Fundamentos de Python (UFCD 10809)
PPT02

Sandra Liliana Meira de Oliveira

PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU

O que é a visualização de Dados?

representação gráfica de dados e informações

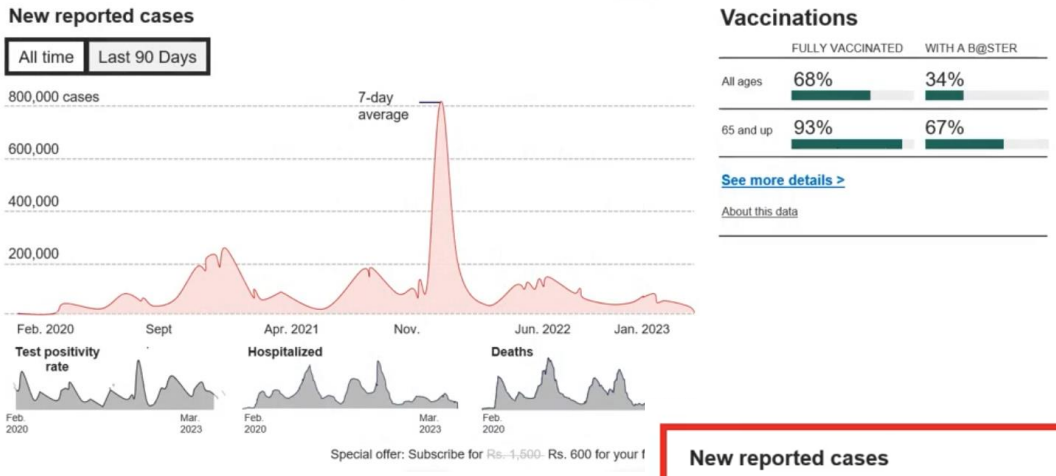
assume formas diferentes

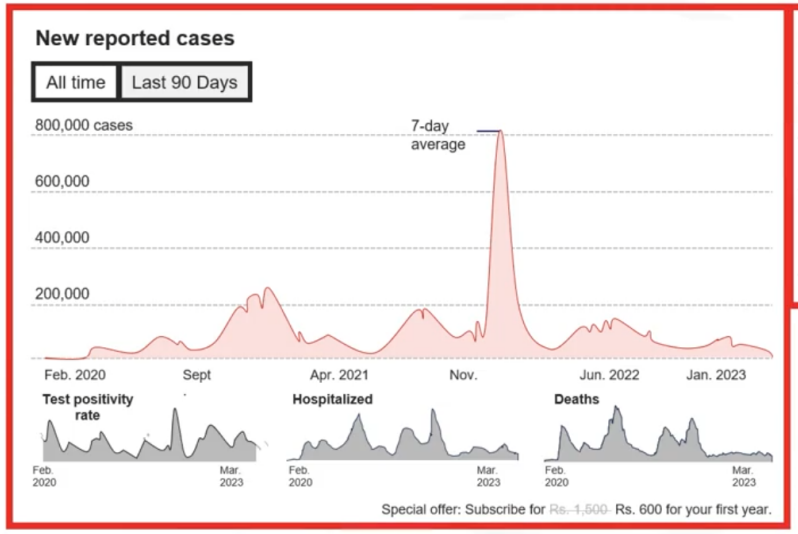
gráficos e diagramas básicos

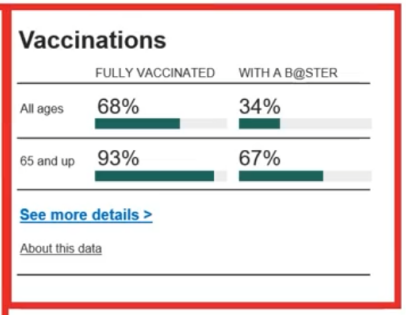
dashboards interativos, mapas e infográficos

