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


filePath = 'heart.csv'
data = pd.read_csv(filePath)

data.head(5)
```

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

```
print("(Rows, columns): " + str(data.shape))
data.columns
```

```
(Rows, columns): (303, 14)
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
data.nunique(axis=0)# returns the number of unique values for each variable
```

```
age          41
sex           2
cp            4
trestbps     49
chol        152
fbs           2
restecg       3
thalach      91
exang         2
oldpeak      40
slope         3
ca            5
thal          4
target        2
dtype: int64
```

```
data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

```
print(data.isna().sum())
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

EDA

```
def EDA(df):

    print('\033[1m' +'EXPLORATORY DATA ANALYSIS :'+ '\033[0m\n')
    print('\033[1m' + 'Shape of the data (rows, columns):' + '\033[0m')
    print(df.shape,
          '\n--------------------------------------------------------------------------------')

    print('\033[1m' + 'All columns from the dataframe :' + '\033[0m')
    print(df.columns,
```

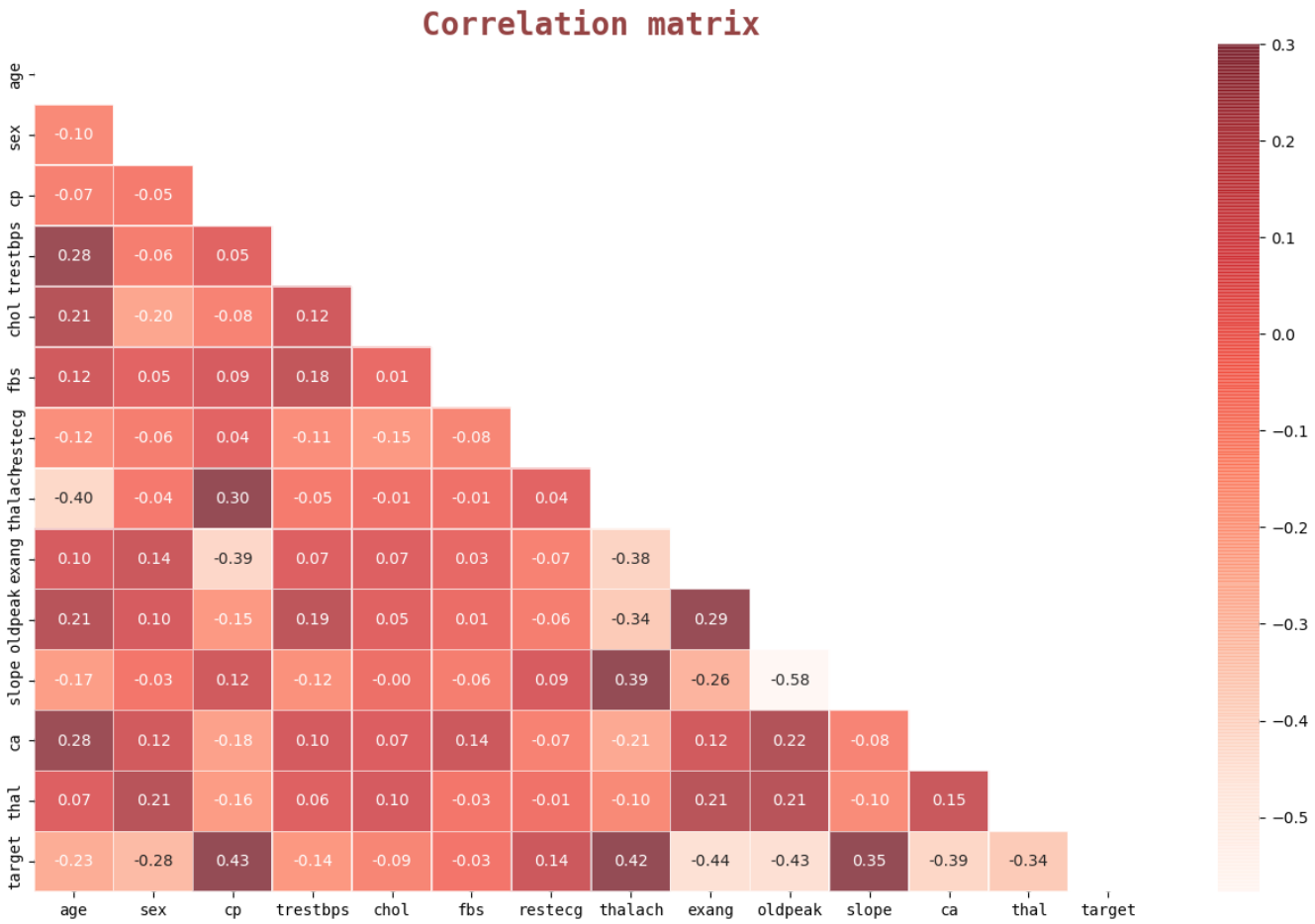
```
        '\n--------------------------------------------------------------------------------')

