INSTITUTO DO EMPREGO E FORMAÇÃO PROFISSIONAL, IP

PPT12– Fundamentos de Python (UFCD 10793)

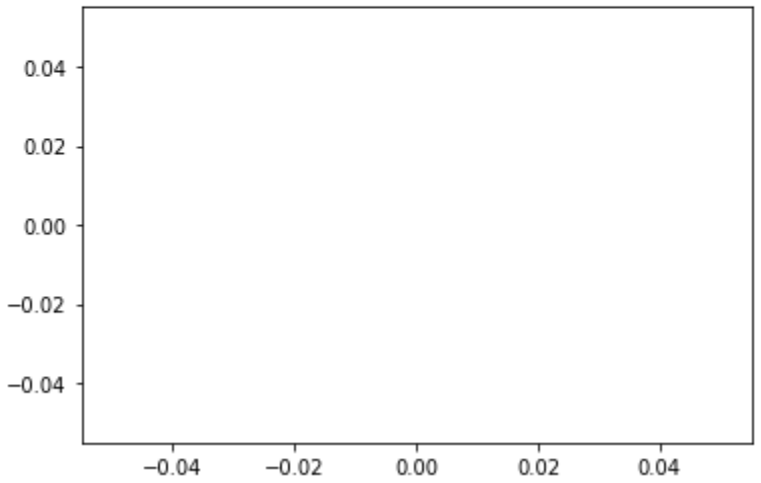
Sandra Liliana Meira de Oliveira

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

Gráficos com Matplotlib – Interface pyplot

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

```
import matplotlib.pyplot as plt


plt.plot()
```

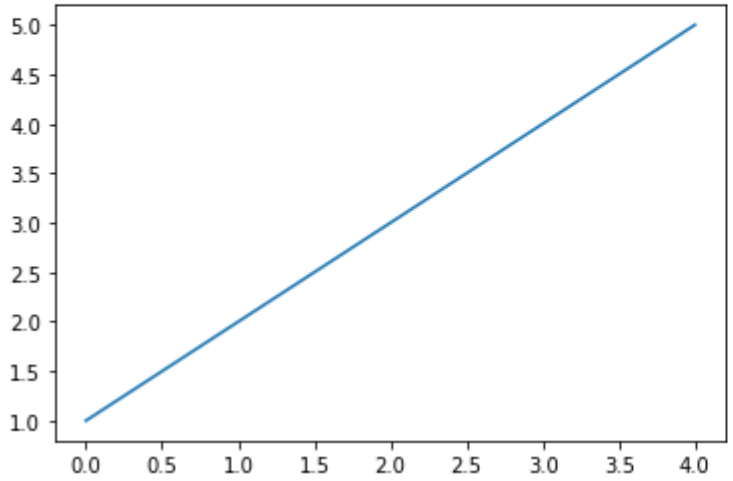


Função Plot

- 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>].

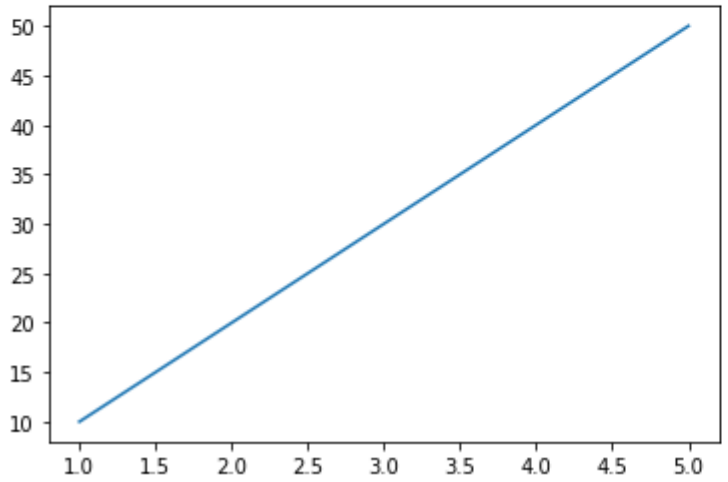
Essa informação retornada não é um erro, trata-se de uma mensagem padrão da Matplotlib informando que um objeto foi criado.