Visualização de Dados

Leveraging the power of data visualization

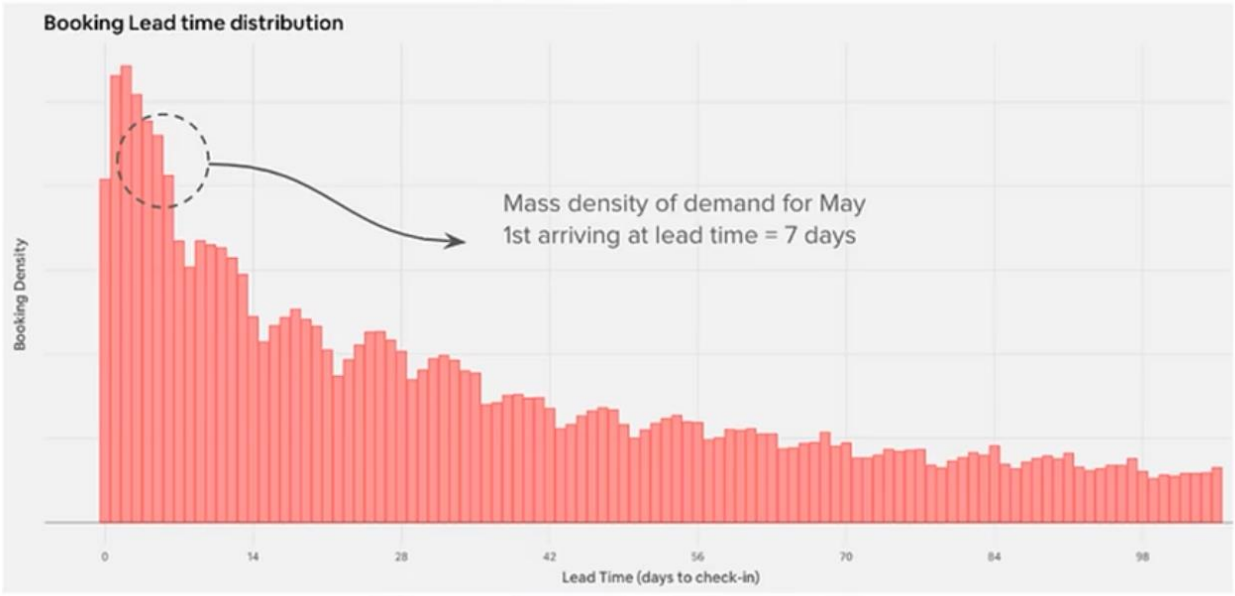




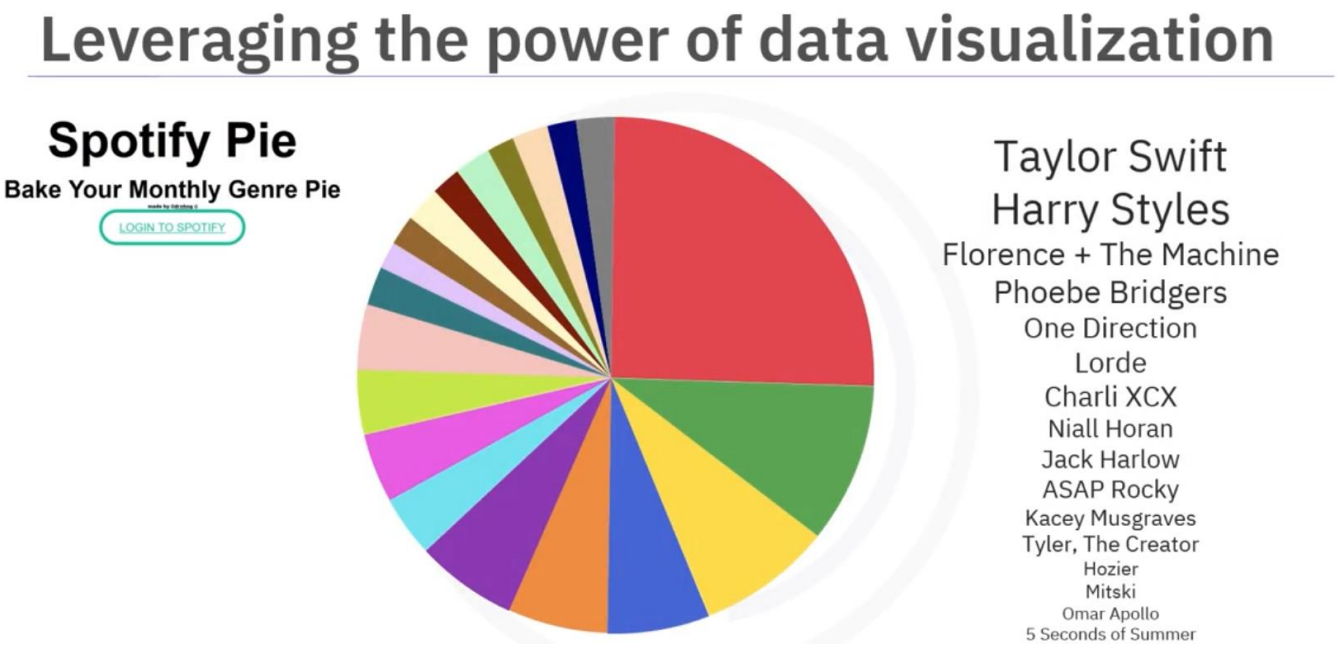


Visualização de Dados

Leveraging the power of data visualization

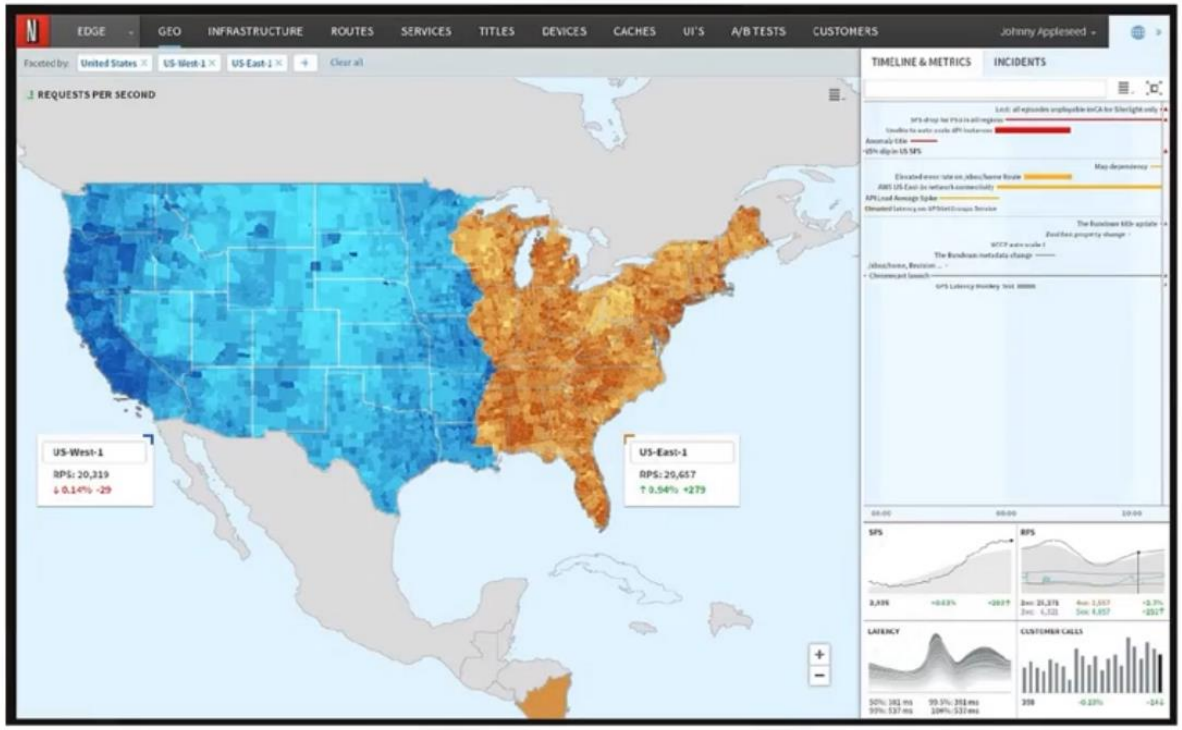


Visualização de Dados



Visualização de Dados

Leveraging the power of data visualization



Visualização de Dados

Visualização de dados é uma ferramenta essencial para quem trabalha com dados.

As empresas podem obter insights sobre tendências de mercado, desempenho financeiro e comportamento dos clientes.

Profissionais da saúde conseguem identificar padrões nos dados dos pacientes e desenvolver planos de tratamento direcionados.

A visualização ajuda a analisar o desempenho dos alunos e a informar decisões pedagógicas, contribuindo para melhores resultados académicos.

No setor público, os governos podem tomar decisões informadas e comunicar os dados ao público.

Cientistas e investigadores podem analisar dados complexos, desenvolver novos insights e partilhar conclusões de forma eficaz.

No setor do entretenimento, a visualização ajuda a determinar classificações e, em última análise, sinaliza aos criadores o que o público deseja.

Visualização de Dados

Embora a visualização de dados se tenha tornado uma ferramenta essencial, nem todas as visualizações são igualmente eficazes.

Para criar visualizações eficientes, é importante seguir as melhores práticas que assegurem que os dados são representados de forma precisa e que a mensagem é transmitida claramente.

Visualização de Dados

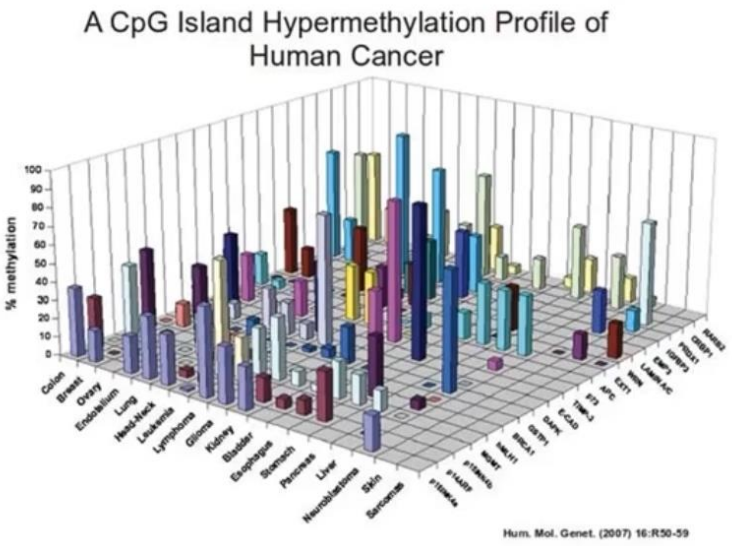
Uma das melhores práticas é escolher o tipo de visualização apropriado para os dados que se pretendem apresentar.

Diferentes tipos de dados requerem técnicas de visualização distintas. Escolher a ferramenta certa é fundamental para comunicar os dados de forma eficaz.

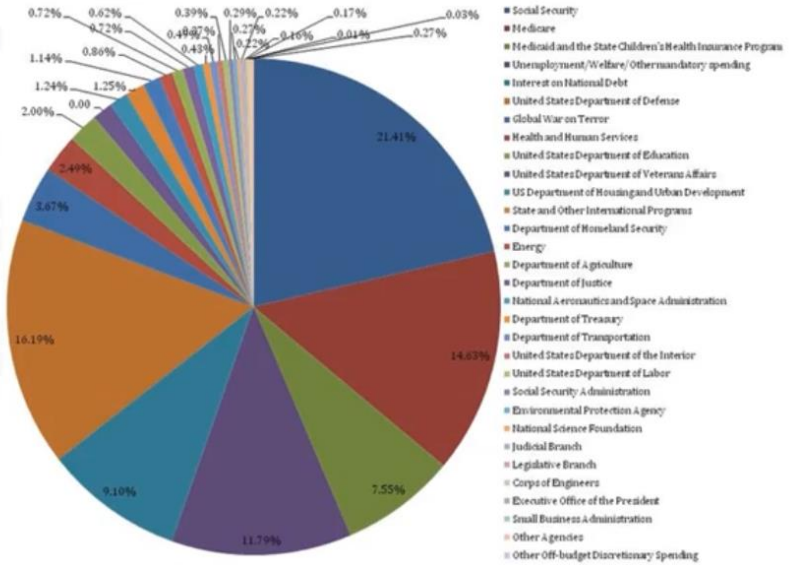
A visualização deve ser simples e fácil de ler.