    print('\033[1m' + 'Datatpes and Missing values:' + '\033[0m')
    print(df.info(),
        '\n--------------------------------------------------------------------------------')

    for col in df.columns:
        print('\033[1m' + 'Unique values in {} :'.format(col) + '\033[0m',len(data[col].uniqu
    print('\n--------------------------------------------------------------------------------')

    print('\033[1m' + 'Summary statistics for the data :' + '\033[0m')
    print(df.describe(include='all'),
        '\n--------------------------------------------------------------------------------')


    print('\033[1m' + 'Memory used by the data :' + '\033[0m')
    print(df.memory_usage(),
        '\n--------------------------------------------------------------------------------')

    print('\033[1m' + 'Number of duplicate values :' + '\033[0m')
    print(df.duplicated().sum())

EDA(data)
```

```
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
None
--------------------------------------------------------------------------------

Unique values in age : 41
Unique values in sex : 2
Unique values in cp : 4
Unique values in trestbps : 49
Unique values in chol : 152
Unique values in fbs : 2
Unique values in restecg : 3
Unique values in thalach : 91
Unique values in exang : 2
Unique values in oldpeak : 40
Unique values in slope : 3
Unique values in ca : 5
Unique values in thal : 4
Unique values in target : 2

--------------------------------------------------------------------------------

Summary statistics for the data :
                age         sex          cp  ...          ca        thal      target
count  303.000000  303.000000  303.000000  ...  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  ...    0.729373    2.313531    0.544554
std      9.082101    0.466011    1.032052  ...    1.022606    0.612277    0.498835
min     29.000000    0.000000    0.000000  ...    0.000000    0.000000    0.000000
25%     47.500000    0.000000    0.000000  ...    0.000000    2.000000    0.000000
50%     55.000000    1.000000    1.000000  ...    0.000000    2.000000    1.000000
```

```
75%     61.000000    1.000000    2.000000  ...    1.000000    3.000000    1.000000
max     77.000000    1.000000    3.000000  ...    4.000000    3.000000    1.000000

[8 rows x 14 columns]
--------------------------------------------------------------------------------

Memory used by the data :
Index        128
age         2424
sex         2424
cp          2424
trestbps    2424
chol        2424
fbs         2424
restecg     2424
thalach     2424
exang       2424
oldpeak     2424
slope       2424
ca          2424
thal        2424
target      2424
dtype: int64
--------------------------------------------------------------------------------

