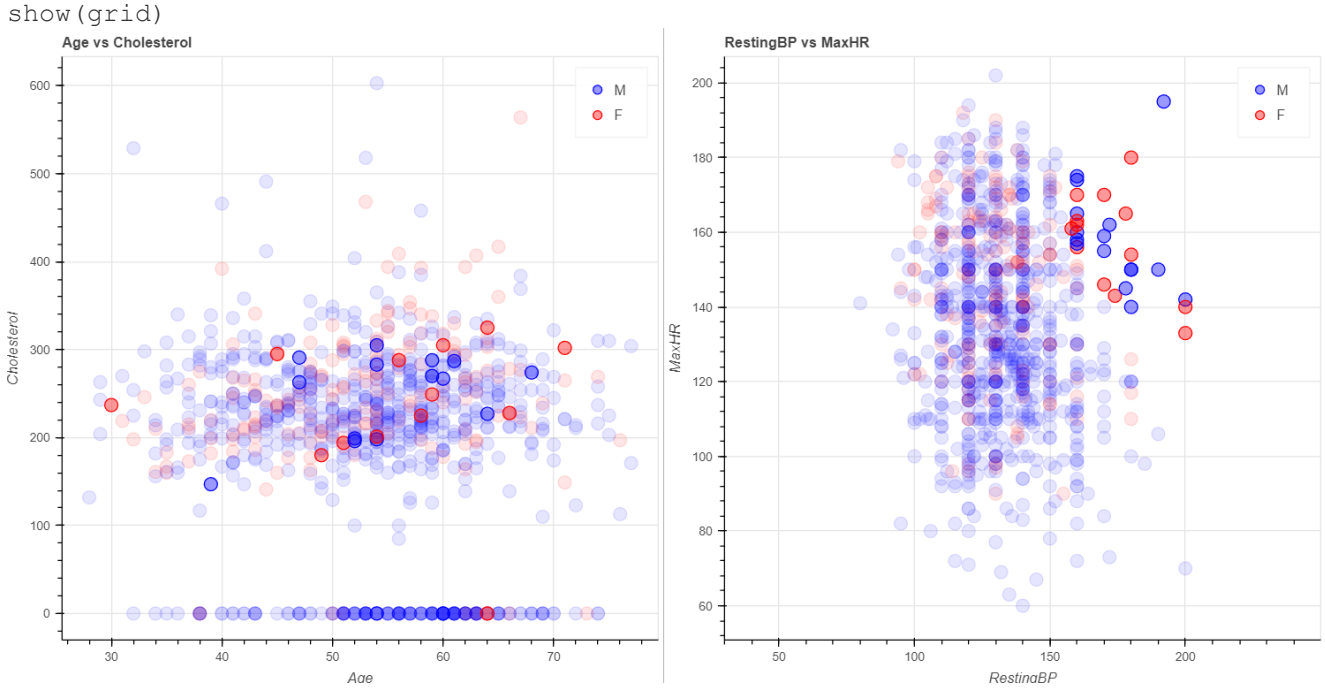
Exercício 4.1

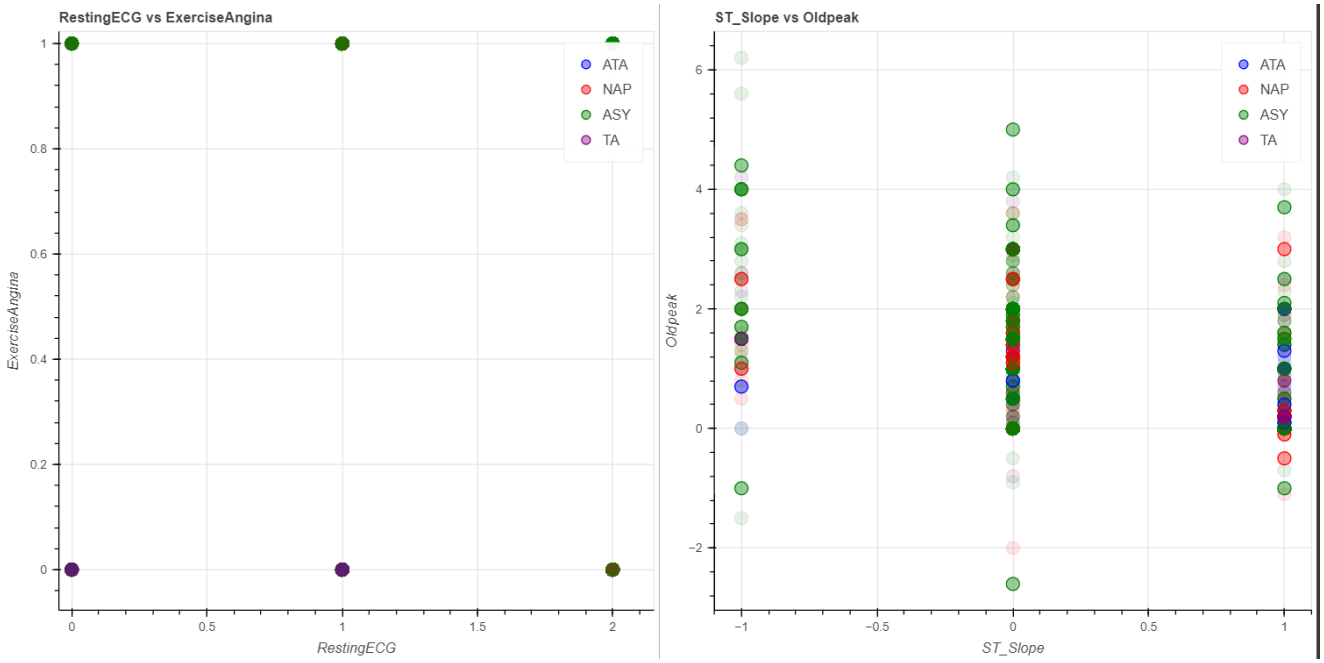
```
heart = pd.read_csv('/content/heart.csv')
import plotly.graph_objects as go

data = heart.head(5)
categories = ["RestingBP", "Cholesterol", "MaxHR"]

fig = go.Figure()

for index, row in data.iterrows():
    values = [row[var] for var in categories]
    fig.add_trace(go.Scatterpolar(r=values, theta=categories, fill='toself',
name=f'Paciente {index + 1}', opacity=0.3))

fig.show()
```

Exercício 4.2

```
from bokeh.layouts import gridplot
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource
from bokeh.transform import factor_cmap, factor_mark

colors = factor_cmap('Sex', palette=['blue', 'red'], factors=['M', 'F'])
source = ColumnDataSource(heart)

p1 = figure(title='Age vs Cholesterol',
tools='box_select,lasso_select,box_zoom,pan')
p1.xaxis.axis_label = 'Age'
p1.yaxis.axis_label = 'Cholesterol'
p1.scatter(x='Age', y='Cholesterol', source=source, legend_field='Sex',
           fill_alpha=0.4, size=12, color=colors)

p2 = figure(title='RestingBP vs MaxHR',
tools='box_select,lasso_select,box_zoom,pan')
p2.xaxis.axis_label = 'RestingBP'
p2.yaxis.axis_label = 'MaxHR'
p2.scatter(x='RestingBP', y='MaxHR', source=source, legend_field='Sex',
           fill_alpha=0.4, size=12, color=colors)

p1.legend.location = 'top_right'
p2.legend.location = 'top_right'

grid = gridplot([[p1, p2]])

show(grid)
```

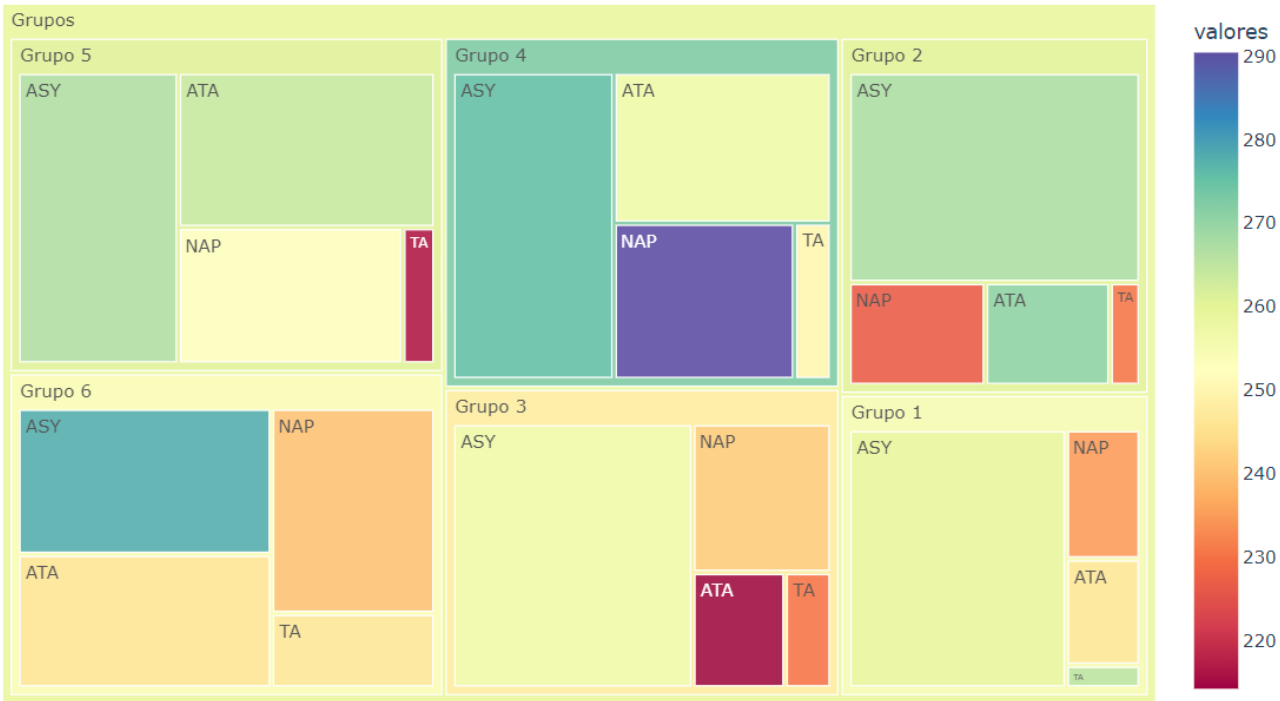


```
from bokeh.layouts import gridplot