Visualização de Dados

Best practices (bad example)



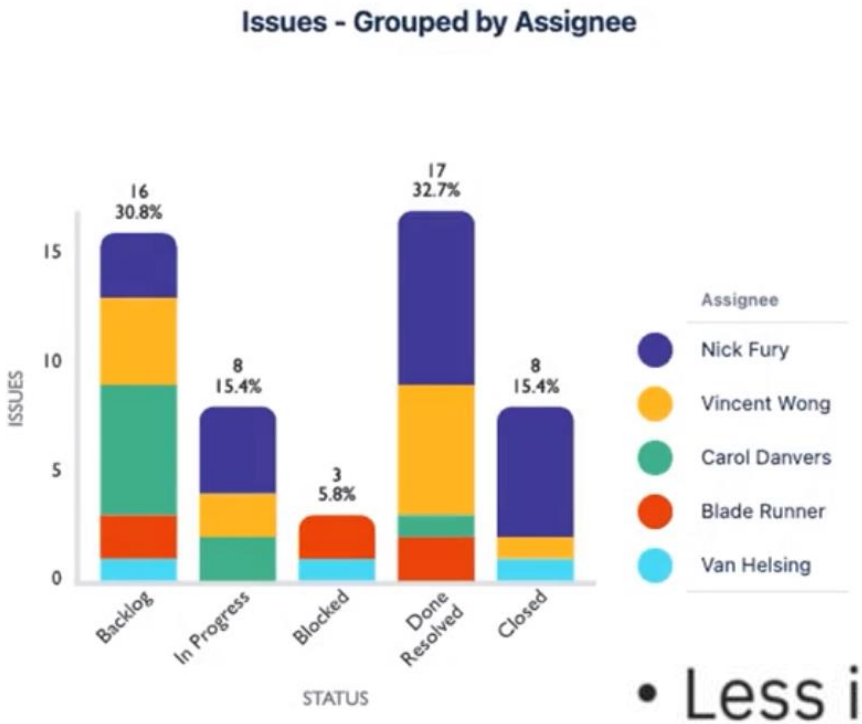
SOURCE: https://www.livestories.com/blog/five-ways-to-fail-data-visualization



SOURCE: https://en.wikipedia.org/wiki/2007_United_States_federal_budget#.

Visualização de Dados

Best practices (good example)



- Less is more effective
- Less is more attractive
- Less is more impactful

Visualização de Dados

Por exemplo, um **gráfico de linhas** é ideal para mostrar tendências ao longo do tempo, enquanto um **gráfico de barras** é mais adequado para comparar valores.

Para assegurar que os dados são representados corretamente, é também importante rotular os eixos de forma clara e fornecer contexto à informação apresentada.

Deve incluir um título claro que resuma a mensagem principal e uma legenda que explique o significado de quaisquer símbolos ou cores utilizadas.

Evitar gráficos ou informações desnecessárias é crucial, pois podem confundir o leitor e desviar a atenção da mensagem principal.

A visualização deve concentrar-se no ponto central, utilizando apenas os dados e etiquetas necessários.

Por fim, é importante ter em conta o público-alvo ao criar a visualização: quem irá visualizar e o que necessita saber.

Bibliotecas que vamos abordar nesta sessão …

MatPlotLib

Pandas

MatPlotLib

Biblioteca de utilização geral que oferece uma grande variedade de gráficos e opções de personalização.

O Matplotlib integra-se bem com outras bibliotecas e frameworks, como o NumPy, Pandas, Seaborn e Plotly, permitindo combinar as forças de diferentes ferramentas para criar visualizações sofisticadas e interativas.

PANDAS

Pandas, utilizada principalmente para a manipulação de dados, mas que também dispõe de funcionalidades de visualização

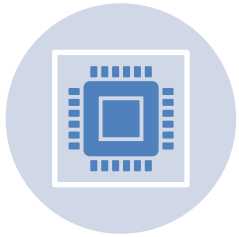
As funções associados aos gráficos do Pandas são construídas em cima do Matplotlib, o que significa que podes aproveitar a versatilidade deste último enquanto manipulas os teus dados com o Pandas.

FT02

Introdução ao MatPlotLib

O MATPLOTLIB É UMA DAS BIBLIOTECAS DE VISUALIZAÇÃO DE DADOS MAIS UTILIZADAS EM PYTHON.

FOI CRIADO POR JOHN HUNTER, UM NEUROBIÓLOGO AMERICANO, QUE, NUMA ALTURA, FAZIA PARTE DE UMA EQUIPA DE INVESTIGAÇÃO QUE ANALISAVA SINAIS DE EEG E ECOG UTILIZANDO UM SOFTWARE PROPRIETÁRIO.



DEVIDO À LIMITAÇÃO DE TER APENAS UMA LICENÇA DO SOFTWARE, OS INVESTIGADORES PRECISAVAM DE UMA ALTERNATIVA QUE PUDESSE SER UTILIZADA POR TODA A EQUIPA.

ASSIM, JOHN HUNTER DECIDIU SUBSTITUIR O SOFTWARE PROPRIETÁRIO POR UMA VERSÃO BASEADA NO MATLAB, QUE PUDESSE SER UTILIZADA POR ELE E PELOS SEUS COLEGAS, SENDO POSTERIORMENTE EXPANDIDA POR VÁRIOS INVESTIGADORES.

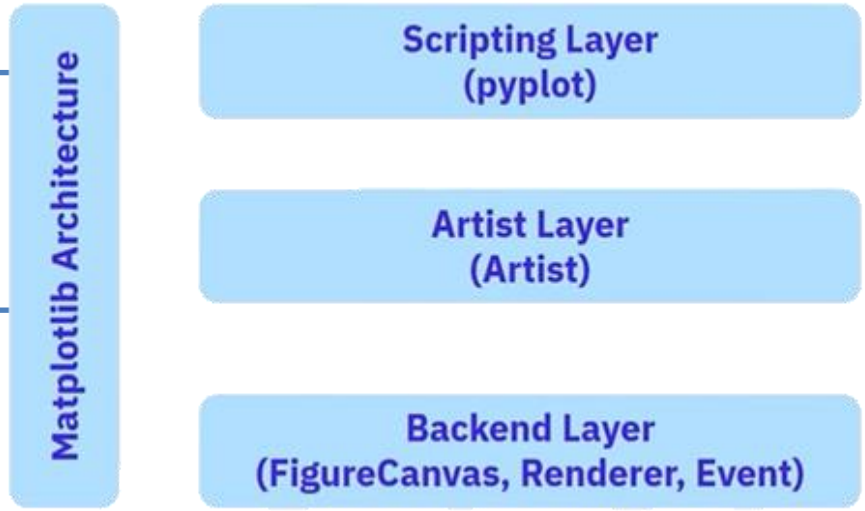
Introdução ao MatPlotLib

Tal como o Matlab, o Matplotlib dispõe de uma interface de scripting para a rápida e fácil geração de gráficos. Quanto à sua arquitetura, o Matplotlib é composto por três camadas principais:

A camada de backend: Responsável por desenhar os gráficos no ecrã.

A camada de artistas: Onde ocorre a maior parte do processamento, sendo responsável por desenhar os elementos do gráfico.

A camada de scripting: Destinada a tarefas quotidianas, simplificando a geração de gráficos com funções de alto nível, como as disponíveis na interface Pyplot.



Introdução ao MatPlotLib – Backend Layer

O backend define a área onde o gráfico é desenhado (o Canvas) e como é desenhado (através do renderer). O gestor de eventos capta as interações do utilizador, como cliques e pressionamento de teclas.

Cada elemento visível num gráfico – título, linhas, etiquetas, imagens – é gerido por um objeto artista.

