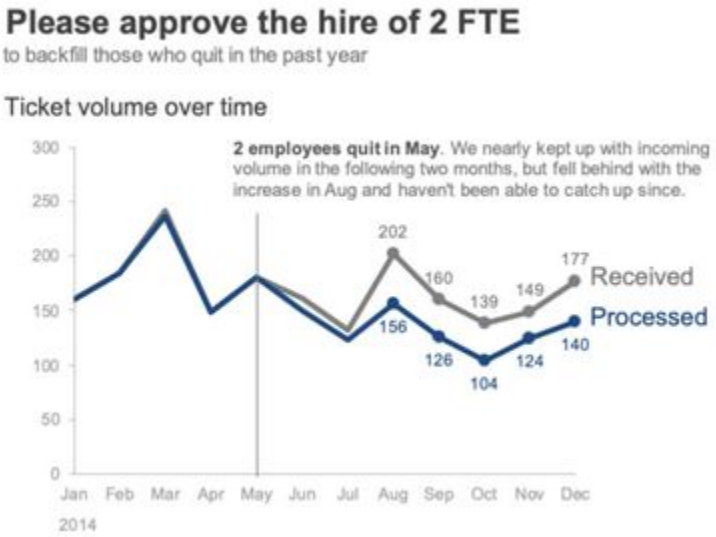
tell a
story



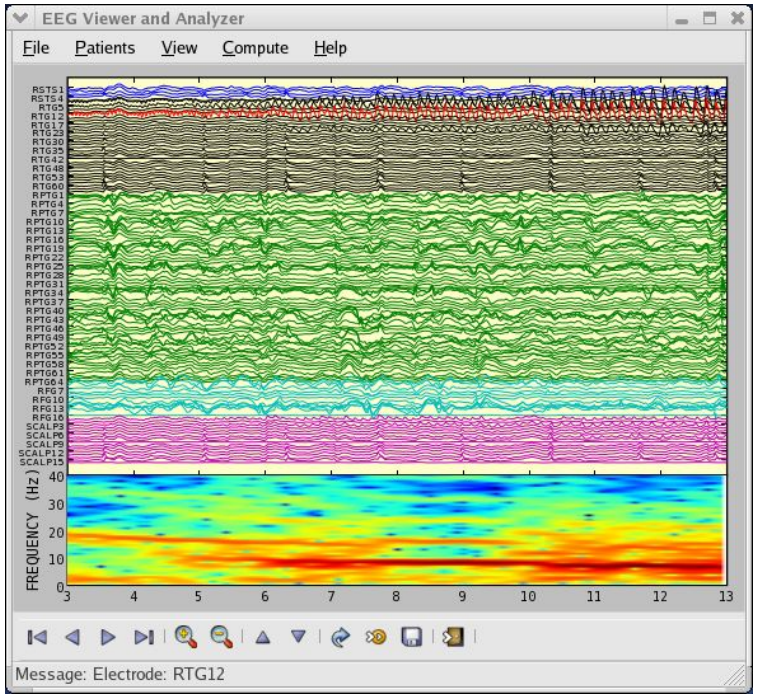
Storytelling



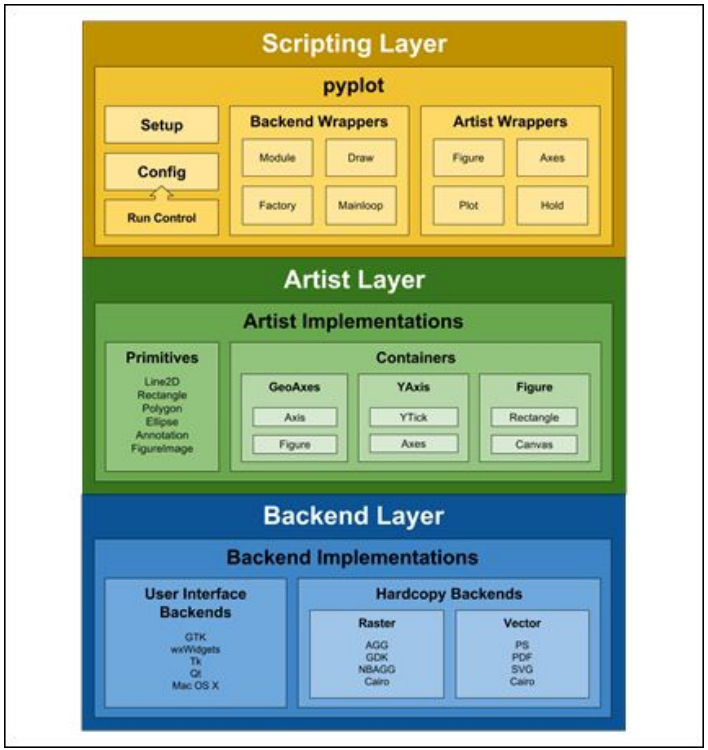
Storytelling



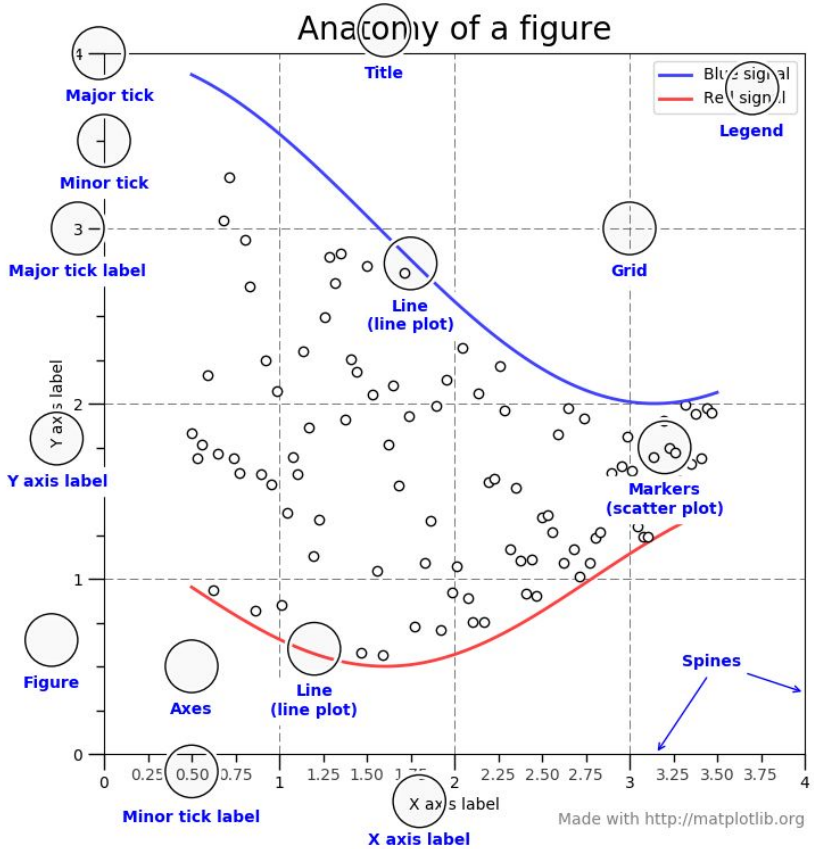
Matplotlib


EEG Viewer and Analyzer
File
Patients
View
Compute
Help
FREQUENCY (Hz)
40
30
20
10
0
3
4
5
6
7
8
9
10
11
12
13
Message: Electrode: RTG12

Matplotlib


Scripting Layer
pyplot
Setup
Config
Run Control
Backend Wrappers
Module
Draw
Factory
Mainloop
Artist Wrappers
Figure
Axes
Plot
Hold
Artist Layer
Artist Implementations
Primitives
Line2D
Rectangle
Polygon
Ellipse
Annotation
FigureImage
Containers
GeoAxes
Axis
Figure
YAxis
YTick
Axes
Figure
Rectangle
Canvas
Backend Layer
Backend Implementations
User Interface Backends
GTK
wxWidgets
Tk
Qt
Mac OS X
Hardcopy Backends
Raster
AGG
GDK
NBAGG
Cairo
Vector
PS
PDF
SVG
Cairo

Matplotlib


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Legend
Grid
Line
(line plot)
Y axis label
Markers
(scatter plot)
Figure
Axes
Line
(line plot)
Spines
Minor tick label
X axis label
Made with http://matplotlib.org

Matplotlib

```
[21]  1  df = pd.read_csv('avocado.csv')
      2  df.head()
```

	Unnamed: 0	Date	AveragePrice	Total Volume	4046	4225	4770	Total Bags	Small Bags	Lar Ba
0	0	2015-12-27	1.33	64236.62	1036.74	54454.85	48.16	8696.87	8603.62	93.
1	1	2015-12-20	1.35	54876.98	674.28	44638.81	58.33	9505.56	9408.07	97.
2	2	2015-12-13	0.93	118220.22	794.70	109149.67	130.50	8145.35	8042.21	103.
3	3	2015-12-06	1.08	78992.15	1132.00	71976.41	72.58	5811.16	5677.40	133.
4	4	2015-11-29	1.28	51039.60	941.48	43838.39	75.78	6183.95	5986.26	197.

Matplotlib

matplotlib.pyplot.figure

```
matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None,
frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kwargs)
```

[source]

Matplotlib

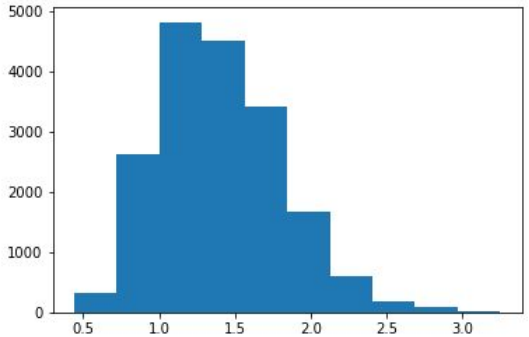
```
[2]  1  plt.figure();
```

```
<Figure size 432x288 with 0 Axes>
```

Matplotlib

```
In [8]: plt.figure()
        plt.hist(df.AveragePrice)
```

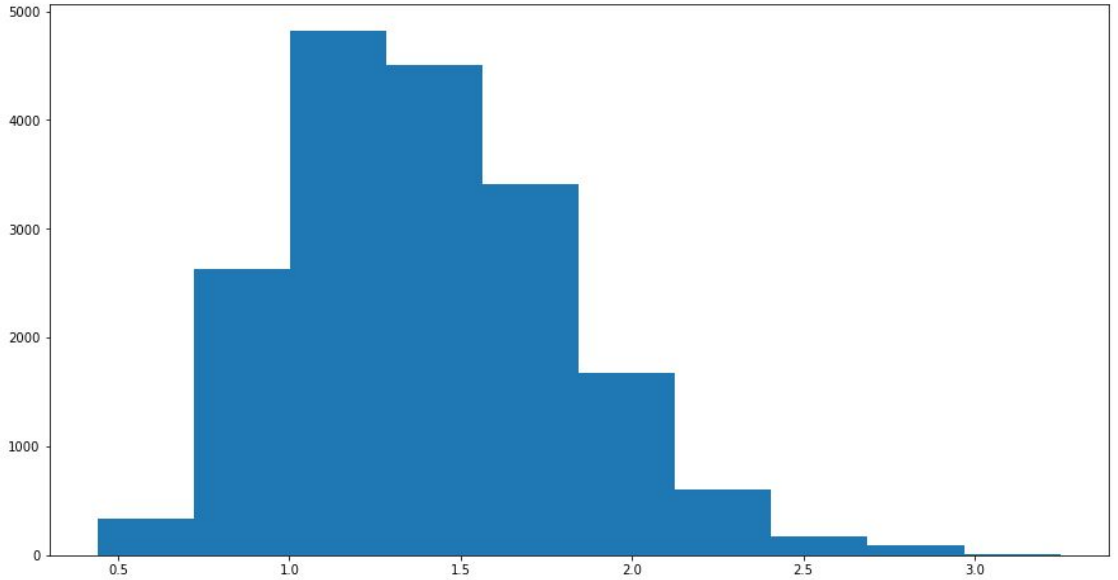
```
Out[8]: (array([ 331., 2632., 4824., 4506., 3412., 1672.,  598.,  177.,   86.,
                  11.]),
         array([0.44 , 0.721, 1.002, 1.283, 1.564, 1.845, 2.126, 2.407, 2.688,
                2.969, 3.25 ]),
         <a list of 10 Patch objects>)
```