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource
from bokeh.transform import factor_cmap, factor_mark

colors = factor_cmap('Sex', palette=['blue', 'red'], factors=['M', 'F'])
source = ColumnDataSource(heart)

p1 = figure(title='Age vs Cholesterol',
tools='box_select,lasso_select,box_zoom,pan')
p1.xaxis.axis_label = 'Age'
p1.yaxis.axis_label = 'Cholesterol'
p1.scatter(x='Age', y='Cholesterol', source=source, legend_field='Sex',
           fill_alpha=0.4, size=12, color=colors)

p2 = figure(title='RestingBP vs MaxHR',
tools='box_select,lasso_select,box_zoom,pan')
p2.xaxis.axis_label = 'RestingBP'
p2.yaxis.axis_label = 'MaxHR'
p2.scatter(x='RestingBP', y='MaxHR', source=source, legend_field='Sex',
           fill_alpha=0.4, size=12, color=colors)

p1.legend.location = 'top_right'
p2.legend.location = 'top_right'

grid = gridplot([[p1, p2]])

show(grid)
```



Exercício 4.3

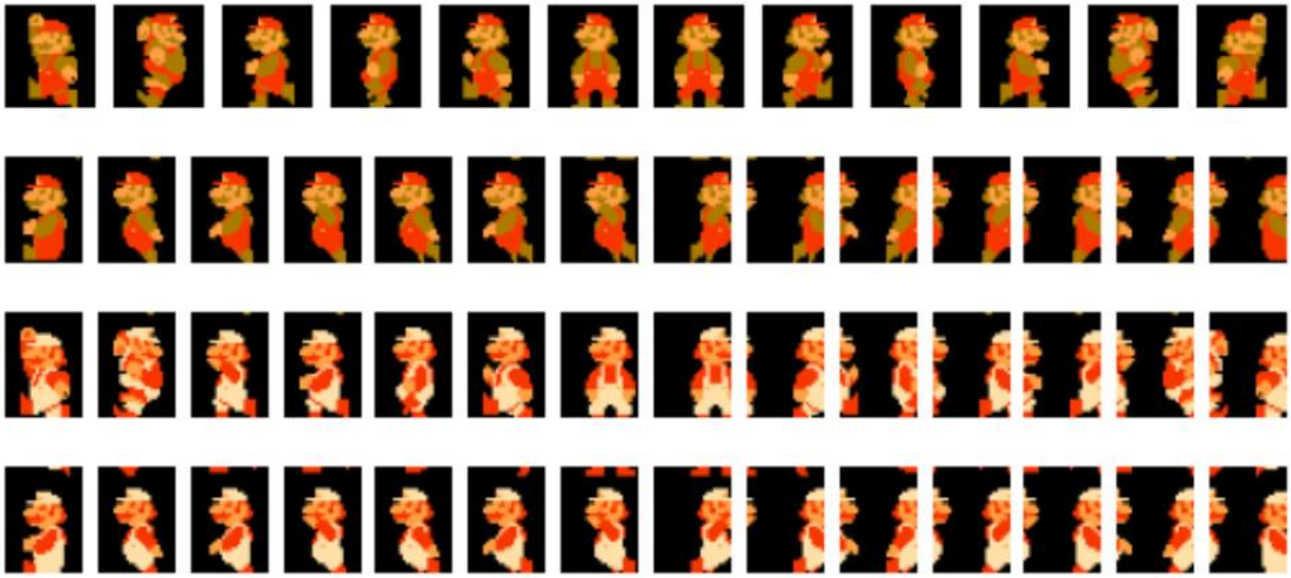
```
import plotly.express as px
import pandas as pd
import plotly.io as pio

from sklearn.linear_model import LinearRegression

modelo = LinearRegression()
modelo.fit(heart[['RestingBP', 'MaxHR']], heart['Cholesterol'])

previsoes = modelo.predict(heart[['RestingBP', 'MaxHR']])
heart['Grupo'] = pd.qcut(previsoes, q=6, labels=['Grupo 1', 'Grupo 2', 'Grupo 3',
'Grupo 4', 'Grupo 5', 'Grupo 6'])

df = heart
setores = heart['ChestPainType']
valores = heart['Cholesterol']
categorias = heart['Grupo']
df = pd.DataFrame(dict(setores = setores, valores = valores, categorias =