Has three built-in abstract interface classes:

1. FigureCanvas: **matplotlib.backend_bases.FigureCanvas**
   - Encompasses the area onto which the figure is drawn
2. Renderer: **matplotlib.backend_bases.Renderer**
   - Knows how to draw on the FigureCanvas
3. Event: **matplotlib.backend_bases.Event**
   - Handles user inputs such as keyboard strokes and mouse clicks

Introdução ao MatPlotLib – Artist Layer

Existem objetos artistas primitivos (como linhas, retângulos, círculos e texto) e compostos (como eixos, ticks, subplots e figuras).

A figura é o objeto principal que contém e gere todos os elementos gráficos, sendo o eixo onde se definem as escalas, os ticks, a grelha e o fundo do gráfico.

- Comprised of one main object – **Artist**:
  - Knows how to use the Renderer to draw on the canvas.
- Title, lines, tick labels, and images, all correspond to individual **Artist** instances.
- Two types of **Artist** objects:
  1. **Primitive**: Line2D, Rectangle, Circle, and Text.
  2. **Composite**: Axis, Tick, Axes, and Figure.
- Each *composite* artist may contain other *composite* artists as well as *primitive* artists.

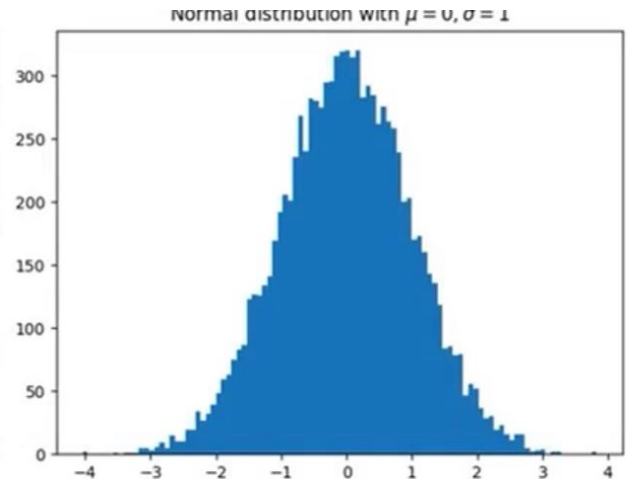
Introdução ao MatPlotLib – Scripting Layer

Para tarefas quotidianas, a camada de scripting, nomeadamente a **interface Pyplot**, é a mais indicada.

Esta interface cria automaticamente um **Canvas** e uma figura, ligando-os e permitindo a criação rápida e simples de gráficos.

Por exemplo, para gerar um histograma de 10.000 números aleatórios, importa-se a interface Pyplot e usa-se a função hist, que cria um conjunto de retângulos (barras) para cada intervalo de dados.

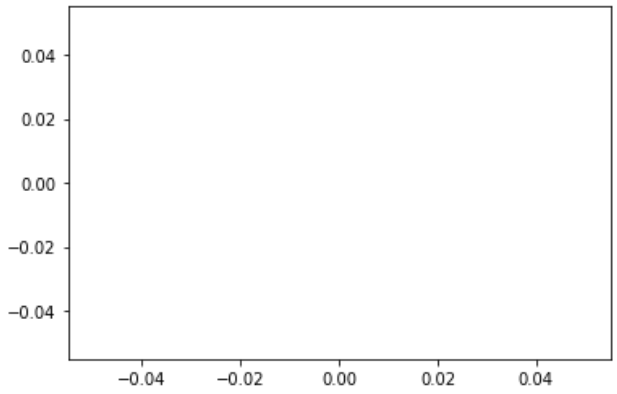
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randn(10000)
plt.hist(x, 100)
plt.title(r'Normal distribution with $\mu=0, \sigma=1$')
plt.savefig('matplotlib_histogram.png')
plt.show()
```



Gráficos com Matplotlib – Interface pyplot

- Ao importar a biblioteca e o interface pyplot usaremos o alias plt

```
import matplotlib.pyplot as plt

plt.plot()
```

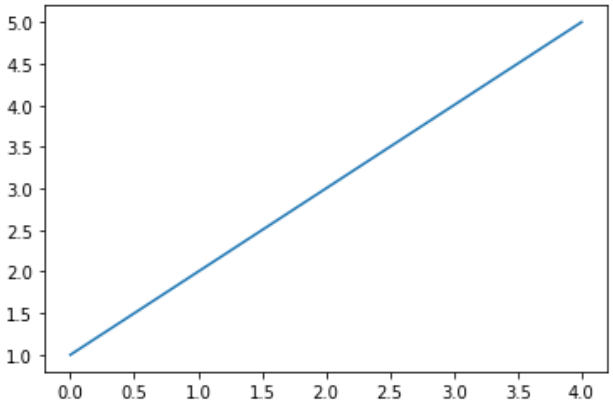


Função Plot

- A função plot() é utilizada para renderizar uma imagem, no exemplo anterior mostra a saída padrão da função. Vamos adicionar alguns dados para criar o nosso primeiro gráfico:

Função Plot

```
plt.plot([1, 2, 3, 4, 5])
[<matplotlib.lines.Line2D at 0x7f4caa06a810>]
```



Geramos o nosso primeiro gráfico com dados em sequência de 1 a 5 que criou uma linha reta.

Ao executar a instrução de plot no notebook recebemos como retorno uma mensagem não muito amigável [<matplotlib.lines.Line2D at 0x7f4caa06a810>].

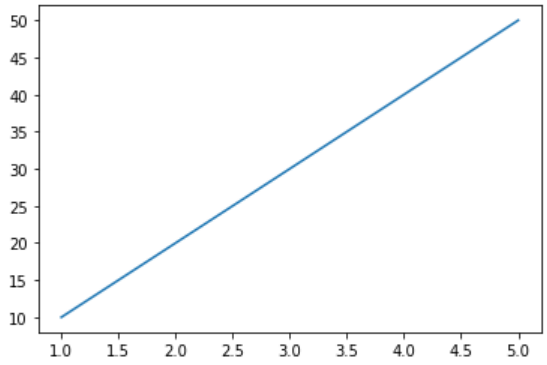
Essa informação retornada não é um erro, trata-se de uma mensagem padrão da Matplotlib informando que um objeto foi criado.

Essa mensagem pode ser escondida de duas formas, utilizando a função plt.show() ou adicionando um ; no final da instrução de plot.

Função Plot

- Vamos agora criar 2 listas de dados, dessa vez armazenadas em variáveis:

```
x = [1, 2, 3, 4, 5]
y = [10, 20, 30, 40, 50]

plt.plot(x, y);
```



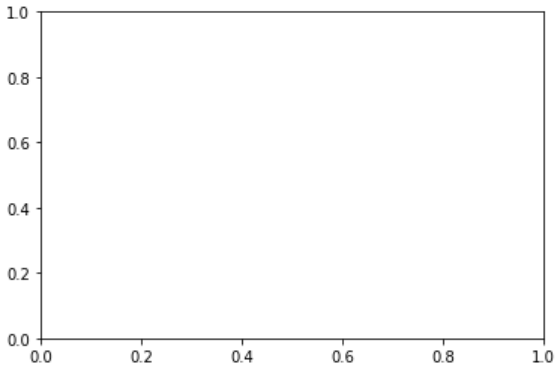
Inserimos dados em ambos os eixos, x e y;

O eixo dos y assumir os valores de 10 a 50

Figure

- Podemos, também, usar abordagem POO em vez do interface Pyplot (que utilizamos no import)

```
fig = plt.figure()
ax = fig.add_subplot()
plt.show()
```

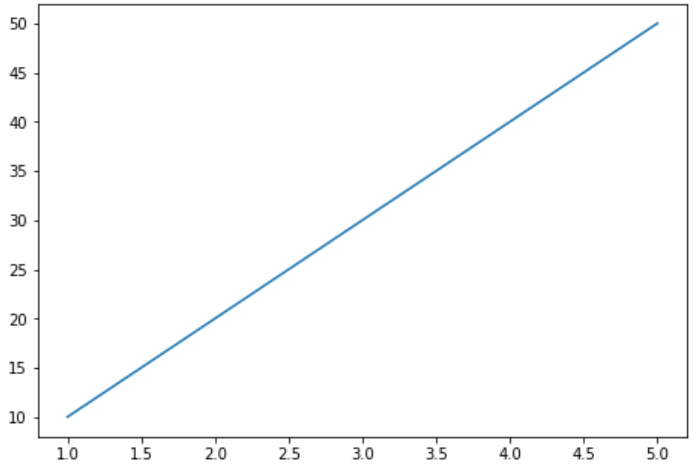


Aqui é necessário definir o objeto figure, para então adicionarmos um “plot” à figura criada.

Estamos a usar plt.show() para ocultar a mensagem padrão

Figure – Construção do gráfico