Number of duplicate values :
1
```

```
# Correlation matrix

df1 = data.copy()

cols = df1.columns

plt.figure(figsize = (16, 10), dpi = 100)

corr = df1.corr()

mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True


sns.heatmap(corr,
            mask = mask,
            cmap = 'Reds',
            vmax=.3,
            annot = True,
            linewidths = 0.5,
            fmt = ".2f",
            alpha = 0.7)

hfont = {'fontname':'monospace'}
plt.xticks(**hfont)
```

```
plt.yticks(**hfont)
plt.title('Correlation matrix',
          family = 'monospace',
          fontsize = 20,
          weight = 'semibold',
          color = '#964545')

plt.show()
```

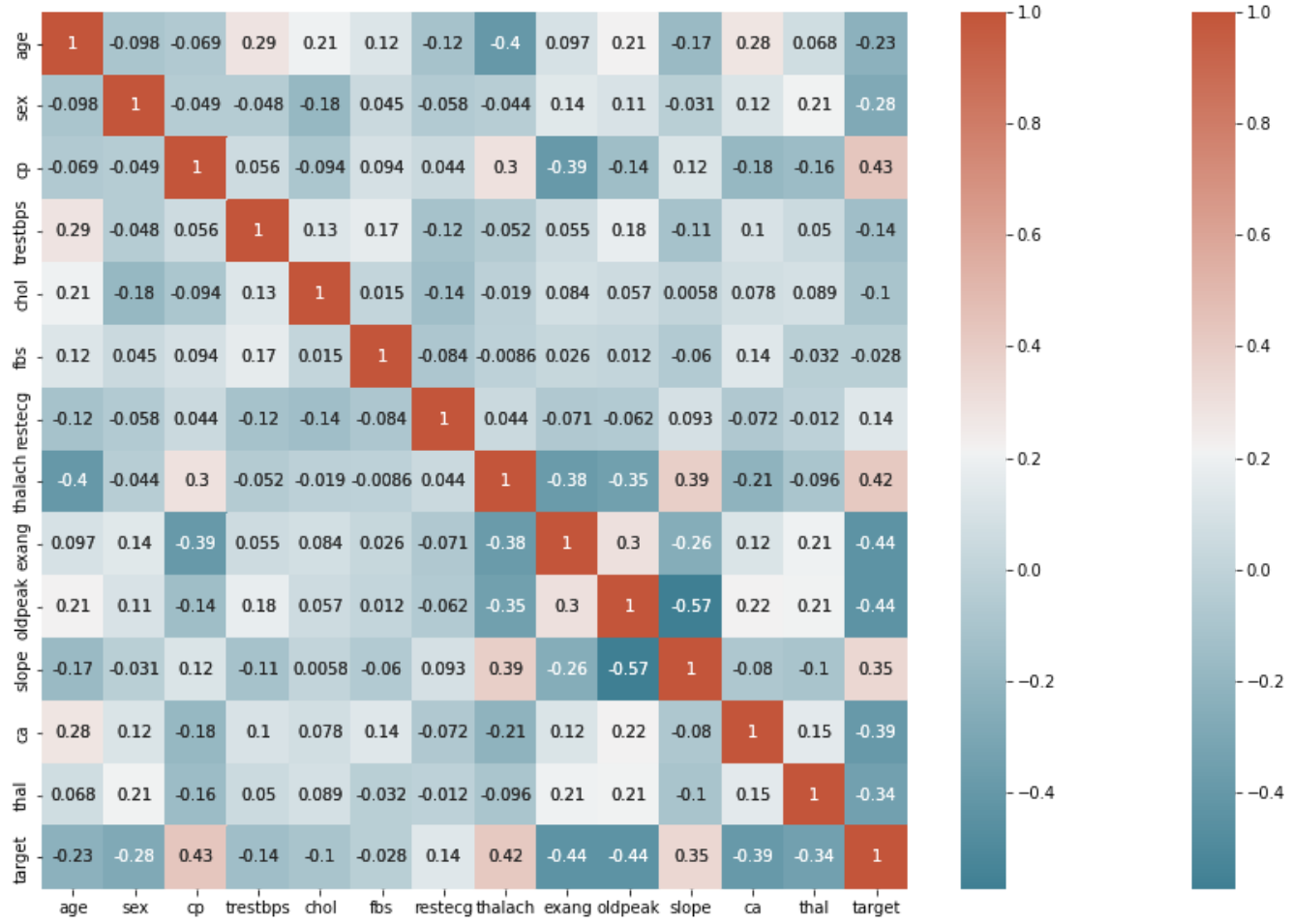


```
# Dealing with outliers

def outlier(df):
        df_ = df.copy()
        df = df.drop(['sex', 'cp', 'fbs', 'restecg', 'thalach',
       'exang', 'slope', 'ca', 'thal', 'target'], axis=1)