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);
```



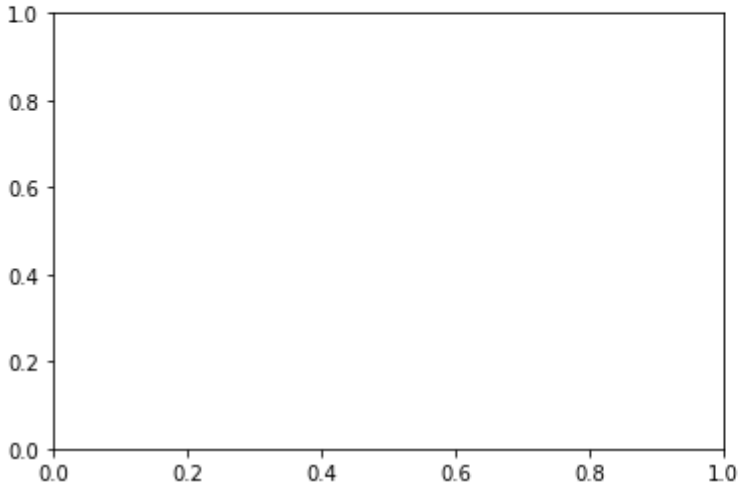
Inserimos dados em ambos os eixos, x e y;

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()
```

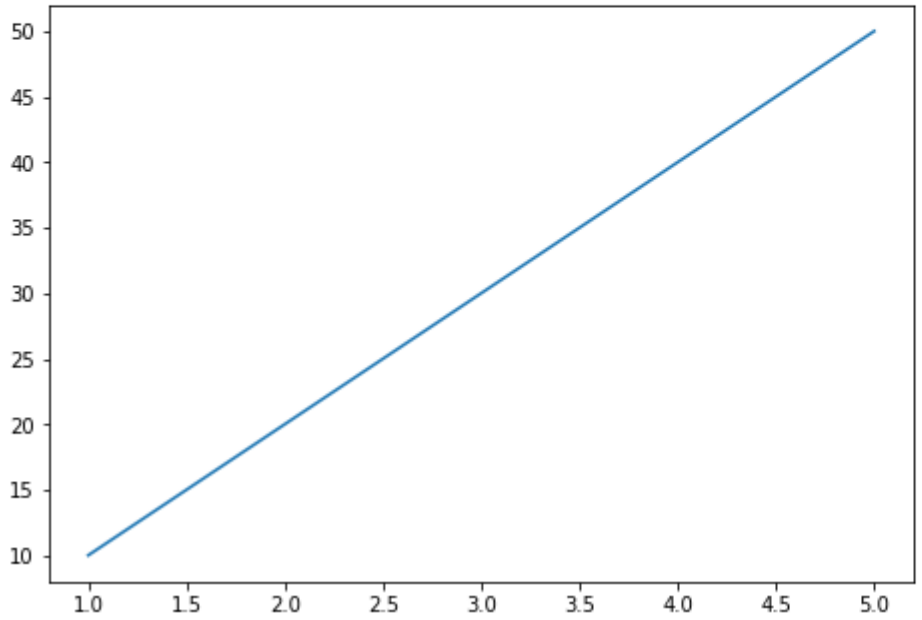


Aqui é necessário definir o objeto figure, para então adicionarmos um “plot” à figura criada.