Matplotlib

```
In [9]: plt.figure(figsize=(15,8))
        plt.hist(df.AveragePrice)
```

```
Out[9]: (array([ 331., 2632., 4824., 4506., 3412., 1672.,  598.,  177.,   86.,
                  11.]),
         array([0.44 , 0.721, 1.002, 1.283, 1.564, 1.845, 2.126, 2.407, 2.688,
                2.969, 3.25 ]),
         <a list of 10 Patch objects>)
```



Matplotlib

matplotlib.pyplot.bar

```
matplotlib.pyplot.bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)
```

[source]

Matplotlib

In [10]:
```
## agrupando os dados necessários

data = (

    df.loc[:,['year','AveragePrice']]
    .groupby('year')
    .mean()

)
```

Matplotlib

In [11]:
```
## criando o array de valores necessarios para o tamanho do plt.bar

tamanho = data.values.flatten()
tamanho
```

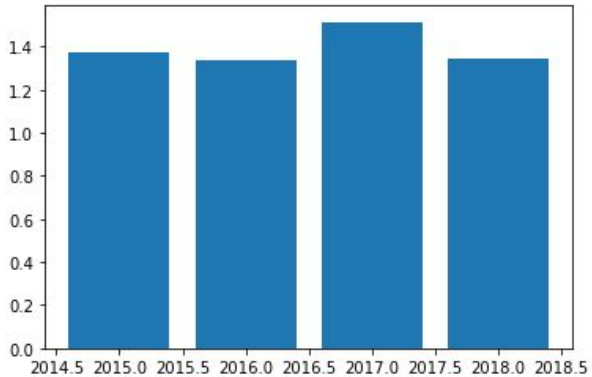
Out[11]:
```
array([1.37559038, 1.3386396 , 1.51512758, 1.34753086])
```

In [12]:
```
## criando a lista de valores necessarios para os indices do plt.bar

x = list(data.index)
x
```

Out[12]:
```
[2015, 2016, 2017, 2018]
```

Matplotlib

In [13]:
```
## criando um plt.bar com tamanho e x

plt.figure()
plt.bar(x = x, height= tamanho)
plt.show()
```



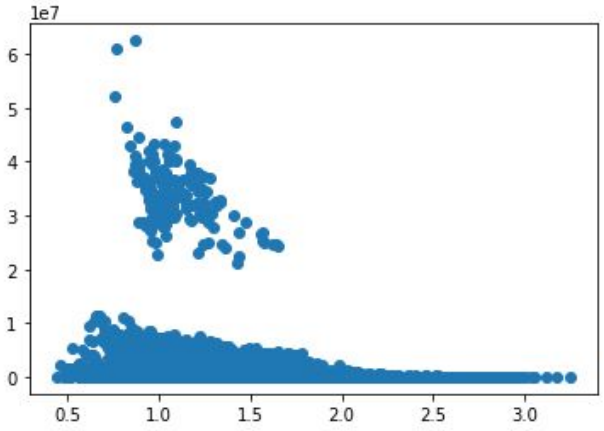
Matplotlib

matplotlib.pyplot.scatter

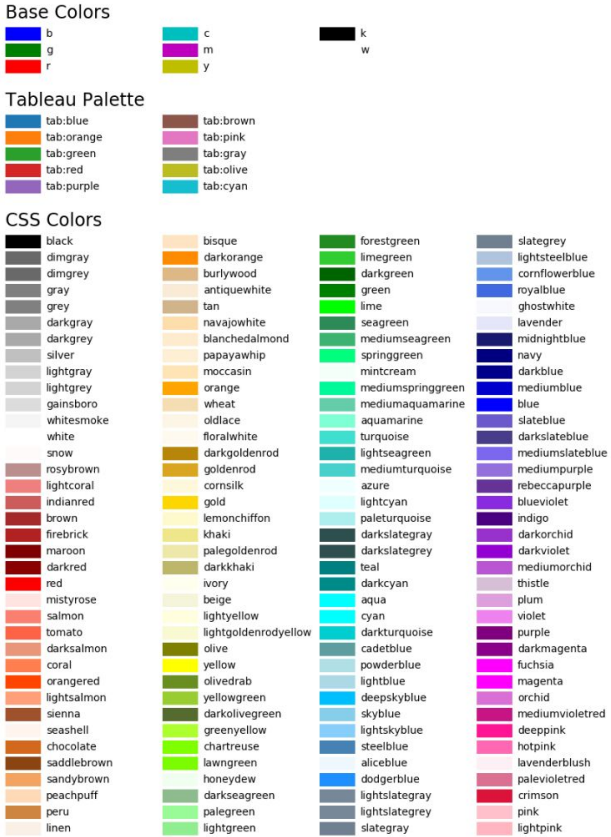
```
matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None,
vmax=None, alpha=None, linewidths=None, verts=<deprecated parameter>, edgecolors=None, *,
plotnonfinite=False, data=None, **kwargs)                                              [source]
```

Matplotlib

In [14]:
```
## criando um scatterplot com AvaragePrice

plt.figure()
plt.scatter(x=df['AveragePrice'],y=df['Total Volume'])
plt.show()
```



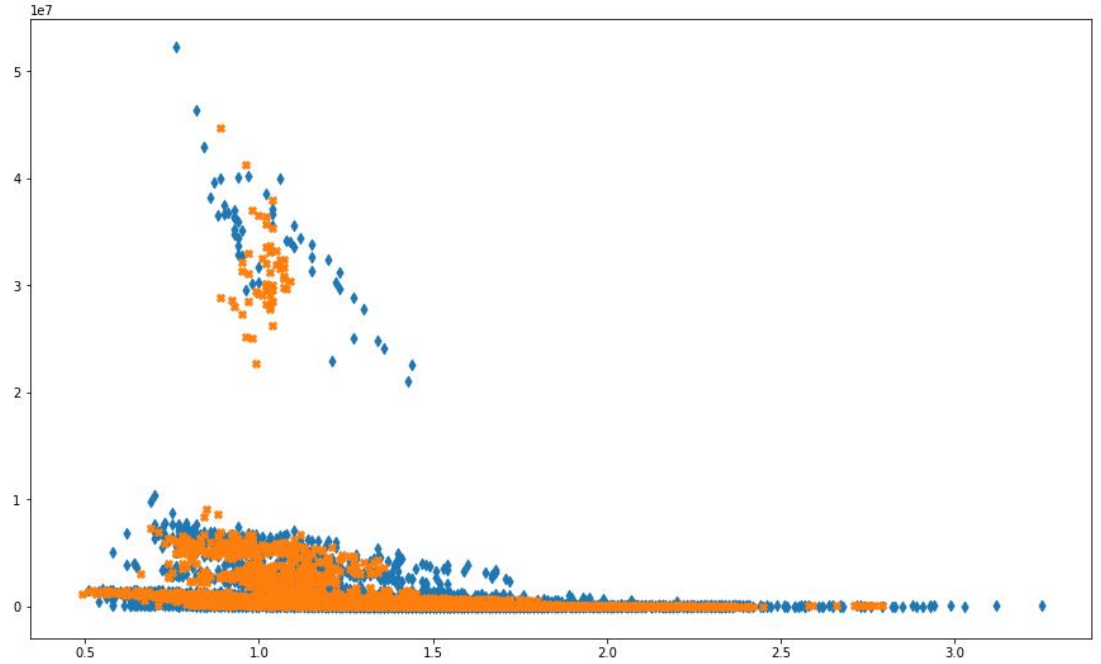
Matplotlib



marker	symbol	description
"."	●	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	Y	tri_down
"2"	⅄	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	●	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star
"h"	⬢	hexagon1
"H"	⬣	hexagon2
"+"	+	plus
"x"	×	x
"X"	✖	x (filled)
"D"	◆	diamond
"d"	♦	thin_diamond

Matplotlib

In [15]:
```
## adicionando dois dados num só grafico, adicionando cor, adicionando marcadores

plt.figure(figsize=(15,9))
plt.scatter(x = df[df.year == 2016].AveragePrice, y = df[df.year == 2016]['Total Volume'],
            c = 'tab:blue' , marker = 'd' )
plt.scatter(x = df[df.year == 2015].AveragePrice, y = df[df.year == 2015]['Total Volume'],
            c = 'tab:orange' , marker = 'X' )
plt.show()
```



Matplotlib

matplotlib.pyplot.legend

`matplotlib.pyplot.legend(*args, **kwargs)`

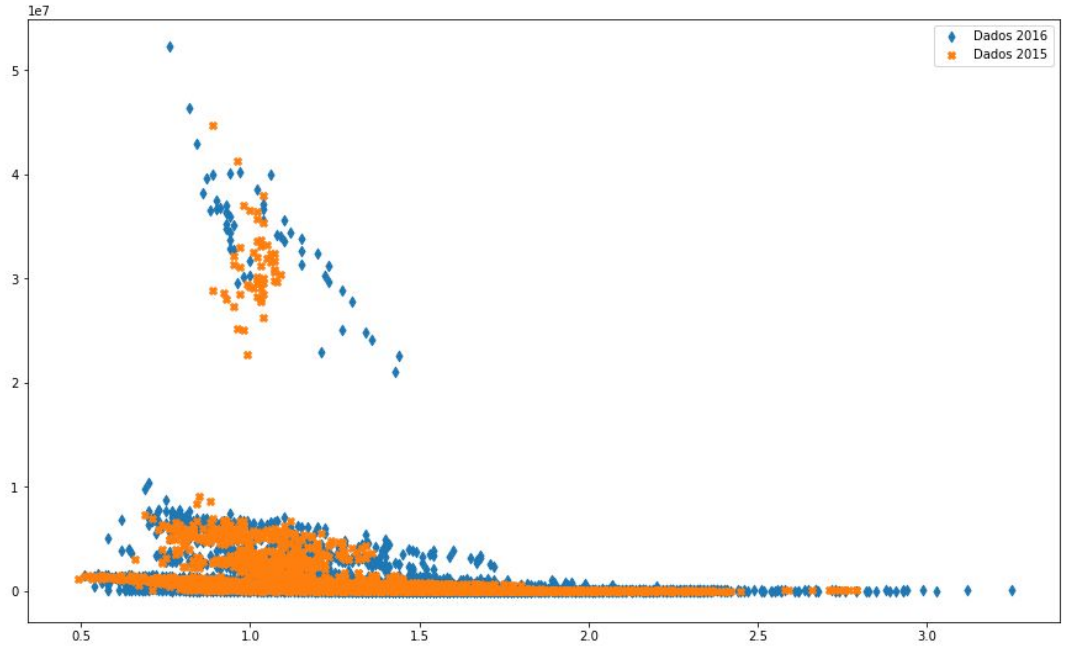
Place a legend on the axes.

