Revision: 09.03.2024

ASNI Med, Phase II

Revision: 09.03.2024, Stand: 13.02.2024, geändert am: 18.02.2024

Gut: https://www.kaggle.com/code/azizozmen/heart-failure-predict-8-classification-techniques

Analyse: https://www.kaggle.com/code/bennyfung/heart-failure-ensemble-by-voting-auc-85

Analyse: https://www.kaggle.com/code/arezoodahesh/heart-failure-prediction-with-ensemble-models

Analyse: https://www.kaggle.com/code/mohamedwasef/heart-disease-eda-with-ml

Analyse: https://www.kaggle.com/code/akashkotal/heart-disease-eda-with-7-machine-learning-model

Analyse: https://www.kaggle.com/code/gaganmaahi224/eda-detailed-explanation-of-knn-algorithm

Analyse: https://www.kaggle.com/code/pythonafroz/eda-heart-disease-prediction-roc-pr-curve

+EDA: https://www.kaggle.com/code/madhurpant/heart-disease-eda

+EDA: https://www.kaggle.com/code/alexandermaklak/eda-randomforest-88-heart-disease/edit

-EDA: https://www.kaggle.com/code/priyaaggarwal23/heart-failure-prediction-eda

-EDA: https://www.kaggle.com/code/taichiuemura/heart-failure-prediction-eda

+-: https://www.kaggle.com/code/gaganmaahi224/heart-failure-85-accuracy-eda-plotly-visualisation

+-: https://www.kaggle.com/code/khsamaha/heart-failure-dataset-eda-py

Gut in Rennen: https://www.kaggle.com/code/aletbm/cardiovascular-diseases-eda-modeling

Gut-: https://www.kaggle.com/code/varunsaikanuri/heart-failure-analysis-and-prediction

Analyse: https://www.kaggle.com/code/ohseokkim/heart-disease-could-our-model-save-lives

GridSearchCV + KFold CV: The Right Way: https://www.kaggle.com/code/marcinrutecki/gridsearchcv-kfold-cv-the-right-way

Outlier detection methods-: https://www.kaggle.com/code/marcinrutecki/outlier-detection-methods

----

0: https://www.kaggle.com/code/ritikaxg/heart-disease-eda

1: https://www.kaggle.com/code/aathikm/heart-failure-eda-data-visualization

1: https://www.kaggle.com/code/asifpervezpolok/heart-failure-prediction-with-eda

1+: https://www.kaggle.com/code/hendabdelnasser/heart-failure-prediction-model-knn-lr

0: https://www.kaggle.com/code/abdelkreem1mahmoud2/the-best-model-for-predicting-heart-disease

0: https://www.kaggle.com/code/minaemil329/heartdisease-analysis-visualization-and-classify

0: https://www.kaggle.com/code/syifanurlaila/heart-failure-prediction

0: https://www.kaggle.com/code/sonalisingh1411/automatic-exploratory-data-analysis

1+ :https://www.kaggle.com/code/mehrdadsadeghi/heart-failure-prediction-eda-5-models-cv-score

Evaluation Metriken im Maschinenlernen, Gut: https://www.kaggle.com/code/pythonafroz/evaluation-metrics-used-in-machine-learning

Categorical to Numerical Encoding Methods: https://www.kaggle.com/code/pythonafroz/categorical-to-numerical-encoding-methods¶

Seeborn: https://www.kaggle.com/code/saurav9786/seaborn-tutorial

http://suanlab.com/assets/youtubes/dv/Seaborn.pdf https://www.analyticsvidhya.com/blog/2022/07/step-by-step-exploratory-data-analysis-eda-using-python/

Dist: https://plotly.com/python/distplot/

Confusion_matrix für Analyse gut: https://www.kaggle.com/code/arezoodahesh/heart-failure-prediction-with-ensemble-models

Confusion_matrix : https://medium.com/@szabo.bibor/how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e

file:///C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/OUTPUT-RESSOURCE/Report/EDA_ReportFinal/++EDA_ChartFinal20240226.html

```
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import cross_val_score
from collections import Counter

import plotly.graph_objects as go
import plotly.offline as po
from plotly.offline import download_plotlyjs, init_notebook_mode,
plot, iplot
import matplotlib.pyplot as plt
import dash

import random
import plotly.figure_factory as ff
from plotly import tools
from plotly.subplots import make_subplots
from plotly.offline import iplot
import warnings
warnings.filterwarnings("ignore")
pd.set_option("display.max_rows",None)
from sklearn import preprocessing
%matplotlib inline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import datasets, linear_model, metrics
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from IPython.core.display import HTML


path="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report"
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"
```

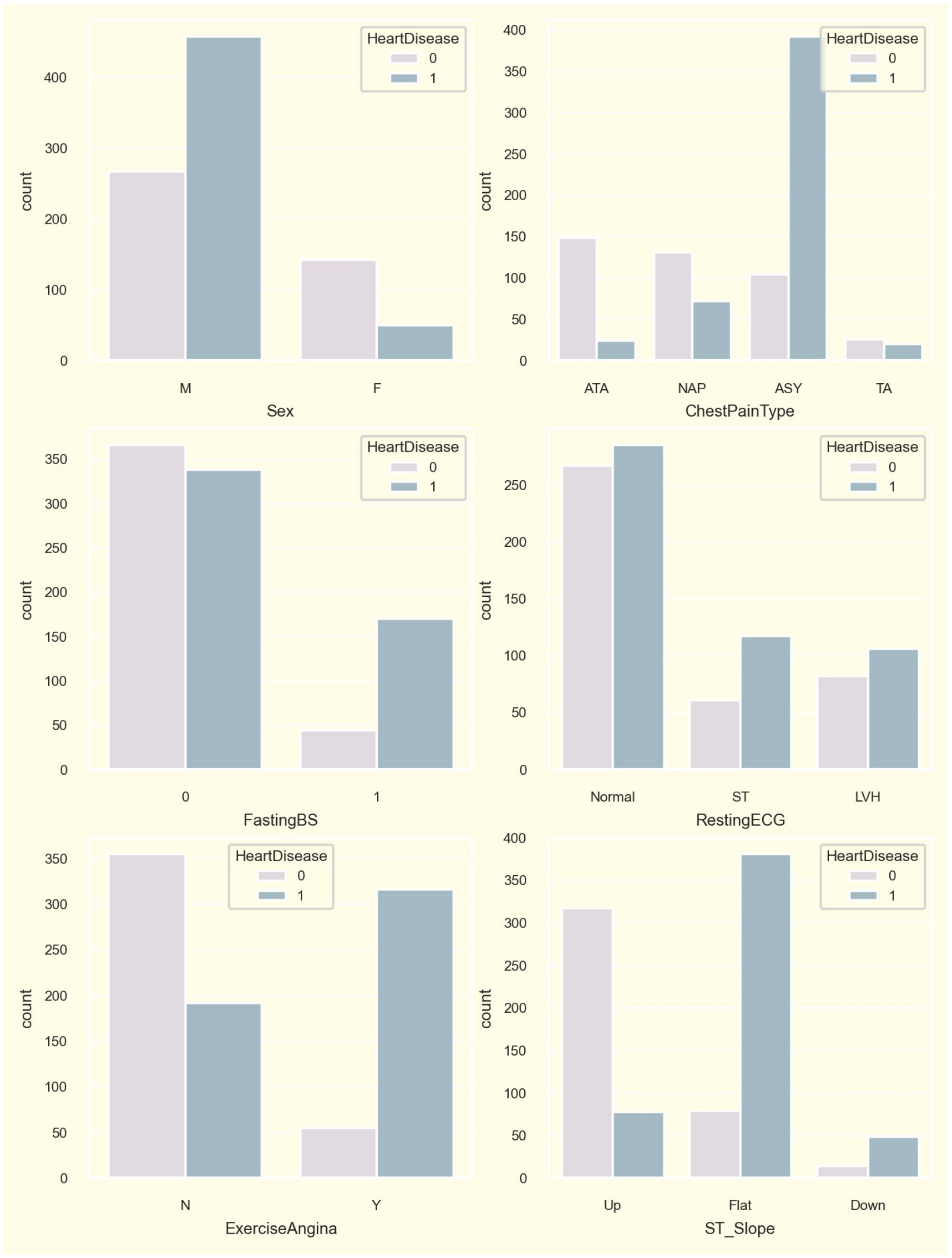
```
C:\Users\satur\anaconda3\Lib\site-packages\pandas\core\arrays\
masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of
'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

```
try:
    raw_df = pd.read_csv(path+'/data/heart.csv')
except:
    raw_df =pd.read_csv(path+'/data/heart.csv')

df=raw_df
he=raw_df.head(40)
#print(raw_df.head(10))

des0=raw_df[raw_df['HeartDisease']==0].describe().T.applymap('{:,.2f}'
.format)
des1=raw_df[raw_df['HeartDisease']==1].describe().T.applymap('{:,.2f}'
.format)

import plotly.io as pio
pio.renderers.default = "browser"
#pio.renderers
path="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report"
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"
fig = go.Figure(data=[go.Table(
                                 header=dict(
                                              values=list(df.columns),
# Header values
                                              line_color='black', #
Line Color of header
                                              fill_color='orange', #
background color of header
                                              align='center', # Align
header at center
                                              height=30, # Height of
Header
                                              font=dict(color='white',
size=12), # Font size & color of header text
                                             ), cells=dict(values=[
                                                     df.Age , # Column
values
                                                     df.Sex,
                                                     df.ChestPainType,
                                                     df.RestingBP,
                                                     df.Cholesterol,
                                                     df.FastingBS,
                                                     df.RestingECG,
                                                     df.MaxHR,

df.ExerciseAngina,
                                                     df.Oldpeak,
                                                     df.ST_Slope,
                                                     df.HeartDisease
                                                    ],line_color='dark
grey', # Line color of the cell

                                            fill_color='lightcyan',
                                            align='left',
                                            font=dict(color='black',
size=16),
                                            height=30
                                           )
                               )
                     ]
               )

fig.update_layout(width=1200, height=1800, title_text="Таблица1.
Исходные даные для выборочных пациентов", font=dict(color="darkred",
size=18))
fig.show()
pio.write_image(fig, pathIm+'/EDA1.png', format='png',scale=6)
```

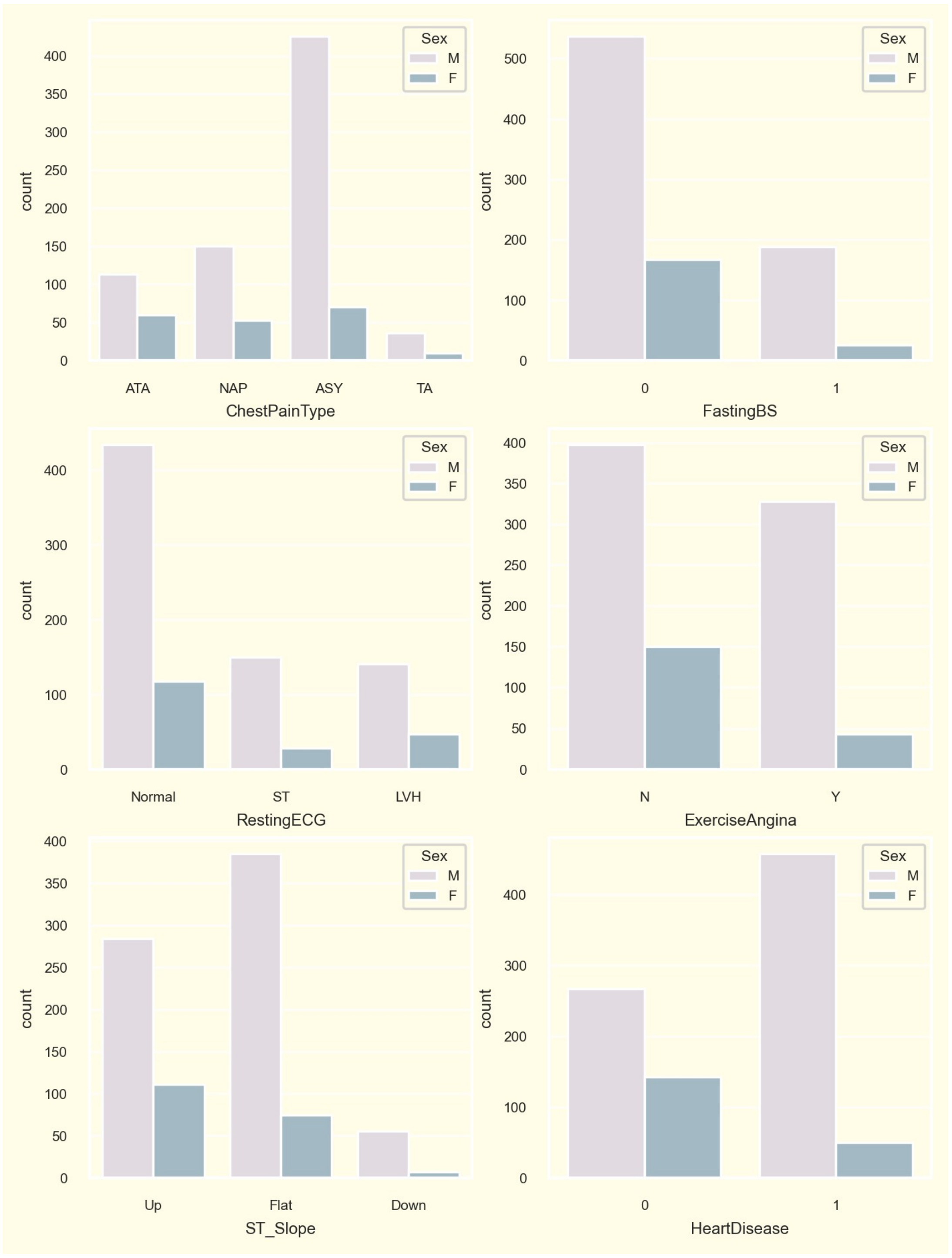
file:///C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/OUTPUT-RESSOURCE/Report/EDA_ChartFinal.html

```
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
fig=px.pie(raw_df,values='HeartDisease',names='ChestPainType',

template='plotly_dark',color_discrete_sequence=px.colors.sequential.Rd
Bu,
           title='The effect of the type of chest pain on the
disease')
fig.update_traces(textposition='inside',textinfo='percent+label')
fig.show()
fig.update_layout(width=1000, height=800)
fig.write_image(pathIm + '/EDA21.png', scale=4)

fig=px.pie(raw_df,values='HeartDisease',names='ST_Slope',hole=.4,templ
ate='plotly_dark',title='The effect of the the slope of the peak
exercise on the disease',)
fig.update_traces(textposition='inside',textinfo='percent+label')
fig.update_layout(annotations=[dict(text='ST slope', x=0.5, y=0.5,
font_size=20, showarrow=False)])
fig.show()
fig.update_layout(width=1000, height=1000)
fig.write_image(pathIm + '/EDA22.png',scale=4)

df=raw_df
colors = px.colors.cyclical.Twilight
fig = make_subplots(rows=1,cols=2,
                    subplot_titles=('Countplot',
                                    'Percentages'),
                    specs=[[{"type": "xy"},
                            {'type':'domain'}]])

fig.add_trace(go.Bar(y = df['Sex'].value_counts().values.tolist(),
                      x = df['Sex'].value_counts().index,
                      text=df['Sex'].value_counts().values.tolist(),
              textfont=dict(size=15),
                      textposition = 'outside',
                      showlegend=False,
              marker = dict(color = colors,
                            line_color = 'black',
                            line_width=3)),row = 1,col = 1)
fig.add_trace((go.Pie(labels=df['Sex'].value_counts().keys(),

values=df['Sex'].value_counts().values,textfont = dict(size = 16),
                     hole = .4,
                     marker=dict(colors=colors),
                     textinfo='label+percent',
                     hoverinfo='label')), row = 1, col = 2)
fig.update_yaxes(range=[0,800])
#Changing plot & figure background
fig.update_layout(
                    paper_bgcolor= '#FFFDE7',
                    plot_bgcolor= '#FFFDE7',
                    title=dict(text = "Gender
Distribution",x=0.5,y=0.95),
                    title_font_size=30
                  )
iplot(fig)
fig.write_image(pathIm + '/EDA23.png',scale=4)

colors = px.colors.cyclical.Twilight
fig = make_subplots(rows=1,cols=2,
                    subplot_titles=('Countplot',
                                    'Percentages'),
                    specs=[[{"type": "xy"},
                            {'type':'domain'}]])
fig.add_trace(go.Bar(y =
df['HeartDisease'].value_counts().values.tolist(),
                      x = df['HeartDisease'].value_counts().index,

text=df['HeartDisease'].value_counts().values.tolist(),
              textfont=dict(size=15),
                      textposition = 'outside',
                      showlegend=False,
              marker = dict(color = colors,
                            line_color = 'black',
                            line_width=3)),row = 1,col = 1)
fig.add_trace((go.Pie(labels=df['HeartDisease'].value_counts().keys(),

values=df['HeartDisease'].value_counts().values,textfont = dict(size =
16),
                     hole = .4,
                     marker=dict(colors=colors),
                     textinfo='label+percent',
                     hoverinfo='label')), row = 1, col = 2)
fig.update_yaxes(range=[0,550])
#Changing plot & figure background
fig.update_layout(
                    paper_bgcolor= '#FFFDE7',
                    plot_bgcolor= '#FFFDE7',
                    title=dict(text = "HeartDisease
Distribution",x=0.5,y=0.95),
                    title_font_size=30
                  )
iplot(fig)
fig.write_image(pathIm + '/EDA24.png',scale=4)