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()
```

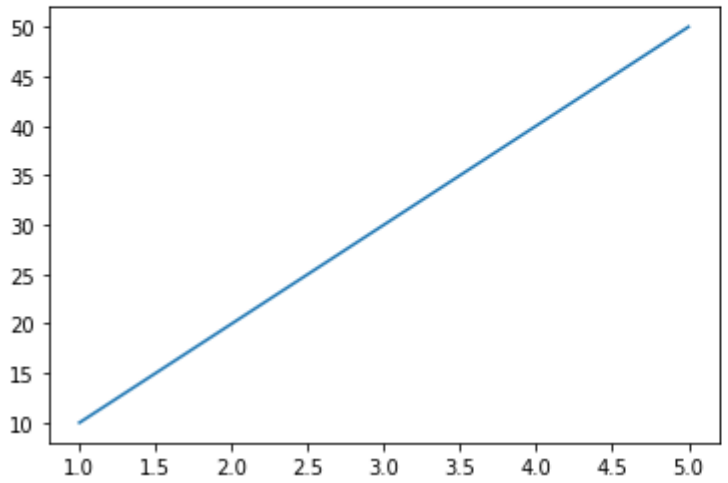


utilizamos os dados das variáveis que criamos anteriormente

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

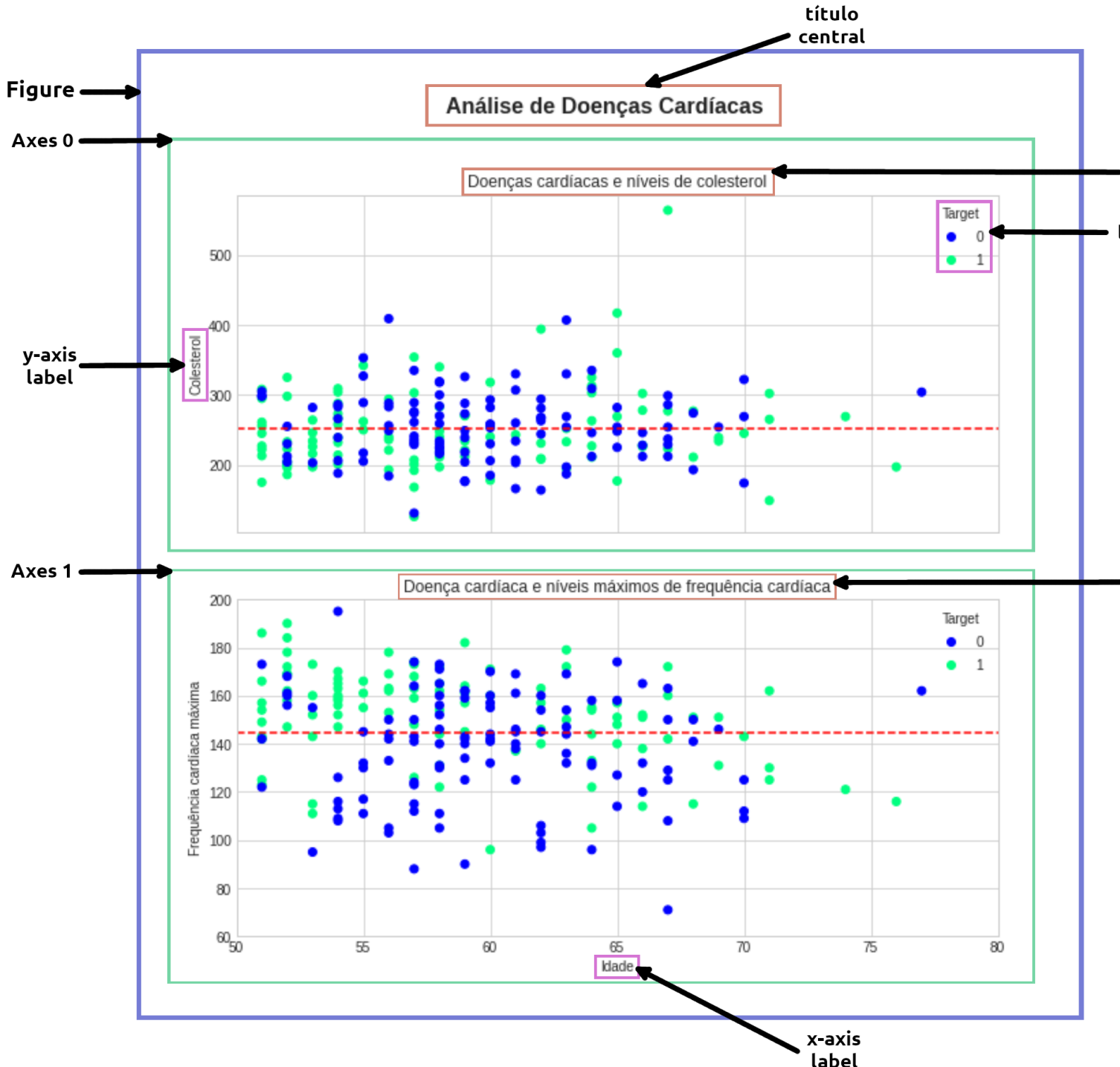
Figure – Construção do gráfico – Outra Forma

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



Permite explorar muitas configurações da Matplotlib

Matplotlib - Anatomia de um gráfico



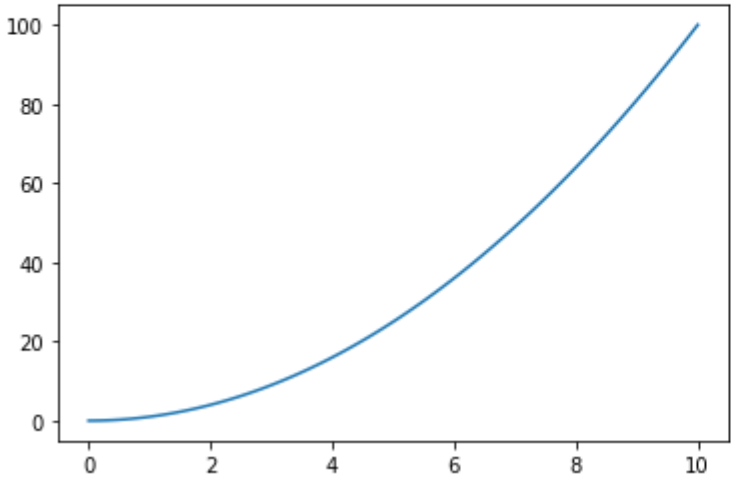
WorkFlow

