



Universidade Federal Fluminense



TÓPICOS ESP. SIST. INFORMAÇÃO



Prof.^aLeila Weitzel

framework

- Os frameworks de deep learning são bibliotecas de software que fornecem ferramentas e recursos para construir e treinar modelos de deep learning, tornando o processo mais eficiente e acessível.

framework

- **TensorFlow:**

- **Características:**

- Flexível: Pode ser usado para tarefas complexas, desde modelos de visão computacional até processamento de linguagem natural.
 - Escalabilidade: Suporta treinamento em GPUs e TPUs para acelerar o processo.
 - Comunidade ativa: Grande comunidade de usuários, com amplo suporte e recursos online.
 - Produção pronta: Ferramentas para implementar modelos em produção, como TensorFlow Serving.

- **Exemplos de uso:**

- Reconhecimento de imagens (Google Photos, carros autônomos)
 - Classificação de texto (filtragem de spam, tradução automática)
 - Previsão (previsão de séries temporais, análise de mercado)

framework

- PyTorch:

- **Características:**

- **Facilidade de uso:** Interface intuitiva e amigável para desenvolvedores.
 - **Dinamicidade:** Permite a definição de gráficos computacionais dinâmicos, tornando-o ideal para pesquisa.
 - **Comunidade vibrante:** Grande comunidade de desenvolvedores, com fóruns ativos e recursos de aprendizado.
 - **Popular na pesquisa:** Largamente utilizado na comunidade acadêmica devido à sua flexibilidade.

- **Exemplos de uso:**

- Processamento de linguagem natural (chatbots, modelos de linguagem)
 - Visão computacional (segmentação de imagens, detecção de objetos)
 - Redes neurais generativas (GANs)

framework

- Keras:

- Características:

- Simples e intuitivo: API de alto nível que simplifica o desenvolvimento de modelos.
 - Facilidade de uso: Fácil de aprender e implementar, ideal para iniciantes.
 - Modular: Permite a criação de modelos complexos combinando camadas e funções.
 - Suporte para vários backends: Pode ser usado com TensorFlow, Theano e CNTK.

- Exemplos de uso:

- Classificação de imagens (classificação de dígitos, reconhecimento facial)
 - Previsão de séries temporais (previsão de preços de ações)
 - Sistemas de recomendação (recomendação de produtos em lojas online)

framework

- Theano:

- Características:

- Eficiência computacional: Permite a otimização automática de código para GPUs.
 - Definido por grafo: Permite a definição de modelos através de grafos computacionais.
 - Flexibilidade: Permite a criação de modelos personalizados e inovadores.
 - Foco em pesquisa: Utilizado em pesquisas acadêmicas, embora não seja tão popular para desenvolvimento.

- Exemplos de uso:

- Modelos de visão computacional (reconhecimento de objetos, detecção de faces)
 - Modelos de processamento de linguagem natural (tradução automática, análise de sentimentos)
 - Redes neurais recorrentes (RNNs)

framework

- MXNet:

- Características:

- Desempenho e escalabilidade: Permite o treinamento em GPUs e CPUs, com escalabilidade para clusters.
 - Eficiência de memória: Utiliza técnicas de otimização para minimizar o uso de memória.
 - Suporte a múltiplas linguagens: Permite o desenvolvimento em Python, C++, R, Julia e JavaScript.
 - Comunidade ativa: Grande comunidade de usuários, com recursos e documentação abrangentes.

- Exemplos de uso:

- Visão computacional (reconhecimento de objetos, segmentação de imagens)
 - Processamento de linguagem natural (tradução automática, análise de sentimentos)
 - Sistemas de recomendação (recomendação de música e filmes)

framework

- CNTK:

- Características:

- Alto desempenho: Projetado para treinamentos de larga escala e execução eficiente.
 - Escalabilidade: Suporta treinamento em múltiplas GPUs e clusters.
 - Orientado a grafos: Permite a definição de modelos complexos através de grafos computacionais.
 - Linguagens de programação: Suporta C++, Python e C#.

- Exemplos de uso:

- Reconhecimento de fala (assistentes virtuais, tradução de voz)
 - Processamento de linguagem natural (classificação de texto, análise de sentimentos)
 - Modelos de previsão (previsão de demanda, análise de riscos)