# Using facet_row and or facet_col arguments to create Sub plots
fig = px.scatter(df,
                 x=df.Age,
                 y=df.Cholesterol,
                 color=df.HeartDisease,
                 facet_col=df.FastingBS,
                 facet_row=df.Sex,
                 color_discrete_map={1: "#FF5722",0: "#7CB342"},
                 width=950,
                 height=800,
                 title="HeartDisease Data")

fig.update_layout(
                    plot_bgcolor= "#dcedc1",
                    paper_bgcolor="#FFFDE7",
                 )


fig.show()
fig.write_image(pathIm + '/EDA25.png',scale=4)
```

```
cat = ['Sex', 'ChestPainType','FastingBS','RestingECG',
                          'ExerciseAngina',
'ST_Slope','HeartDisease']
num = ['Age','RestingBP','Cholesterol','MaxHR','Oldpeak']

import seaborn as sns
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
'#fffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale = 0.55)
fig, ax = plt.subplots(3, 2, figsize = (6.5, 9))
for indx, (column, axes) in list(enumerate(list(zip(cat,
ax.flatten())))):
    if column not in 'HearDisease':
        sns.countplot(ax = axes, x = df[column], hue =
df['HeartDisease'], palette = colors, alpha = 1)
else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

axes_legend = ax.flatten()
axes_legend[1].legend(title = 'HeartDisease', loc = 'upper right')
axes_legend[2].legend(title = 'HeartDisease', loc = 'upper right')
plt.show()
fig.savefig(pathIm + '/EDA26.png')
```

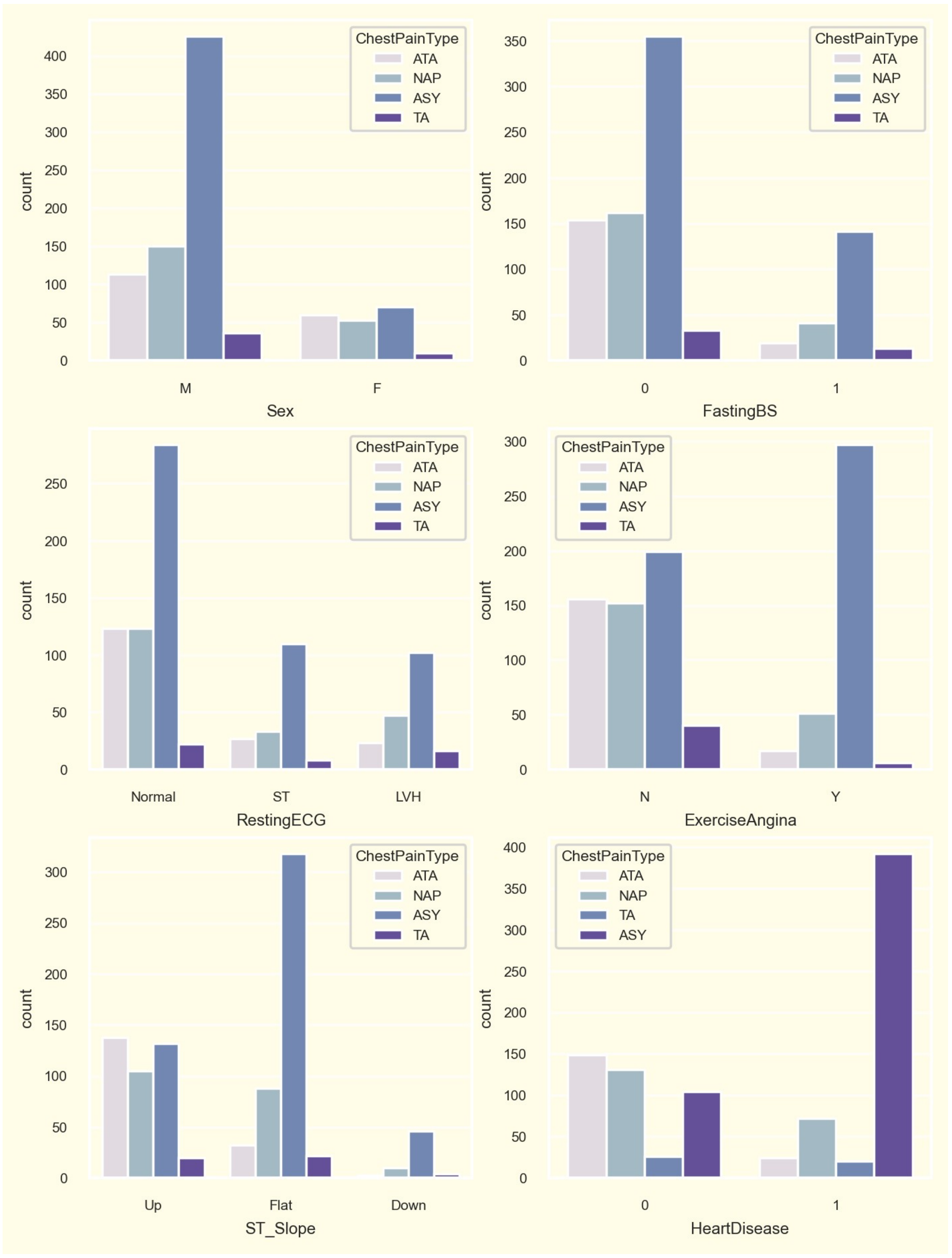


```
import seaborn as sns
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
'#fffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale = 0.55)
fig, ax = plt.subplots(3, 2, figsize = (6.5, 9))
for indx, (column, axes) in list(enumerate(list(zip(cat[1:],
ax.flatten())))):
    sns.countplot(ax = axes, x = df[column], hue = df['Sex'], palette
= colors, alpha = 1)
else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]
axes_legend = ax.flatten()
axes_legend[1].legend(title = 'Sex', loc = 'upper right')
axes_legend[2].legend(title = 'Sex', loc = 'upper right')
plt.show()
fig.savefig(pathIm + '/EDA27.png')
```



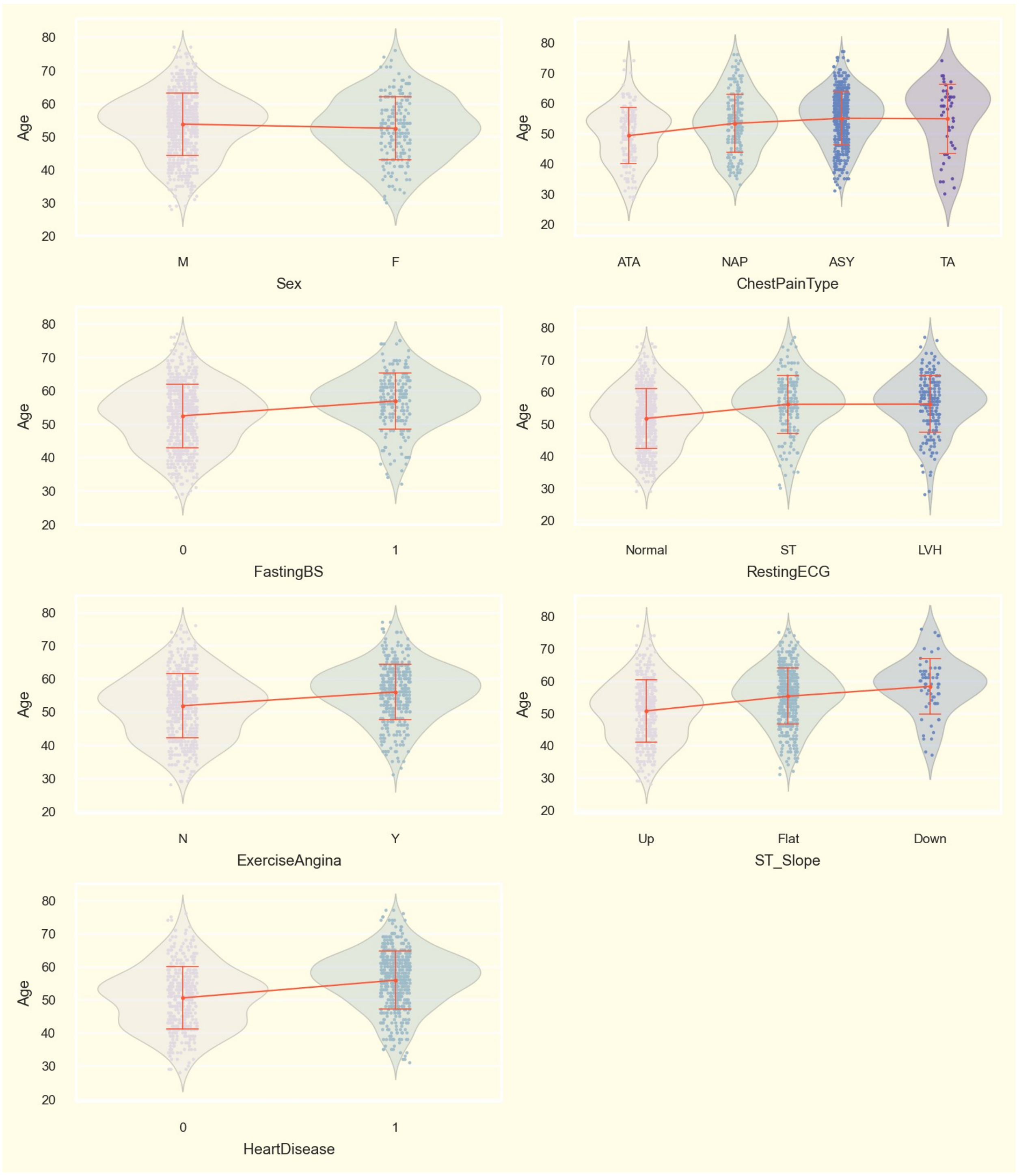
```
import seaborn as sns
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
'#fffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale = 0.55)
fig, ax = plt.subplots(3, 2, figsize = (6.5, 9))
cat2 = []
```

```python
for i in cat:
    if i not in 'ChestPainType':
        cat2.append(i)
for indx, (column, axes) in list(enumerate(list(zip(cat2,
ax.flatten())))):
    sns.countplot(ax = axes, x = df[column], hue =
df['ChestPainType'], palette = colors, alpha = 1)
else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]
axes_legend = ax.flatten()
axes_legend[1].legend(title = 'ChestPainType', loc = 'upper right')
axes_legend[2].legend(title = 'ChestPainType', loc = 'upper right')
plt.show()
fig.savefig(pathIm + '/EDA28.png')
```



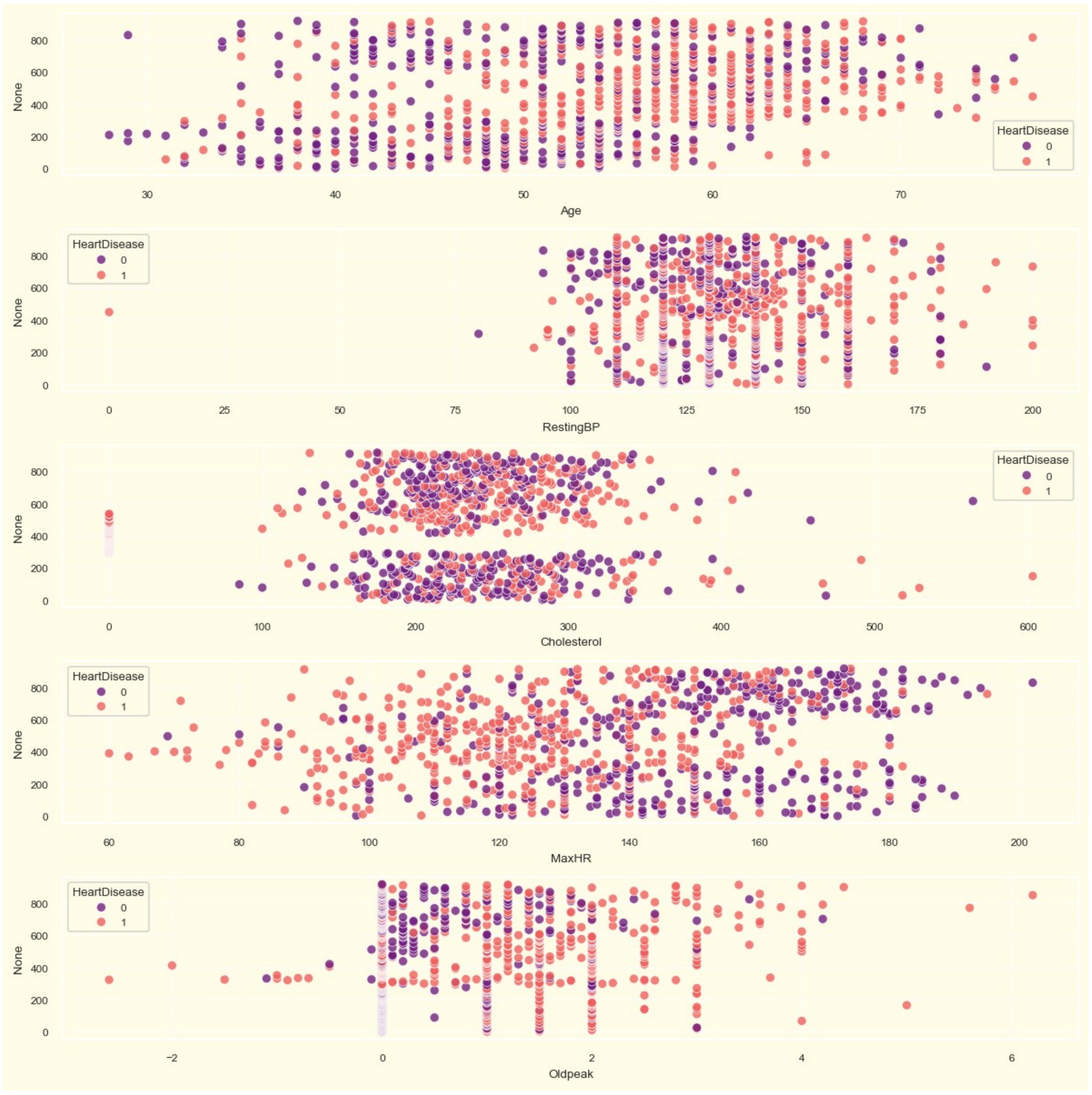
```
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
'#fffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale = 0.55)
fig, ax = plt.subplots(4, 2, figsize = (6.5, 7.5))
for indx, (column, axes) in list(enumerate(list(zip(cat,
ax.flatten())))):
```

```python
    sns.violinplot(ax = axes, x = df[column],
                   y = df['Age'],
                   scale = 'width', linewidth = 0.5,
                   palette = colors, inner = None)

    plt.setp(axes.collections, alpha = 0.3)

    sns.stripplot(ax = axes, x = df[column],
                  y = df['Age'],
                  palette = colors, alpha = 0.9,
                  s = 1.5, jitter = 0.07)
    sns.pointplot(ax = axes, x = df[column],
                  y = df['Age'],
                  color = '#ff5736', scale = 0.25,
                  estimator = np.mean, ci = 'sd',
                  errwidth = 0.5, capsize = 0.15, join = True)

    plt.setp(axes.lines, zorder = 100)
    plt.setp(axes.collections, zorder = 100)

else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA29.png')
```

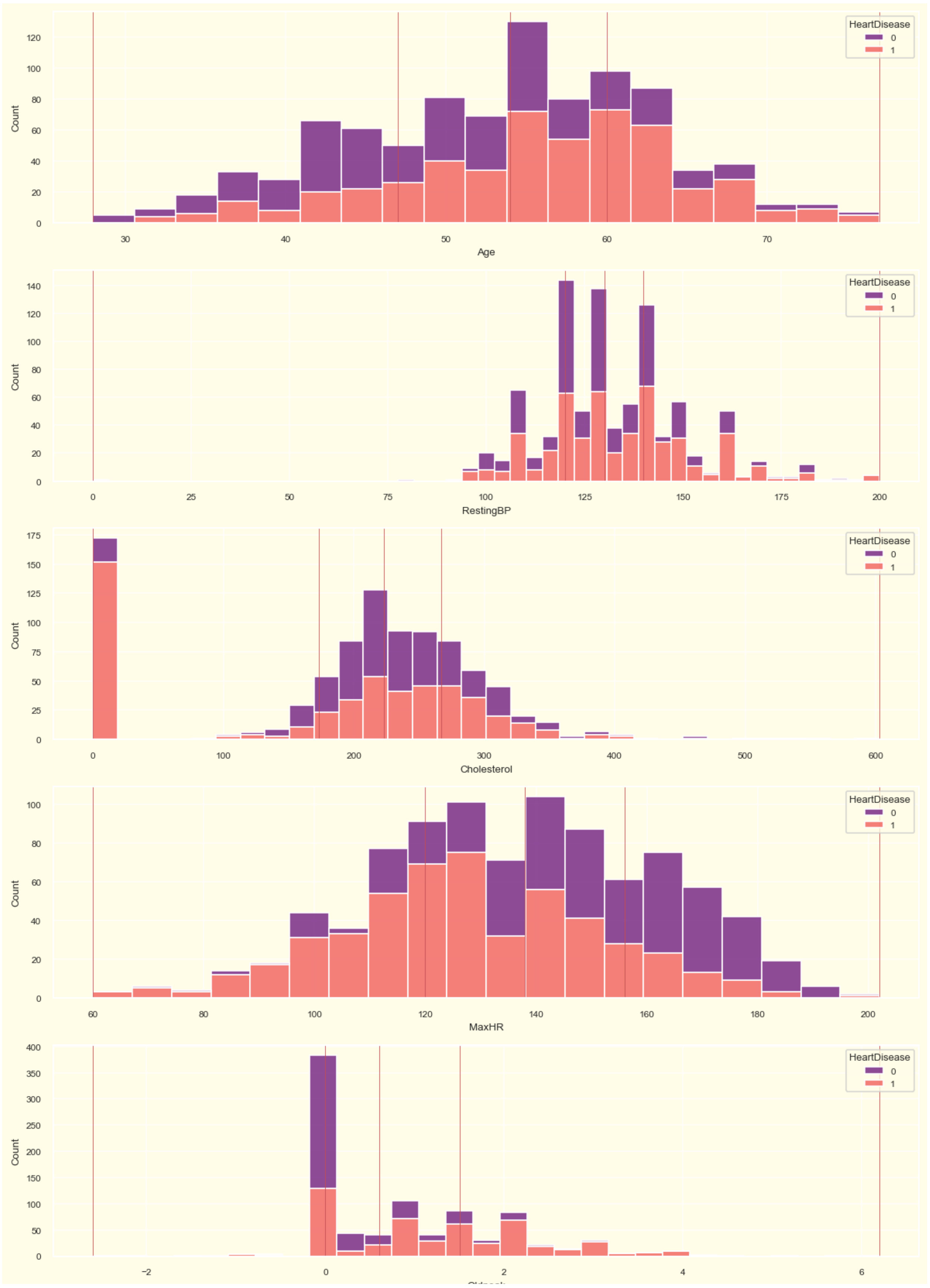