categorias))
df['Grupos'] = 'Grupos'
fig = px.treemap(df, path = ['Grupos', 'categorias', 'setores'],
values = 'valores', color_continuous_scale = 'spectral', color = 'valores')
fig.update_traces(root_color = 'lightgrey', opacity = 0.9)
fig.update_layout(margin = dict(t = 25, l = 25, r = 25, b = 25))
fig.show()
```



Exercício 4.4

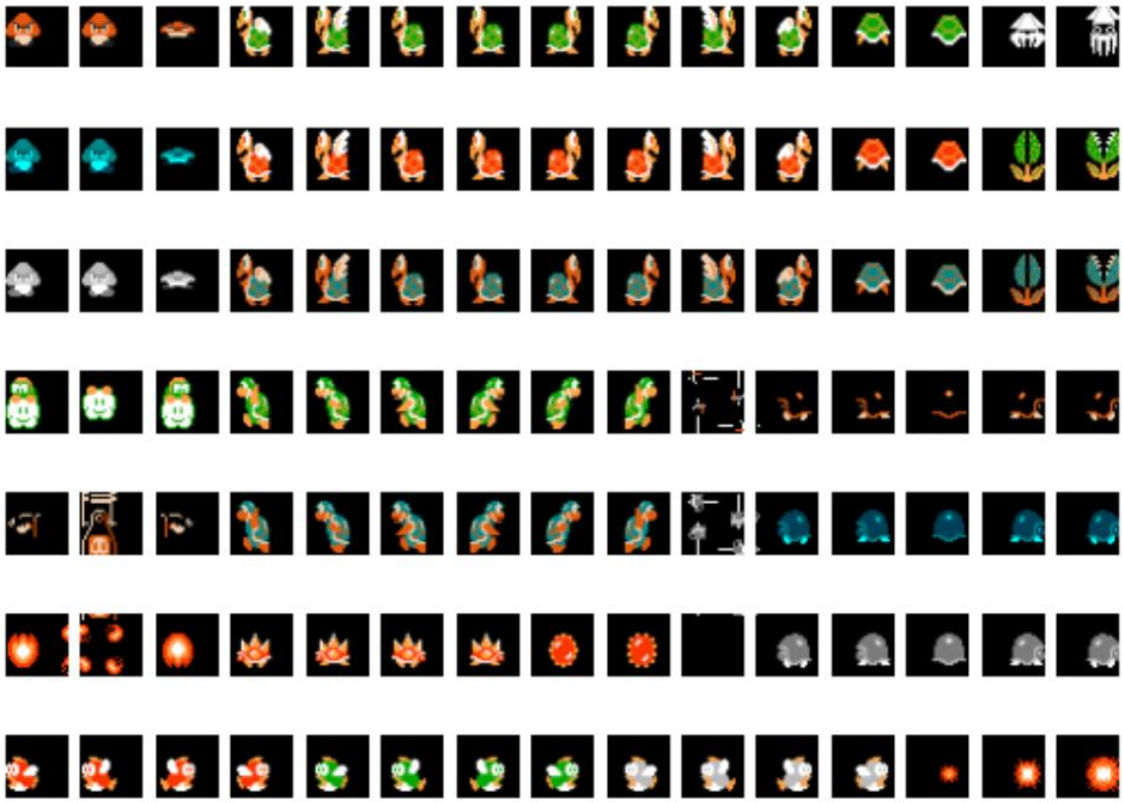
```
marios = Image.open('mario sheet.png').convert('RGB')

marios = np.asarray(marios)
plt.imshow(marios)

altura, largura, _ = marios.shape
intervalos = 4
altura_intervalo = altura // intervalos

imagens_cortadas = []

for i in range(intervalos):
    y1 = i * altura_intervalo
    y2 = (i + 1) * altura_intervalo
    imagem_cortada = marios[y1:y2, :, :]
    imagens_cortadas.append(imagem_cortada)

def dividir_verticalmente(imagem, intervalos):
    altura, largura, _ = imagem.shape
    largura_intervalo = largura // intervalos
    imagens_divididas = []
    for i in range(intervalos):
        x1 = i * largura_intervalo
        x2 = (i + 1) * largura_intervalo
        imagem_dividida = imagem[:, x1:x2, :]
        imagens_divididas.append(imagem_dividida)
    return imagens_divididas


imagem1_dividida = dividir_verticalmente(imagens_cortadas[0], 12)
imagens2_divididas = dividir_verticalmente(imagens_cortadas[1], 14)
imagens3_divididas = dividir_verticalmente(imagens_cortadas[2], 14)
imagens4_divididas = dividir_verticalmente(imagens_cortadas[3], 14)


for i, imagem in enumerate(imagem1_dividida):
    plt.subplot(1, len(imagem1_dividida), i + 1)