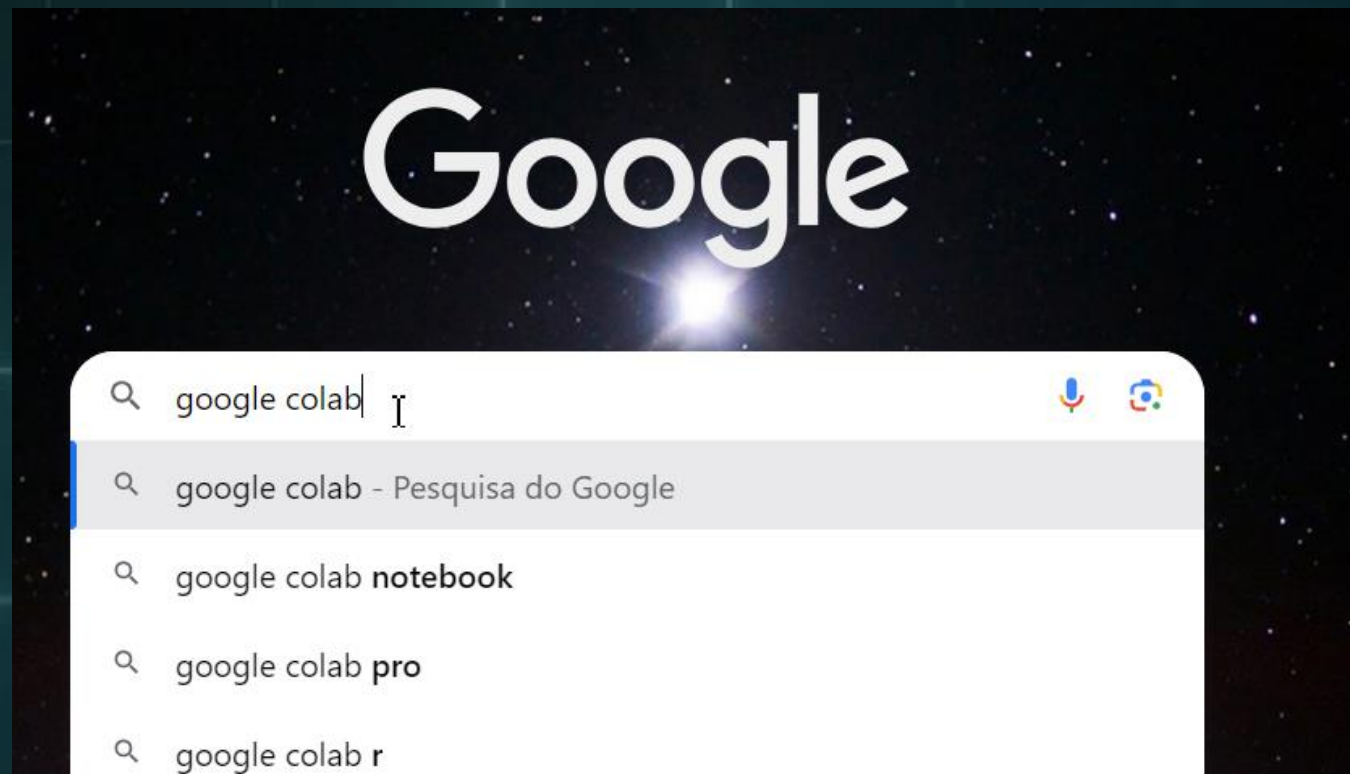
A escolha do framework depende da necessidade específica do projeto.

Considerar fatores como facilidade de uso, performance, escalabilidade e comunidade é crucial.


Cada framework oferece recursos únicos e vantagens distintas, permitindo a escolha ideal para o desenvolvimento de modelos de deep learning eficientes e inovadores.

Ambiente de programação

- Editor de Python
- Colab do Google, pesquisa no próprio Google search




Ambiente de programação


 google colab

Todas Vídeos Imagens Notícias Livros Mais Ferramentas

Aproximadamente 49.000.000 resultados (0,42 segundos)

 Google
<https://colab.research.google.com> · Traduzir esta página

Google Colab
Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your own Colab ...

 colab.google
<https://colab.google> · Traduzir esta página

Google colab
Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs.



Damos-lhe as boas-vindas ao Colaboratory

Ficheiro Editar Ver Inserir Tempo de execução Ferramentas Ajuda



Índice



Introdução



Ciência de dados



Aprendizagem automática



Mais recursos

Exemplos em destaque

+ Secção



Abrir bloco de notas

Exemplos >

Recente >

Google Drive >

GitHub >

Carregar >



Pesquisar blocos de notas



Título

Abertos
recentemente ↓

Primeiro acesso
↑↓



[neuralmind/bert-base-portuguese-cased-BE...](#)

11 de fevereiro

12/12/2023



[Cópia de neuralmind/bert-base-portuguese-...](#)

11 de fevereiro

18/12/2023



[Cópia de Bert versão Leila](#)

11 de fevereiro

12/09/2023



[teste-neuralmind/bert-base-portuguese-cas...](#)

6 de fevereiro

5 de janeiro



[hate-bert.ipynb](#)