```
sns.set_theme(rc = {'figure.dpi': 120, 'axes.labelsize': 8,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
'#fffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale = 0.65)

fig, ax = plt.subplots(5, 1, figsize = (10, 10))
```

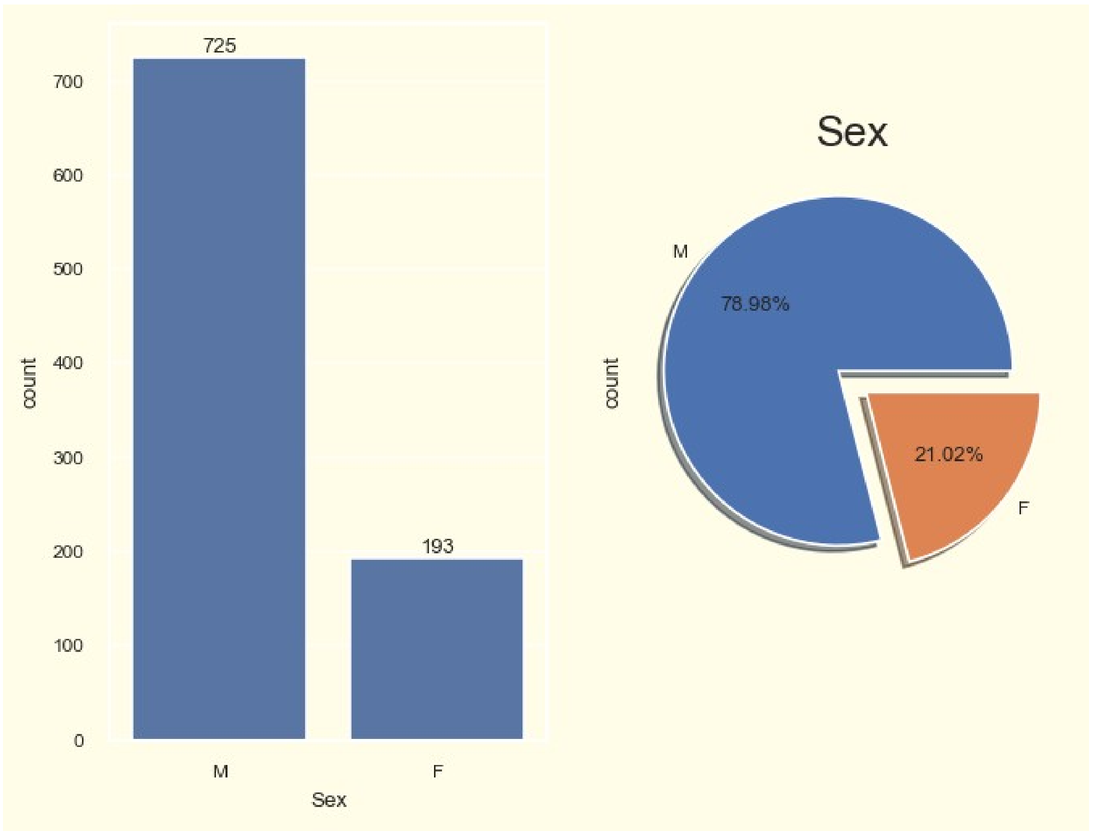
```
for indx, (column, axes) in list(enumerate(list(zip(num,
ax.flatten())))):

    sns.scatterplot(ax = axes, y = df[column].index, x = df[column],
                    hue = df['HeartDisease'], palette = 'magma', alpha
= 0.8)

else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA30.png')
```



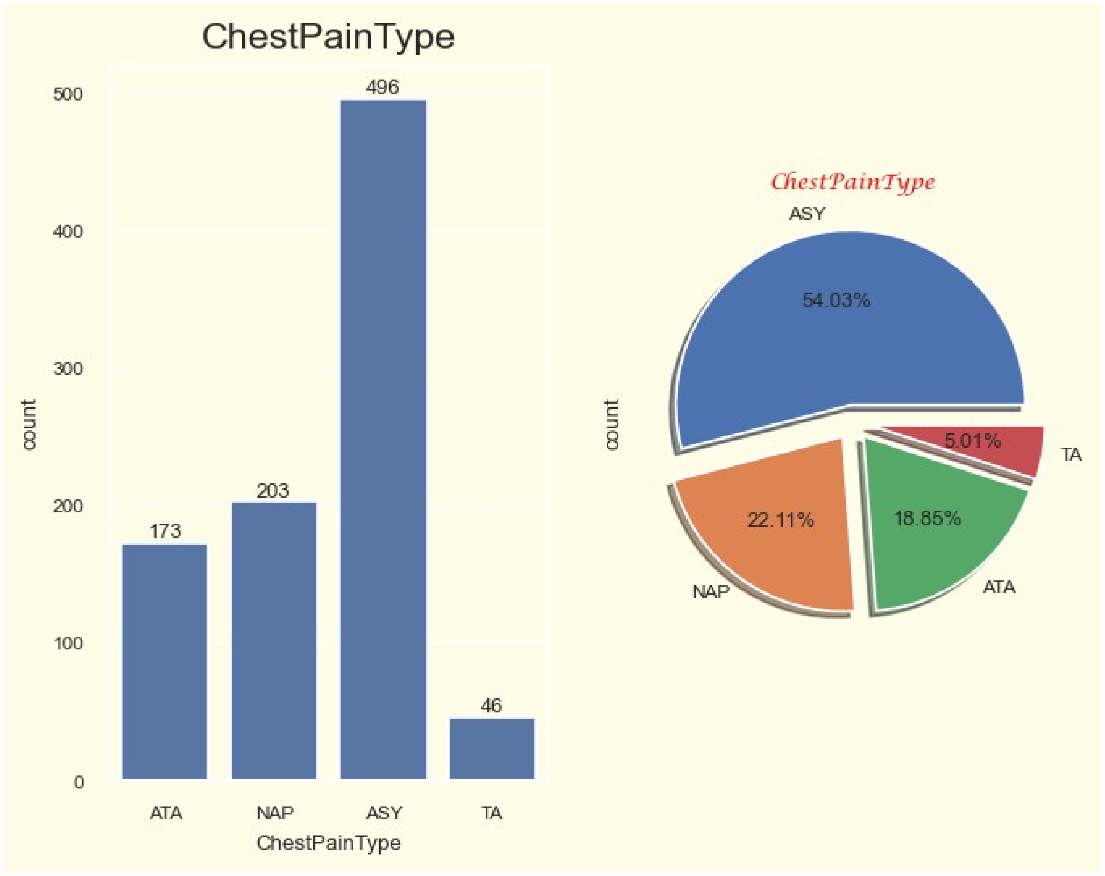
```
sns.set_theme(rc = {'figure.dpi': 120, 'axes.labelsize': 8,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
'#fffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale = 0.65)

fig, ax = plt.subplots(5, 1, figsize = (10, 14))

for indx, (column, axes) in list(enumerate(list(zip(num,
ax.flatten())))):

    sns.histplot(ax = axes, x = df[column], hue = df['HeartDisease'],
                 palette = 'magma', alpha = 0.8, multiple = 'stack')
    legend = axes.get_legend() # sns.hisplot has some issues with
legend
    handles = legend.legendHandles
    legend.remove()
    axes.legend(handles, ['0', '1'], title = 'HeartDisease', loc =
'upper right')
    Quantiles = np.quantile(df[column], [0, 0.25, 0.50, 0.75, 1])

    for q in Quantiles: axes.axvline(x = q, linewidth = 0.5, color =
'r')

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA31.png')
```


HeartDisease
0
1
Count
Age
RestingBP
Cholesterol
MaxHR

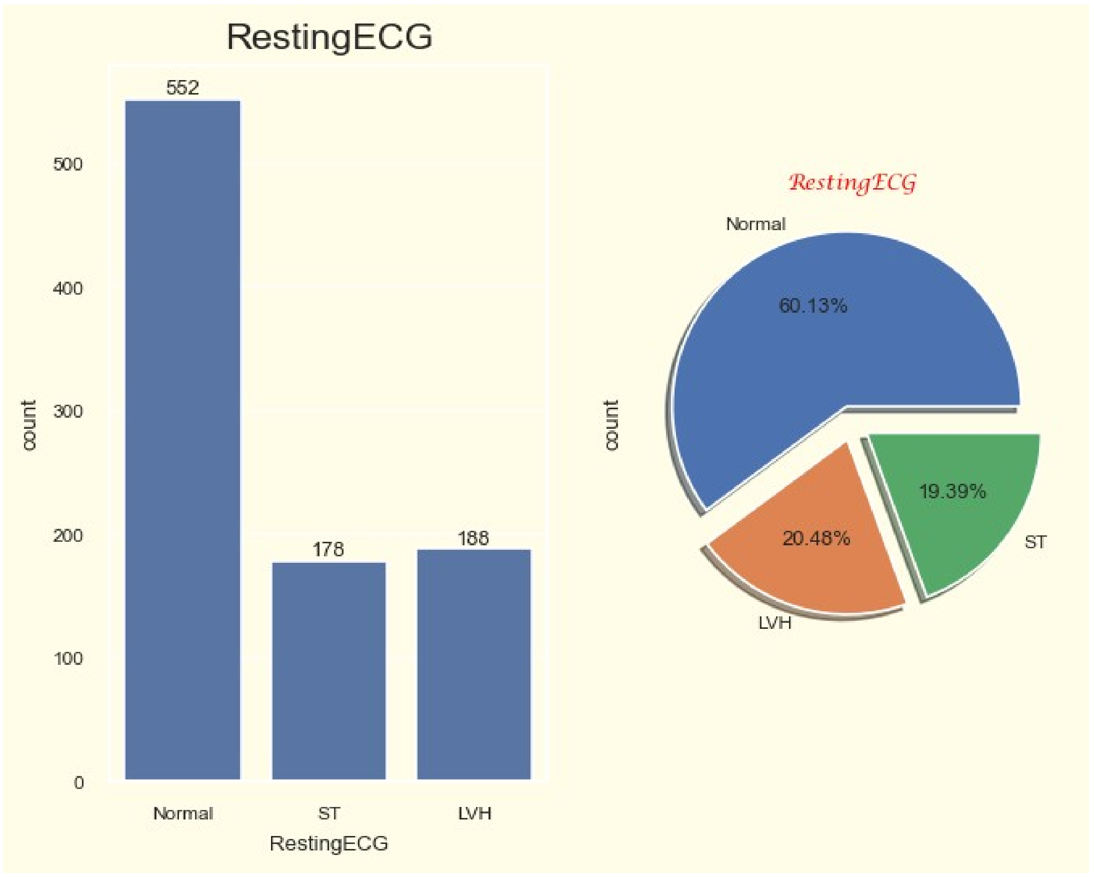
```
df2 = df.groupby('Sex').agg({'Age' : 'mean',
"ChestPainType":'count','RestingBP':'mean','Cholesterol':'mean',

'FastingBS':'sum','RestingECG':'count','MaxHR':'mean','ExerciseAngina'
:'count','Oldpeak':'mean',
                            'ST_Slope':'count','HeartDisease':'sum'})
df2
fig=px.bar(data_frame=df2, barmode='group',
       title = "<b>Gender wise Analyzing</b>",template="plotly_dark")
fig.show()
fig.write_image(pathIm + '/EDA32.png',scale=4)

ax = plt.subplot(1,2,1)
ax = sns.countplot(x='Sex', data=raw_df)
ax.bar_label(ax.containers[0])
ax =plt.subplot(1,2,2)
ax=raw_df['Sex'].value_counts().plot.pie(explode=[0.1,
0.1],autopct='%1.2f%%',shadow=True);
ax.set_title(label = "Sex", fontsize = 16) #,color='Red',font='Lucida
Calligraphy')
plt.savefig(pathIm + '/EDA33.png')
```



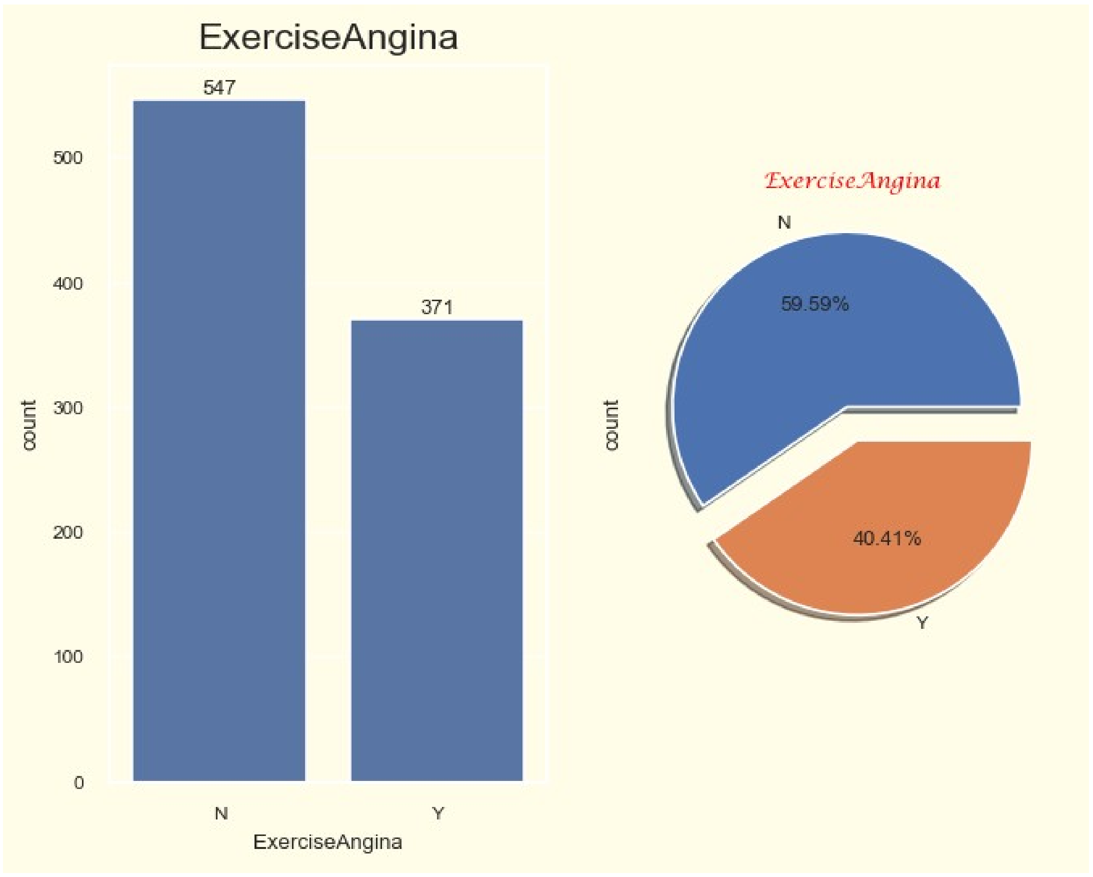
```
heart=raw_df
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='ChestPainType', data=heart)
ax.bar_label(ax.containers[0])
plt.title("ChestPainType", fontsize=14)
ax =plt.subplot(1,2,2)
ax=heart['ChestPainType'].value_counts().plot.pie(explode=[0.1,
0.1,0.1,0.1],autopct='%1.2f%%',shadow=True);
ax.set_title(label = "ChestPainType", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA34.png')
```



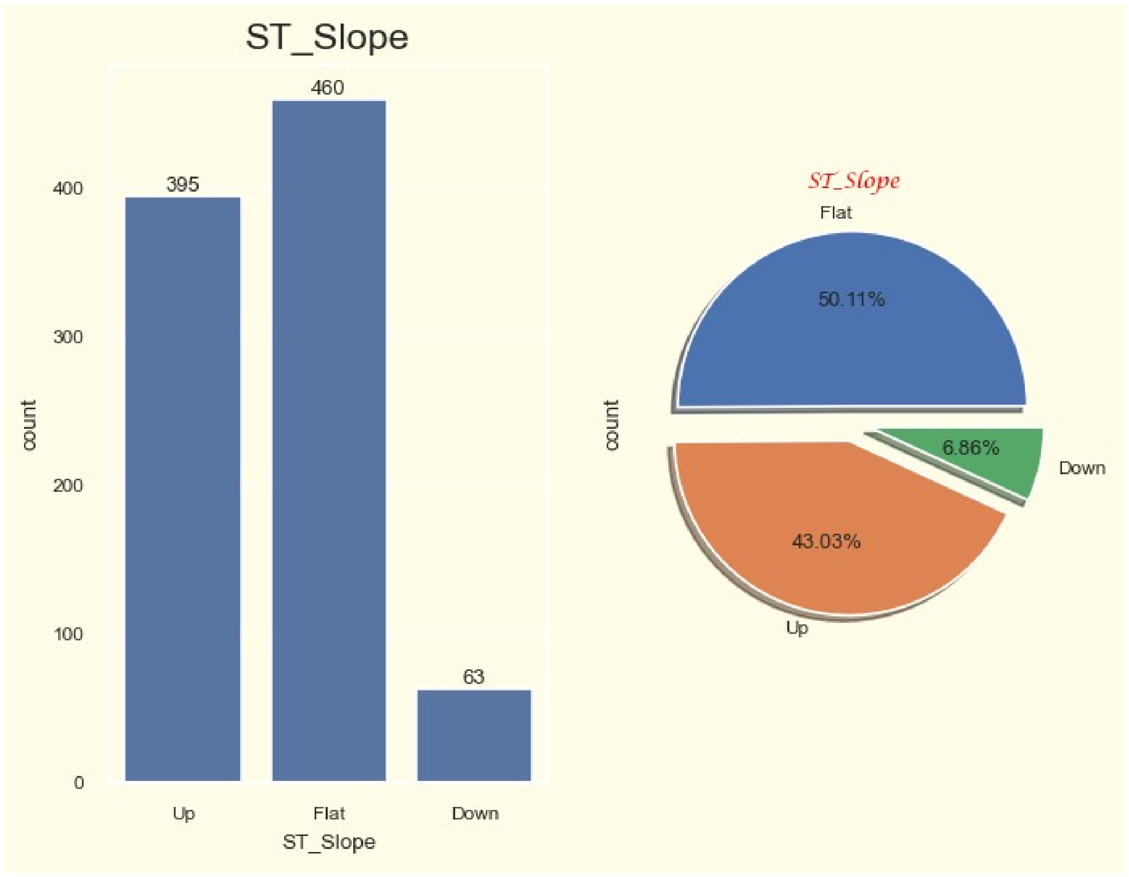
```
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='RestingECG', data=heart)
ax.bar_label(ax.containers[0])
plt.title("RestingECG", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['RestingECG'].value_counts().plot.pie(explode=[0.1,
0.1,0.1],autopct='%1.2f%%',shadow=True);
ax.set_title(label = "RestingECG", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA35.png')
```



```
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='ExerciseAngina', data=heart)
ax.bar_label(ax.containers[0])
plt.title("ExerciseAngina", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['ExerciseAngina'].value_counts().plot.pie(explode=[0.1,
0.1],autopct='%1.2f%%',shadow=True);
ax.set_title(label = "ExerciseAngina", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA36.png')
```



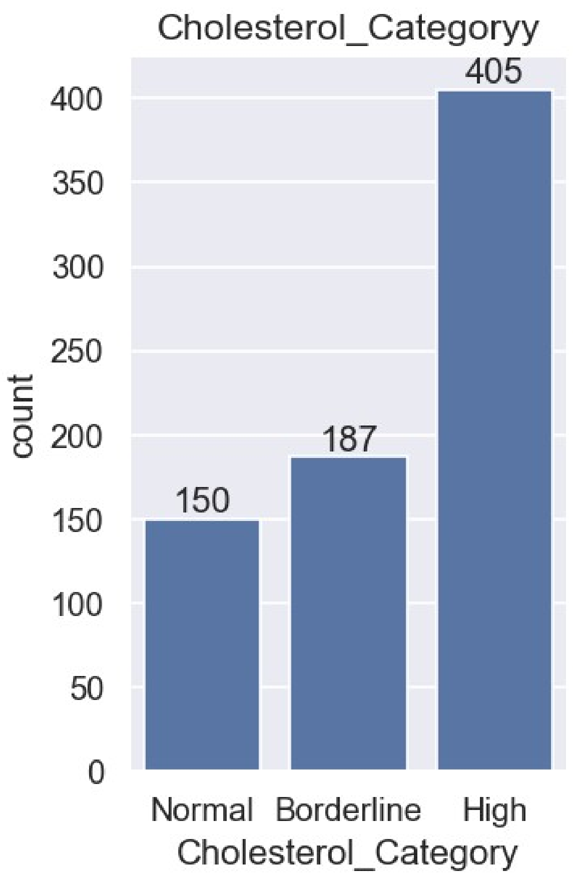
```
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='ST_Slope', data=heart)
ax.bar_label(ax.containers[0])
plt.title("ST_Slope", fontsize=14)


ax =plt.subplot(1,2,2)
ax=heart['ST_Slope'].value_counts().plot.pie(explode=[0.1,
0.1,0.1],autopct='%1.2f%%',shadow=True);
ax.set_title(label = "ST_Slope", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA37.png')
```

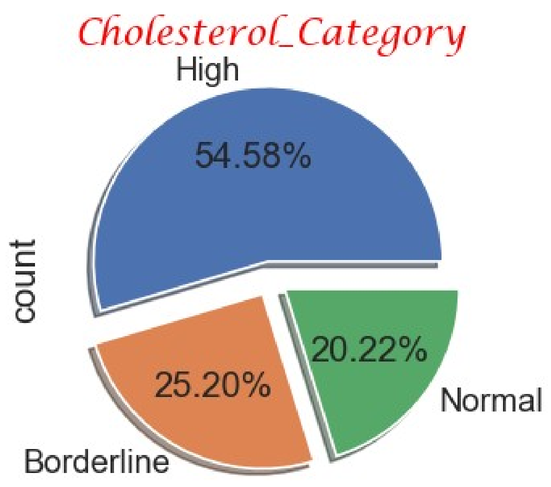