Call signatures:

```
legend()
legend(labels)
legend(handles, labels)
```

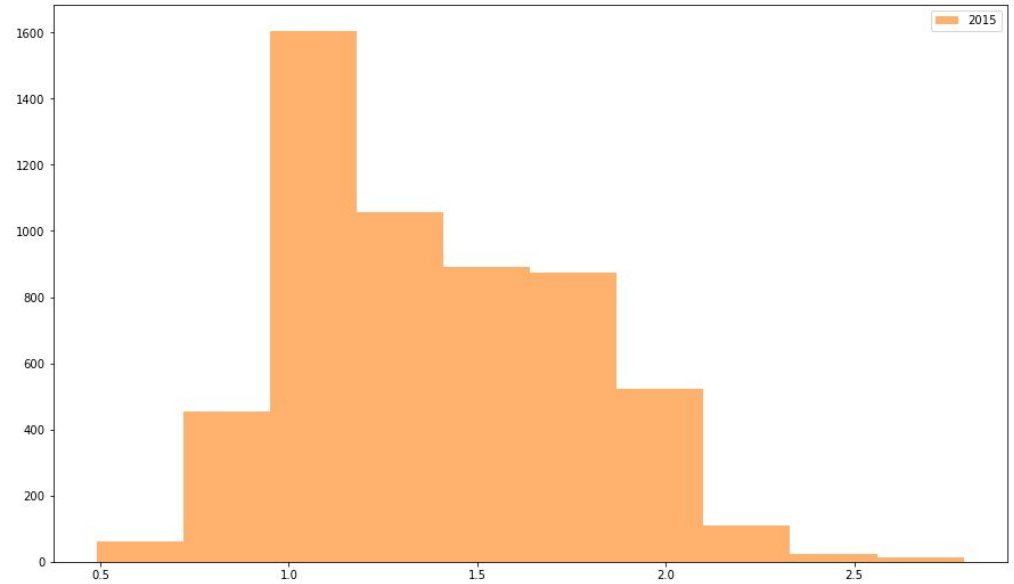
The call signatures correspond to three different ways how to use this method.

In [16]:
```
## adicionando legendas

plt.figure(figsize=(15,9))
plt.scatter(x = df[df.year == 2016].AveragePrice, y = df[df.year == 2016]['Total Volume'],
            c = 'tab:blue' , marker = 'd' , label='Dados 2016')
plt.scatter(x = df[df.year == 2015].AveragePrice, y = df[df.year == 2015]['Total Volume'],
            c = 'tab:orange' , marker = 'X', label='Dados 2015')

plt.legend()
plt.show()
```



Matplotlib

In [17]:
```
## criando um histograma

plt.figure(figsize=(15,9))
plt.hist(x = df[df['year'] == 2015]['AveragePrice'],color='tab:orange',alpha=0.6,label='2015')
plt.legend()
plt.show()
```



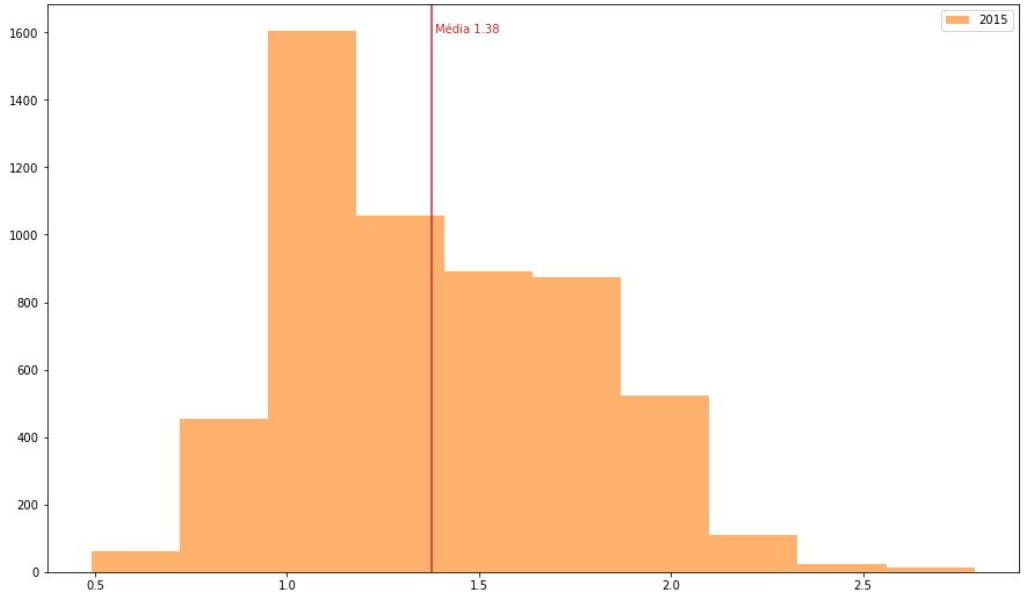
Matplotlib

matplotlib.pyplot.axvline

```
matplotlib.pyplot.axvline(x=0, ymin=0, ymax=1, **kwargs)
```

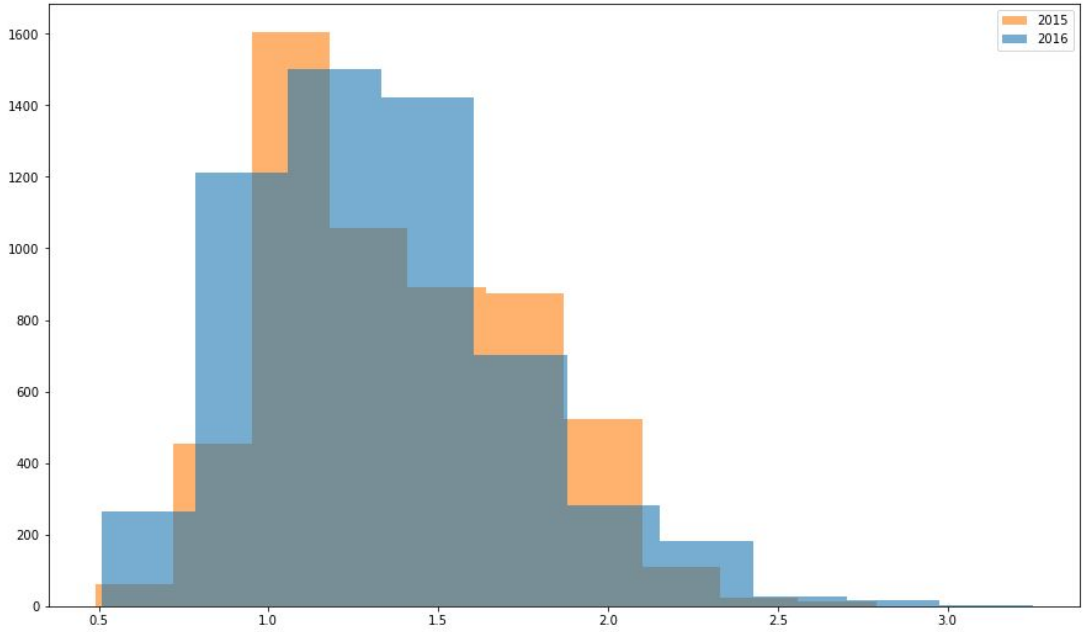
Matplotlib

In [18]:
```
## encontrado a media dos valores

media = df[df['year'] == 2015]['AveragePrice'].mean()

plt.figure(figsize=(15,9))
plt.hist(x = df[df['year'] == 2015]['AveragePrice'],color='tab:orange',alpha=0.6,label='2015')
plt.axvline(x = media ,color='tab:red')
plt.text(x =  media + 0.01, y = 1600, s = f'Média {round(media,2)}', fontdict = {'color':'tab:red'})

plt.legend()
plt.show()
```



Matplotlib

In [19]:
```
## criando mais de um histograma na mesma figura

plt.figure(figsize=(15,9))
plt.hist(x = df[df['year'] == 2015]['AveragePrice'],color='tab:orange',alpha=0.6,label='2015')
plt.hist(x = df[df['year'] == 2016]['AveragePrice'],color='tab:blue',alpha=0.6,label='2016')
plt.legend()
plt.show()
```



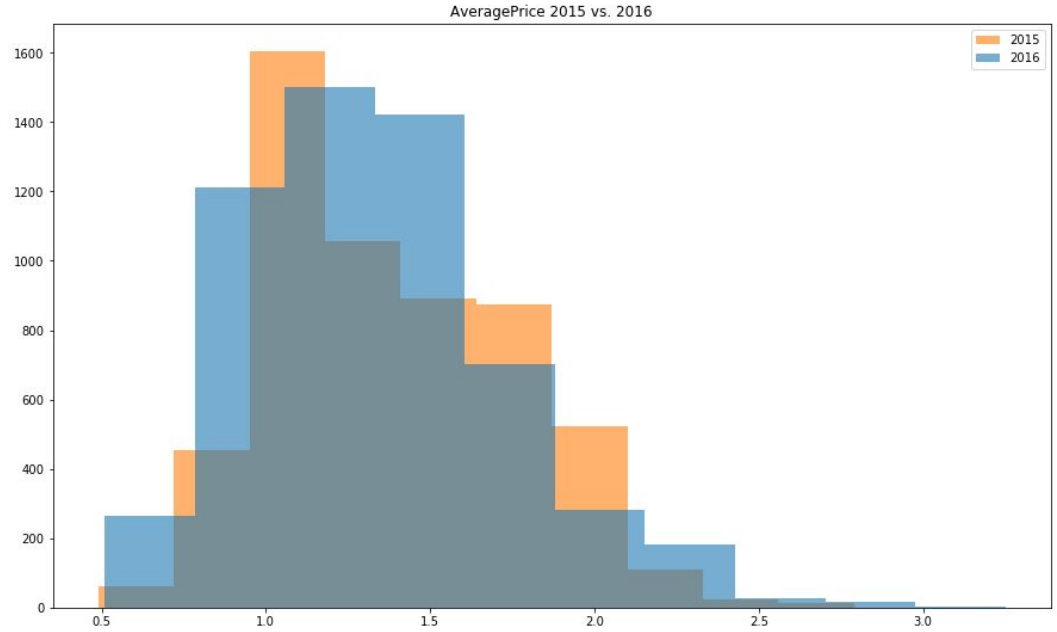
Matplotlib

matplotlib.pyplot.title

```
matplotlib.pyplot.title(label, fontdict=None, loc='center', pad=None, **kwargs)
```

Matplotlib

In [22]:
```
## gerando o gráfico acima com título

plt.figure(figsize=(15,9))
plt.hist(x = df[df['year'] == 2015]['AveragePrice'],color='tab:orange',alpha=0.6,label='2015')
plt.hist(x = df[df['year'] == 2016]['AveragePrice'],color='tab:blue',alpha=0.6,label='2016')
plt.title('AveragePrice 2015 vs. 2016',fontdict={'color':'black'})
plt.legend()
plt.show()
```



Matplotlib

matplotlib.pyplot.plot

```
matplotlib.pyplot.plot(*args, scalex=True, scaley=True, data=None, **kwargs)
```

Plot y versus x as lines and/or markers.

Parameters:

x, y : array-like or scalar

The horizontal / vertical coordinates of the data points. *x* values are optional and default to `range(len(y))`.

Commonly, these parameters are 1D arrays.

They can also be scalars, or two-dimensional (in that case, the columns represent separate data sets).

These arguments cannot be passed as keywords.