6 de fevereiro

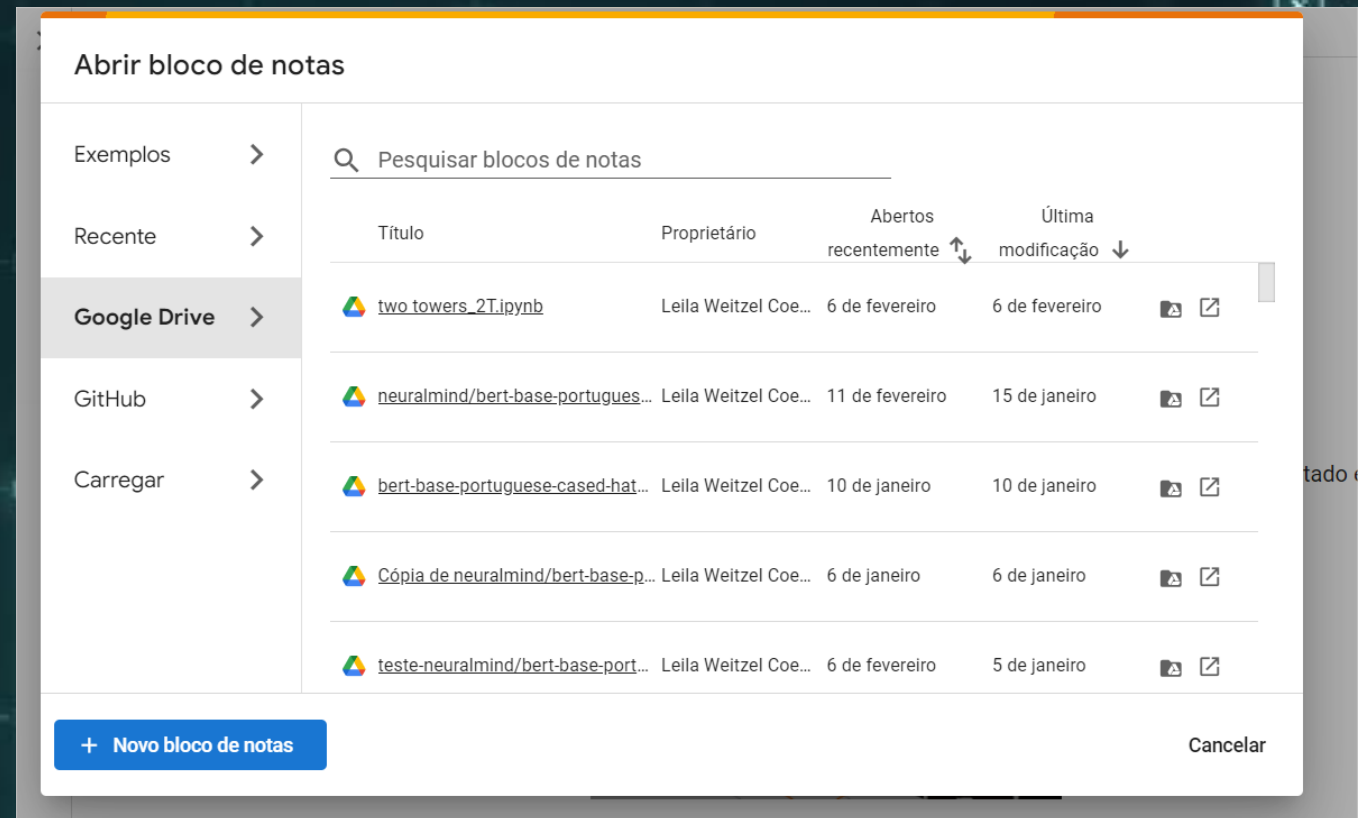
4 de janeiro



+ Novo bloco de notas

Cancelar

- Abrir seu Google Drive e abrir os exemplos que foram dados.
- Todos os Arquivos são disponibilizados no Classroom, no drive da disciplina
- Vocês devem fazer uma cópia para o drive de vocês dos exercícios e exemplos dados



Dúvidas ler tutorial

<https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar>



+ Código + Texto

0s



```
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
```

```
scipy: 1.10.1
numpy: 1.22.4
```

0s

[2]

```
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
```

```
matplotlib: 3.5.3
pandas: 1.3.5
```

- Executar tudo Ctrl+F9
- Executar antes Ctrl+F8
- Executar a célula em foco Ctrl+Enter
- Executar seleção Ctrl+Shift+Enter
- Executar após Ctrl+F10
- Interromper execução Ctrl+M I
- Reiniciar ambiente de execução Ctrl+M .
- Reiniciar e executar tudo
- Desconectar e excluir ambiente de execução
- Alterar o tipo de ambiente de execução
- Gerenciar sessões
- Ver registros do ambiente de execução

Ver

Inserir

Ambiente de execução

Ferramentas

Ajuda

Todas as alterações fora

Executar tudo

Ctrl+F9

Executar antes

Ctrl+F8

Executar a célula em foco

Ctrl+Enter

Executar seleção

Ctrl+Shift+Enter

Executar após

Ctrl+F10

Interromper execução

Ctrl+M I

Reiniciar ambiente de execução

Ctrl+M .