```
#heart=pd.read_csv('data/heart.csv')
sns.set(font_scale=1.1)
heart["Cholesterol_Category"]= pd.cut(heart["Cholesterol"] ,bins=[0,
200, 230 , 500] ,labels=["Normal","Borderline","High" ] )
print("Value Counts :\n\
n",heart['Cholesterol_Category'].value_counts())

heart.head()
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='Cholesterol_Category', data=heart)
ax.bar_label(ax.containers[0])
plt.title("Cholesterol_Categoryy", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['Cholesterol_Category'].value_counts().plot.pie(explode=[0.1,
0.1,0.1],autopct='%1.2f%%',shadow=True);
ax.set_title(label = "Cholesterol_Category", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA38.png')
```

```
Value Counts :

 Cholesterol_Category
High            405
```

```
Borderline    187
Normal        150
Name: count, dtype: int64
```

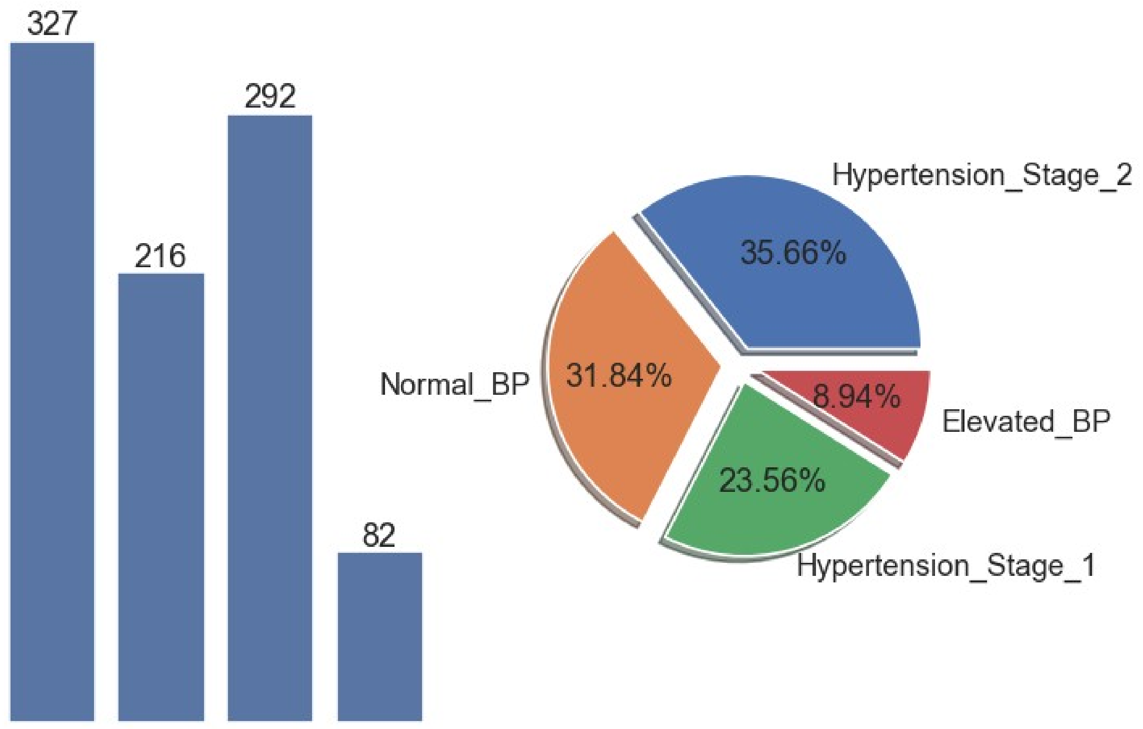




```
heart["RestingBP_Category"]= pd.cut(heart["RestingBP"] ,bins=[0,120,
129 ,
139,200] ,labels=["Normal_BP","Elevated_BP","Hypertension_Stage_1",
"Hypertension_Stage_2"] )
print("Value Counts :\n\n",heart['RestingBP_Category'].value_counts())

heart.sample(5)

heart['RestingBP_Category'] =
heart['RestingBP_Category'].astype(object)

Value Counts :

 RestingBP_Category
Hypertension_Stage_2    327
Normal_BP               292
Hypertension_Stage_1    216
```

```
Elevated_BP              82
Name: count, dtype: int64
```

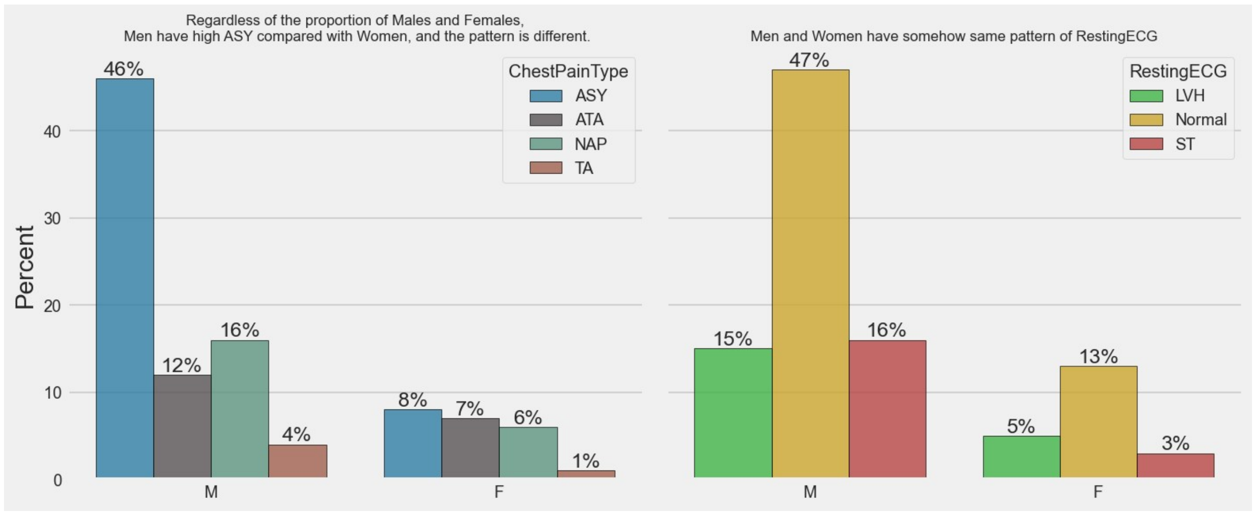
```
plt.rcParams['legend.fontsize'] = 7
sns.set(font_scale=1.0)
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='RestingBP_Category', data=heart)
ax.bar_label(ax.containers[0])
plt.axis('off');

ax =plt.subplot(1,2,2)
ax=heart['RestingBP_Category'].value_counts().plot.pie(explode=[0.1,
0.1,0.1,0.1],autopct='%1.2f%%',shadow=True);
plt.axis('off');
plt.savefig(pathIm + '/EDA39.png')
```



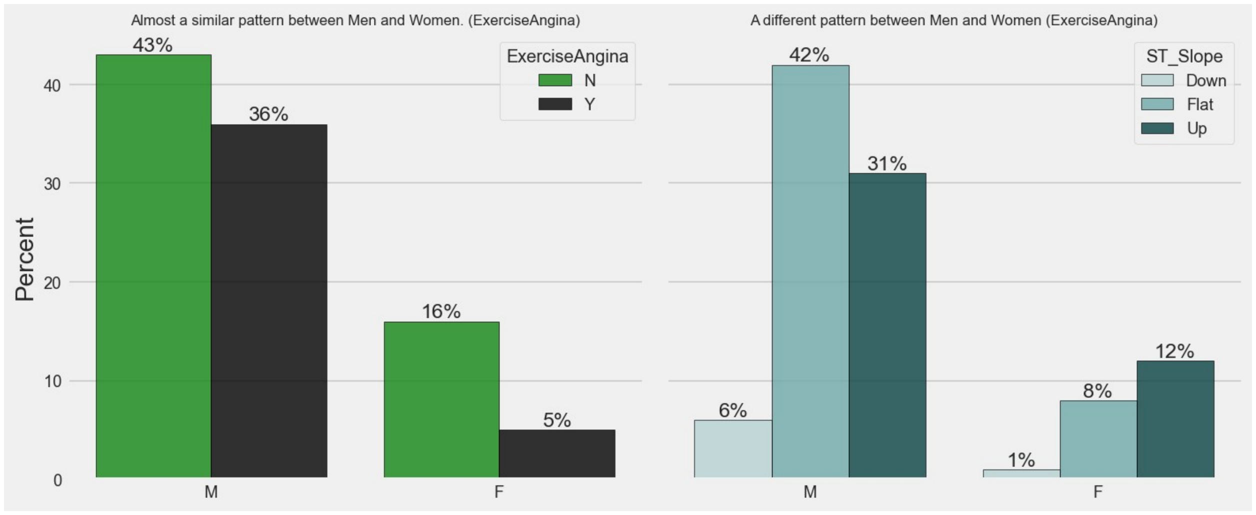
```
df = heart # pd.read_csv('data/heart.csv')
male_df = df[df['Sex'] == 'M']
female_df = df[df['Sex'] == 'F']

## Grouping Datasets
male_cp_fbs = male_df.groupby(['ChestPainType',
'FastingBS']).size().reset_index().rename(columns={0: 'count'})
female_cp_fbs = female_df.groupby(['ChestPainType',
'FastingBS']).size().reset_index().rename(columns={0: 'count'})
```

```
male_st_ecg = male_df.groupby(['ST_Slope',
'RestingECG']).size().reset_index().rename(columns={0: 'count'})
female_st_ecg = female_df.groupby(['ST_Slope',
'RestingECG']).size().reset_index().rename(columns={0: 'count'})

male_ea_cp = male_df.groupby(['ExerciseAngina',
'ChestPainType']).size().reset_index().rename(columns={0: 'count'})
female_ea_cp = female_df.groupby(['ExerciseAngina',
'ChestPainType']).size().reset_index().rename(columns={0: 'count'})

## Creating Sunburst Figures
sb1 = px.sunburst(male_cp_fbs, values='count', path=['ChestPainType',
'FastingBS'])
sb2 = px.sunburst(female_cp_fbs, values='count',
path=['ChestPainType', 'FastingBS'])

sb3 = px.sunburst(male_st_ecg, values='count', path=['ST_Slope',
'RestingECG'])
sb4 = px.sunburst(female_st_ecg, values='count', path=['ST_Slope',
'RestingECG'])

sb5 = px.sunburst(male_ea_cp, values='count', path=['ExerciseAngina',
'ChestPainType'])
sb6 = px.sunburst(female_ea_cp, values='count',
path=['ExerciseAngina', 'ChestPainType'])

## Subplots
fig = make_subplots(rows=3, cols=2, specs=[
    [{"type": "sunburst"}, {"type": "sunburst"}],
    [{"type": "sunburst"}, {"type": "sunburst"}],
    [{"type": "sunburst"}, {"type": "sunburst"}]
], subplot_titles=("Male Chest Pain with Fasting Blood Sugar", "Female
Chest Pain with Fasting Blood Sugar",
                   "Male ST Slope with Resting ECG", "Female ST Slope
with Resting ECG",
                   "Male Exercise Angina with Chest Pain Type",
"Female Exercise Angina with Chest Pain Type"))

## Plotting Figures
fig.add_trace(sb1.data[0], row=1, col=1)
fig.add_trace(sb2.data[0], row=1, col=2)
fig.add_trace(sb3.data[0], row=2, col=1)
fig.add_trace(sb4.data[0], row=2, col=2)
fig.add_trace(sb5.data[0], row=3, col=1)
fig.add_trace(sb6.data[0], row=3, col=2)

fig.update_traces(textinfo="label+percent parent")

# Update title and height
fig.update_layout(title_text="Male vs Female Sunburst", title_x=0.5,
                   height=1200, width=1200, template='plotly_dark',
showlegend=False,
        font=dict(
            family="Rubik",
            size=14)
)

fig.show()
fig.write_image(pathIm + '/EDA40.png',scale=6)

heart_dft= heart
#print(heart_dft.head())
RestingECG_vs_Sex = (
    heart_dft[["RestingECG", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="RestingECG")
)
RestingECG_vs_Sex["Pct"] = RestingECG_vs_Sex["Pct"].round(2) * 100
RestingECG_vs_Sex.sort_values(by="Pct", ascending=False)

ChestPainType_vs_Sex = (
    heart_dft[["ChestPainType", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="ChestPainType")
)
ChestPainType_vs_Sex["Pct"] = ChestPainType_vs_Sex["Pct"].round(2) *
100
#ChestPainType_vs_Sex

plt.style.use("fivethirtyeight")
fig, ax = plt.subplots(1, 2, figsize=(12, 5), sharey=True)
palette4 = {"ASY": "#1b85b8", "ATA": "#5a5255", "NAP": "#559e83",
"TA": "#ae5a41"}
palette5 = {"LVH": "#2dc937", "Normal": "#e7b416", "ST": "#cc3232"}

sns.barplot(
    data=ChestPainType_vs_Sex,
    x="Sex",
    hue="ChestPainType",
    #errorbar=None,
    y="Pct",
    palette=palette4,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[0],
)
```

```python
for ax1 in [ax[0]]:
    for container in ax1.containers:
        values2 = container.datavalues
        labels = ["{:g}%".format(val) for val in values2]
        ax1.bar_label(container, labels=labels)

ax[0].set_ylabel("Percent")
ax[0].set_xlabel("")
ax[0].set_title(
    "Regardless of the proportion of Males and Females,\n Men have
high ASY compared with Women, and the pattern is different.",
    fontsize=10,
)


sns.barplot(
    data=RestingECG_vs_Sex,
    x="Sex",
    hue="RestingECG",
    #errorbar=None,
    y="Pct",
    palette=palette5,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[1],
)

for ax2 in [ax[1]]:
    for container in ax2.containers:
        values3 = container.datavalues
        labels = ["{:g}%".format(val) for val in values3]
        ax2.bar_label(container, labels=labels)

ax[1].set_ylabel("")
ax[1].set_xlabel("")
ax[1].set_title("Men and Women have somehow same pattern of
RestingECG", fontsize=10)


plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA41.png')
```



```
ExerciseAngina_vs_Sex = (
    heart_dft[["ExerciseAngina", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="ExerciseAngina")
)
ExerciseAngina_vs_Sex["Pct"] = ExerciseAngina_vs_Sex["Pct"].round(2) *
100


ST_Slope_vs_Sex = (
    heart_dft[["ST_Slope", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="ST_Slope")
)
ST_Slope_vs_Sex["Pct"] = ST_Slope_vs_Sex["Pct"].round(2) * 100

plt.style.use("fivethirtyeight")
fig, ax = plt.subplots(1, 2, figsize=(12, 5), sharey=True)

palette6 = {
    "Y": "#000000",
    "N": "#009900",
}

palette7 = {"Down": "#b2d8d8", "Flat": "#66b2b2", "Up": "#004c4c"}

sns.barplot(
    data=ExerciseAngina_vs_Sex,
    x="Sex",
    hue="ExerciseAngina",
    #errorbar=None,
    y="Pct",
    palette=palette6,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[0],
)

for ax3 in [ax[0]]:
    for container in ax3.containers:
        values2 = container.datavalues
        labels = ["{:g}%".format(val) for val in values2]
        ax3.bar_label(container, labels=labels)

ax[0].set_ylabel("Percent")
ax[0].set_xlabel("")
ax[0].set_title(
    "Almost a similar pattern between Men and Women.
(ExerciseAngina)", fontsize=10
)


sns.barplot(
    data=ST_Slope_vs_Sex,
    x="Sex",
    hue="ST_Slope",
    #errorbar=None,
    y="Pct",
    palette=palette7,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[1],
)

for ax4 in [ax[1]]:
    for container in ax4.containers:
        values3 = container.datavalues
        labels = ["{:g}%".format(val) for val in values3]
        ax4.bar_label(container, labels=labels)

ax[1].set_ylabel("")
ax[1].set_xlabel("")
ax[1].set_title(
    "A different pattern between Men and Women (ExerciseAngina)",
fontsize=10
)


plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA42.png')
```

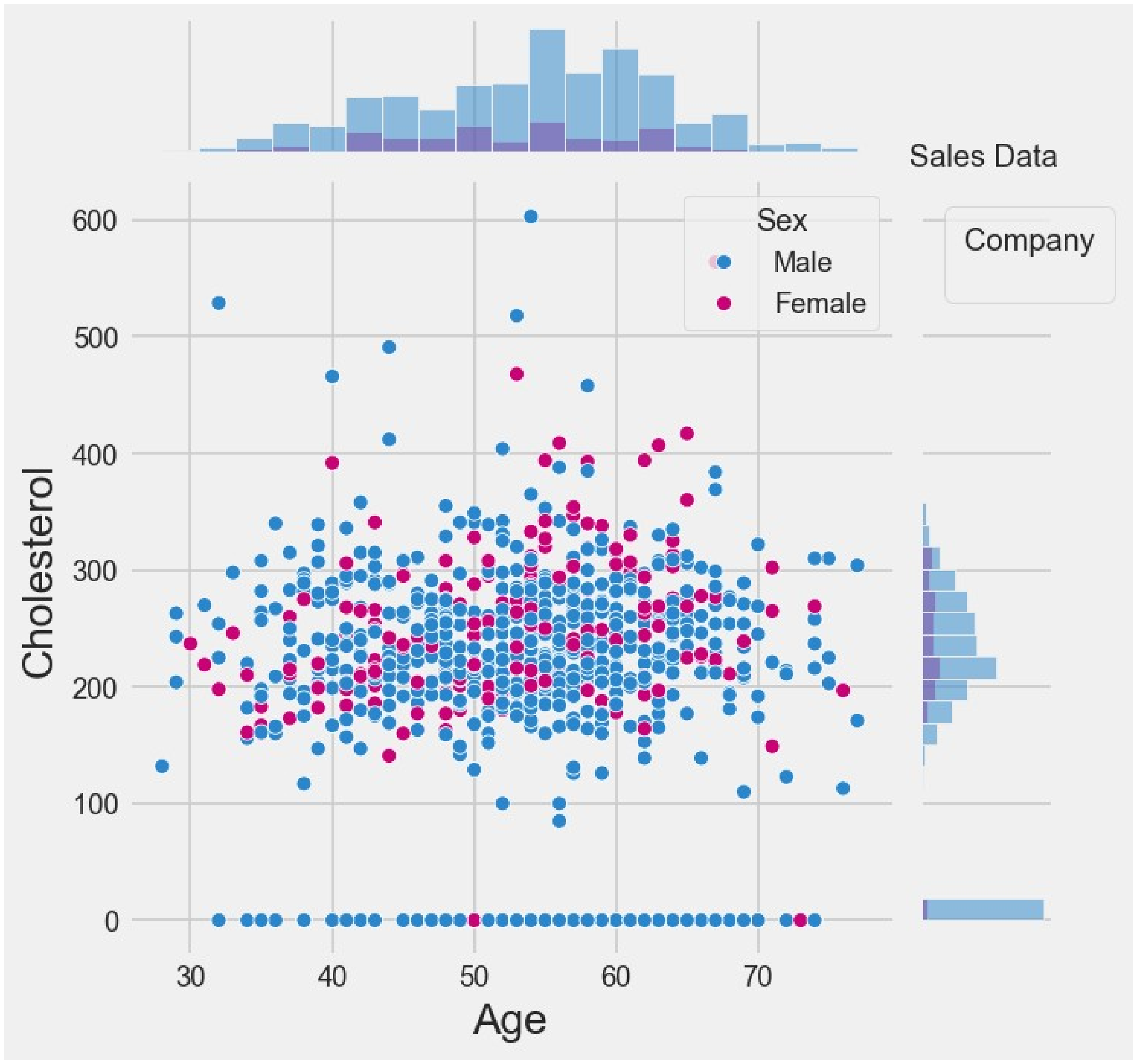