        q1 = df.quantile(0.25)
        q3 = df.quantile(0.75)

        iqr = q3 - q1

        lower_limit = q1 -(1.5 * iqr)
        upper_limit = q3 +(1.5 * iqr)


        for col in df.columns:
            for i in range(0,len(df[col])):
                if df[col][i] < lower_limit[col]:
                    df[col][i] = lower_limit[col]

                if df[col][i] > upper_limit[col]:
                    df[col][i] = upper_limit[col]


        for col in df.columns:
            df_[col] = df[col]

        return(df_)

data = outlier(data)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: SettingWithCopyWarning
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user
```

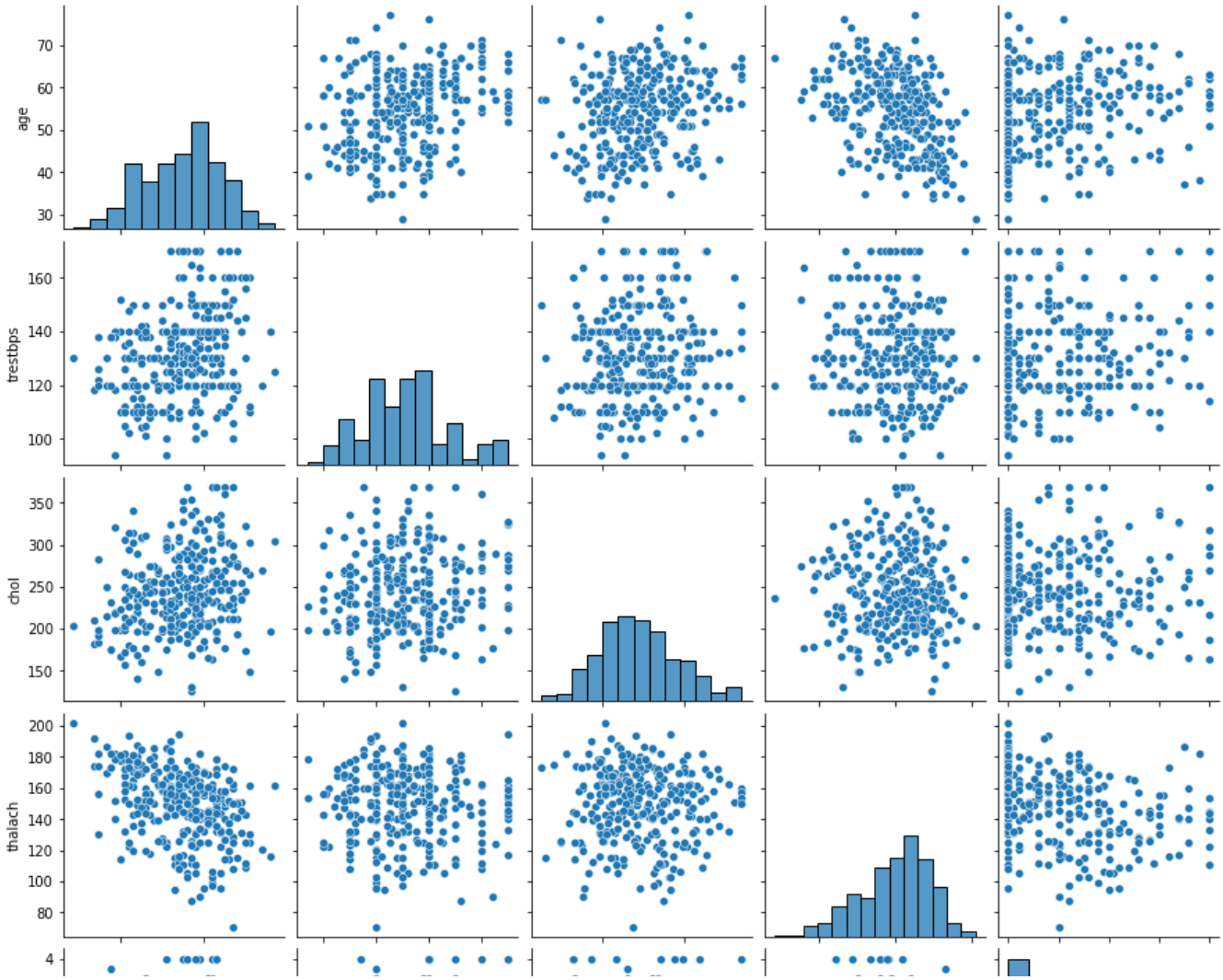
```
corr = data.corr()
plt.subplots(figsize=(15,10))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.di
sns.heatmap(corr, xticklabels=corr.columns,
            yticklabels=corr.columns,
            annot=True,
            cmap=sns.diverging_palette(220, 20, as_cmap=True))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbda4f6d210>
```



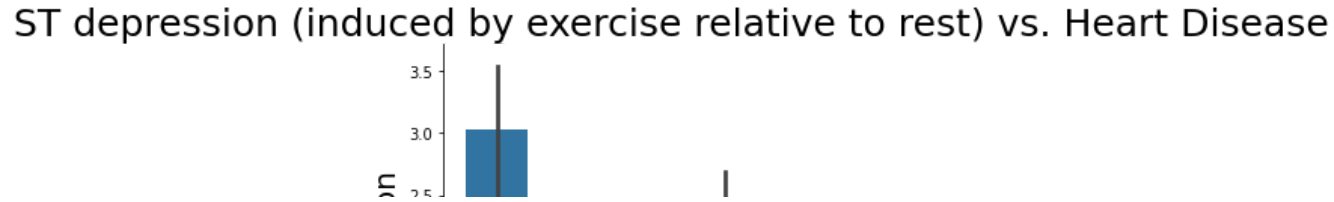