Reiniciar e executar tudo

Desconectar e excluir ambiente de execução

Alterar o tipo de ambiente de execução

Gerenciar sessões

Ver registros do ambiente de execução

ib

plotlib

plotlib: {}

das

das: {}'.format(pandas.__version__)

: 3.5.3

3.5

Configurações de notebook

Acelerador de hardware

None

None

GPU

TPU



la de código ao salvar este

Cancelar

Salvar



+ Código + Texto

✓
0s

```
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
```

```
scipy: 1.10.1
numpy: 1.22.4
```

✓
0s

```
[2] # matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
```

```
matplotlib: 3.5.3
pandas: 1.3.5
```




- Executar tudo Ctrl+F9
- Executar antes Ctrl+F8
- Executar a célula em foco Ctrl+Enter
- Executar seleção Ctrl+Shift+Enter
- Executar após Ctrl+F10
- Interromper execução Ctrl+M I
- Reiniciar ambiente de execução Ctrl+M .
- Reiniciar e executar tudo
- Desconectar e excluir ambiente de execução
- Alterar o tipo de ambiente de execução
- Gerenciar sessões
- Ver registros do ambiente de execução

Sessões ativas

Título	Última execução	RAM usada	
 bibliotecas.ipynb Sessão atual	há 0 minuto	0.89 GB	ENCERRAR

FECHAR





 bibliotecas.ipynb ☆

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda [Todas as alterações foram salvas](#)

Comentário


Compartilhar


Abrir configurações



+ Código + Texto

0s



```
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
```


scipy: 1.10.1
numpy: 1.22.4


[2]

```
# matplotlib
```

Comentário

Compartilhar


Abrir configurações



RAM

Disco

↑ ↓ ↻ 💬 ⚙ 📄 🗑 ⋮

Configurações

Site

Editor

Colab Pro

GitHub

Diversos

Tema

adaptive

- ☒ Exibir notificações na área de trabalho quando as execuções forem concluídas
- ☐ Os notebooks novos usam saídas particulares (omitem saídas ao salvar)

Layout padrão da página

horizontal

Custom snippet notebook URL

- ☐ Usar um notebook temporário como página de destino padrão.

Cancelar

Salvar

Configurações

Site

Editor


Colab Pro

GitHub

Diversos

Conta do GitHub

Sua conta do GitHub está conectada com sua conta do Google Drive.

[Revogar acesso em github.com](#) 

[Esquecer conta do GitHub](#)

☒ Acessar repositórios e organizações particulares
[Mais informações](#)

Cancelar

Salvar

Estruturas em Python

- Tupla: leitura apenas



```
a = (1, 2, 3)
print (a)
```

(1, 2, 3)

The image shows a code snippet from a Python IDE. It contains two lines of code: `a = (1, 2, 3)` and `print (a)`. The output of the code is `(1, 2, 3)`. The numbers 1, 2, and 3 in the code are highlighted in green. The output is displayed below the code in a separate line.

Estruturas em Python

- Lista: As listas usam a notação de colchete e podem ser indexadas usando a notação de matriz

✓
0s



```
mylist = [1, 2, 3]
print("Zeroth Value:", mylist[0])
mylist.append(4)
print("List Length:", len(mylist))
for value in mylist:
    print (value)
```

```
Zeroth Value: 1
List Length: 4
1
2
3
4
```