```
heart_dft=heart
print(heart_dft.head())
sex_color = dict({"Male": "#2986cc", "Female": "#c90076"})
plt.style.use("fivethirtyeight")
heart_dft["Sex"] = heart_dft["Sex"].map({"M": "Male", "F": "Female"})
heart_dft["Sex"]
heart_dft["HeartDisease"] = heart_dft["HeartDisease"].map({0: "No", 1:
"Yes"})
filtheart_dft = heart_dft["Cholesterol"] > 0
heart_dft_chol_n0 = heart_dft[filtheart_dft]

g=sns.JointGrid(
    data=heart_dft, x="Age", y="Cholesterol", hue="Sex",
palette=sex_color
).plot(sns.scatterplot, sns.histplot)

plt.legend(title='Company', fontsize=20)
plt.xlabel('Agex', fontsize=10);
plt.ylabel('Cholesterolx', fontsize=10);
plt.title('Sales Data', fontsize=12)
plt.tick_params(axis='both', which='major', labelsize=10)
#plt.show()
g.savefig(pathIm + '/EDA43.png')
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```

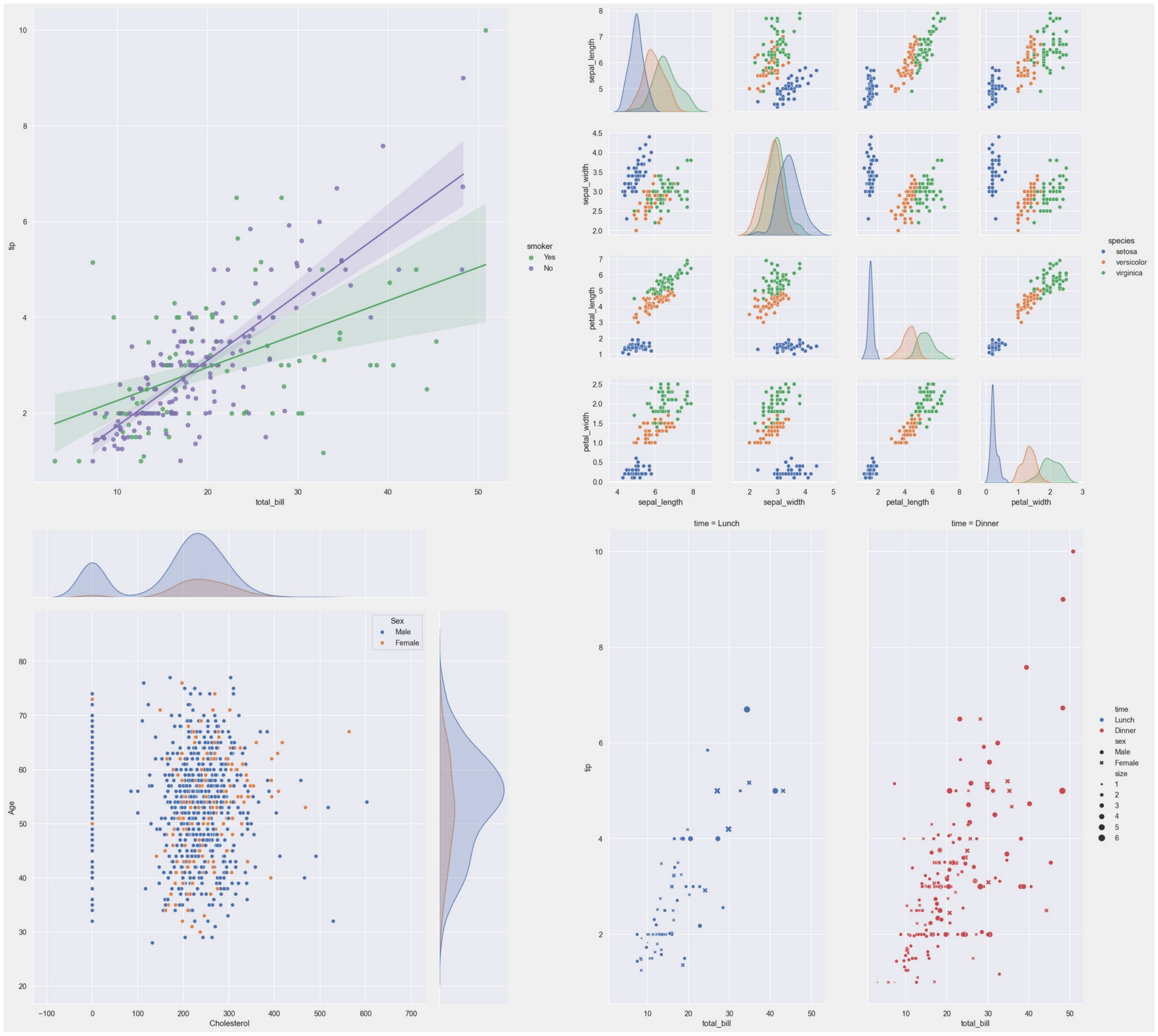
```
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG
MaxHR  \
0   40   M           ATA        140          289          0     Normal
```

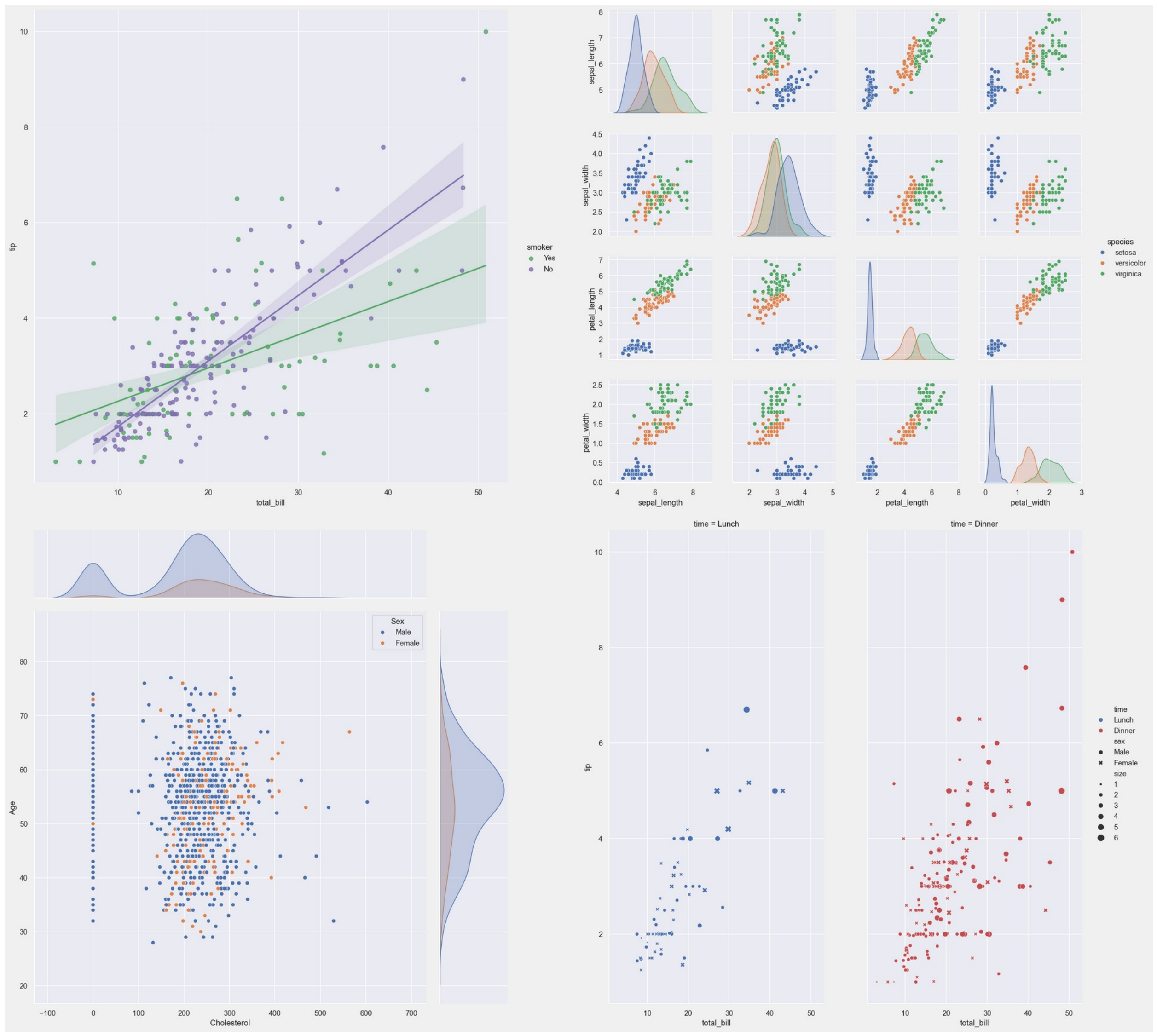
```
172
1   49   F             NAP        160          180          0     Normal
156
2   37   M             ATA        130          283          0         ST
98
3   48   F             ASY        138          214          0     Normal
108
4   54   M             NAP        150          195          0     Normal
122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease Cholesterol_Category
\
0              N      0.0       Up             0                 High

1              N      1.0     Flat             1               Normal

2              N      0.0       Up             0                 High

3              Y      1.5     Flat             1           Borderline

4              N      0.0       Up             0               Normal


     RestingBP_Category
0  Hypertension_Stage_2
1  Hypertension_Stage_2
2  Hypertension_Stage_1
3  Hypertension_Stage_1
4  Hypertension_Stage_2
```



```
import seaborn as sns
import patchworklib as pw
sns.set_theme()
pw.overwrite_axisgrid()

raw_df = heart
iris = sns.load_dataset("iris")
tips = sns.load_dataset("tips")

# An lmplot
g0 = sns.lmplot(x="total_bill", y="tip", hue="smoker", data=tips,
                palette=dict(Yes="g", No="m"))
g0 = pw.load_seaborngrid(g0, label="g0")

# A Pairplot
g1 = sns.pairplot(iris, hue="species")
```

```
g1 = pw.load_seaborngrid(g1, label="g1")

# A relplot
g2 = sns.relplot(data=tips, x="total_bill", y="tip", col="time",
hue="time",
                 size="size", style="sex", palette=["b", "r"],
sizes=(10, 100))
g2 = pw.load_seaborngrid(g2, label="g2")

# A JointGrid
#g3 = sns.jointplot(x='Cholesterol',y='FastingBS',data=raw_df,
hue="Sex")
g3 = sns.jointplot(x='Cholesterol',y='Age',data=raw_df, hue="Sex")
g3 = pw.load_seaborngrid(g3, label="g3")
(((g0|g1)["g0"]/g3)["g3"]|g2).savefig(pathIm + '/EDA44.png')
```

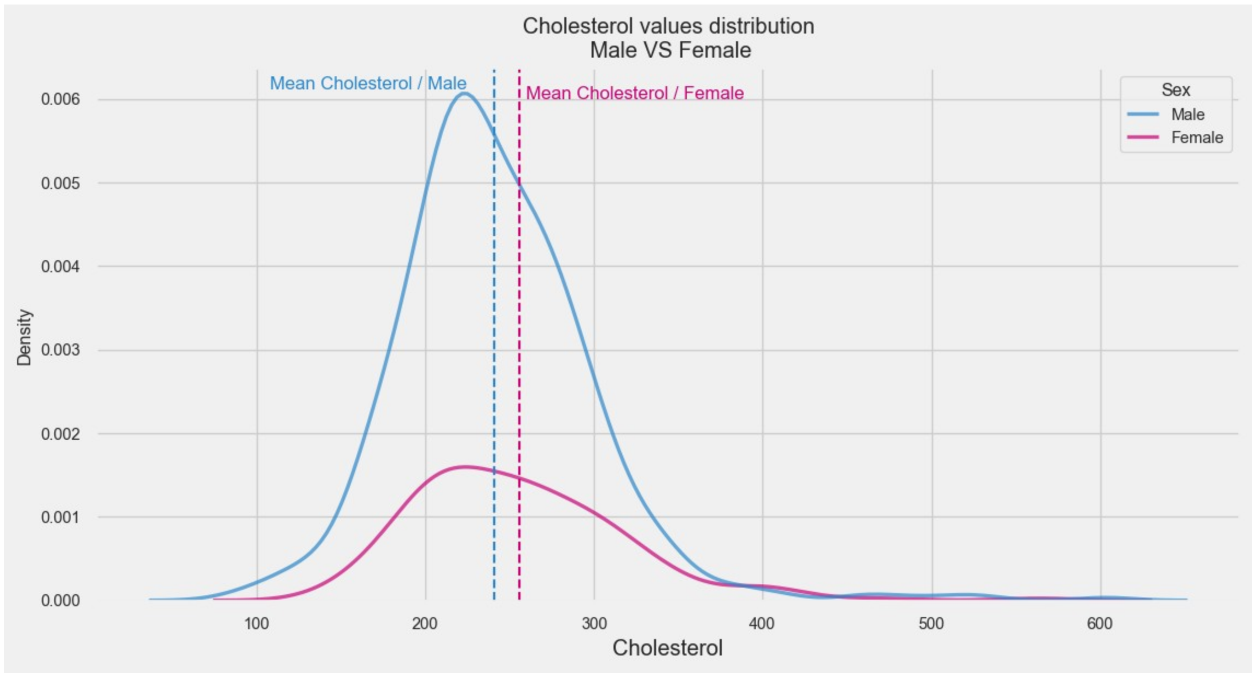




```
try:
    heart_dft = pd.read_csv(path+'/data/heart.csv')
except:
    heart_dft=pd.read_csv(path+'/data/heart.csv')

#print(heart_dft.head())

sex_color = dict({"Male": "#2986cc", "Female": "#c90076"})
plt.style.use("fivethirtyeight")
heart_dft["Sex"] = heart_dft["Sex"].map({"M": "Male", "F": "Female"})
heart_dft["Sex"]
#print(heart_dft["Sex"].head())
heart_dft["HeartDisease"] = heart_dft["HeartDisease"].map({0: "No", 1:
"Yes"})
#heart_dft.info()
```

```python
filtheart_dft = heart_dft["Cholesterol"] > 0
heart_dft_chol_n0 = heart_dft[filtheart_dft]
#heart_dft_chol_n0.head()

Chol_mean_f = (
    heart_dft_chol_n0[["Sex", "Cholesterol"]]
    .groupby(["Sex"])
    .mean("Cholesterol")
    .loc["Female", "Cholesterol"]
).round()

Chol_mean_m = (
    heart_dft_chol_n0[["Sex", "Cholesterol"]]
    .groupby(["Sex"])
    .mean("Cholesterol")
    .loc["Male", "Cholesterol"]
).round()

print("for Female =", Chol_mean_f, "for Male =", Chol_mean_m)

plt.figure(figsize=(10, 5))
sns.set_context("paper")

kdeplt = sns.kdeplot(
    data=heart_dft_chol_n0,
    x="Cholesterol",
    hue="Sex",
    palette=sex_color,
    alpha=0.7,
    lw=2,
)

kdeplt.set_title("Cholesterol values distribution\n Male VS Female",
fontsize=12)
kdeplt.set_xlabel("Cholesterol", fontsize=12)
plt.axvline(x=Chol_mean_f, color="#c90076", ls="--", lw=1.3)
plt.axvline(x=Chol_mean_m, color="#2986cc", ls="--", lw=1.3)
plt.text(108, 0.00612, "Mean Cholesterol / Male", fontsize=10,
color="#2986cc")
plt.text(260, 0.006, "Mean Cholesterol / Female", fontsize=10,
color="#c90076")
kdeplt.figure.savefig(pathIm + '/EDA45.png')
```

```
for Female = 256.0 for Male = 241.0
```