Matplotlib

In [23]:
```
## gerando os dados necessários para o próximo gráfico

dados = (


  df[['Date','Total Bags','year','region']]
  .query('year == 2015')
  .query("region == 'LasVegas'")
  .drop(['year','region'],axis=1)
  .assign(Date = lambda x: pd.to_datetime(x.Date,infer_datetime_format=True))
  .sort_values('Date')

)

print(dados.shape)
dados
```

```
(104, 2)
```

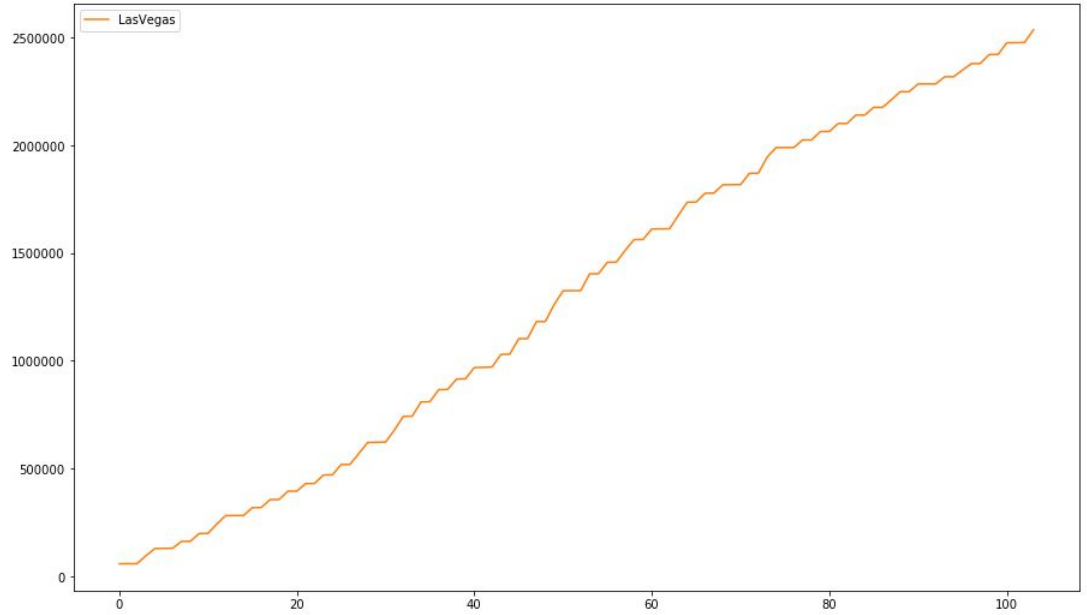
Out[23]:

	Date	Total Bags
1143	2015-01-04	58638.68
10269	2015-01-04	457.12
10268	2015-01-11	33.33
1142	2015-01-11	37144.36
1141	2015-01-18	33175.55
...	...	...
10220	2015-12-13	509.14
1093	2015-12-20	54103.94
10219	2015-12-20	407.88
10218	2015-12-27	1015.50
1092	2015-12-27	58612.10

104 rows × 2 columns

Matplotlib

In [24]:
```
## gerando o gráfico de linhas com a soma cumulativa

soma_cumulativa = dados['Total Bags'].cumsum()

plt.figure(figsize=(15,9))
plt.plot(range(104),soma_cumulativa,color='tab:orange',label='LasVegas')
plt.title('AveragePrice 2015 vs. 2016',fontdict={'color':'white'})
plt.legend()
plt.show()
```



Matplotlib

In [19]:
```
## definindo os dados de albany

dados_albany = (


  df[['Date','Total Bags','year','region']]
  .query('year == 2015')
  .query("region == 'Albany'")
  .drop(['year','region'],axis=1)
  .assign(Date = lambda x: pd.to_datetime(x.Date,infer_datetime_format=True))
  .sort_values('Date')


)

print(dados_albany.shape)
dados
```

```
(104, 2)
```

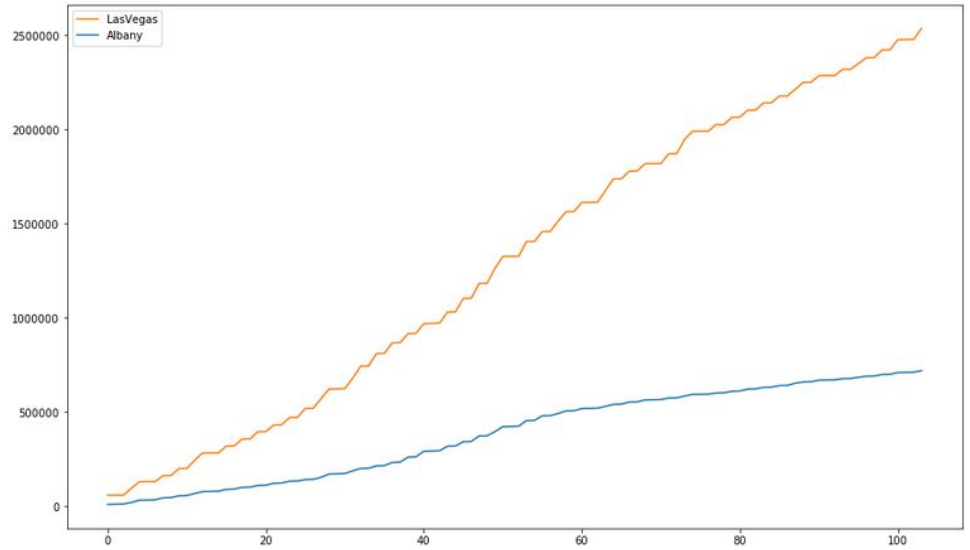
Out[19]:

	Date	Total Bags
1143	2015-01-04	58638.68
10269	2015-01-04	457.12
10268	2015-01-11	33.33
1142	2015-01-11	37144.36
1141	2015-01-18	33175.55
...	...	...
10220	2015-12-13	509.14
1093	2015-12-20	54103.94
10219	2015-12-20	407.88
10218	2015-12-27	1015.50
1092	2015-12-27	58612.10

104 rows × 2 columns

Matplotlib

In [20]:
```
## gerando o gráfico de linhas com a soma cumulativa

soma_cumulativa = dados['Total Bags'].cumsum()
soma_cumulativa_albany = dados_albany['Total Bags'].cumsum()

plt.figure(figsize=(15,9))
plt.plot(range(104),soma_cumulativa,color='tab:orange',label='LasVegas')
plt.plot(range(104),soma_cumulativa_albany,color='tab:blue',label='Albany')
plt.title('Total Bags LasVegas vs Albany (2015)',fontdict={'color':'white'})
plt.legend()
plt.show()
```



Matplotlib

matplotlib.pyplot.axhline

`matplotlib.pyplot.axhline(y=0, xmin=0, xmax=1, **kwargs)` [source]

Add a horizontal line across the axis.

Parameters:

y : scalar, optional, default: 0

y position in data coordinates of the horizontal line.

xmin : scalar, optional, default: 0

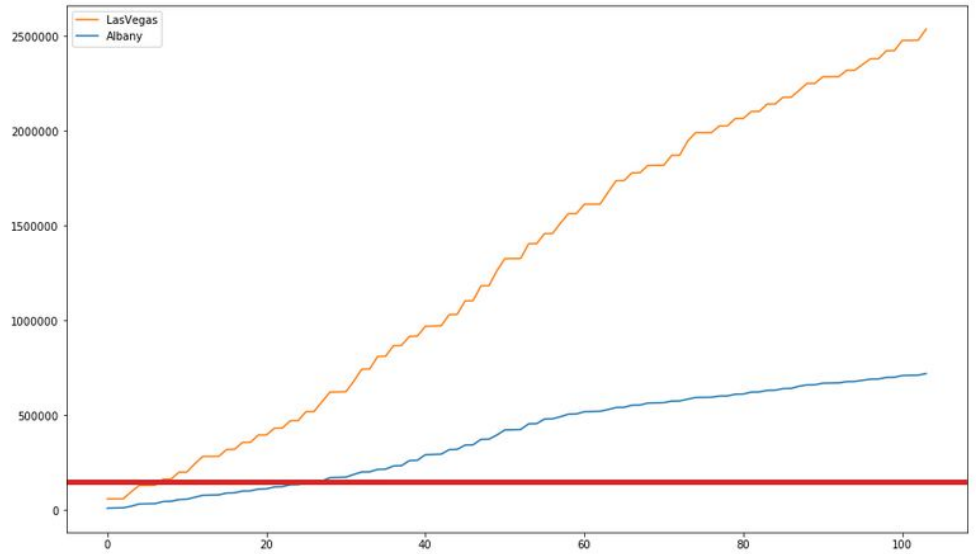
Should be between 0 and 1, 0 being the far left of the plot, 1 the far right of the plot.

xmax : scalar, optional, default: 1

Should be between 0 and 1, 0 being the far left of the plot, 1 the far right of the plot.

Matplotlib

In [23]:
```
## adicionando uma linha horizontal no valor 150000

soma_cumulativa = dados['Total Bags'].cumsum()
soma_cumulativa_albany = dados_albany['Total Bags'].cumsum()

plt.figure(figsize=(15,9))
plt.plot(range(104),soma_cumulativa,color='tab:orange',label='LasVegas')
plt.plot(range(104),soma_cumulativa_albany,color='tab:blue',label='Albany')
plt.title('Total Bags LasVegas vs Albany (2015)',fontdict={'color':'white'})

plt.axhline(y=150000,color='tab:red',lw=5)

plt.legend()
plt.show()
```



Matplotlib

matplotlib.pyplot.subplots

`matplotlib.pyplot.`**subplots**(*nrows=1*, *ncols=1*, *sharex=False*, *sharey=False*, *squeeze=True*, *subplot_kw=None*, *gridspec_kw=None*, ***fig_kw*) [source]

Create a figure and a set of subplots.

This utility wrapper makes it convenient to create common layouts of subplots, including the enclosing figure object, in a single call.

Returns:

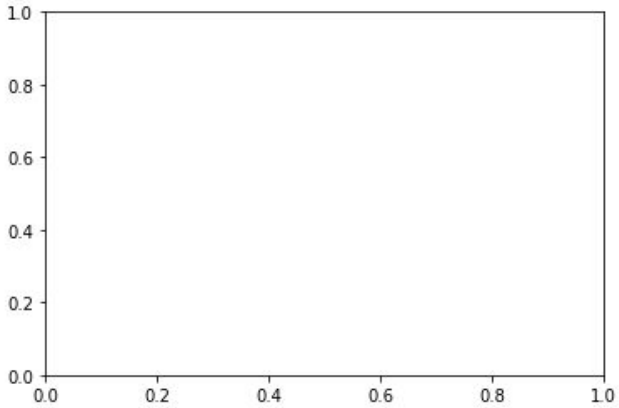
fig : `Figure`

ax : `axes.Axes` object or array of Axes objects.

ax can be either a single `Axes` object or an array of Axes objects if more than one subplot was created. The dimensions of the resulting array can be controlled with the squeeze keyword, see above.

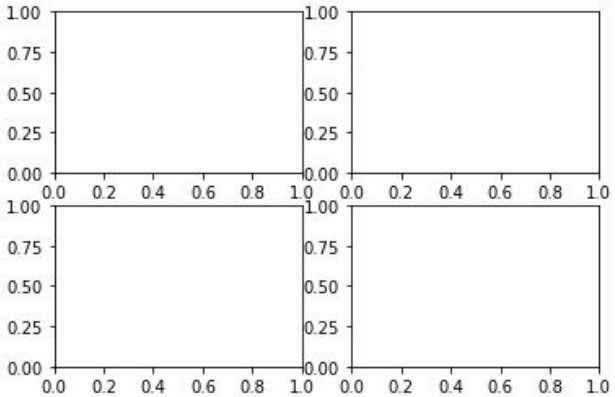
Matplotlib

In [24]:
```
## criando um subplot vazio

plt.subplots()
plt.show()
```



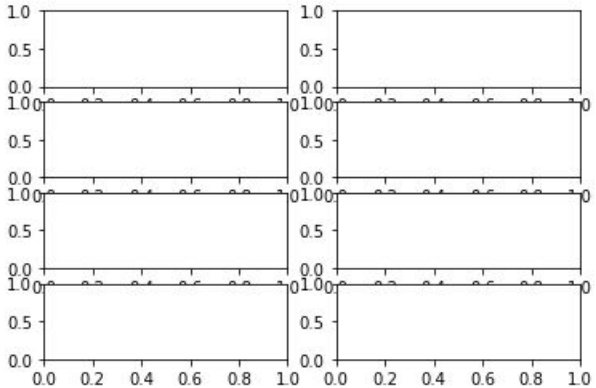
Matplotlib

In [25]:
```
## criando um subplot com 4 axes (2 linhas e 2 colunas)

plt.subplots(2,2)
plt.show()
```