```
subData = data[['age','trestbps','chol','thalach','oldpeak']]
sns.pairplot(subData)
```

```
<seaborn.axisgrid.PairGrid at 0x7fbda4daf450>
```



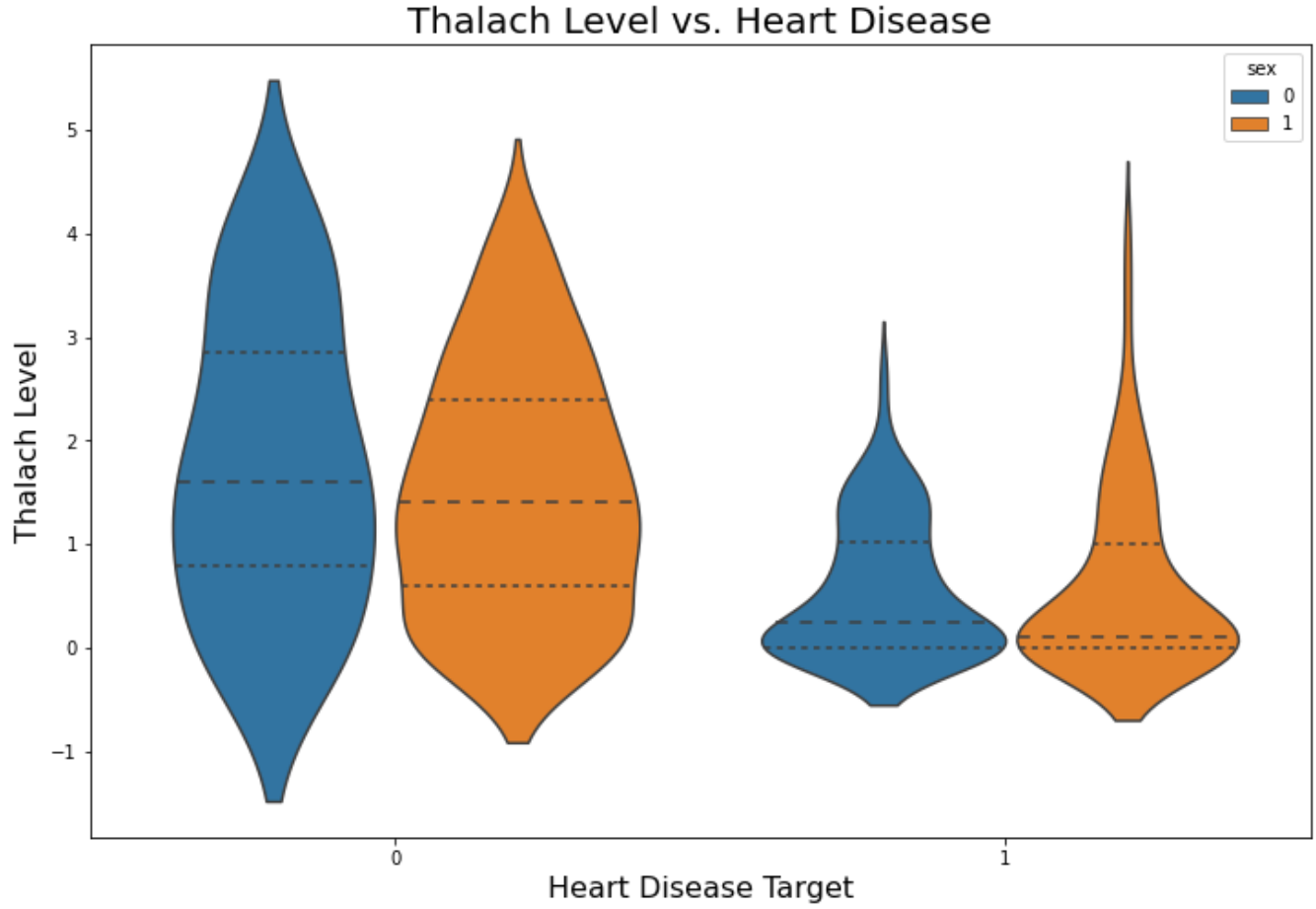

```
sns.catplot(x="target", y="oldpeak", hue="slope", kind="bar", data=data);

plt.title('ST depression (induced by exercise relative to rest) vs. Heart Disease',size=25)
plt.xlabel('Heart Disease',size=20)
plt.ylabel('ST depression',size=20)
```