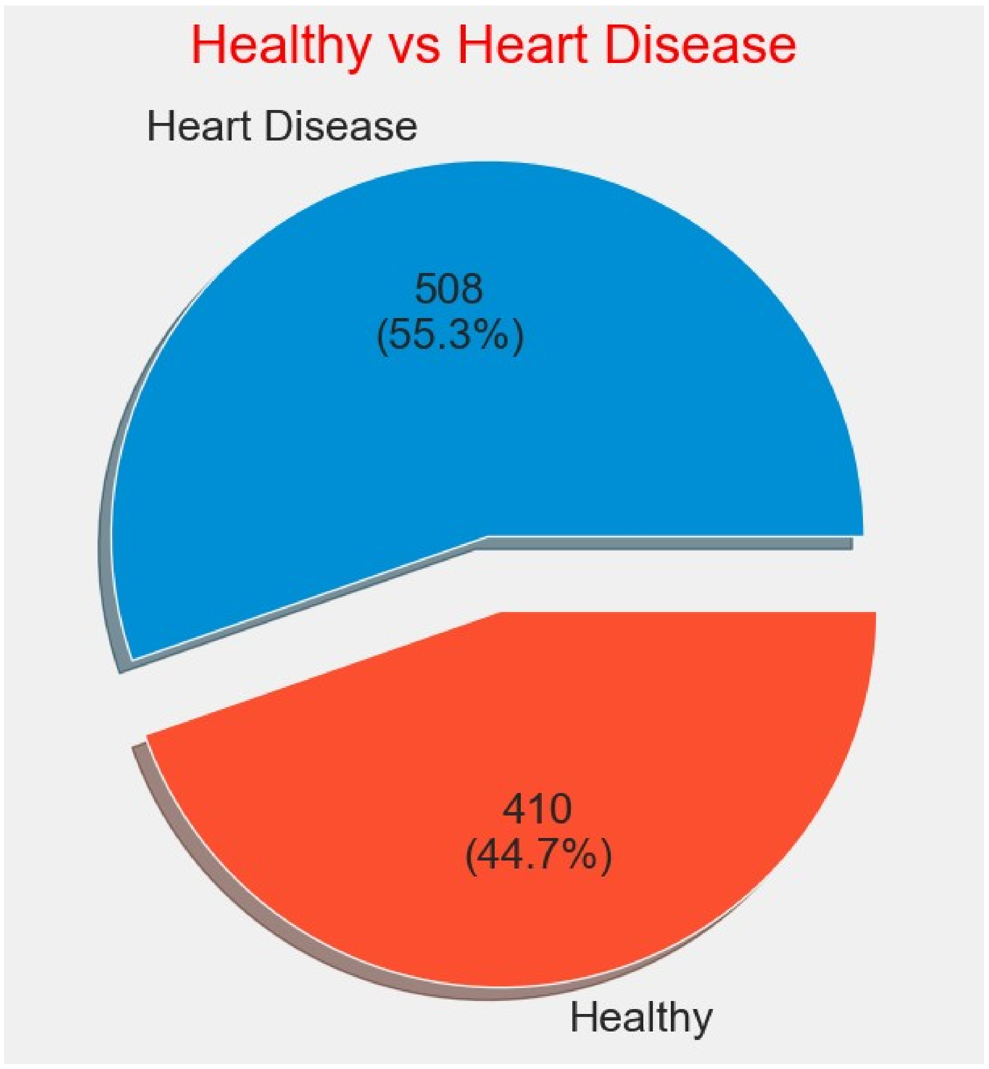


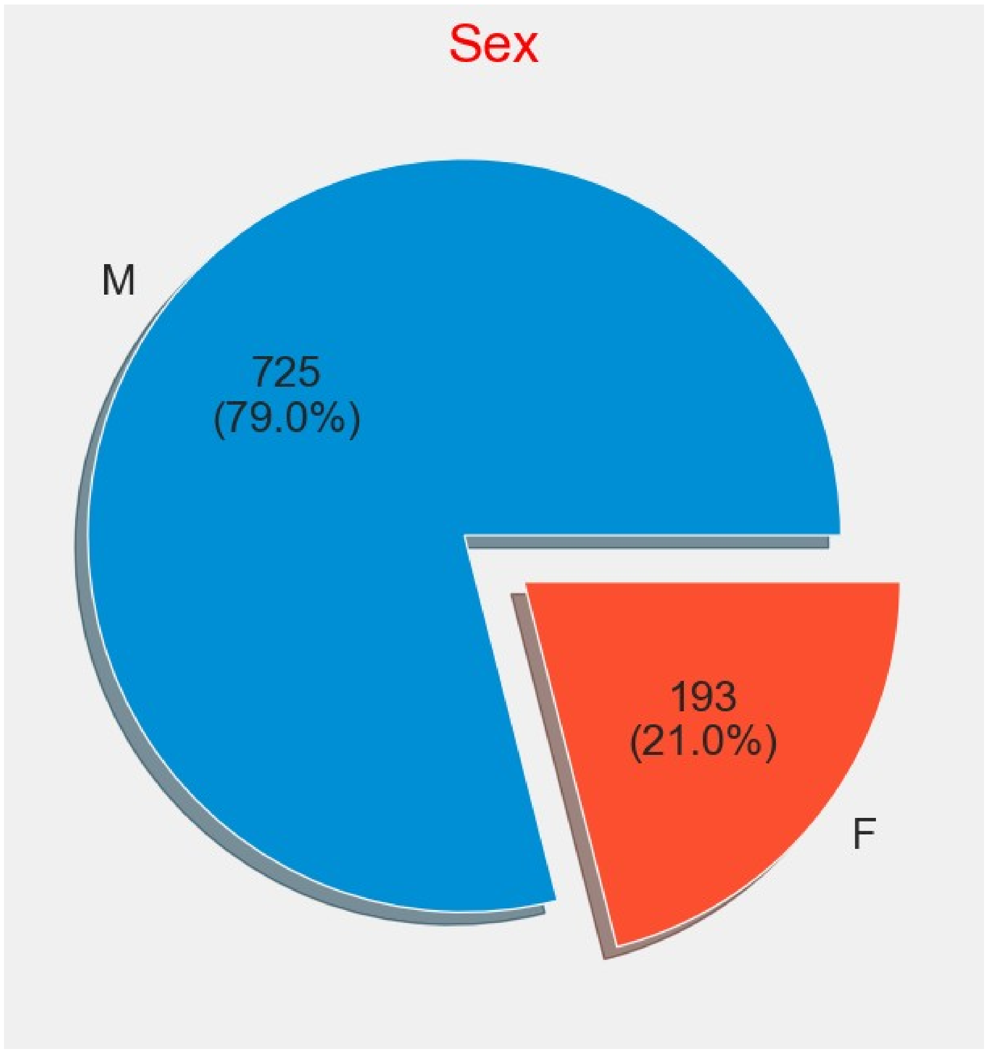
```
import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
colorscale = ['#f2e5ff','#ffffff']

des=raw_df.describe().T.applymap('{:,.2f}'.format)
des.to_csv(path+'/data/des.csv')
try:
    df_sample = pd.read_csv(path+'/data/des.csv')
except:
    df_sample = pd.read_csv(path+'/data/des.csv')
df_sample.rename(columns = {'Unnamed: 0':'Variable'}, inplace = True)
#print(df_sample)

fig = go.Figure(data=[
      go.Table(
      header = dict(
        values = [['<b>Variable</b>'],
                  ['<b>count</b>'],
                  ['<b>mean</b>'],
                  ['<b>std</b>'],
                  ['<b>min</b>'],
                  ['<b>25% </b>'],
                  ['<b>50%</b>'],
                  ['<b>75%</b>'],
                  ['<b>max</b>'],
                 ],
        line_color='darkslategray',
        fill_color='#4d004c',

align=['center','center','center','center','center','center','center',
'center','center'],
        font=dict(color='white', size=14),
        height=30
     ),
         cells=dict(values=df_sample.values.transpose(),
                    fill_color = [colorscale*des1.shape[0]],
                    align='right', font=dict(color='black', size=16),
                    height=30
                   )
             )
      ])

fig.update_layout(
    height=800,
    width=1200,
    showlegend=True,
    title_text="Таблица 2.Дескриптивная статистика для всех
пациентов",
    font=dict(color="darkred", size=18)
)

fig.update_layout(width=1200, height=800)
fig.show(renderer="svg")
fig.write_image(pathIm + '/EDA2.png',scale=4)
```

```python
import plotly.figure_factory as ff
import pandas as pd
import plotly.io as pio
pio.renderers

des=raw_df[raw_df['HeartDisease']==0].describe().T.applymap('{:,.2f}'.
format)
des.to_csv(path+'/data/des.csv')
try:
    df_sample = pd.read_csv(path+'/data/des.csv')
except:
    df_sample = pd.read_csv(path+'/data/des.csv')
df_sample.rename(columns = {'Unnamed: 0':'Variable'}, inplace = True)

colorscale = [[0, '#4d004c'],[.5, '#f2e5ff'],[1, '#ffffff']]
columnwidth=[15,10,10,10,10,10,10,10,10]
fig =  ff.create_table(df_sample, colorscale=colorscale)

fig.update_layout(width=1200, height=350,
    title_text="Таблица 3. Основная статистика для переменной
HeartDisease='Healthy'",
    margin = {'t':50, 'b':50},
    font=dict(color="darkred", size=18)
)
```

```
fig.show()
fig.write_image(pathIm+'/EDA3.png',scale=6)

######################################################################
#####################
des=raw_df[raw_df['HeartDisease']==1].describe().T.applymap('{:,.2f}'.
format)
des.to_csv(path+'/data/des.csv')
try:
    df_sample = pd.read_csv(path+'/data/des.csv')
except:
    df_sample = pd.read_csv(path+'/data/des.csv')
df_sample.rename(columns = {'Unnamed: 0':'Variable'}, inplace = True)

colorscale = [[0, '#4d004c'],[.5, '#f2e5ff'],[1, '#ffffff']]
columnwidth=[15,10,10,10,10,10,10,10,10]
fig =  ff.create_table(df_sample, colorscale=colorscale)

fig.update_layout(width=1200, height=350,
    title_text="Таблица 4. Основная статистика для переменной
HeartDisease='Heart Disease'",
    margin = {'t':50, 'b':50},
    font=dict(color="darkred", size=18)
)

fig.show()
fig.write_image(pathIm+'/EDA4.png',scale=6)

from matplotlib import *
from matplotlib.colors import ListedColormap
import matplotlib
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
import pandas as pd
import numpy as np
pio.renderers

def auto_fmt (pct_value):
    return '{:.0f}\n({:.1f}
%)'.format(raw_df['HeartDisease'].value_counts().sum()*pct_value/
100,pct_value)

try:
```

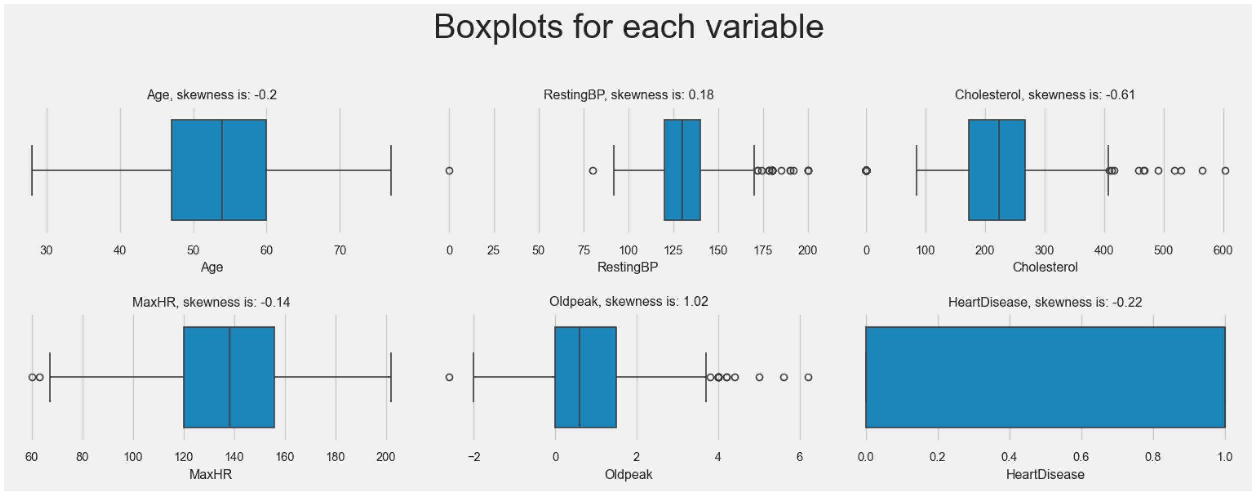
```python
    raw_df = pd.read_csv(path+'/data/heart.csv')
except:
    raw_df = pd.read_csv(path+'/data/heart.csv')
HDValues={
    0:'Healthy',
    1:'Heart Disease'
    }

df = raw_df.HeartDisease.replace(HDValues)
#print(df.head())

fig=plt.figure(figsize=(6, 6))
matplotlib.rcParams.update({'font.size': 15})

df.value_counts().plot.pie(explode=[0.1, 0.1],

                                       autopct=auto_fmt,
                                       textprops={'fontsize': 16},
                                       shadow=True)

plt.title('Healthy vs Heart Disease', color='Red',pad=15,
fontsize=20);
plt.axis('off');
fig.show()
plt.savefig(pathIm + '/EDA5.png')


plt.figure(figsize=(6, 6))
matplotlib.rcParams.update({'font.size': 15})

raw_df.Sex.value_counts().plot.pie(explode=[0.1, 0.1],
                                       #autopct='%1.2f%%',
                                       autopct=auto_fmt,
                                       textprops={'fontsize': 16},
                                       shadow=True)
plt.title('Sex', color='Red',pad=10, fontsize=20);
plt.axis('off');

fig.show()
plt.savefig(pathIm + '/EDA6.png')
```


Healthy vs Heart Disease
Heart Disease
508
(55.3%)
410
(44.7%)
Healthy

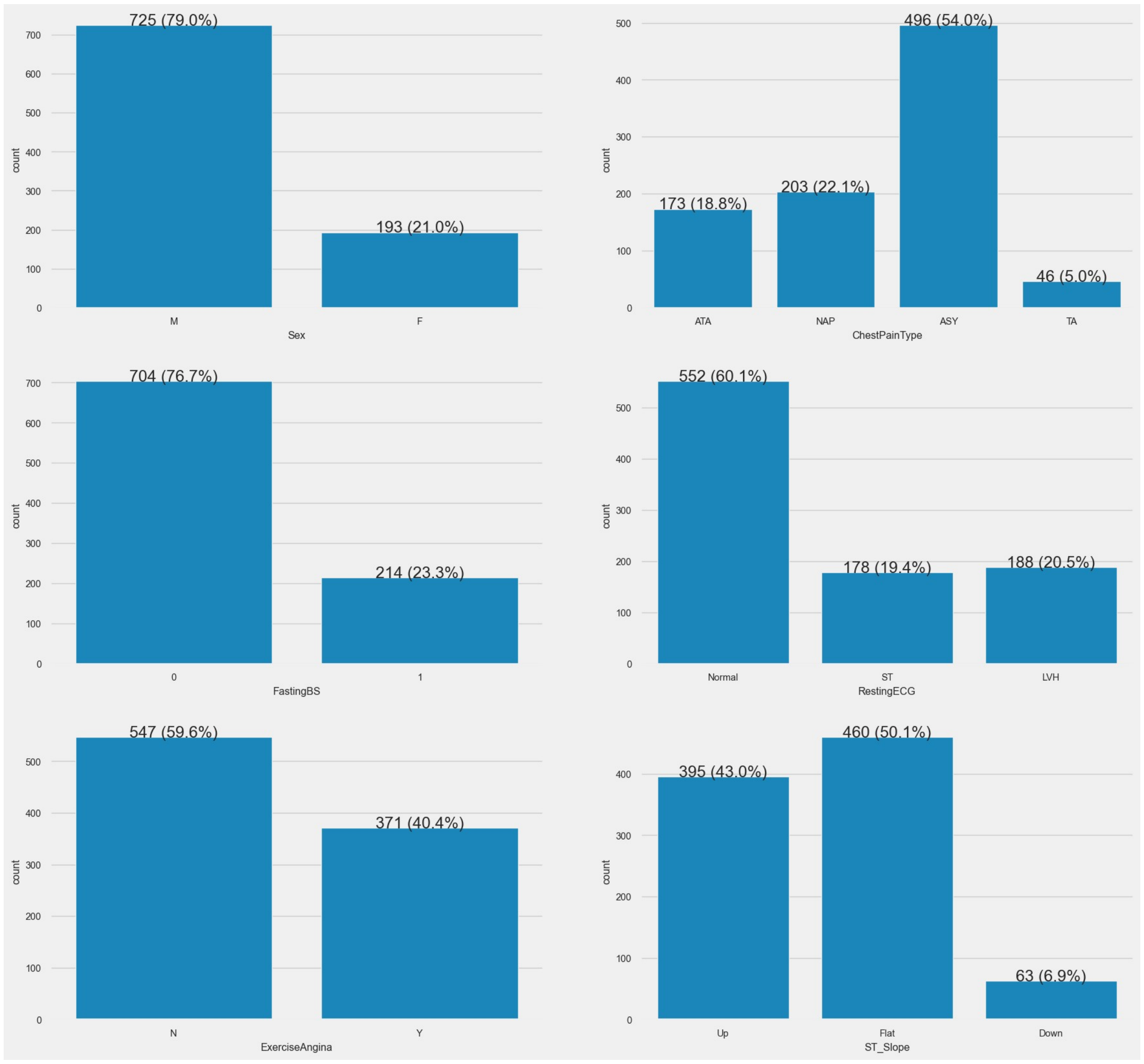


```
numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])

# checking boxplots
def boxplots_custom(dataset, columns_list, rows, cols, suptitle):
    fig, axs = plt.subplots(rows, cols, sharey=True, figsize=(13,5))
    fig.suptitle(suptitle,y=1, size=25)
    axs = axs.flatten()
    for i, data in enumerate(columns_list):
        sns.boxplot(data=dataset[data], orient='h', ax=axs[i])
        axs[i].set_title(data + ', skewness is:
'+str(round(dataset[data].skew(axis = 0, skipna = True),2)))

boxplots_custom(dataset=raw_df, columns_list=numerical_columns,
rows=2, cols=3, suptitle='Boxplots for each variable')
plt.tight_layout()
```

```
#fig.show()
plt.savefig(pathIm + '/EDA7.png')
```

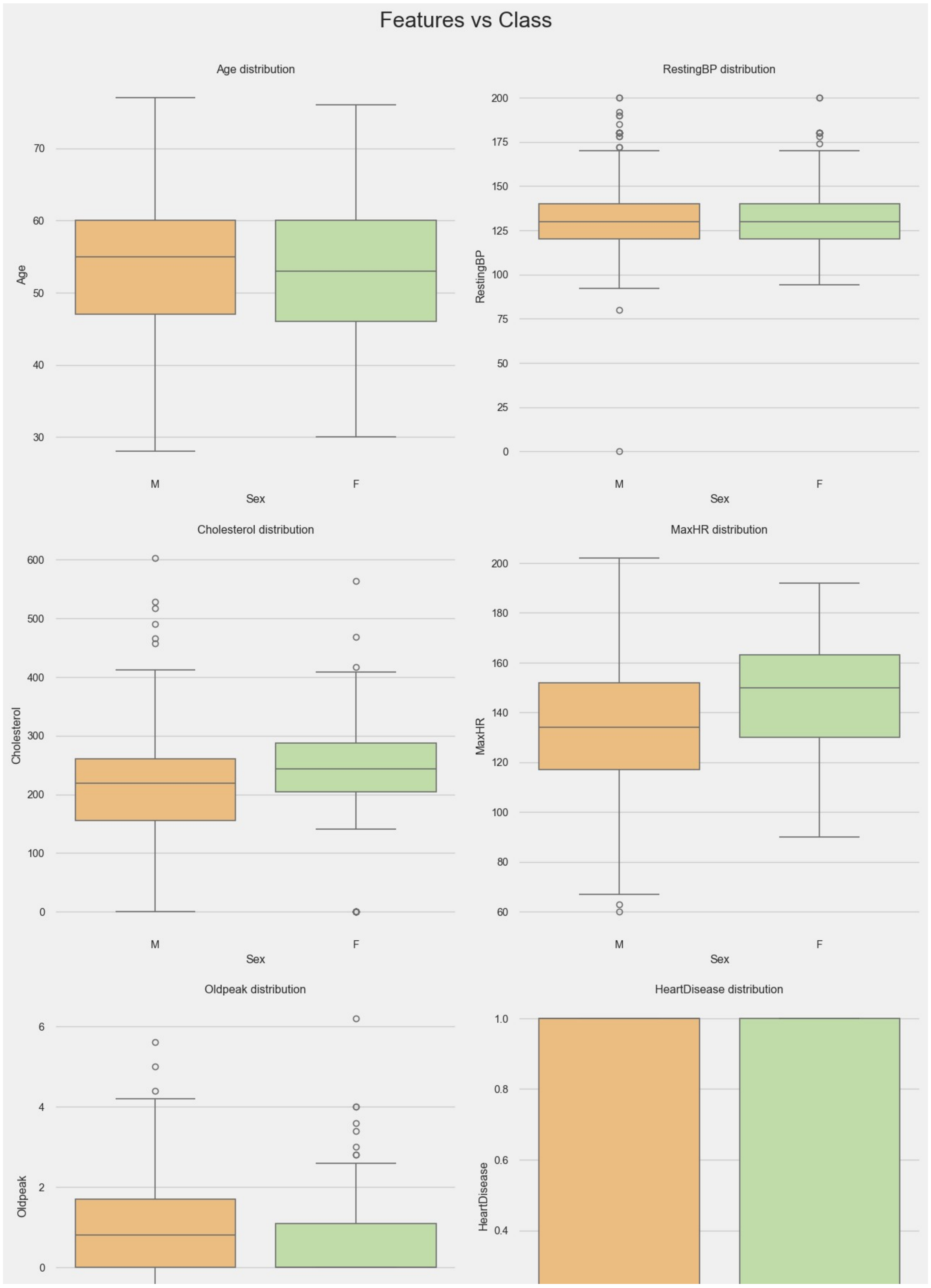


```
fig, ax = plt.subplots (3, 2, figsize=(16, 16))
ax_rst = []

for i in range(len(categorical_columns)):
    axs = sns.countplot(data=raw_df, x
=raw_df[categorical_columns[i]], ax=ax[int(i/2),i % 2])
    ax_rst.append(axs)
    total = raw_df[categorical_columns[i]].value_counts().sum()
    for p in axs.patches:
        value_pct = '{:.0f} ({:.1f}%)'.format(p.get_height(), 100 *
p.get_height()/total)
        x = p.get_x() + p.get_width()/2
        y = p.get_height()
        axs.annotate(value_pct, (x, y),ha='center')
plt.savefig(pathIm + '/EDA8.png')
```