Estruturas em Python

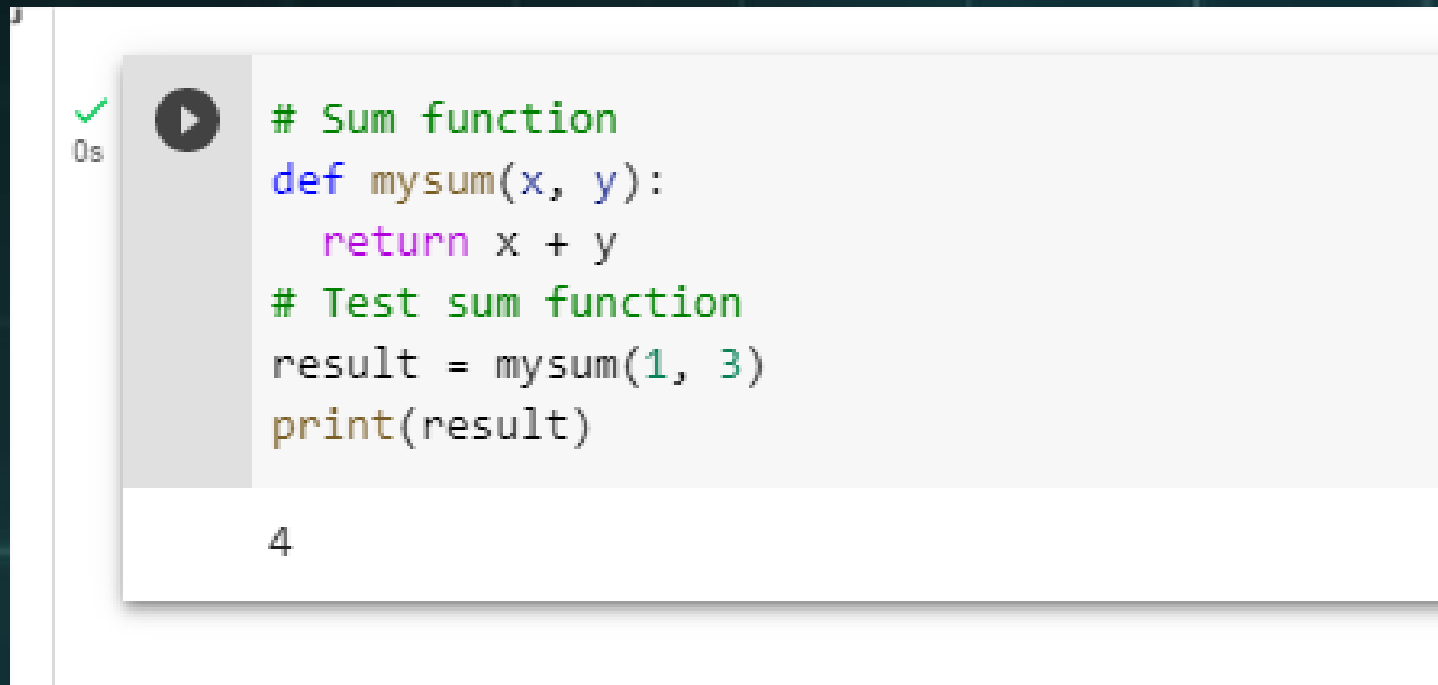
- Dicionário: Os dicionários são mapeamentos como pares de valores-chave.

```
▶ mydict = {'a': 1, 'b': 2, 'c': 3}
print("A value: ", mydict['a'])
mydict['a'] = 11
print("A value:", mydict['a'])
print("Keys:", mydict.keys())
print("Values:", mydict.values())
for key in mydict.keys():
    print (mydict[key])
```

```
A value: 1
A value: 11
Keys: dict_keys(['a', 'b', 'c'])
Values: dict_values([11, 2, 3])
11
2
3
```


Estruturas em Python

- função



A screenshot of a Python code execution window. On the left, there is a green checkmark and a play button icon. Below the play button, it says '0s'. The main area contains the following Python code:

```
# Sum function
def mysum(x, y):
    return x + y
# Test sum function
result = mysum(1, 3)
print(result)
```

Below the code, the output '4' is displayed.

bibliotecas

- Numpy

✓
0s



```
import numpy
mylist = [[1, 2, 3], [3, 4, 5]]
myarray = numpy.array(mylist)
print(myarray)
print(myarray.shape)
print("First row: ", myarray[0])
print("Last row: %s", myarray[-1])
print("Specific row and col: ", myarray[0, 2])
print("Whole col: %s", myarray[:, 2])
```

```
[[1 2 3]
 [3 4 5]]
(2, 3)
First row: [1 2 3]
Last row: %s [3 4 5]
Specific row and col: 3
Whole col: %s [3 5]
```