```
Text(26.426458333333343, 0.5, 'ST depression')
```



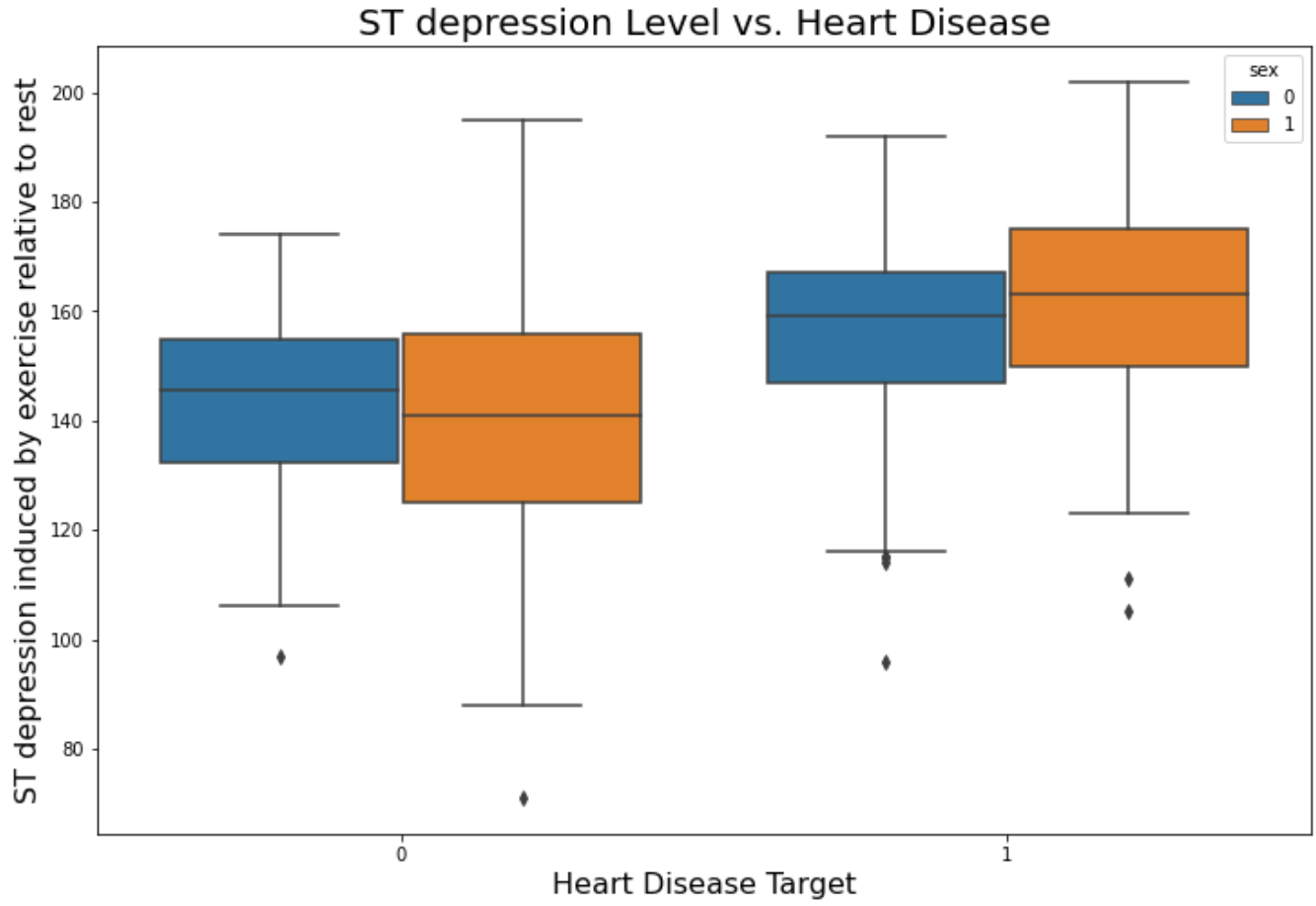

```
plt.figure(figsize=(12,8))
sns.violinplot(x= 'target', y= 'oldpeak',hue="sex", inner='quartile',data= data )
print("thalach: The person's maximum heart rate achieved\n")
plt.title("Thalach Level vs. Heart Disease",fontsize=20)
plt.xlabel("Heart Disease Target", fontsize=16)
plt.ylabel("Thalach Level", fontsize=16)
```

```
thalach: The person's maximum heart rate achieved

Text(0, 0.5, 'Thalach Level')
```




```
plt.figure(figsize=(12,8))
sns.boxplot(x= 'target', y= 'thalach',hue="sex", data=data )
plt.title("ST depression Level vs. Heart Disease", fontsize=20)
plt.xlabel("Heart Disease Target",fontsize=16)
plt.ylabel("ST depression induced by exercise relative to rest", fontsize=16)
```

```
Text(0, 0.5, 'ST depression induced by exercise relative to rest')
```



Filtering data by positive & negative Heart Disease patient

```
# Filtering data by POSITIVE Heart Disease patient
pos_data = data[data['target']==1]
pos_data.describe()
```