```
fig = plt.figure()
ax = fig.add_axes([1, 1, 1, 1])
ax.plot(x, y)
plt.show()
```

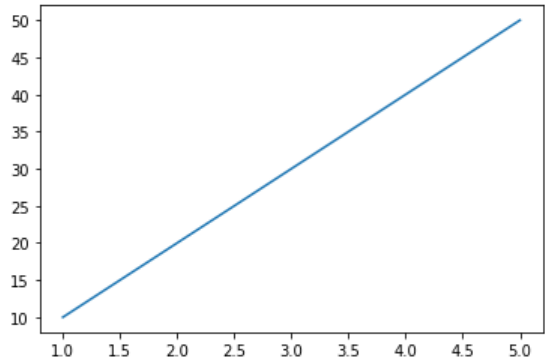


utilizamos os dados das variáveis que criamos anteriormente

axes é uma camada da figura onde um gráfico será posicionado.

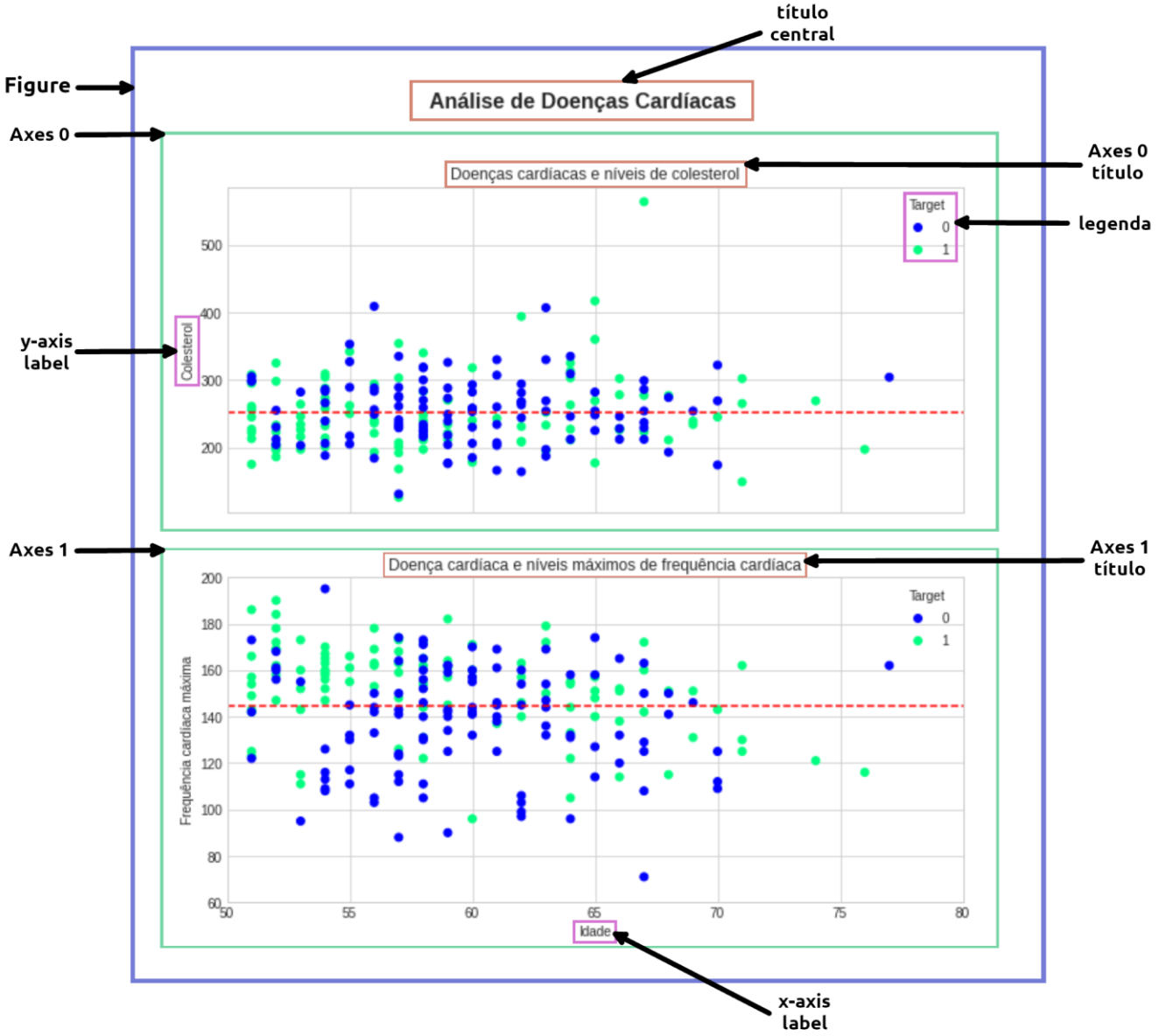
Figure – Construção do gráfico – Outra Forma

```
fig, ax = plt.subplots()
ax.plot(x, y);
```



Permite explorar muitas configurações da Matplotlib

Matplotlib - Anatomia de um gráfico



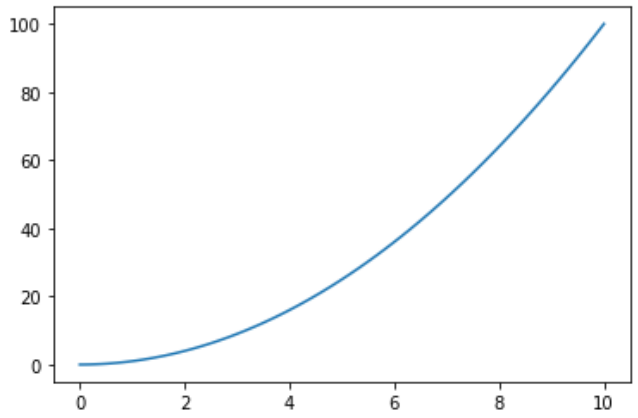
WorkFlow

```python
# importar da biblioteca
import matplotlib.pyplot as plt

# preparar os dados
x = [1, 2, 3, 4, 5]
y = [10, 20, 30, 40, 50]

# configuração do gráfico
fig, ax = plt.subplots(figsize=(10, 10))

# desenhar os dados
ax.plot(x, y)

# customizar o gráfico
ax.set(title="Exemplo de plot customizado", xlabel="axis x", ylabel="axis y")

# exibindo o plot
plt.show()
```

Gráficos de Linha

```
import numpy as np

x = np.linspace(0, 10, 100)
x[:10]

array([0.        , 0.1010101 , 0.2020202 , 0.3030303 , 0.4040404 ,
       0.50505051, 0.60606061, 0.70707071, 0.80808081, 0.90909091])
```

```
fig, ax = plt.subplots()
ax.plot(x, x**2);
```



Scatter – Gráfico de Dispersão