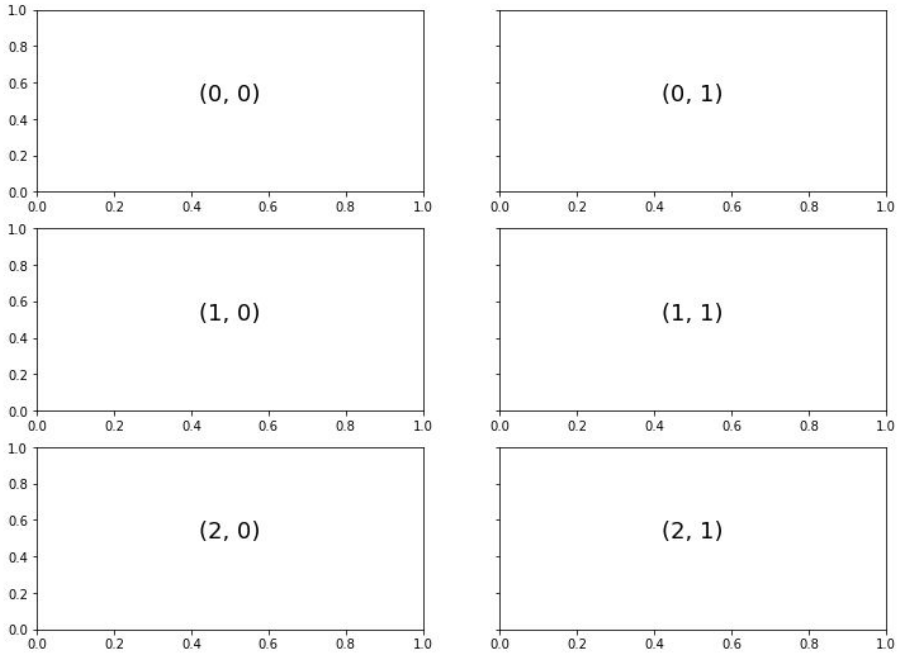
Matplotlib

In [26]:
```
## guardando os retornos em varíaveis que podemos acessar e controlar

f, ax = plt.subplots(4,2)
plt.show()
```



Matplotlib

In [28]:
```
## acessando os gráficos e adicionando os textos

f, ax = plt.subplots(3,2,figsize=(12,9),sharey=True)


for i in range(3):
  for j in range(2):
    ax[i,j].text(0.5, 0.5, str((i, j)),fontsize=18, ha='center')


plt.show()
```



Matplotlib

In [29]:
```
## definindo os dados que vamos usar
subplot_data = df[['AveragePrice','year']]
subplot_data
```

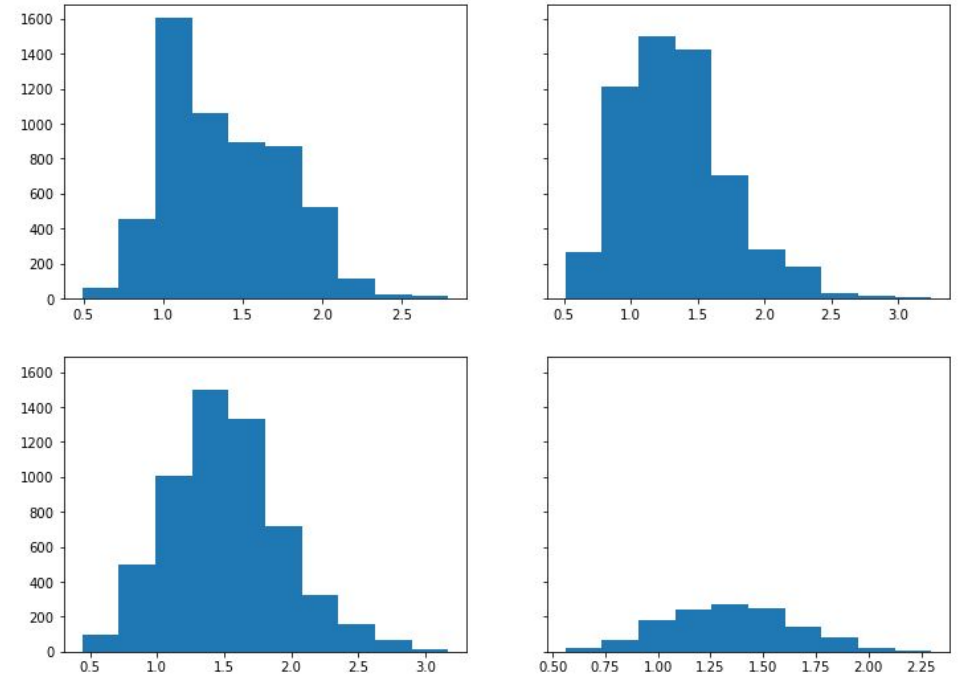
Out[29]:

	AveragePrice	year
0	1.33	2015
1	1.35	2015
2	0.93	2015
3	1.08	2015
4	1.28	2015
...	...	...
18244	1.63	2018
18245	1.71	2018
18246	1.87	2018
18247	1.93	2018
18248	1.62	2018

18249 rows × 2 columns

Matplotlib

In [30]:
```
## criando os gráficos em cada um dos eixos

f, ax = plt.subplots(2,2,figsize=(12,9),sharey=True)

ax[0,0].hist(subplot_data.query('year == 2015').AveragePrice)
ax[0,1].hist(subplot_data.query('year == 2016').AveragePrice)
ax[1,0].hist(subplot_data.query('year == 2017').AveragePrice)
ax[1,1].hist(subplot_data.query('year == 2018').AveragePrice)

plt.show()
```



Matplotlib

matplotlib
Version 3.3.0

Fork me on GitHub

Installation | Documentation | Examples | Tutorials | Contributing | Search

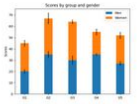
home | contents » modules | index

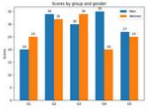
Gallery

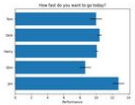
This gallery contains examples of the many things you can do with Matplotlib. Click on any image to see the full image and source code.

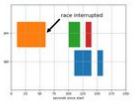
For longer tutorials, see our tutorials page. You can also find external resources and a FAQ in our user guide.

Lines, bars and markers

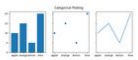

Stacked bar chart

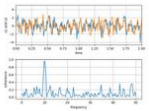

Grouped bar chart with labels

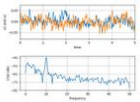

Horizontal bar chart

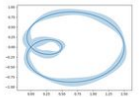


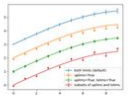
Broken Barh

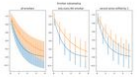

Plotting categorical variables

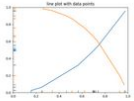

Plotting the coherence of two signals


CSD Demo


Curve with error band


Errorbar limit selection


Errorbar subsampling


EventCollection Demo

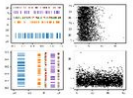

Eventplot Demo

Table of Contents

Related Topics

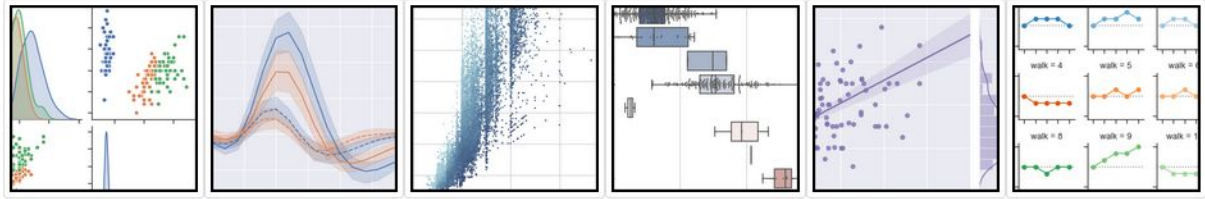
Documentation overview

Show Page Source

Seaborn

seaborn 0.10.1 Gallery Tutorial API Site Page Search

seaborn: statistical data visualization



Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package. You can browse the example gallery to see what you can do with seaborn, and then check out the tutorial and API reference to find out how.

To see the code or report a bug, please visit the github repository. General support issues are most at home on stackoverflow, where there is a seaborn tag.

Contents

- Introduction
- Release notes
- Installing
- Example gallery
- Tutorial
- API reference

Features

- Relational: API | Tutorial
- Categorical: API | Tutorial
- Distribution: API | Tutorial
- Regression: API | Tutorial
- Multiples: API | Tutorial
- Style: API | Tutorial
- Color: API | Tutorial

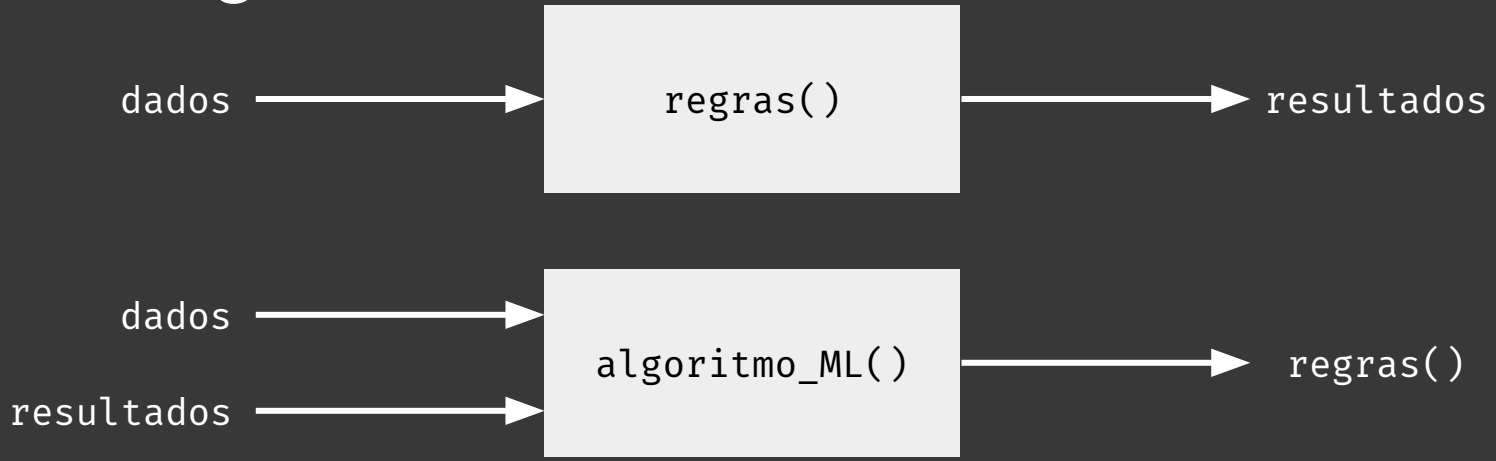
Back to top

Machine Learning

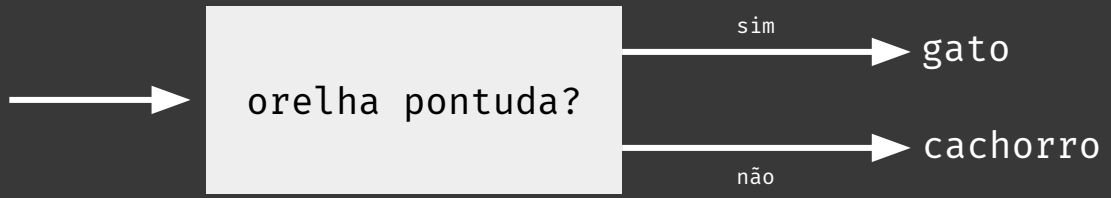
Machine Learning

“Machine learning is the science of getting computers to act without being explicitly programmed.”