```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));
```

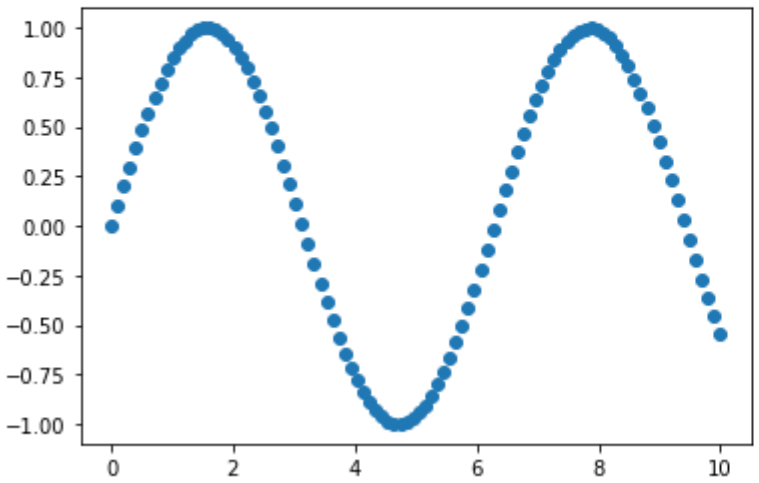
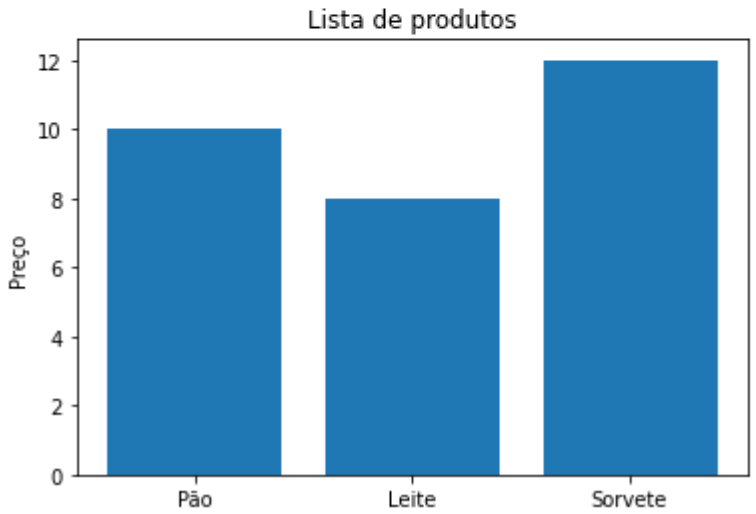
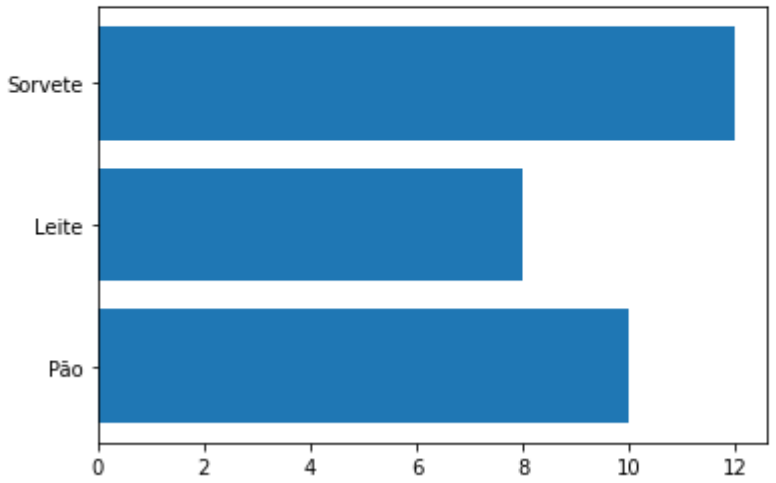


Gráfico de Barras

```
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");
```

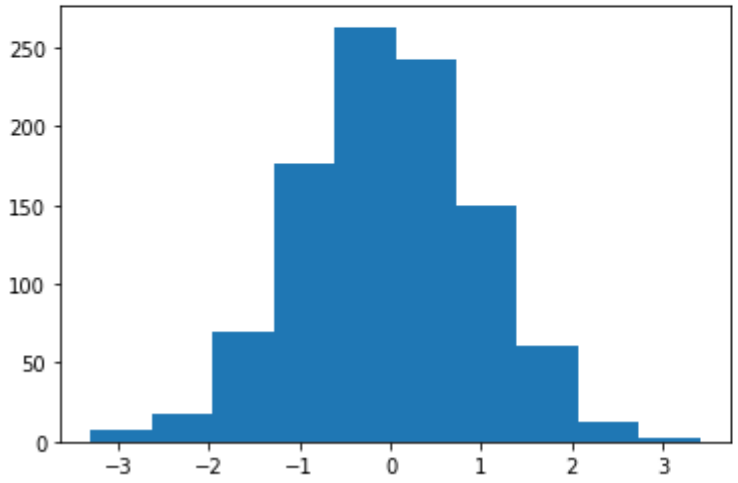