```
fig, ax = plt.subplots()
ax.scatter(x, np.sin(x));
```

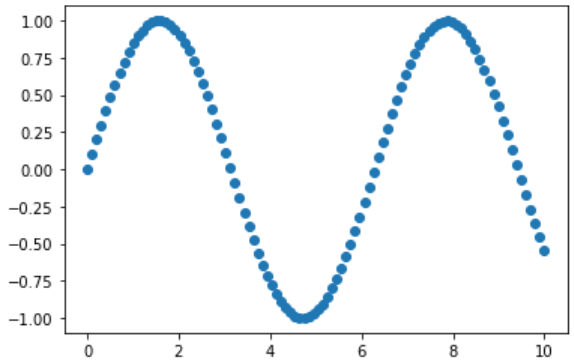
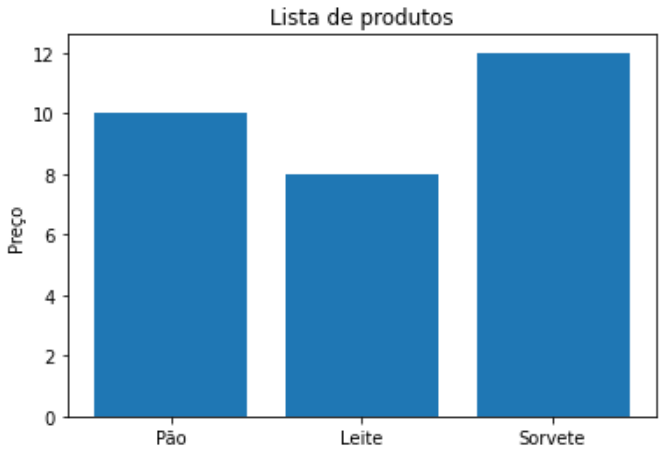
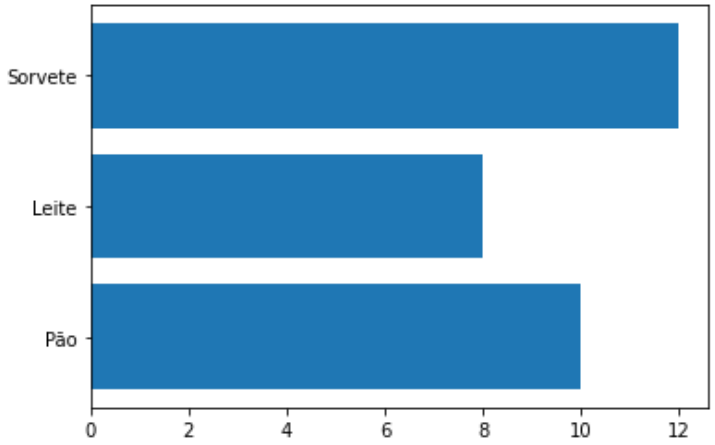


Gráfico de Barras

```
produtos = {"Pão": 10,
            "Leite": 8,
            "Sorvete": 12}

fig, ax = plt.subplots()
ax.bar(produtos.keys(), produtos.values())
ax.set(title="Lista de produtos", ylabel="Preço");
```

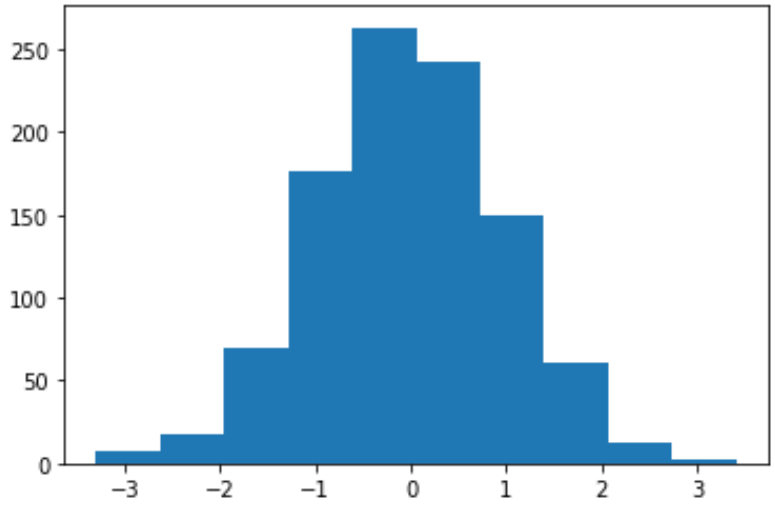



```
fig, ax = plt.subplots()
ax.barh(list(produtos.keys()), list(produtos.values()));
```



Histograma

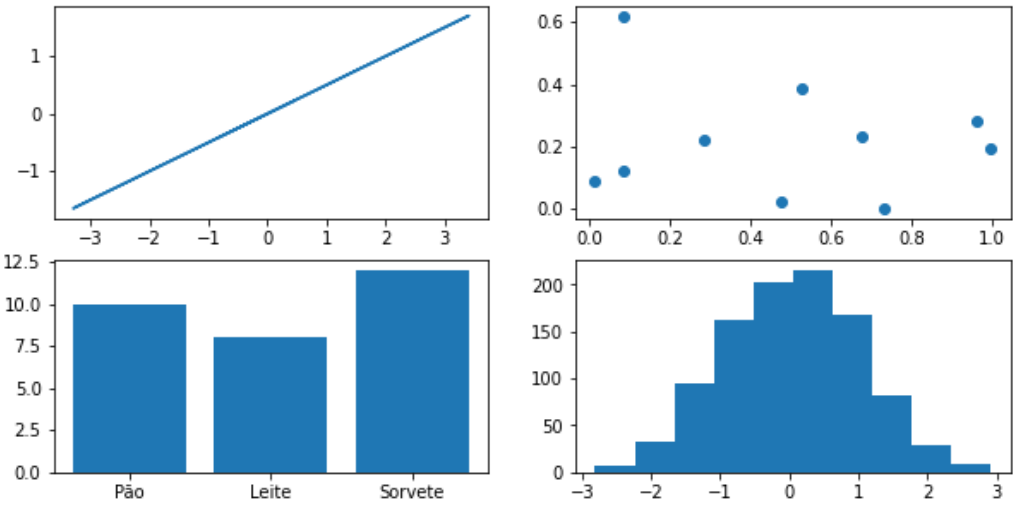
```
x = np.random.randn(1000)


fig, ax = plt.subplots()
ax.hist(x);
```



Subplots

Desenhamos os dados em casa *axis* da figura:

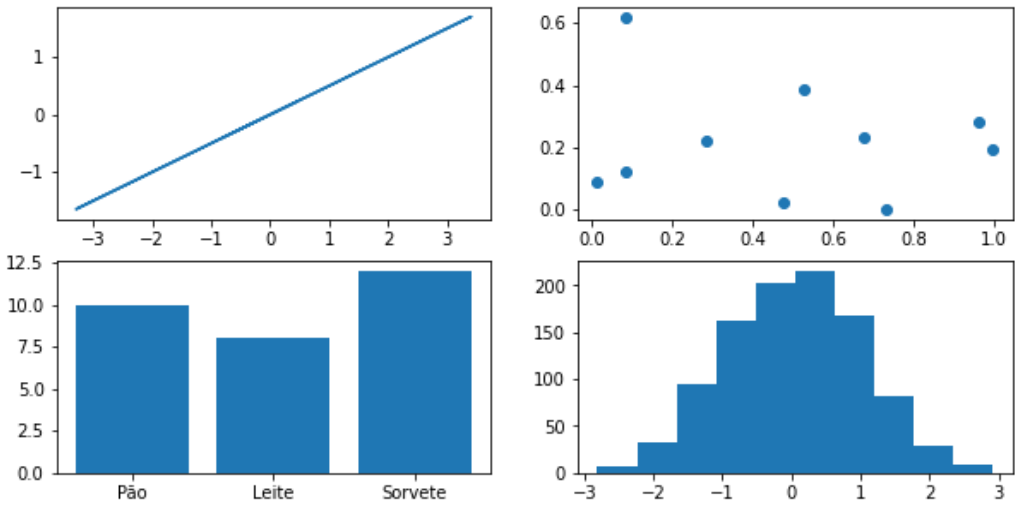
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2,
                                             ncols=2,
                                             figsize=(10, 5))
ax1.plot(x, x/2); #line
ax2.scatter(np.random.random(10), np.random.random(10)) #scatter
ax3.bar(produtos.keys(), produtos.values()) #bar
ax4.hist(np.random.randn(1000)); #hist
```



Subplots

```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(10, 5))


ax[0, 0].plot(x, x/2) #line
ax[0, 1].scatter(np.random.random(10), np.random.random(10)) #scatter
ax[1, 0].bar(produtos.keys(), produtos.values()) #bar
ax[1, 1].hist(np.random.randn(1000)); #hist
```