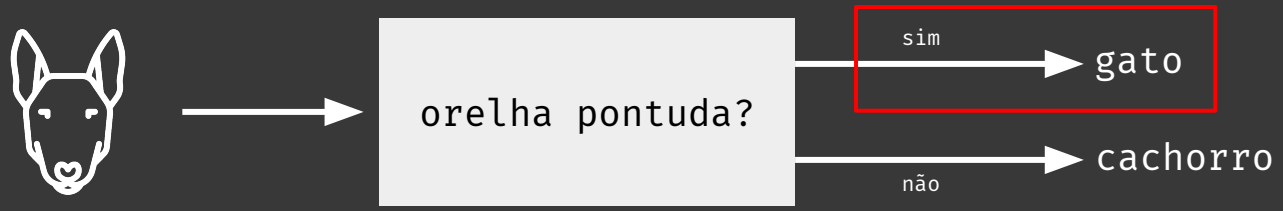
Machine Learning



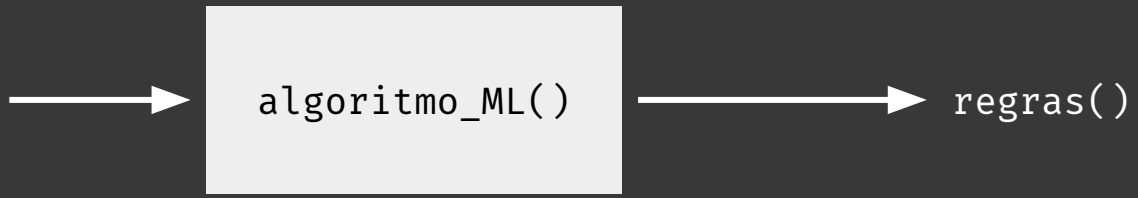
Machine Learning



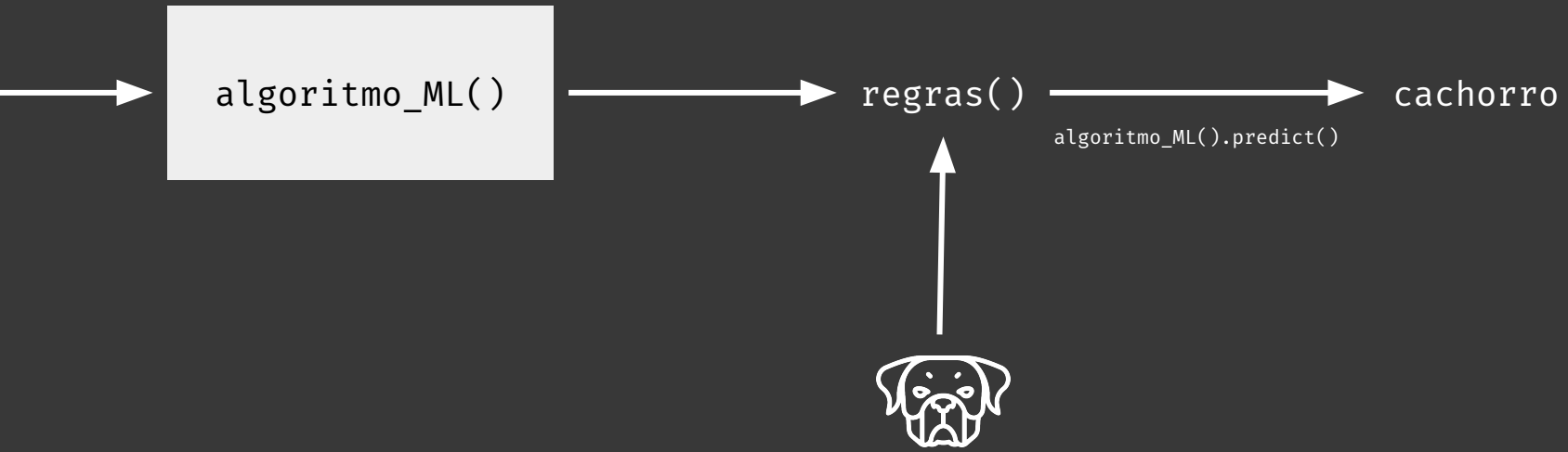
Machine Learning



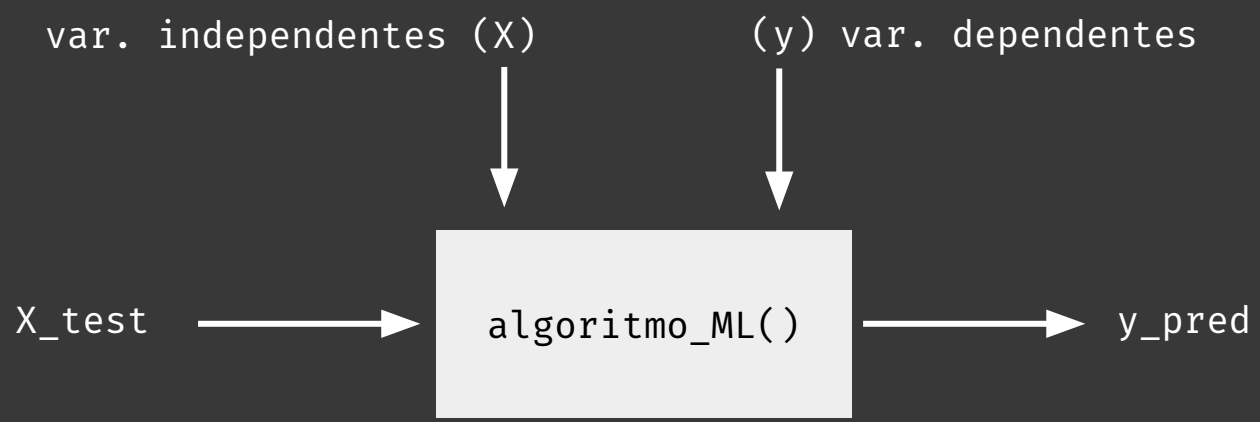
Machine Learning



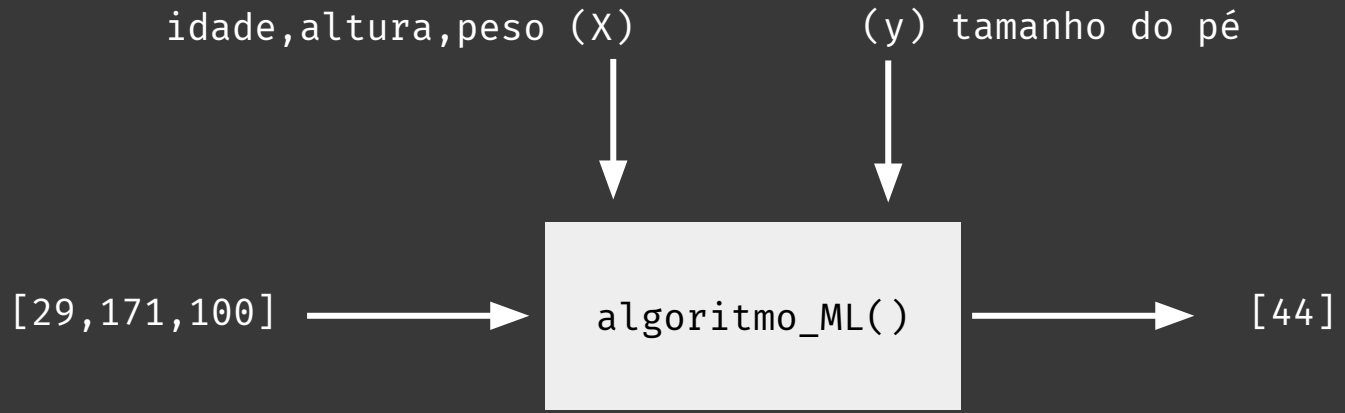
Machine Learning



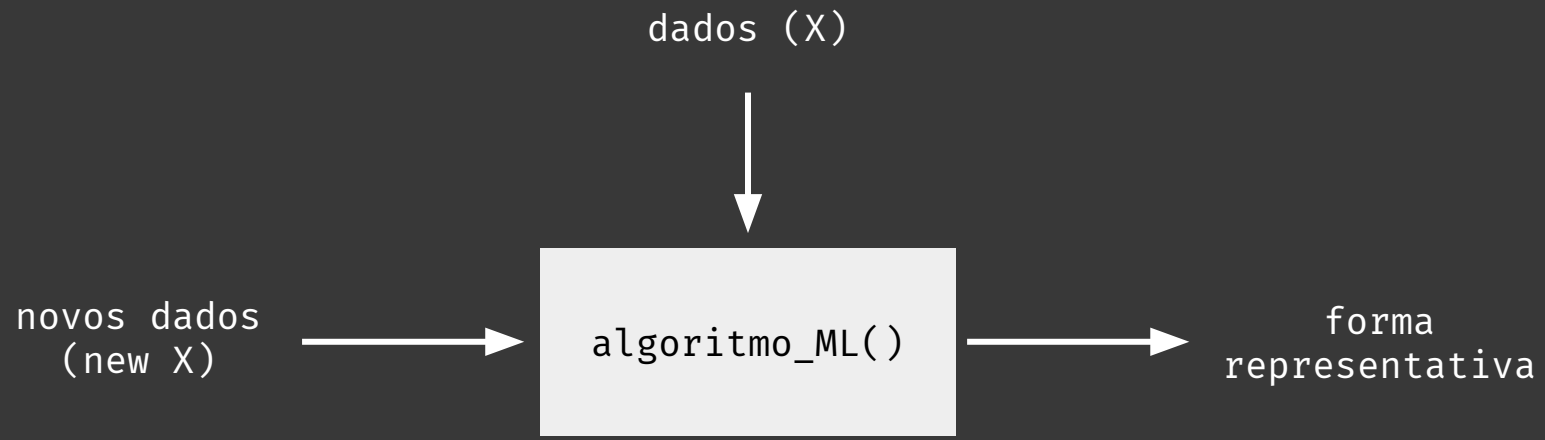
Aprendizagem Supervisionada



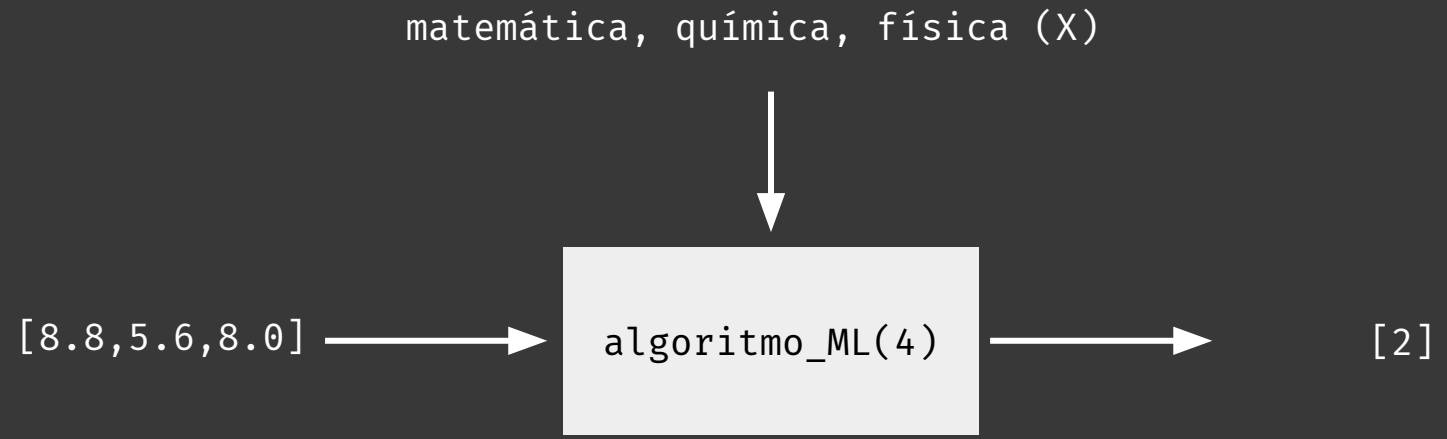
Aprendizagem Supervisionada



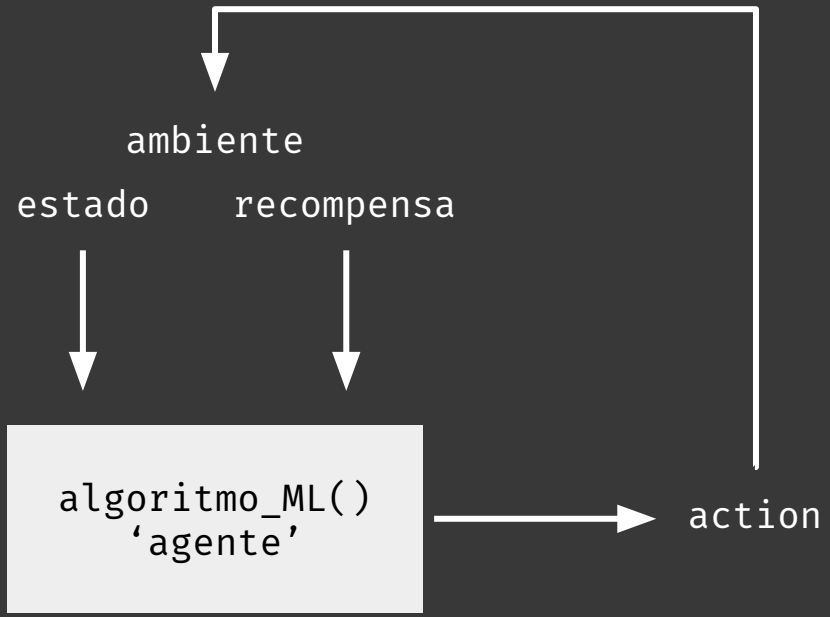
Aprendizagem Supervisionada



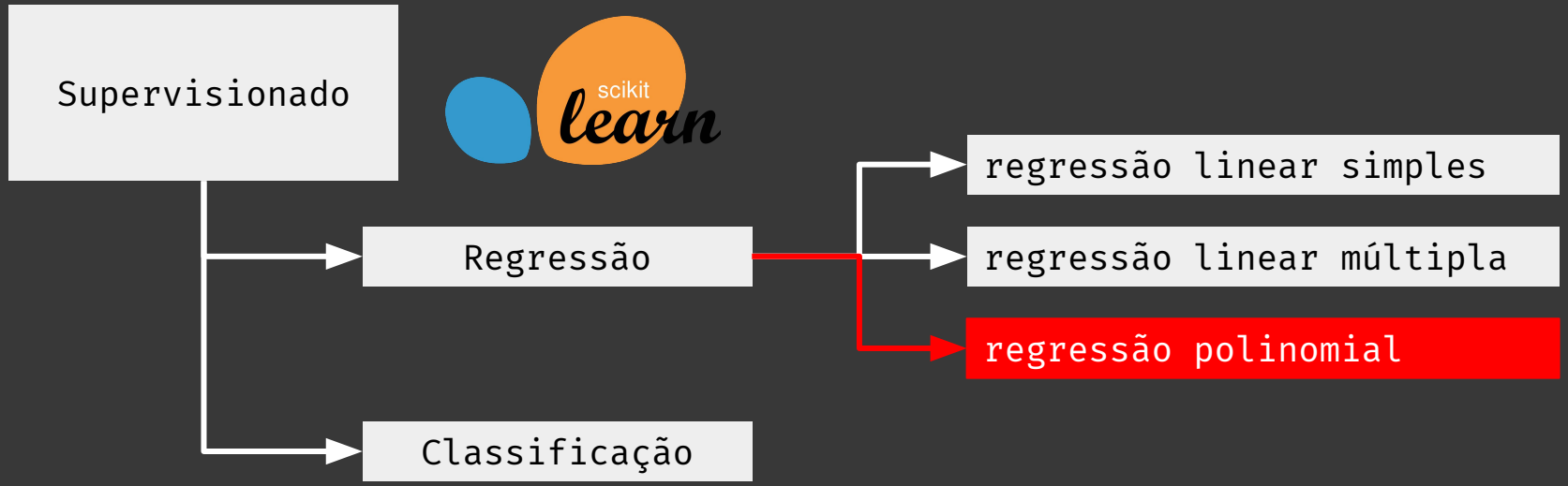