    plt.imshow(imagem)
    plt.axis('off')

plt.show()

for i, imagem in enumerate(imagens2_divididas):
    plt.subplot(1, len(imagens2_divididas), i + 1)
    plt.imshow(imagem)
    plt.axis('off')

plt.show()

for i, imagem in enumerate(imagens3_divididas):
    plt.subplot(1, len(imagens3_divididas), i + 1)
```

```
    plt.imshow(imagem)
    plt.axis('off')

plt.show()

for i, imagem in enumerate(imagens4_divididas):
    plt.subplot(1, len(imagens4_divididas), i + 1)
    plt.imshow(imagem)
    plt.axis('off')

plt.show()
```



```
todas_as_imagens = imagem1_dividida + imagens2_divididas + imagens3_divididas +
imagens4_divididas
arrays_de_imagens = [np.asarray(imagem) for imagem in todas_as_imagens]
df = pd.DataFrame({'Imagens': arrays_de_imagens})
df['Resposta'] = "Mario"

import numpy as np


intervalo = range(30, 100, 1)
melhor_acuracia = 0
melhor_limiar_media = None
melhor_limiar_desvio_padrao = None

for limiar_media in intervalo:
    for limiar_desvio_padrao in intervalo:
        df['Classificacao'] = df['Imagens'].apply(classificar_como_mario,
args=(limiar_media, limiar_desvio_padrao))
        acuracia = sum(df['Resposta'] == df['Classificacao']) / len(df)

        if acuracia > melhor_acuracia:
```

```
            melhor_acuracia = acuracia
            melhor_limiar_media = limiar_media
            melhor_limiar_desvio_padrao = limiar_desvio_padrao

def classificar_como_mario(imagem, limiar_media, limiar_desvio_padrao):
    media = np.mean(imagem)