Usamos índices para desenhar os dados



Customizar gráficos

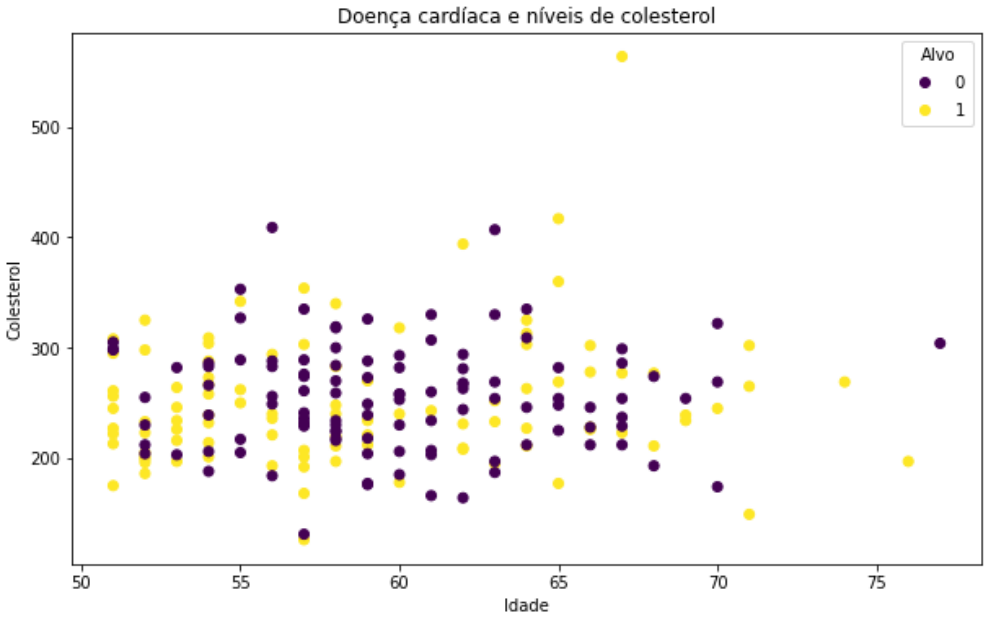
```
df = pd.read_csv("heart-disease.csv")
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Vamos iniciar nosso gráfico com uma análise de dados em pacientes com mais de 50 anos

Customizar gráficos

```
fig, ax = plt.subplots(figsize=(10, 6))

scatter = ax.scatter(mais_de_50["age"],
                     mais_de_50["chol"],
                     c=mais_de_50["target"])

ax.set(title="Doença cardíaca e níveis de colesterol",
       xlabel="Idade",
       ylabel="Colesterol")

ax.legend(*scatter.legend_elements(), title="Alvo");
```



Customizar gráficos

```
fig, ax = plt.subplots(figsize=(10, 6))

scatter = ax.scatter(mais_de_50["age"],
                     mais_de_50["chol"],
                     c=mais_de_50["target"])

ax.set(title="Doença cardíaca e níveis de colesterol",
       xlabel="Idade",
       ylabel="Colesterol")

ax.legend(*scatter.legend_elements(), title="Alvo");
```

primeiro criamos um objeto do tipo subplots e definimos o tamanho final da nossa imagem, no caso (10x6)

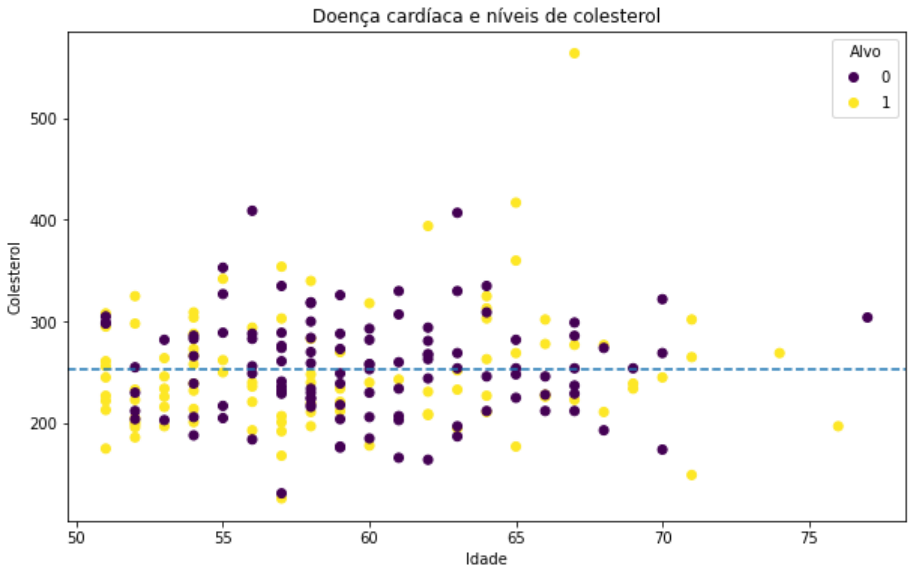
Na configuração dos axes ou escolhemos o tipo de gráfico scatter e preenchemos os valores dos eixos, x e y inserindo respetivamente as colunas de idade e colesterol

O último parâmetro na configuração de scatter o c é um marcador de cores que recebe como argumento valores em escala ou sequência de números, que serão mapeados para cores, nesse caso inserimos a coluna target que possui apenas dois tipos de valores 0 ou 1 para exibir roxo nos casos onde o paciente não é um possível alvo de uma doença cardíaca e amarelo para os casos onde o paciente é um possível alvo.

A função set() permite configurar o título da nossa imagem e os labels dos eixos x e y, respetivamente idade e colesterol. E a última função inserida legend() foi utilizada para configurar a nossa legenda que aparece no canto superior direito com o título de Alvo.

Podemos Customizar um pouco mais

```
# Criar o gráfico
fig, ax = plt.subplots(figsize=(10, 6))

# Desenhar os dados
scatter = ax.scatter(mais_de_50["age"],
                     mais_de_50["chol"],
                     c=mais_de_50["target"])

# Customizar o gráfico
ax.set(title="Doença cardíaca e níveis de
colesterol",
       xlabel="Idade",
       ylabel="Colesterol")
ax.legend(*scatter.legend_elements(), title="Alvo")

# Adicionar a linha média horizontal para colesterol
ax.axhline(mais_de_50["chol"].mean(),
           linestyle="--");
```



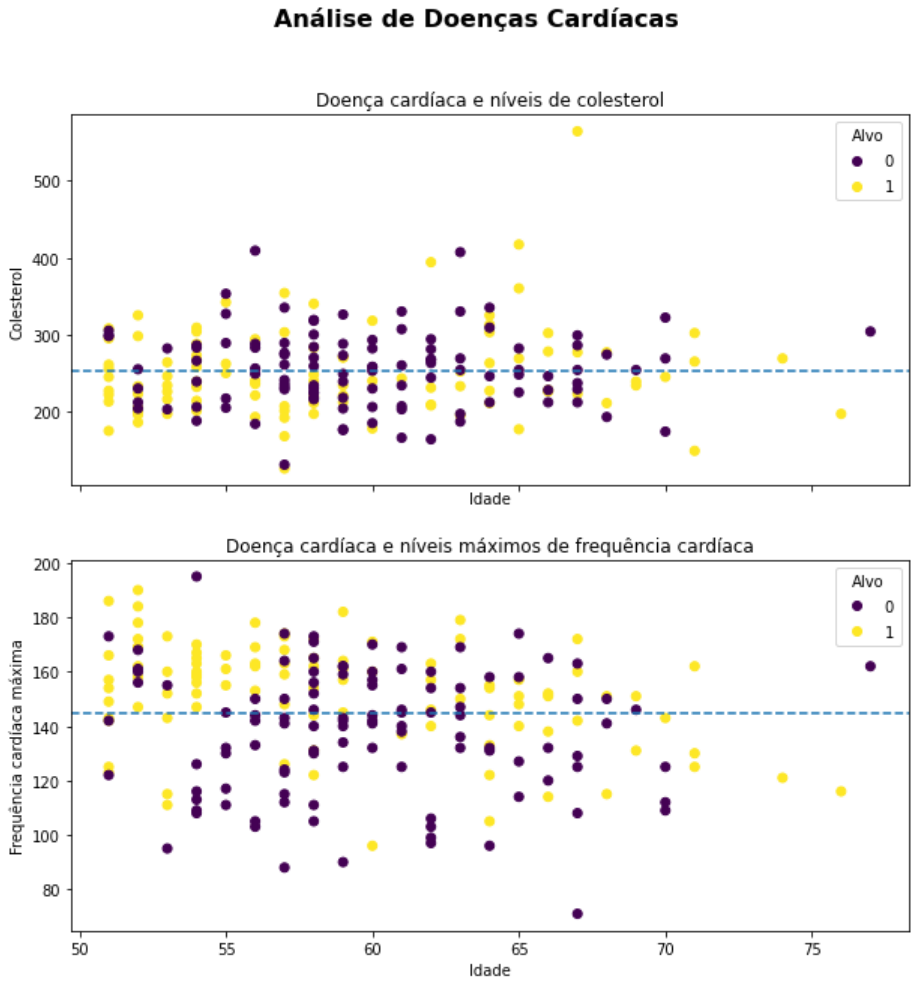
a função axhline() gera uma linha horizontal, para os dados que lhe são passados

Adicionar outro gráfico