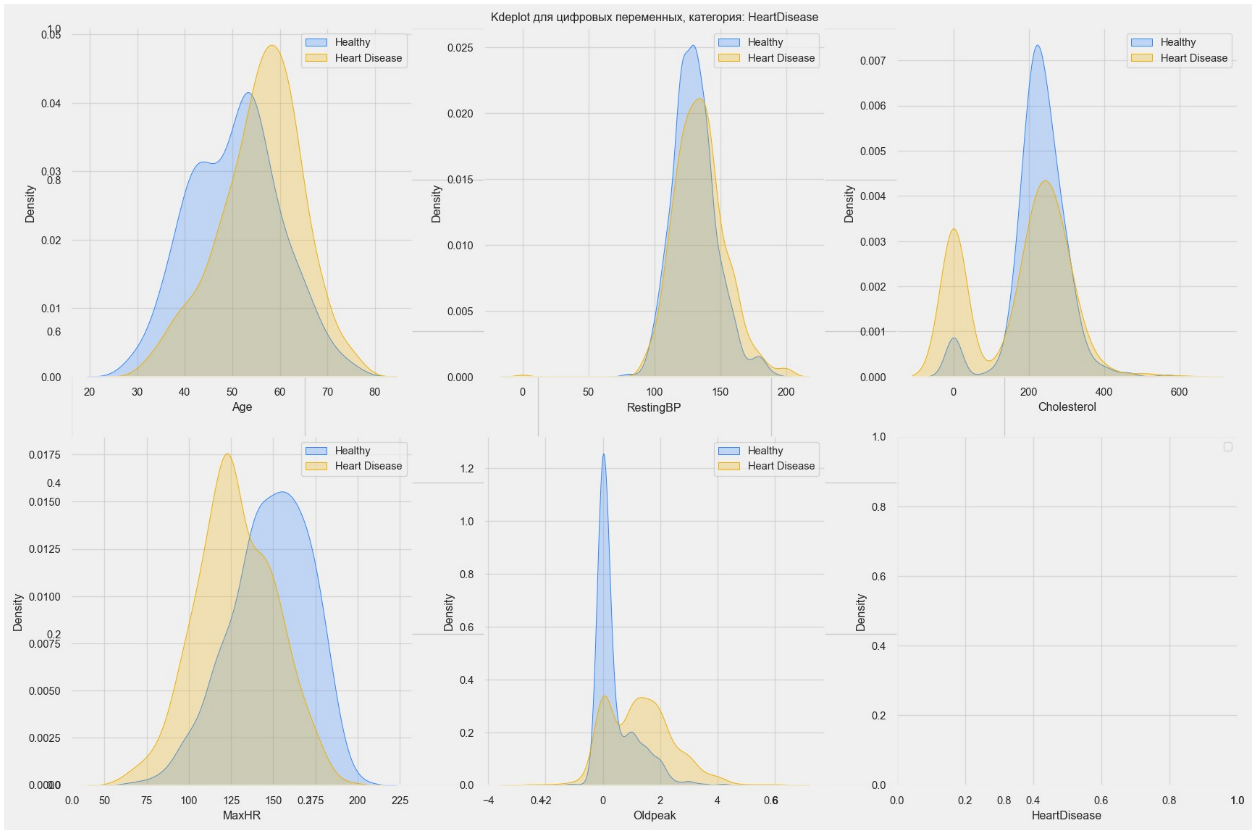
```
fig, axes = plt.subplots(nrows=3, ncols=2,figsize=(11,17))
fig.suptitle('Features vs Class\n', size = 18)

sns.boxplot(ax=axes[0, 0], data=raw_df, x='Sex', y='Age',
palette='Spectral')
axes[0,0].set_title("Age distribution");


sns.boxplot(ax=axes[0,1], data=raw_df, x='Sex', y='RestingBP',
palette='Spectral')
axes[0,1].set_title("RestingBP distribution");


sns.boxplot(ax=axes[1, 0], data=raw_df, x='Sex', y='Cholesterol',
palette='Spectral')
```

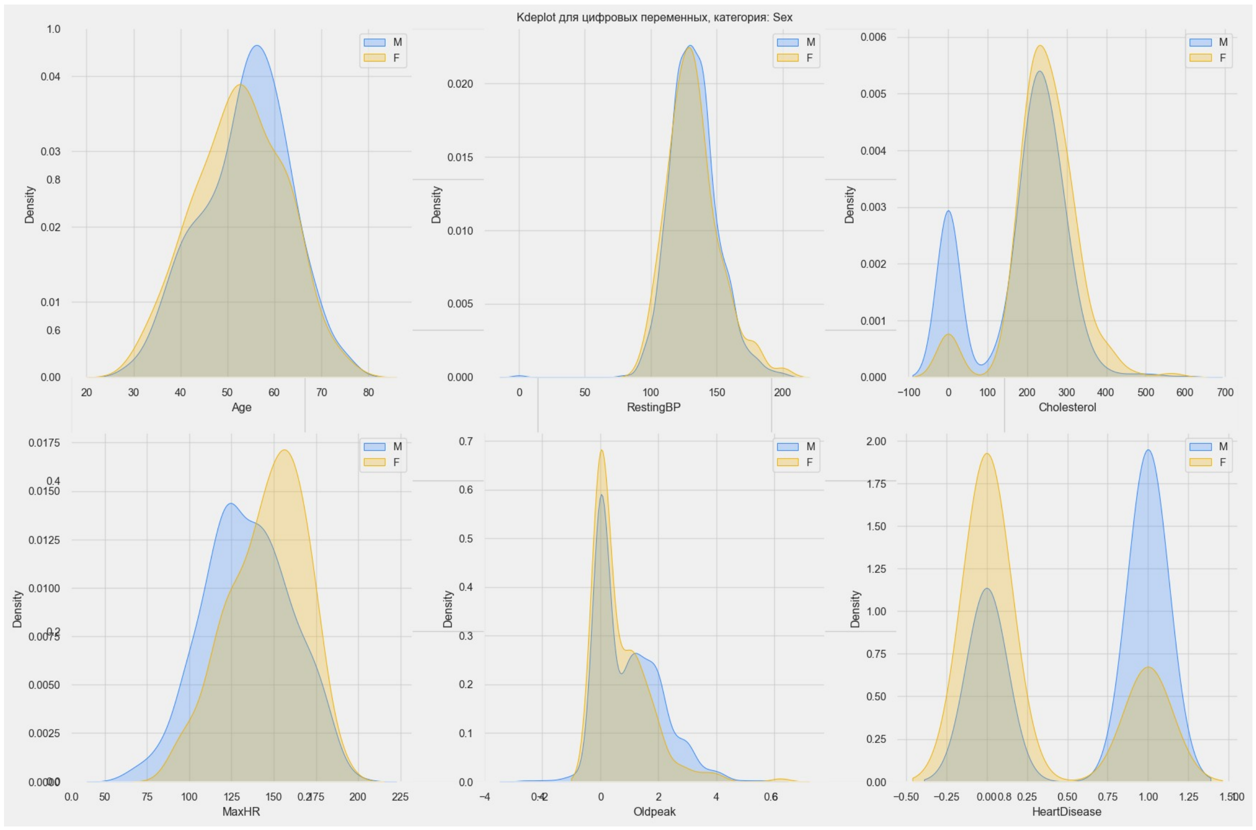
```
axes[1,0].set_title("Cholesterol distribution");

sns.boxplot(ax=axes[1, 1], data=raw_df, x='Sex', y='MaxHR',
palette='Spectral')
axes[1,1].set_title("MaxHR distribution");

sns.boxplot(ax=axes[2, 0], data=raw_df, x='Sex', y='Oldpeak',
palette='Spectral')
axes[2,0].set_title("Oldpeak distribution");

sns.boxplot(ax=axes[2, 1], data=raw_df, x='Sex', y='HeartDisease',
palette='Spectral')
axes[2,1].set_title("HeartDisease distribution");

plt.tight_layout()
#fig.show()
plt.savefig(pathIm + '/EDA9.png')
```


Features vs Class
Age distribution
Age
Sex
M
F
RestingBP distribution
RestingBP
Cholesterol distribution
Cholesterol
MaxHR distribution
MaxHR
Oldpeak distribution
Oldpeak
HeartDisease distribution
HeartDisease

```
numeric_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
#categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
try:
    raw_df = pd.read_csv(path+'/data/heart.csv')
except:
    raw_df = pd.read_csv(path+'/data/heart.csv')


fig = plt.figure(figsize=(15, 10))
plt.title('Kdeplot для цифровых переменных, категория: HeartDisease')
rows, cols = 2, 3
for idx, num in enumerate(numeric_columns[:30]):
    ax = fig.add_subplot(rows, cols, idx+1)
    ax.grid(alpha = 0.7, axis ="both")
    sns.kdeplot(x = num, fill = True,color ="#3386FF",linewidth=0.6,
data = raw_df[raw_df['HeartDisease']==0], label = "Healthy")
    sns.kdeplot(x = num, fill = True,color ="#EFB000",linewidth=0.6,
data = raw_df[raw_df['HeartDisease']==1] , label = "Heart Disease")
    ax.set_xlabel(num)
    ax.legend()

fig.tight_layout()
fig.show()
plt.savefig(pathIm + '/EDA10.png')

fig = plt.figure(figsize=(15, 10))
plt.title('Kdeplot для цифровых переменных, категория: Sex')
rows, cols = 2, 3
for idx, num in enumerate(numeric_columns[:30]):
    ax = fig.add_subplot(rows, cols, idx+1)
    ax.grid(alpha = 0.7, axis ="both")
    sns.kdeplot(x = num, fill = True,color ="#3386FF",linewidth=0.6,
data = raw_df[raw_df['Sex']=="M"], label = "M")
    sns.kdeplot(x = num, fill = True,color ="#EFB000",linewidth=0.6,
data = raw_df[raw_df['Sex']=="F"], label = "F")
    ax.set_xlabel(num)
    ax.legend()

fig.tight_layout()
fig.show()
plt.savefig(pathIm + '/EDA11.png')
```

```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```


Kdeplot для цифровых переменных, категория: HeartDisease
Healthy
Heart Disease
Density
Age
RestingBP
Cholesterol
MaxHR
Oldpeak
HeartDisease


Kdeplot для цифровых переменных, категория: Sex
M
F
Density
Age
RestingBP
Cholesterol
MaxHR
Oldpeak
HeartDisease

https://www.kaggle.com/code/hendabdelnasser/heart-failure-prediction-model-knn-lr

```
from matplotlib import *
from matplotlib.colors import ListedColormap
import matplotlib
import matplotlib.pyplot as plt
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import plotly as pplt
from matplotlib.colors import ListedColormap
import seaborn as sns
import pandas as pd
import numpy as np

raw_df = pd.read_csv(path+'/data/heart.csv')
fig = plt.figure(figsize=(25, 10))
fig = px.scatter_3d(raw_df,
                    x='RestingBP',
                    y='Age',
                    z='Sex',
                    color='HeartDisease')


fig.show()
fig.write_html(pathIm + '/Buble3D.html')
```

```
<Figure size 3000x1200 with 0 Axes>
```

```
with
open("C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image/EDA12.png",
'wb') as f:
    f.write(pplt.io.to_image(fig, width=1200, height=800,
format='png'))
```

https://www.kaggle.com/code/aadharshviswanath/heart-failure-detailed-eda-and-model-building

```
df = pd.read_csv(path+'/data/heart.csv')
#print(df.head())

def replace_zero_cholesterol(df):
    # Step 1: Create age groups and calculate average cholesterol for
each group
    age_bins = [10, 20, 30, 40, 50, 60, 70, 80]
    age_labels = [f'{start}-{end}' for start, end in zip(age_bins[:-
1], age_bins[1:])]
    df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins,
labels=age_labels, right=False)
    average_cholesterol_by_age = df.groupby('AgeGroup')
['Cholesterol'].mean()

    # Step 2: Replace zero values in 'Cholesterol' with average values
based on age groups
    def replace_zero(row):
        if row['Cholesterol'] == 0:
            return average_cholesterol_by_age[row['AgeGroup']]
        else:
            return row['Cholesterol']

    df['Cholesterol'] = df.apply(replace_zero, axis=1)

    # Drop the temporary 'AgeGroup' column
    df.drop(columns=['AgeGroup'], inplace=True)

# Example usage:
replace_zero_cholesterol(df)

fig = px.scatter(df, y = 'Age',x='Cholesterol', color='Cholesterol' )
fig.update_layout(title=f'Buble Chart Cholesterol')
fig.show(renderer="svg")

import plotly.io as pio
pio.write_image(fig, pathIm + '/EDA13.png', width=1200, height=800,
format='png', scale=6)
```

https://www.kaggle.com/code/azizozmen/heart-failure-predict-8-classification-techniques

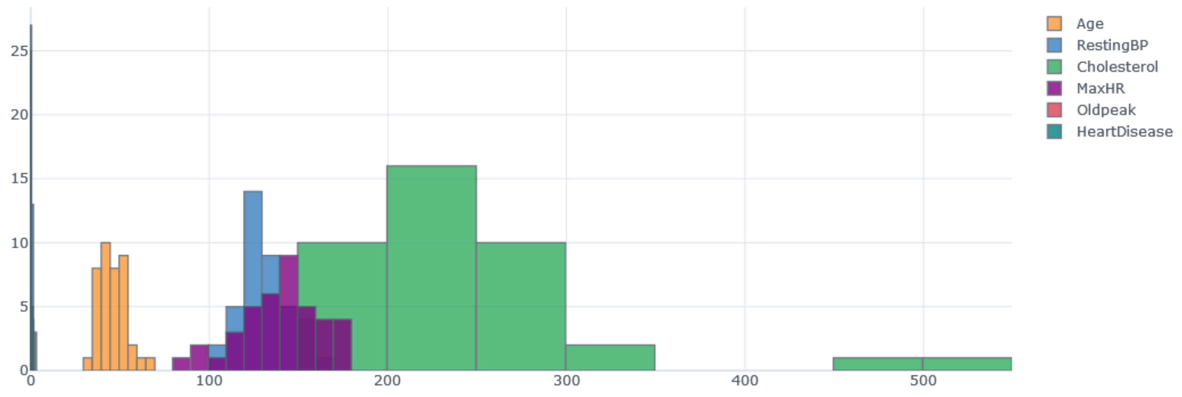
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import scipy.stats as stats
from sklearn.compose import make_column_transformer
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier, GradientBoostingRegressor,
ExtraTreesRegressor, AdaBoostClassifier
from sklearn.feature_selection import SelectKBest, SelectPercentile,
f_classif, f_regression, mutual_info_regression
from xgboost import XGBRegressor, XGBClassifier
from xgboost import plot_importance
from sklearn.pipeline import Pipeline
from sklearn.tree import plot_tree
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import LogisticRegression
```

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

#importing plotly and cufflinks in offline mode
import cufflinks as cf
import plotly.offline
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)

import warnings
warnings.filterwarnings('ignore')
warnings.warn("this will not show")
plt.rcParams["figure.figsize"] = (10,6)
pd.set_option('max_colwidth',200)

pd.set_option('display.max_columns', 200)
pd.set_option('display.float_format', lambda x: '%.3f' % x)

import colorama
from colorama import Fore, Style  # maakes strings colored
from termcolor import colored
```

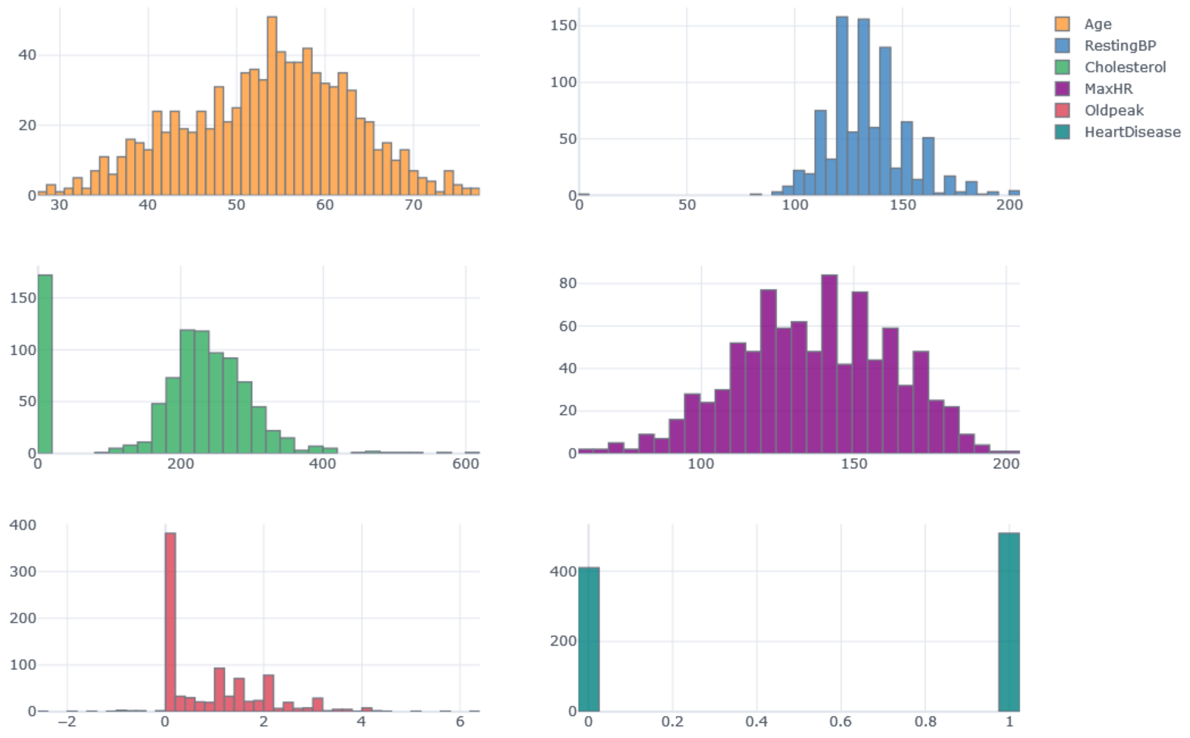
import dataframe_image as dfi
dfi.export(df[df['HeartDisease']==1].describe().T.style.background_gradient(subset=['mean','std','50%','count'],cmap='RdPu'), pathIm+"/EDA18.png", table_conversion="matplotlib")