Aprendizagem Não-Supervisionada



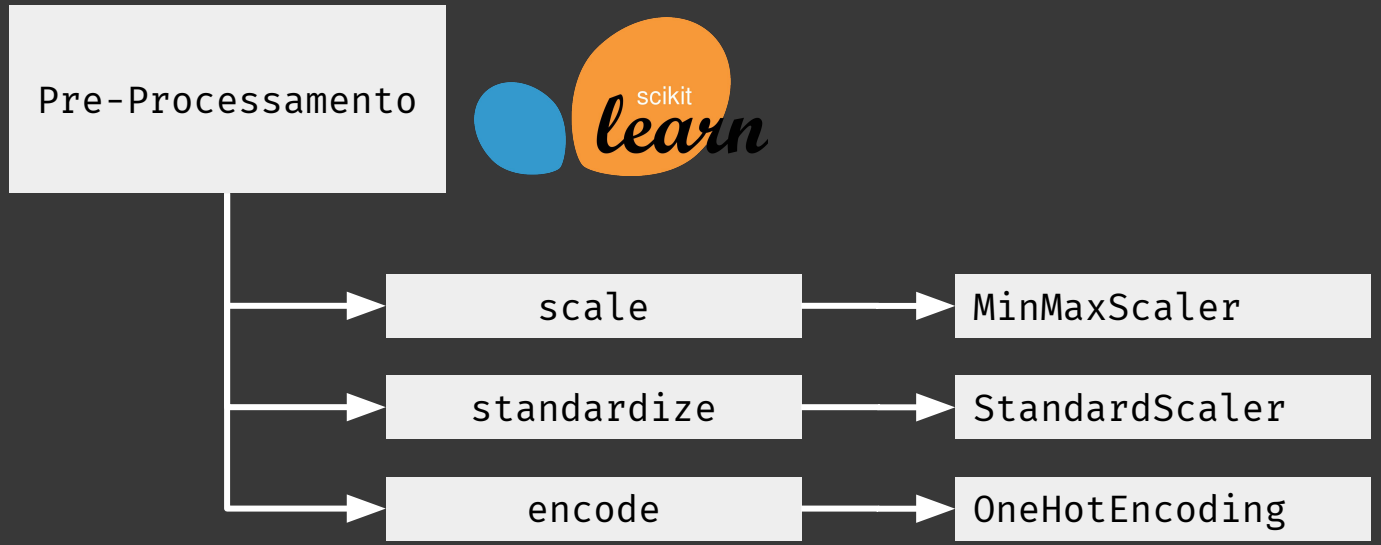
Aprendizagem por Reforço



Nosso escopo

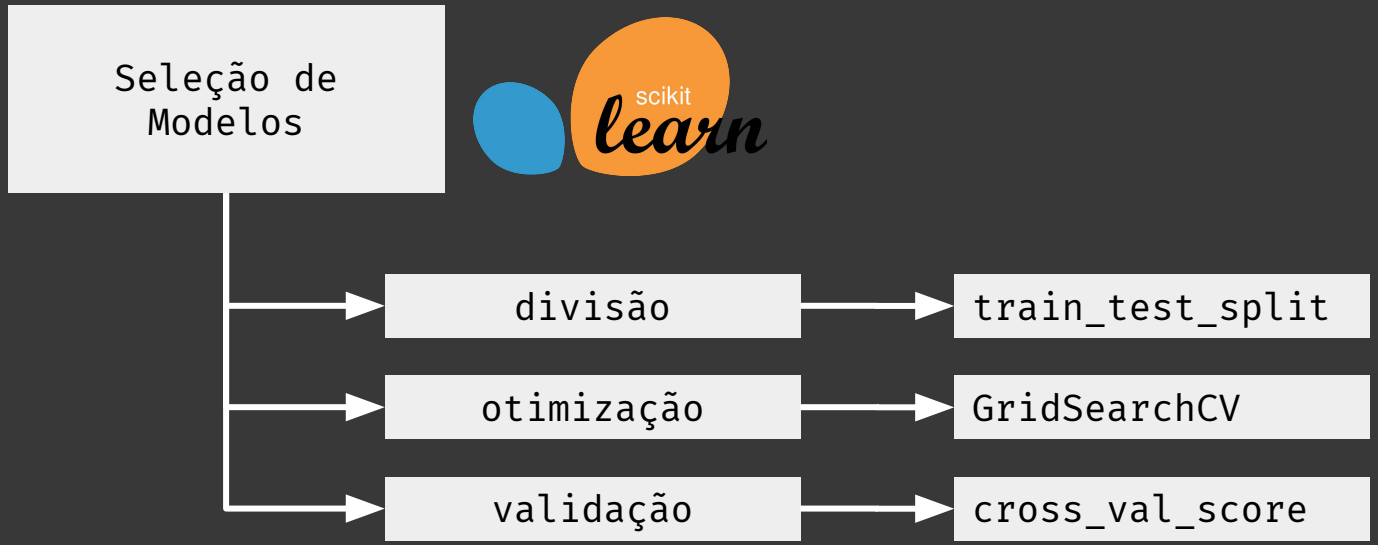


Nosso escopo



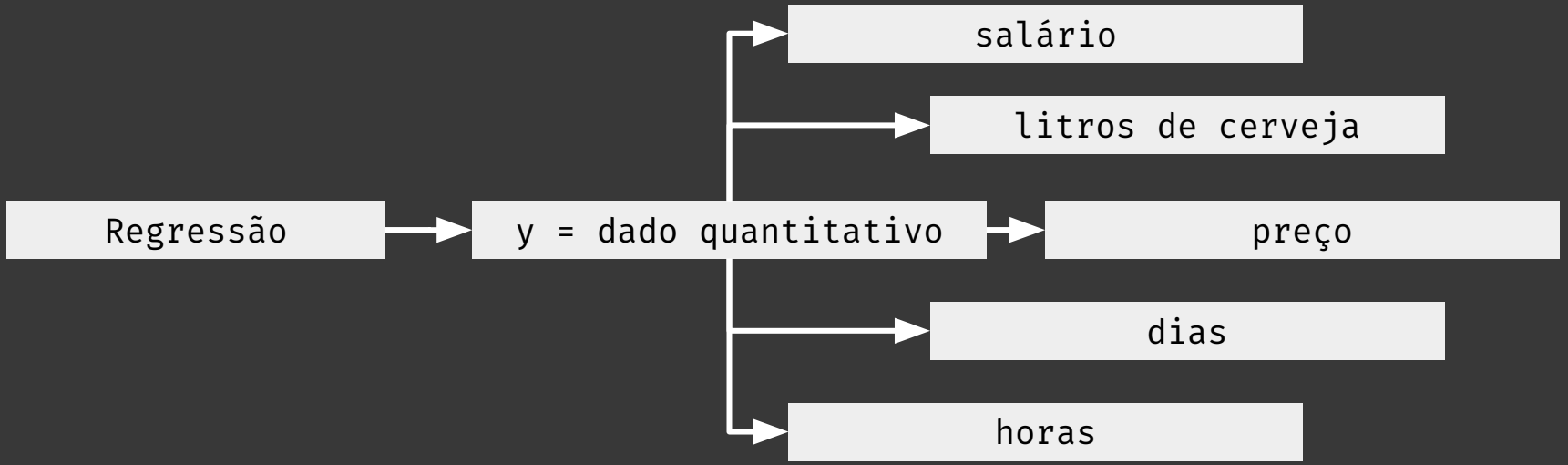
Machine Learning

Nosso escopo




Regressão!

Regressão


Regressão
y = dado quantitativo
salário
litros de cerveja
preço
dias
horas

Regressão

Posso descobrir o valor de Y com X?
Posso descobrir o tamanho do seu pé (Y) de acordo com a sua altura (X) ?
Por quanto vou vender essa casa?
Qual o limite devo dar para meu cliente?
Quanto de cerveja vai ser bebido hoje no bar?