bibliotecas

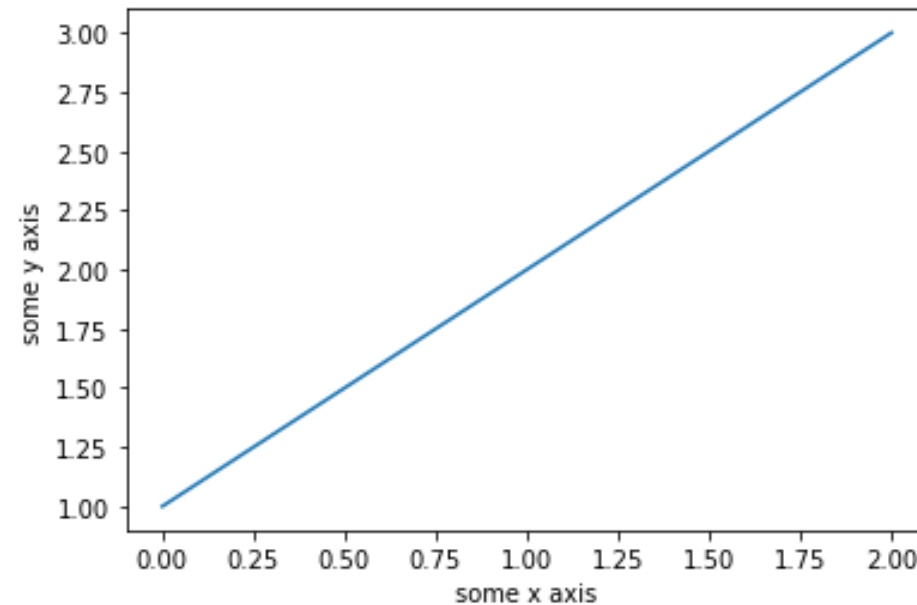
- Matplotlib

✓
1s



```
# basic line plot
import matplotlib.pyplot as plt
import numpy
myarray = numpy.array([1, 2, 3])
plt.plot(myarray)
plt.xlabel('some x axis')
plt.ylabel('some y axis')
plt.show()
```

As é um alias = apelido

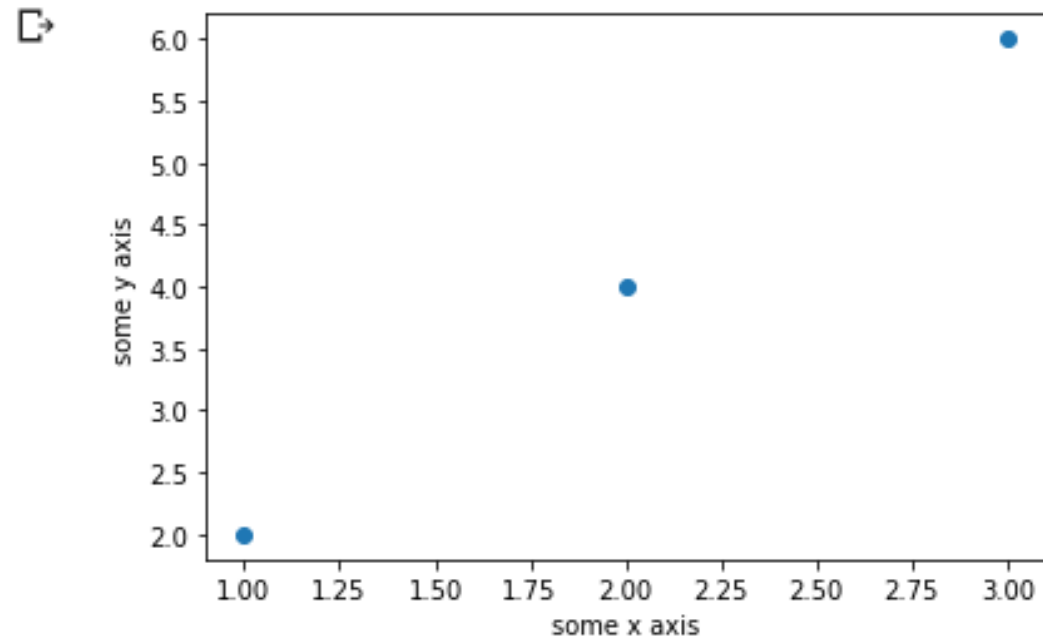


```
import numpy as np
```

bibliotecas

- Matplotlib
-

```
# basic scatter plot
import matplotlib.pyplot as plt
import numpy
x = numpy.array([1, 2, 3])
y = numpy.array([2, 4, 6])
plt.scatter(x,y)
plt.xlabel('some x axis')
plt.ylabel('some y axis')
plt.show()
```



bibliotecas

- Pandas
- Series
- unidimensional

✓
0s



```
# series
import numpy
import pandas
myarray = numpy.array([1, 2, 3])
rownames = ['a', 'b', 'c']
myseries = pandas.Series(myarray, index=rownames)
print(myseries)
```

```
a    1
b    2
c    3
dtype: int64
```


Estruturas em Python

- Pandas
- Dataframe multidimensional

```
# dataframe
import numpy
import pandas
myarray = numpy.array([[1, 2, 3], [4, 5, 6]])
rownames = ['a', 'b']
colnames = ['one', 'two', 'three']
mydataframe = pandas.DataFrame(myarray, index=rownames, columns=colnames)
print(mydataframe)
```

	one	two	three
a	1	2	3
b	4	5	6

Ler arquivos

- Os arquivos estão sempre no formato **csv** (arquivo texto onde as colunas são separados por virgula)
- Atenção se abrir no Excel este arquivo , automaticamente é trocado a virgula por ponto e virgula... Então não abrir....

Ler arquivos

- 1. Carregar arquivos CSV com a biblioteca padrão Python.
- 2. Carregar arquivos CSV com o NumPy.
- 3. Carregar arquivos CSV com **Pandas**.