    desvio_padrao = np.std(imagem)

    if media > melhor_limiar_media and desvio_padrao > melhor_limiar_desvio_padra:
        return "Mario"
    else:
        return "Não Mario"

enemies = Image.open('enemies sheet.png').convert('RGB')

enemies = np.asarray(enemies)
plt.imshow(enemies)

altura, largura, _ = enemies.shape
intervalos_horizontais = 7
intervalos_verticais = 15
altura_intervalo = altura // intervalos_horizontais
largura_intervalo = largura // intervalos_verticais
imagens_cortadas_horizontais = []
imagens_cortadas_enemies = []

for i in range(intervalos_horizontais):
    y1 = i * altura_intervalo
    y2 = (i + 1) * altura_intervalo
    imagem_cortada = enemies[y1:y2, :, :]
    imagens_cortadas_horizontais.append(imagem_cortada)

for imagem in imagens_cortadas_horizontais:
    for i in range(intervalos_verticais):
        x1 = i * largura_intervalo
        x2 = (i + 1) * largura_intervalo
        imagem_cortada = imagem[:, x1:x2, :]
        imagens_cortadas_enemies.append(imagem_cortada)

for i, imagem in enumerate(imagens_cortadas_enemies):
    plt.subplot(intervalos_horizontais, intervalos_verticais, i + 1)
    plt.imshow(imagem)
    plt.axis('off')
plt.show()
```



```
df_inimigos = pd.DataFrame({'Imagens': imagens_cortadas_enemies})
df_inimigos['Resposta'] = "Não Mario"
df = pd.concat([df, df_inimigos], axis=0)
df['Classificacao'] = df['Imagens'].apply(classificar_como_mario)
```

```
import matplotlib.pyplot as plt

def exibir_imagens_grid(df):