```
fig, ax = plt.subplots()
ax.barh(list(produtos.keys()), list(produtos.values()));
```



Histograma

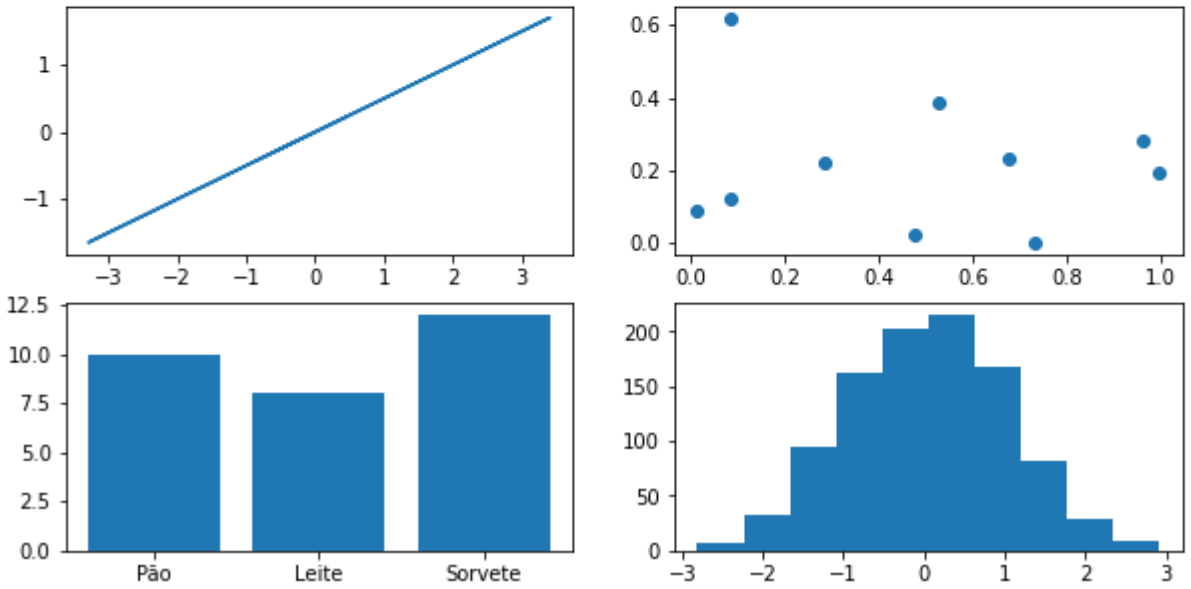
```
x = np.random.randn(1000)


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



Subplots

Desenhamos os dados em casa *axis* da figura:

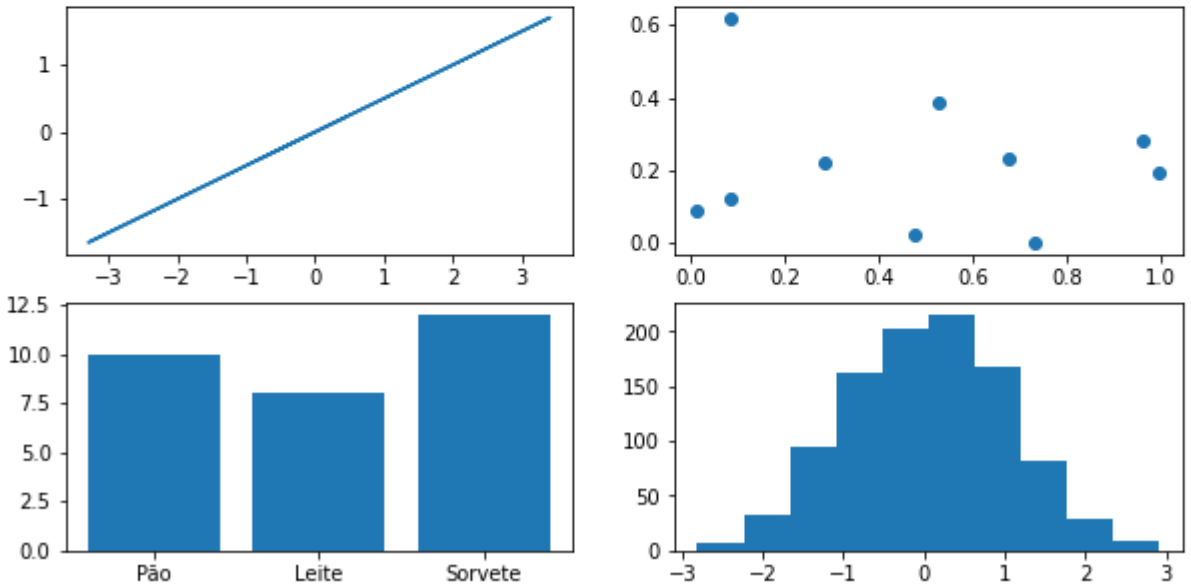
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2,
                                             ncols=2,
                                             figsize=(10, 5))
ax1.plot(x, x/2); #line
ax2.scatter(np.random.random(10), np.random.random(10)) #scatter
ax3.bar(produtos.keys(), produtos.values())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