Ler arquivos

- Os comentários em um arquivo CSV são indicados por um hash (#) no início de uma linha.
- Se você tiver comentários em seu arquivo, dependendo do método utilizado para carregar seu dados, você pode precisar indicar se deve ou não esperar comentários e o caráter a esperar para significar uma linha de comentário.

Ler arquivos

- **Duas formas de leitura de dados:**
 - Por meio de leitura do seu próprio dataset.
 - Por meio de **load** pois tem algumas bibliotecas que já têm dados pré-coletados


```
✓ [28] import pandas as pd
0s
df = pd.read_csv('/content/arquivo1.csv')
```

Lendo um arquivo de upload

```
✓ [29] df.shape
0s
(1795, 7)
```

```
✓ [30] df.head()
0s
```

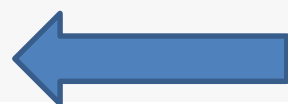
	id	my_datetime	event	country	user_id	source	topic
0	0	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
1	1	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
2	2	2018-01-01 00:04:01	read	country_7	2458151263	AdWords	Africa
3	3	2018-01-01 00:04:02	read	country_7	2458151264	AdWords	Europe
4	4	2018-01-01 00:05:03	read	country_8	2458151265	Reddit	North America

```
✓ 4s [24] import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing

data = fetch_california_housing(as_frame=True)
```

Lendo um arquivo por
load do sklearn

```
✓ 0s ▶ data.data
```



	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
--	--------	----------	----------	-----------	------------	----------	----------	-----------



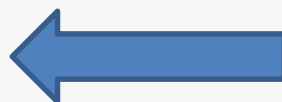
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24



0s



data.target



0 4.526

1 3.585

2 3.521

3 3.413

4 3.422

...

20635 0.781

20636 0.771

20637 0.923

20638 0.847

20639 0.894

Name: MedHouseVal, Length: 20640, dtype: float64

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets>

Loaders

<code>datasets.clear_data_home([data_home])</code>	Delete all the content of the data home cache.
<code>datasets.dump_svmlight_file(X, y, f, *[, ...])</code>	Dump the dataset in svmlight / libsvm file format.
<code>datasets.fetch_20newsgroups(*[, data_home, ...])</code>	Load the filenames and data from the 20 newsgroups dataset (classification).
<code>datasets.fetch_20newsgroups_vectorized(*[, ...])</code>	Load and vectorize the 20 newsgroups dataset (classification).
<code>datasets.fetch_california_housing(*[, ...])</code>	Load the California housing dataset (regression).
<code>datasets.fetch_covtype(*[, data_home, ...])</code>	Load the covtype dataset (classification).
<code>datasets.fetch_kddcup99(*[, subset, ...])</code>	Load the kddcup99 dataset (classification).
<code>datasets.fetch_lfw_pairs(*[, subset, ...])</code>	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
<code>datasets.fetch_lfw_people(*[, data_home, ...])</code>	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
<code>datasets.fetch_olivetti_faces(*[, ...])</code>	Load the Olivetti faces data-set from AT&T (classification).
<code>datasets.fetch_openml([name, version, ...])</code>	Fetch dataset from openml by name or dataset id.
<code>datasets.fetch_rcv1(*[, data_home, subset, ...])</code>	Load the RCV1 multilabel dataset (classification).
<code>datasets.fetch_species_distributions(*[, ...])</code>	Loader for species distribution dataset from Phillips et.
<code>datasets.get_data_home([data_home])</code>	Return the path of the scikit-learn data directory.
<code>datasets.load_breast_cancer(*[, return_X_y, ...])</code>	Load and return the breast cancer wisconsin dataset (classification).
<code>datasets.load_diabetes(*[, return_X_y, ...])</code>	Load and return the diabetes dataset (regression).
<code>datasets.load_digits(*[, n_class, ...])</code>	Load and return the digits dataset (classification).
<code>datasets.load_files(container_path, *[, ...])</code>	Load text files with categories as subfolder names.
<code>datasets.load_iris(*[, return_X_y, as_frame])</code>	Load and return the iris dataset (classification).
<code>datasets.load_linnerud(*[, return_X_y, as_frame])</code>	Load and return the physical exercise Linnerud dataset.
<code>datasets.load_sample_image(image_name)</code>	Load the numpy array of a single sample image.