```python
import numpy as np
import pandas as pd
import cufflinks as cf
import matplotlib.pyplot as plt
import plotly.io as pio
cf.go_online()
cf.set_config_file(offline=False)
cf.get_config_file()
path="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report"
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"

raw_df = pd.read_csv(path+'/data/heart.csv')
categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
numerical=numerical_columns
#print(numerical)
df=raw_df[numerical].head(40)
#print(df)
```

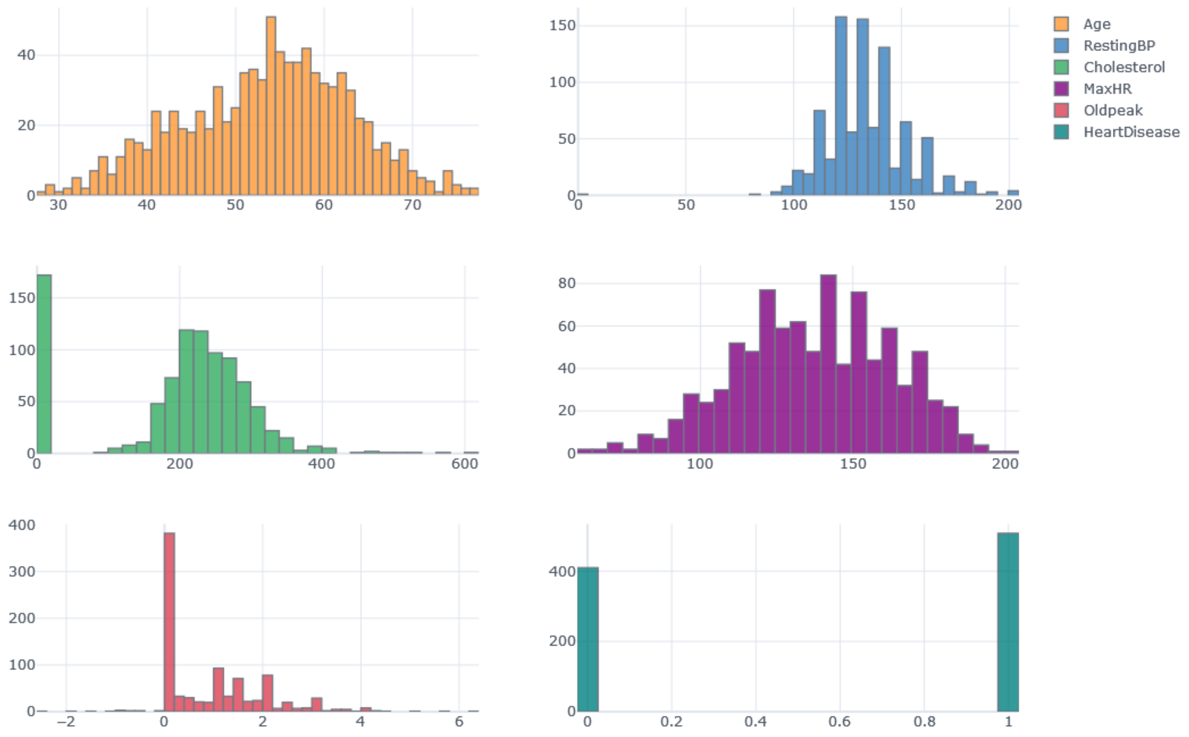
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly as pl
import cufflinks as cf
import plotly.offline as po
po.init_notebook_mode(connected=True)
cf.go_offline()
df.iplot(kind='histogram', theme='white', asImage=True,
dimensions=(800,500))
fig=df.iplot(kind='histogram', theme='white', asFigure=True,
dimensions=(1200,800))
fig.write_image(pathIm + "/EDA14.png")
```



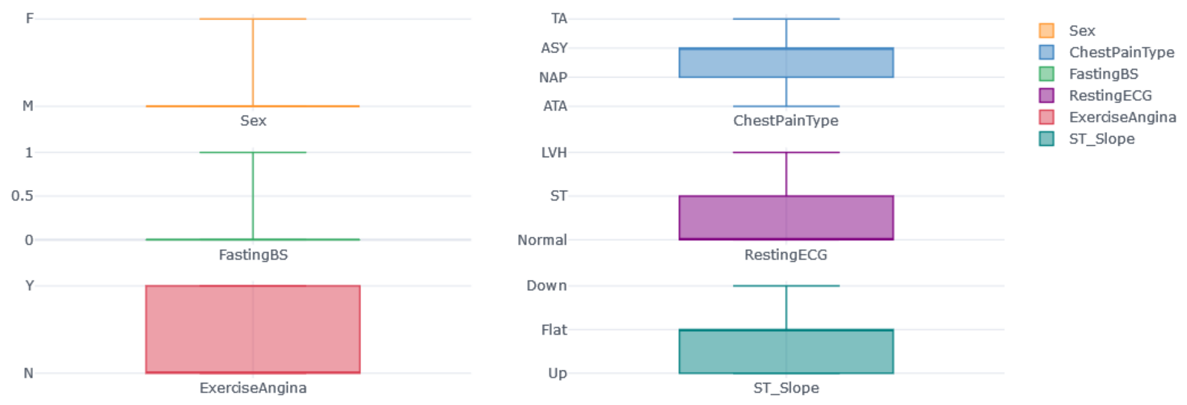
```
#fig=plt.figure(figsize=(16, 10), dpi=600)
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"
raw_df[numerical].iplot(kind='histogram', subplots=True,bins=50,
theme='white',asImage=True,dimensions=(1200,800))
fig=raw_df[numerical].iplot(kind='histogram', subplots=True,bins=50,
theme='white',asFigure=True,dimensions=(1200,800))
fig.write_image(pathIm + "/EDA15.png")
fig.show()
```



Age
RestingBP
Cholesterol
MaxHR
Oldpeak
HeartDisease



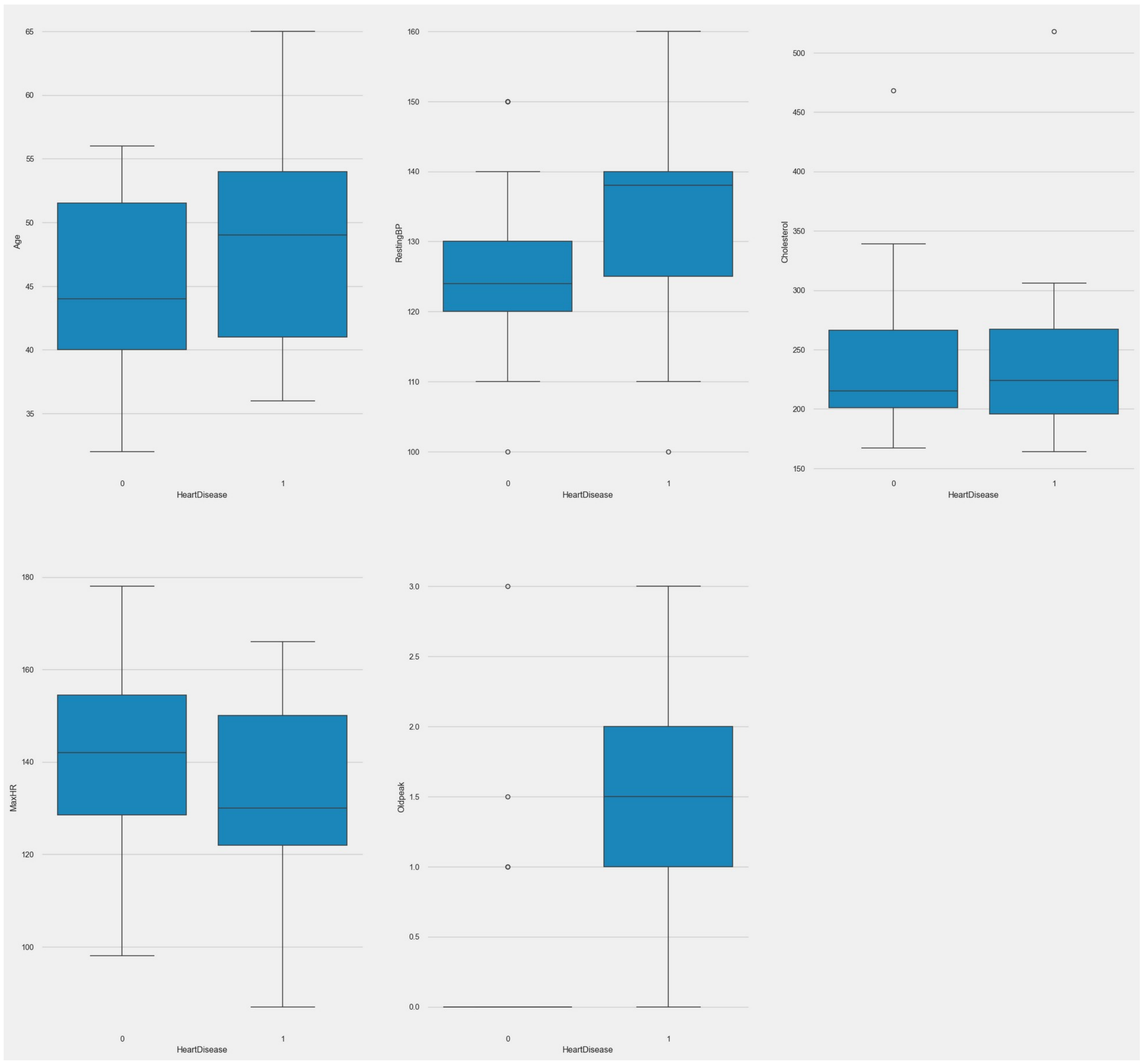
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly as pl
import cufflinks as cf
import plotly.offline as po
po.init_notebook_mode(connected=True)
cf.go_offline()

categorical = list(raw_df.loc[:,['Sex', 'ChestPainType', 'FastingBS',
'RestingECG', 'ExerciseAngina', 'ST_Slope']])
raw_df[categorical].iplot(kind='box', subplots=True,bins=50,
theme='white', asImage=True, dimensions=(800,500))
fig=raw_df[categorical].iplot(kind='box', subplots=True, bins=50,
theme='white', asFigure=True, dimensions=(800,500))
fig.write_image(pathIm + "/EDA16.png")
```



Es muss PivotTable sein!

```
# Es muss PivotTable sein!
import plotly.figure_factory as ff
import pandas as pd

cat=raw_df[categorical].value_counts()
#print(cat)
cat.to_csv(path+"/cat_count.csv")
df_sample = pd.read_csv(path + '/cat_count.csv')

colorscale = [[0, '#4d004c'],[.5, '#f2e5ff'],[1, '#ffffff']]
columnwidth=[10,20,20,10,20,10]

fig =  ff.create_table(df_sample.head(40), colorscale=colorscale)
pio.write_image(fig, pathIm+'/EDA17.png', format='png', width=800,
height=800, scale=1)
fig.show(renderer="svg")
```

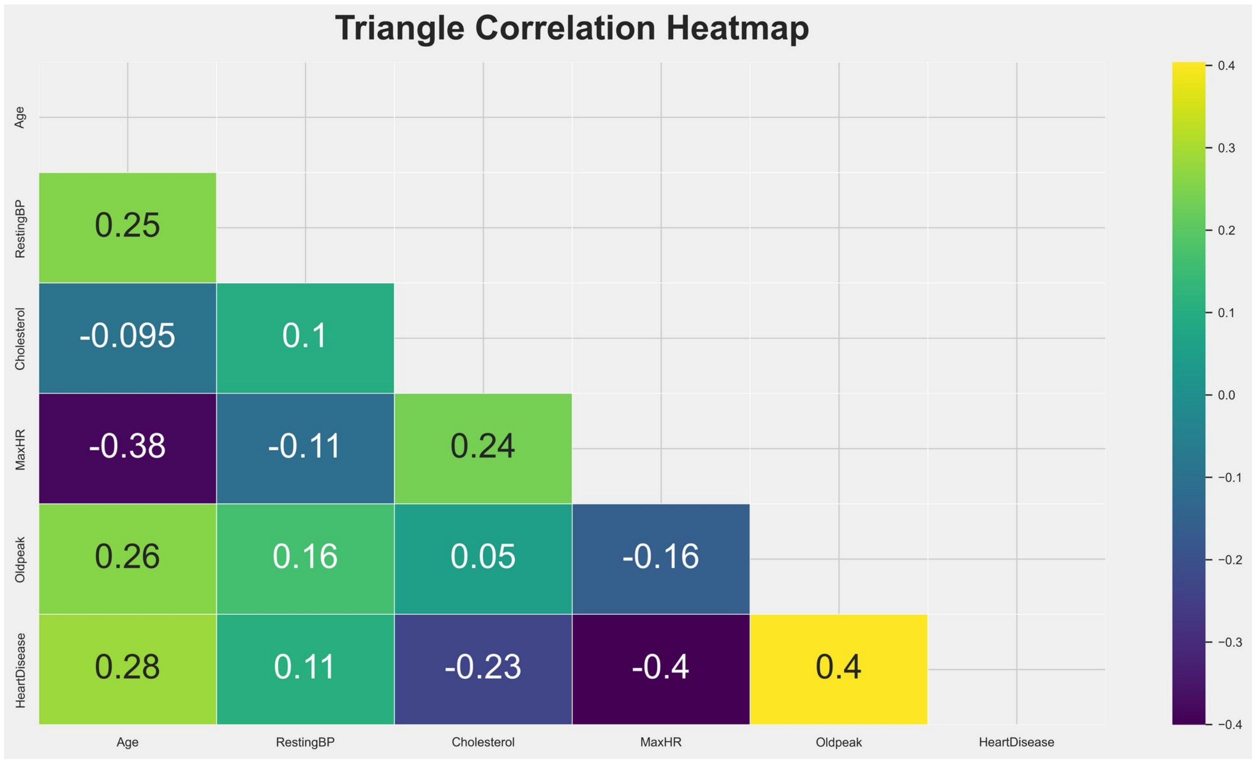
```python
import seaborn as sns
index = 0
plt.figure(figsize=(20,20))
for feature in numerical:
    if feature != "HeartDisease":
        index += 1
        plt.subplot(2, 3, index)
        sns.boxplot(x='HeartDisease', y=feature, data=df)

plt.savefig(pathIm + '/EDA18.png')
```



```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

raw_df = pd.read_csv(path+"/data/heart.csv")
raw_df.head()
```

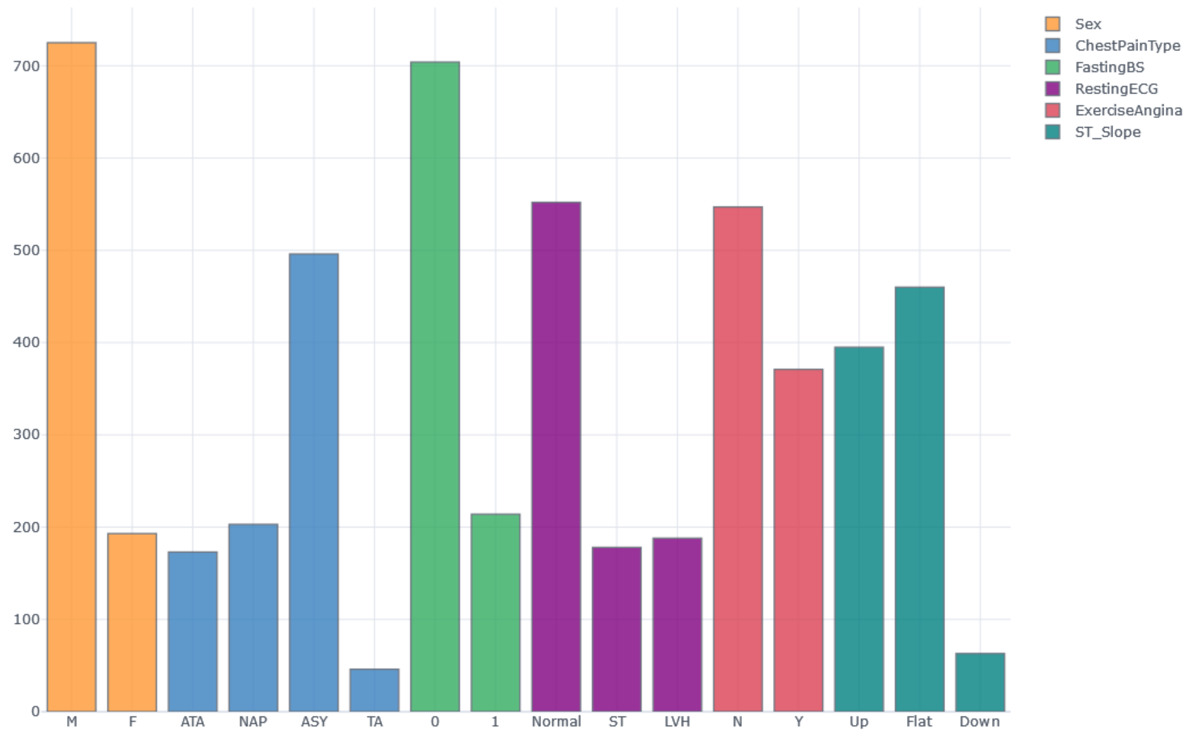
```
   Age Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG
MaxHR  \
0   40   M           ATA        140          289          0     Normal
172
1   49   F           NAP        160          180          0     Normal
156
2   37   M           ATA        130          283          0         ST
98
3   48   F           ASY        138          214          0     Normal
108
4   54   M           NAP        150          195          0     Normal
122

  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0              N    0.000       Up             0
1              N    1.000     Flat             1
2              N    0.000       Up             0
3              Y    1.500     Flat             1
4              N    0.000       Up             0
```

```
import seaborn as sns
import matplotlib.pyplot as plt

categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
numerical=numerical_columns
#print(numerical)
df=raw_df[numerical]
#print(df.head(10))

plt.figure(figsize=(15, 8), dpi=800)
corr_matrix = df.corr()
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
heatmap = sns.heatmap(corr_matrix, annot=True, mask=mask,
cmap='viridis', linewidth=.5, annot_kws={'size': 24})
heatmap.set_title('Triangle Correlation Heatmap',
fontdict={'fontsize':24, 'weight':'bold'}, pad=16);
sns.set(font_scale=1.5)
#plt.savefig('C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\ASSETS\Image\
EDA19.png')
plt.savefig(pathIm+'/EDA19.png')
```



nächste Schritt: https://www.kaggle.com/code/ohseokkim/heart-disease-could-our-model-save-lives

```
df1=raw_df[categorical]
df1.iplot(kind='hist',
theme='white',asImage=True,dimensions=(1200,800))
fig=df1.iplot(kind='hist', theme='white',asFigure=True,
dimensions=(1200,800))
fig.write_image(pathIm + "/EDA20.png")
```



700
600
500
400
300
200
100
0
M
F
ATA
NAP
ASY
TA
0
1
Normal
ST
LVH
N
Y
Up
Flat
Down
Sex
ChestPainType
FastingBS
RestingECG
ExerciseAngina
ST_Slope