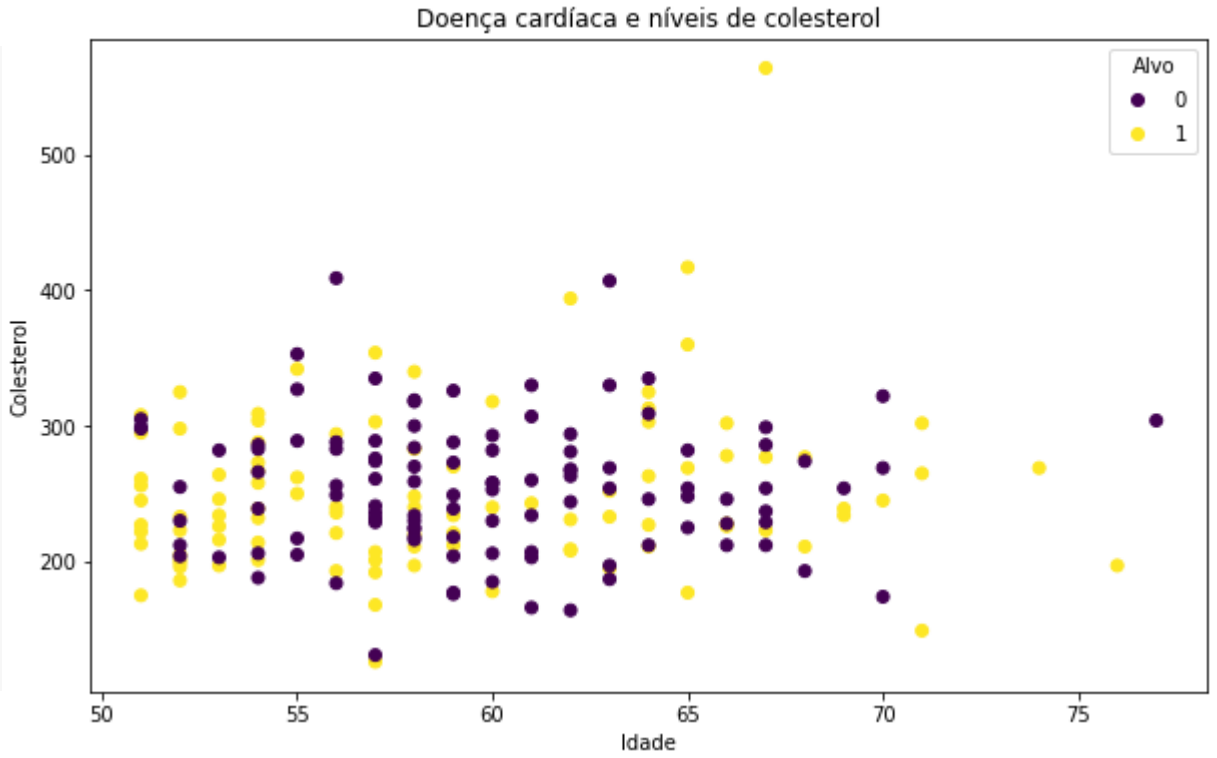
```
df = pd.read_csv("heart-disease.csv")
df.head()
```

	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)