Lendo arquivo de url

```
import pandas as pd
import io
import requests
c = pd.read_csv("https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv")

print(c)
```

	Country	Region
0	Algeria	AFRICA
1	Angola	AFRICA
2	Benin	AFRICA
3	Botswana	AFRICA
4	Burkina	AFRICA
..
189	Paraguay	SOUTH AMERICA
190	Peru	SOUTH AMERICA
191	Suriname	SOUTH AMERICA
192	Uruguay	SOUTH AMERICA
193	Venezuela	SOUTH AMERICA

[194 rows x 2 columns]

Lendo outros formatos

- Ler json
- Ler arff
- Ler excel
- Tutorial no link
 - <https://practicaldatascience.co.uk/data-science/how-to-import-data-into-pandas-dataframes>

Gerar um base de dados

Samples generator

<code>datasets.make_biclusters(shape, n_clusters, *)</code>	Generate a constant block diagonal structure array for biclustering.
<code>datasets.make_blobs([n_samples, n_features, ...])</code>	Generate isotropic Gaussian blobs for clustering.
<code>datasets.make_checkerboard(shape, n_clusters, *)</code>	Generate an array with block checkerboard structure for biclustering.
<code>datasets.make_circles([n_samples, shuffle, ...])</code>	Make a large circle containing a smaller circle in 2d.
<code>datasets.make_classification([n_samples, ...])</code>	Generate a random n-class classification problem.
<code>datasets.make_friedman1([n_samples, ...])</code>	Generate the "Friedman #1" regression problem.
<code>datasets.make_friedman2([n_samples, noise, ...])</code>	Generate the "Friedman #2" regression problem.
<code>datasets.make_friedman3([n_samples, noise, ...])</code>	Generate the "Friedman #3" regression problem.
<code>datasets.make_gaussian_quantiles(*[, mean, ...])</code>	Generate isotropic Gaussian and label samples by quantile.
<code>datasets.make_hastie_10_2([n_samples, ...])</code>	Generate data for binary classification used in Hastie et al. 2009, Example 10.2
<code>datasets.make_low_rank_matrix([n_samples, ...])</code>	Generate a mostly low rank matrix with bell-shaped singular values.
<code>datasets.make_moons([n_samples, shuffle, ...])</code>	Make two interleaving half circles.
<code>datasets.make_multilabel_classification([...])</code>	Generate a random multilabel classification problem.
<code>datasets.make_regression([n_samples, ...])</code>	Generate a random regression problem.
<code>datasets.make_s_curve([n_samples, noise, ...])</code>	Generate an S curve dataset.
<code>datasets.make_sparse_coded_signal(n_samples, ...)</code>	Generate a signal as a sparse combination of dictionary elements.
<code>datasets.make_sparse_spd_matrix([dim, ...])</code>	Generate a sparse symmetric definite positive matrix.
<code>datasets.make_sparse_uncorrelated([...])</code>	Generate a random regression problem with sparse uncorrelated design.
<code>datasets.make_spd_matrix(n_dim, *[, ...])</code>	Generate a random symmetric, positive-definite matrix.
<code>datasets.make_swiss_roll([n_samples, noise, ...])</code>	Generate a swiss roll dataset.

Gerar um base de dados

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=10, centers=3, n_features=2,
                  random_state=0)
print(X.shape)

X, y = make_blobs(n_samples=[3, 3, 4], centers=None, n_features=2,
                  random_state=0)
print(X.shape)
```

(10, 2)

(10, 2)

Gerar um base de dados

```
import pandas as pd
from sklearn.datasets import make_classification

# Gerar um conjunto de dados de classificação
X, y = make_classification(n_samples=1000, n_features=20, n_informative=2,
n_redundant=2, n_classes=2, random_state=42)

# Converter para DataFrame
df = pd.DataFrame(X, columns=[f'feature_{i}' for i in range(X.shape[1])])
df['target'] = y

# Salvar em um arquivo CSV
df.to_csv('classification_data.csv', index=False)

print("Arquivo 'classification_data.csv' criado com sucesso!")
```

Lendo um arquivo csv

- Lendo um arquivo csv que não contem os nomes da coluna.



```
import pandas as pd

names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

df= pd.read_csv( "pima-indians-diabetes.csv", names=names)

df.head(3)
```

	preg	plas	pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1



Machine Learning

- **Modelos preditivos apresentam 6 fases**

1. **Definir o problema:** a fim de compreender melhor os objetivos do projeto.
2. **Analisar os dados:** Utilizar estatísticas descritivas e visualização para melhor compreender os dados
3. **Preparar os dados:** Use transformadores de dados (por exemplo normalização) para melhor expor a estrutura do problema de predição para algoritmos de modelagem.
4. **Avaliar Algoritmos:** Projetar um bateria de teste para avaliar uma série de algoritmos padrão
5. **Melhore os resultados:** Use métodos de ajuste para obter o máximo de performance dos algoritmos em seus dados
6. **Apresente os Resultados:** Finalize o modelo, faça previsões e apresente os resultados

Machine Learning

- Passos do treinamento:

1. Criar o modelo
2. Compilar o modelo : decidir quais hiperparametros usar
3. Treinar o modelo (função fit()) dividir a base em treinamento (70%) e teste (30%).
4. Fazer previsões y_pred().
5. Avaliar o modelo