    imagens_mario = df[df['Resposta'] == 'Mario']['Imagens']
    imagens_nao_mario = df[df['Resposta'] == 'Não Mario']['Imagens']
    respostas = df['Resposta'] == df['Classificacao']

    num_imagens = len(df)
    num_colunas = min(5, num_imagens)
    num_linhas = (num_imagens + num_colunas - 1) // num_colunas

    fig, axs = plt.subplots(num_linhas, num_colunas, figsize=(15, 10))

    for i, imagem in enumerate(imagens_mario):
        linha = i // num_colunas
        coluna = i % num_colunas
        axs[linha, coluna].imshow(imagem)
        axs[linha, coluna].set_title("Mario", loc="center", x=-4.0, y=-.3)
        axs[linha, coluna].axis('off')
        if respostas.iloc[i] == True:
            axs[linha, coluna].text(30, 27, "Correto", fontsize=10, color='green')
```

```
    for i, imagem in enumerate(imagens_nao_mario):
        linha = (i + len(imagens_mario)) // num_colunas
        coluna = (i + len(imagens_mario)) % num_colunas
        axs[linha, coluna].imshow(imagem)
        axs[linha, coluna].set_title("Não Mario", loc="center", x=-4.0, y=-.3)
        axs[linha, coluna].axis('off')
        if respostas.iloc[i + len(imagens_mario)] == True:
            axs[linha, coluna].text(30, 27, "Correto", fontsize=10, color='green')

    plt.subplots_adjust(left=0., right=0.9, wspace=0.4)
    porcentagem_corretas = sum(df['Resposta'] == df['Classificacao']) / len(df)
    plt.figtext(0.1, .92, f"Porcentagem de respostas corretas:
{porcentagem_corretas:.2%}", fontsize=12)
    plt.show()

exibir_imagens_grid(df)
```