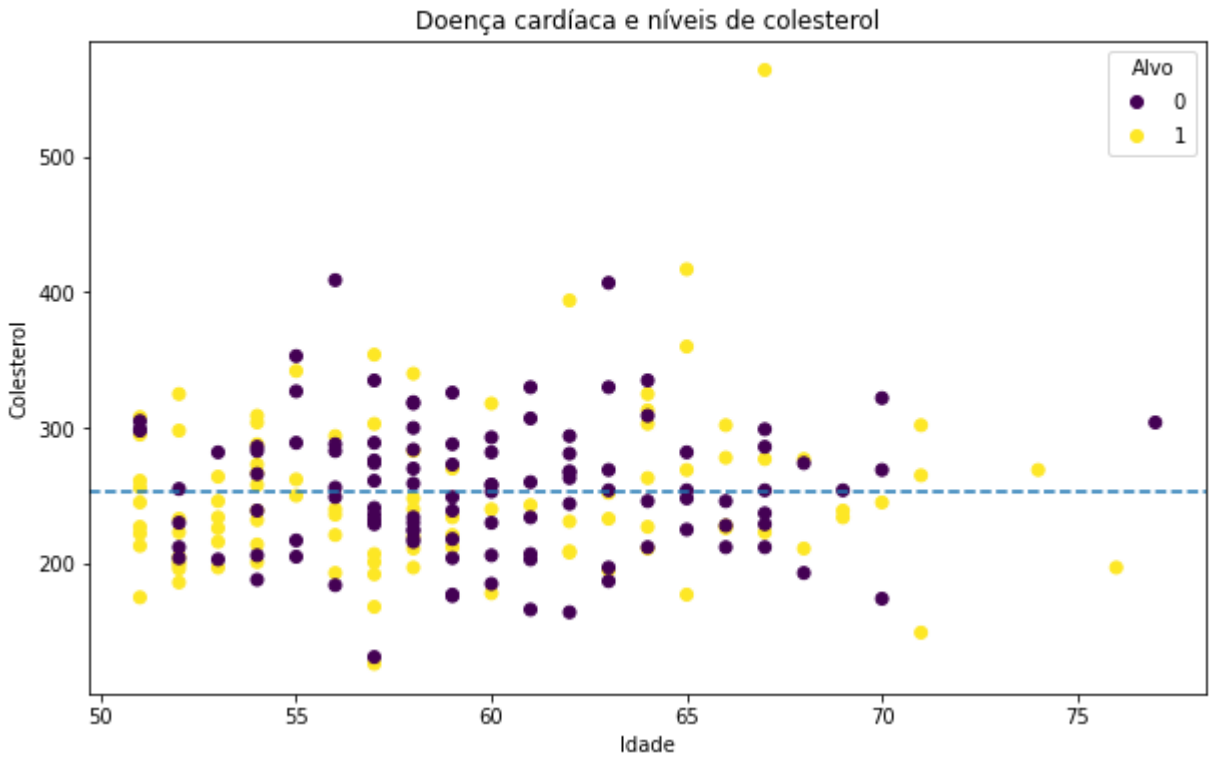
Na configuração dos axes ou escolhemos o tipo de gráfico scatter e preenchemos os valores dos eixos, x e y inserindo respetivamente as colunas de idade e colesterol

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="--");
```



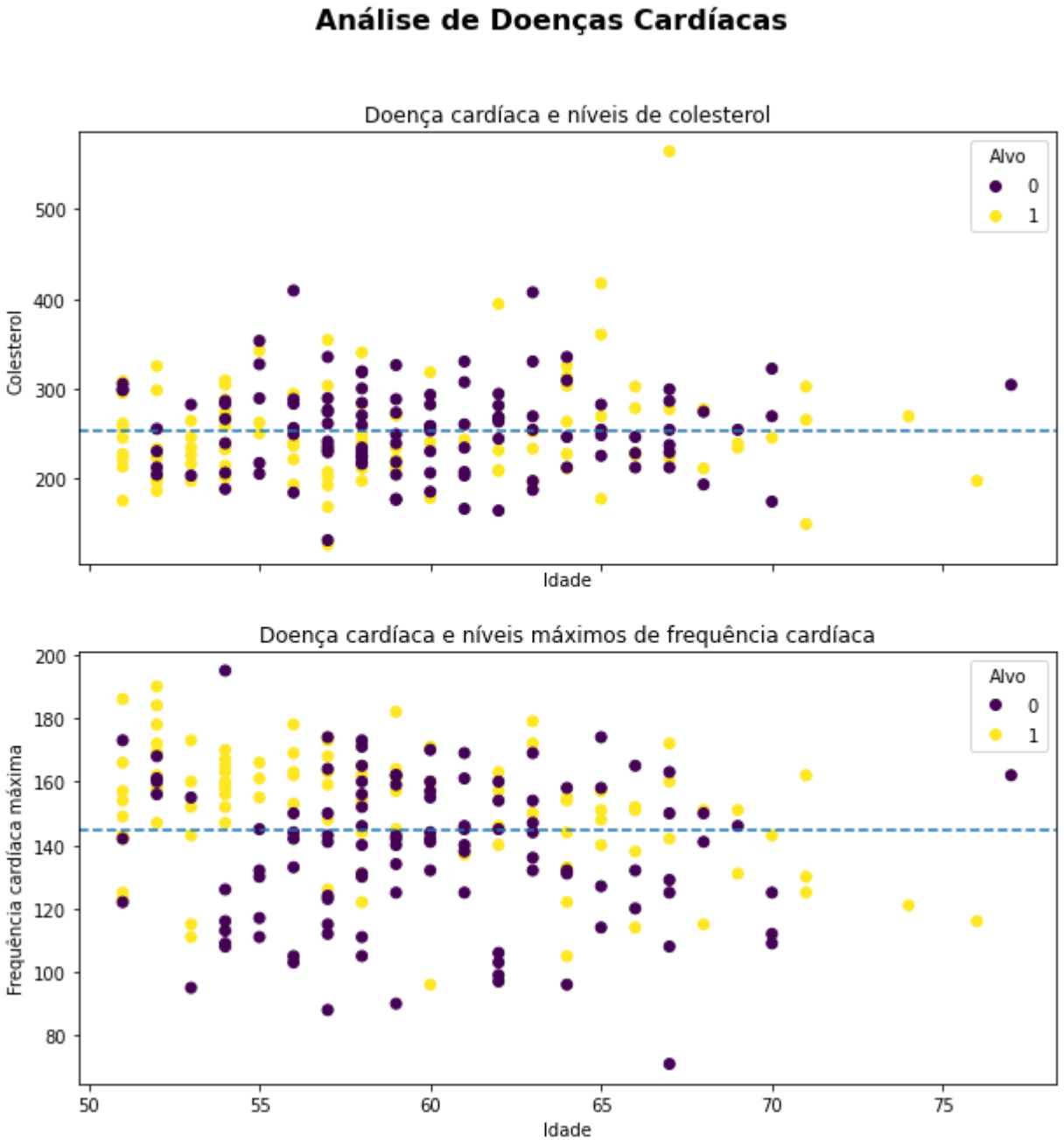
a função axhline() gera uma linha horizontal, para os dados que lhe são passados

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