```
# Criar o plot
fig, (ax0, ax1) = plt.subplots(nrows=2,
                               ncols=1,
                               sharex=True,
                               figsize=(10, 10))

# Adicionar os dados para ax0
scatter = ax0.scatter(mais_de_50["age"],
                      mais_de_50["chol"],
                      c=mais_de_50["target"])

# Customizar ax0
ax0.set(title="Doença cardíaca e níveis de colesterol",
       xlabel="Idade",
       ylabel="Colesterol")
ax0.legend(*scatter.legend_elements(), title="Alvo")

# Adicionar a linha média horizontal para colesterol em ax0
ax0.axhline(mais_de_50["chol"].mean(),
           linestyle="--")
```

```
# Adiciona os dados para ax1
scatter = ax1.scatter(mais_de_50["age"],
                      mais_de_50["thalach"],
                      c=mais_de_50["target"])

# Customizar ax1
ax1.set(title="Doença cardíaca e níveis máximos de
frequência cardíaca",
        xlabel="Idade",
        ylabel="Frequência cardíaca máxima")
ax1.legend(*scatter.legend_elements(), title="Alvo")

# Adicionar a linha média para frequência cardíaca
ax1.axhline(mais_de_50["thalach"].mean(),
            linestyle="--")

# Título da figura
fig.suptitle("Análise de Doenças Cardíacas", fontsize=16,
fontweight="bold");
```

Adicionar outro gráfico



Podemos também customizar cores

```python
# Definir um novo estilo com grid
plt.style.use("seaborn-whitegrid")

# Criar o plot
fig, (ax0, ax1) = plt.subplots(nrows=2,
                               ncols=1,
                               sharex=True,
                               figsize=(10, 10))

# Adicionar os dados para ax0
scatter = ax0.scatter(mais_de_50["age"],
                      mais_de_50["chol"],
                      c=mais_de_50["target"],
                      cmap='winter')

# Customizar ax0
ax0.set(title="Doenças cardíacas e níveis de colesterol",
        ylabel="Colesterol")
ax0.set_xlim([50, 80])
ax0.legend(*scatter.legend_elements(), title="Alvo")

# Adicionando a linha média horizontal para colesterol em ax0
ax0.axhline(mais_de_50["chol"].mean(),
            color="r",
            linestyle="--")
```

```python
# Adicionar os dados para ax1
scatter = ax1.scatter(mais_de_50["age"],
                      mais_de_50["thalach"],
                      c=mais_de_50["target"],
                      cmap='winter')

# Customizar ax1
ax1.set(title="Doença cardíaca e níveis máximos de frequência cardíaca",
        ylabel="Frequência cardíaca máxima",
        xlabel="Idade",
        ylim=[60, 200])
ax1.legend(*scatter.legend_elements(), title="Alvo")

# Adiconar a linha média para frequência cardíaca
ax1.axhline(mais_de_50["thalach"].mean(),
            color="r",
            linestyle="--")

# Título da figura
fig.suptitle("Análise de Doenças Cardíacas", fontsize=16, fontweight="bold");
```

Podemos também customizar cores

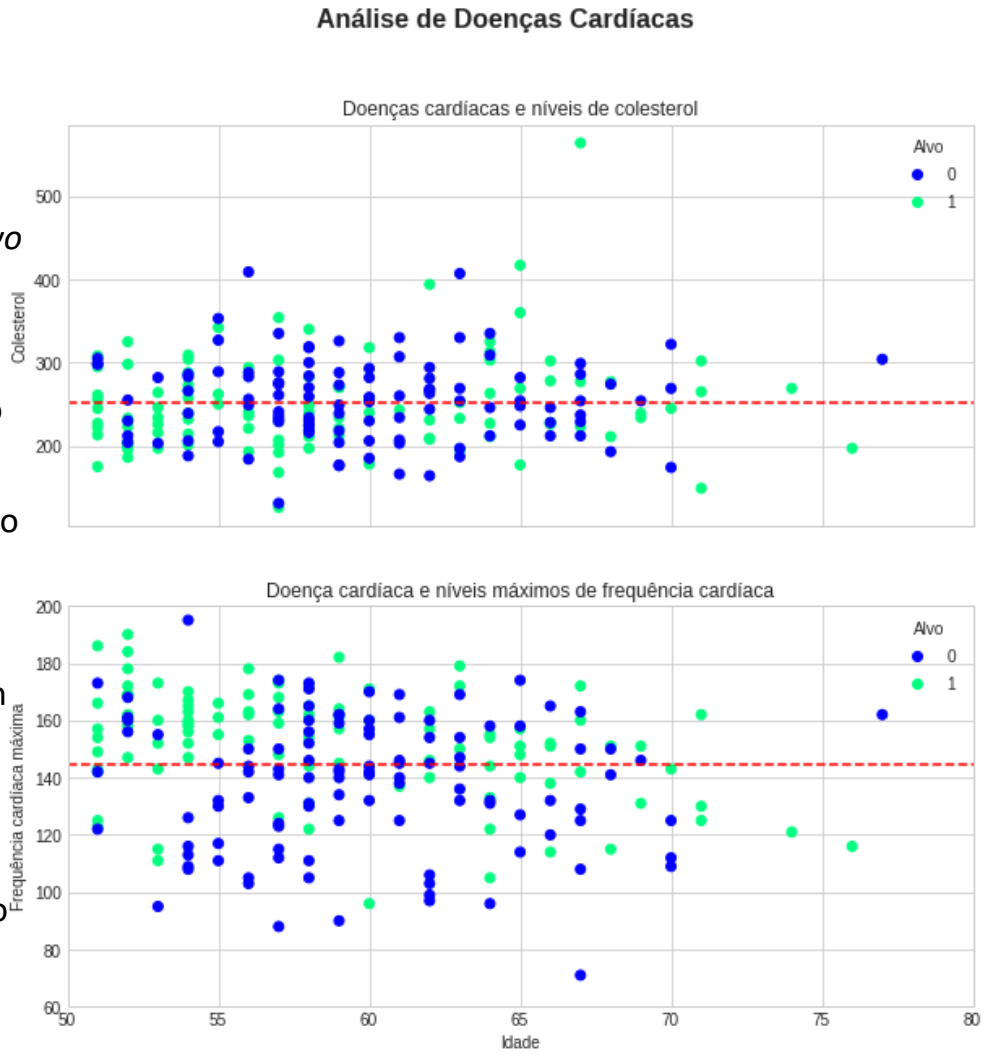
A opção *style.use("seaborn-whitegrid") oferece um novo tema ao gráfico*

Ao configurar os dados para os plots ax0 e ax1, adicionamos a opção cmap ou color map para definir o esquema de cores dos dados relacionados ao alvo

Já na customização dos plots adicionamos limites para o eixo x com set_xlim a variar de 50 anos até 80 anos.

No plot ax1 fizemos o mesmo, mas para o eixo y utilizando ylim e limitamos os valores que representam a frequência cardíaca entre 60 e 200.

Por fim mudamos também as cores das linhas horizontais que marcam as médias de colesterol e frequência cardíaca respetivamente, utilizando a opção color="r" trocamos para cor vermelha.



Documentação

- https://matplotlib.org/stable/index.html