Regressão

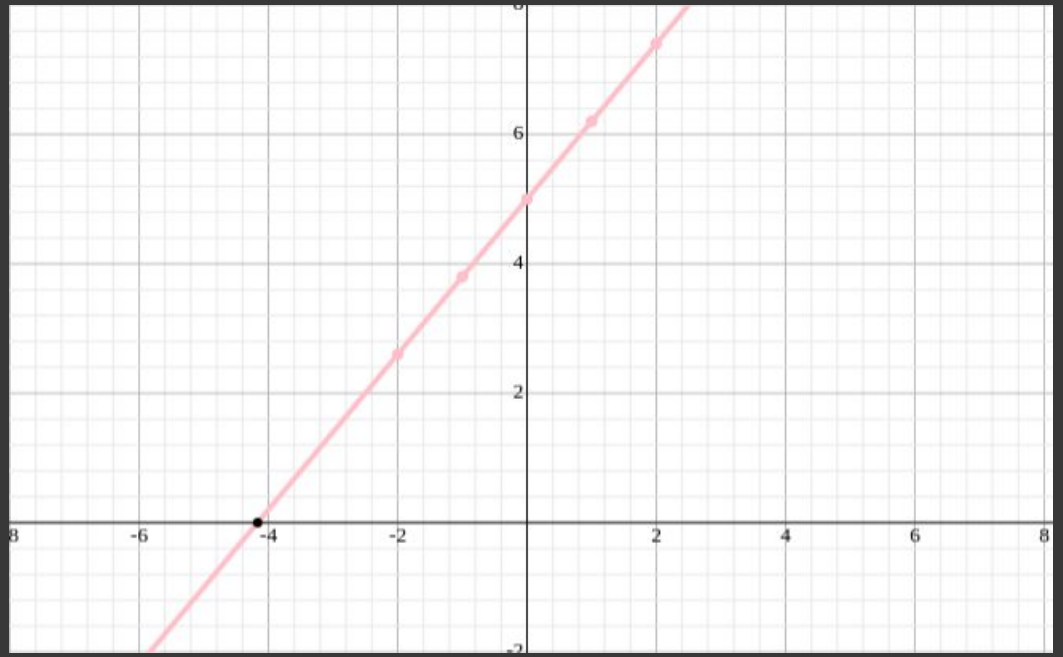
$$y(x) = ax + b$$

Regressão

$$y(x) = 5 + 1.2 * x$$

x	y
0	5
2	7.4
3	8.6
5	11
10	17

Regressão



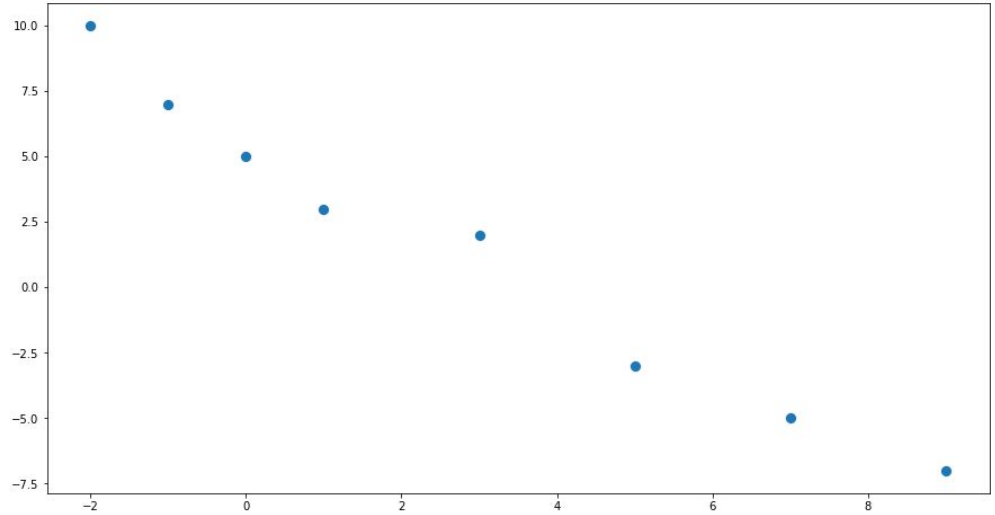
Regressão

$$y(x) = \; ?? + ?? * x$$

x	y
-2	10
-1	7
0	5
1	3
3	2
5	-3
7	-5
9	-7

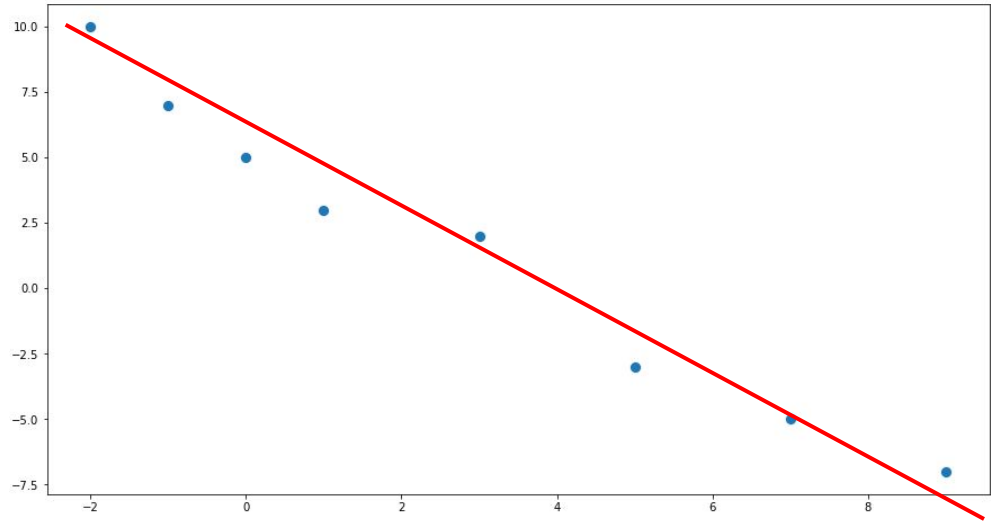
Regressão

In [158]:
```
## verificando os dados para regressão linear simples

x = [-2,-1,0,1,3,5,7,9]
y = [10,7,5,3,2,-3,-5,-7]

plt.figure(figsize=(15,8))
sns.scatterplot(x=x,y=y,s=100)
plt.show()
```



Regressão

In [158]:
```
## verificando os dados para regressão linear simples

x = [-2,-1,0,1,3,5,7,9]
y = [10,7,5,3,2,-3,-5,-7]

plt.figure(figsize=(15,8))
sns.scatterplot(x=x,y=y,s=100)
plt.show()
```



Regressão

resíduo = (y_real - y_pred)

cost_function = Σ(y_real - y_pred)^2

argmin (cost_function)

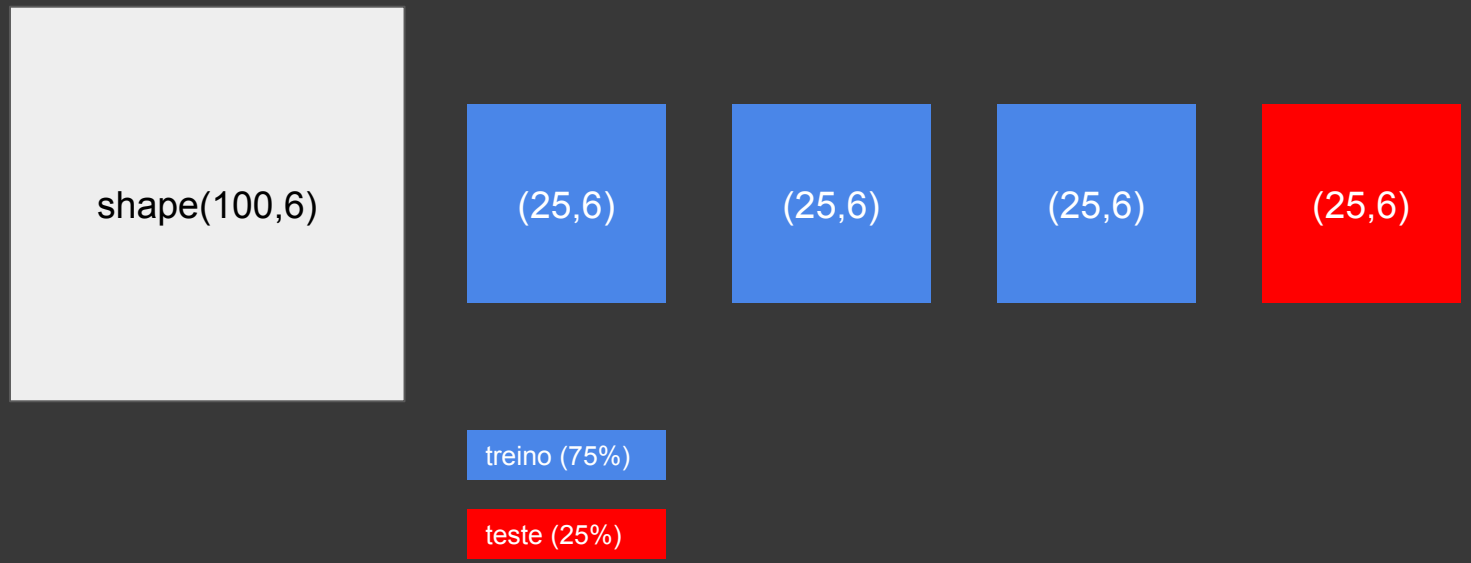
Regressão

https://docs.google.com/spreadsheets/d/1IpZ3PoIkk9HpBvalRvptXt-dFVmKVTLZSzG1xVmm6I/edit#gid=1146422701

Regressão

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \hat{y} \right)^2}_{\text{The square of the difference between actual and predicted}}$$

Treino e Teste



Machine Learning

`model.fit(X_train,y_train)`

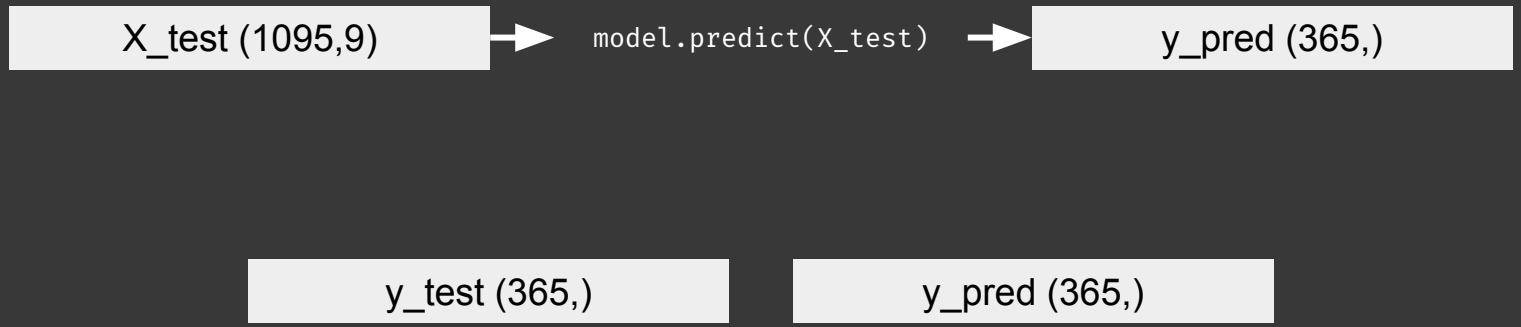
X_train (1095,9)

y_train (1095,)

`model.predict(X_test)`

X_test (365,9)

Machine Learning

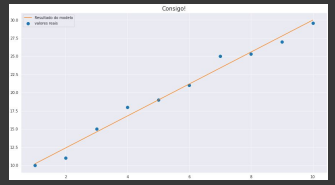


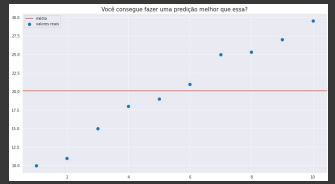
Regressão

HANDS-ON

R2

$$r2 = 1 - \left(\frac{SS_{reg}}{SS_{total}} \right)$$





R2

$$r2 = 1-\left(\frac{100}{101}\right) = ??$$

R2

$$r2 = 1-\left(\frac{250}{110}\right) = ??$$

Under e Overfitting