```
# 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="--")
```

```
# 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

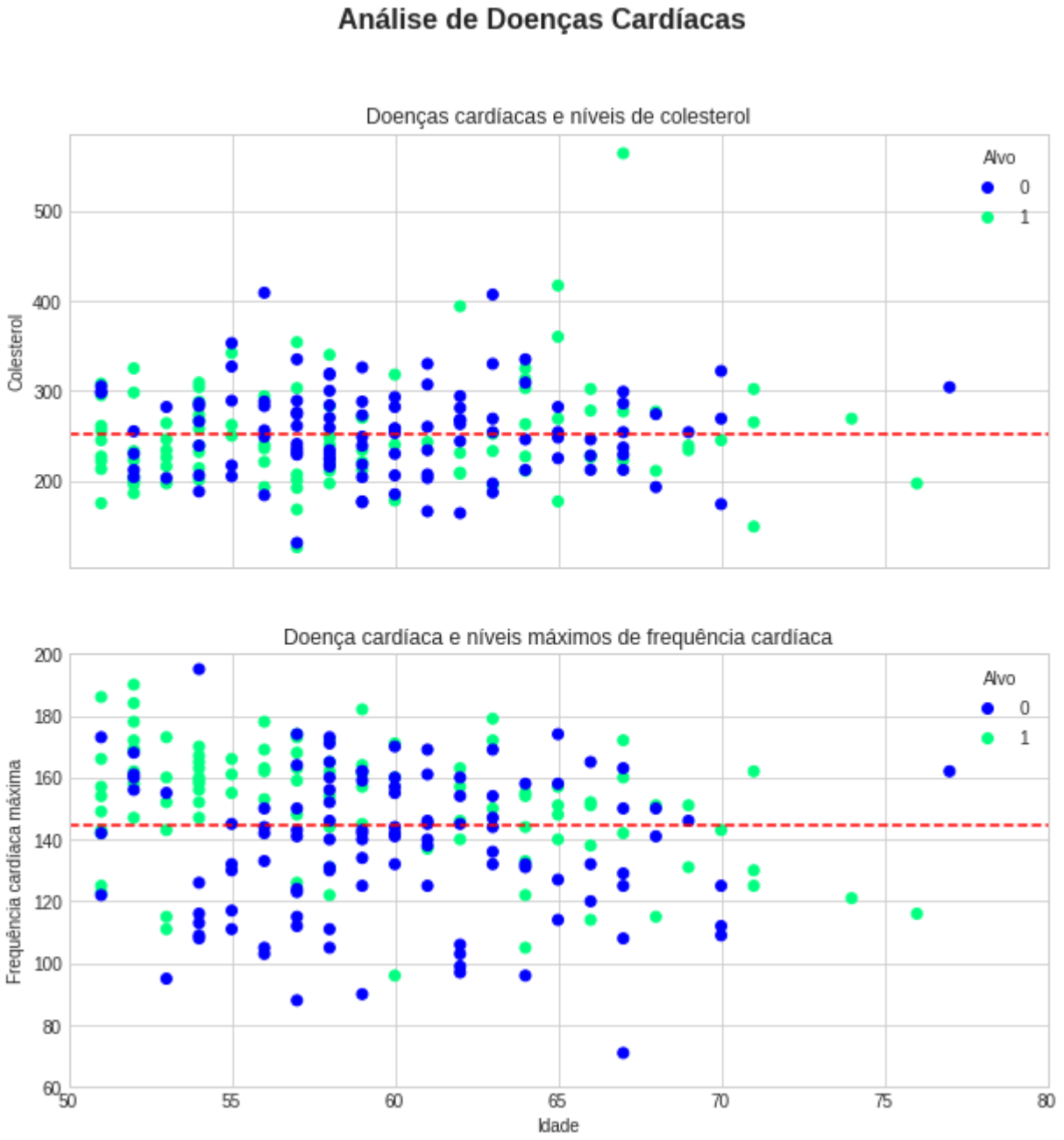
A opção *style.use("seaborn-whitegrid") oferece um novo tema ao gráfico*

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