```
# Filtering data by NEGATIVE Heart Disease patient
neg_data = data[data['target']==0]
neg_data.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg
count	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000
mean	56.601449	0.826087	0.478261	133.789855	250.521739	0.159420	0.449275
std	7.962082	0.380416	0.905920	17.164916	47.842994	0.367401	0.541321
min	35.000000	0.000000	0.000000	100.000000	131.000000	0.000000	0.000000
25%	52.000000	1.000000	0.000000	120.000000	217.250000	0.000000	0.000000
50%	58.000000	1.000000	0.000000	130.000000	249.000000	0.000000	0.000000
75%	62.000000	1.000000	0.000000	144.750000	283.000000	0.000000	1.000000
max	77.000000	1.000000	3.000000	170.000000	369.000000	1.000000	2.000000

```
print("(Positive Patients ST depression): " + str(pos_data['oldpeak'].mean()))
print("(Negative Patients ST depression): " + str(neg_data['oldpeak'].mean()))
```

```
(Positive Patients ST depression): 0.5818181818181817
(Negative Patients ST depression): 1.5536231884057976
```

```
print("(Positive Patients thalach): " + str(pos_data['thalach'].mean()))
print("(Negative Patients thalach): " + str(neg_data['thalach'].mean()))
```

```
(Positive Patients thalach): 158.46666666666667
(Negative Patients thalach): 139.1014492753623
```

Machine Learning & predictive analysis

Prepare Data for Modeling

```
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 1)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Modeling /Training

Random forest

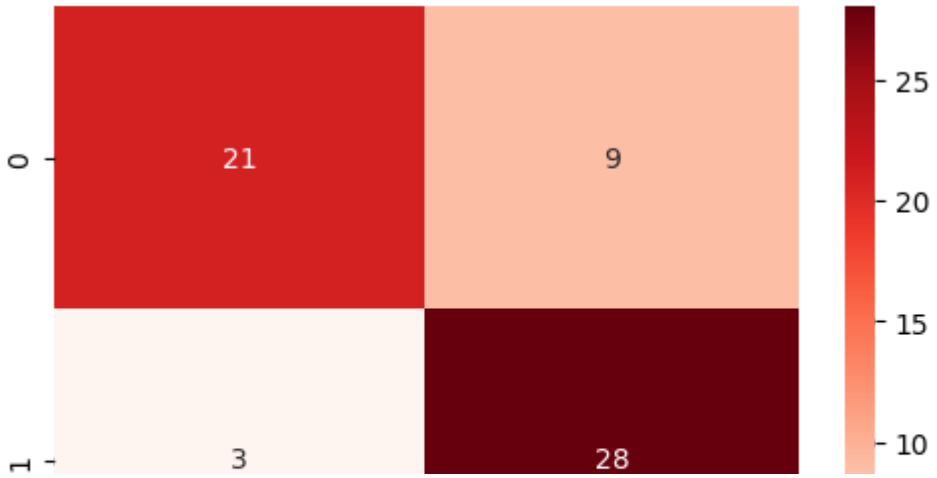
```
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model6 = RandomForestClassifier(random_state=1)# get instance of model
model6.fit(x_train, y_train) # Train/Fit model

y_pred6 = model6.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred6)) # output accuracy
```

```
                  precision    recall  f1-score   support

               0       0.88      0.70      0.78        30
               1       0.76      0.90      0.82        31

        accuracy                           0.80        61
       macro avg       0.82      0.80      0.80        61
    weighted avg       0.81      0.80      0.80        61
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test,y_pred6)
print('\033[1m' +'Confusion Matrix : '+ '\033[0m')
plt.figure(dpi=100)
sns.heatmap(cm, cmap = 'Reds',annot = True, fmt='d')
plt.show()

print('\naccuracy_score')
accuracy_score(y_test, y_pred6)
```

```
Confusion Matrix :
```



Feature Importance

```
# get importance
importance = model6.feature_importances_

# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))

     Feature: 0, Score: 0.07789
     Feature: 1, Score: 0.04292
     Feature: 2, Score: 0.16512
     Feature: 3, Score: 0.07340
     Feature: 4, Score: 0.07773
     Feature: 5, Score: 0.00864
     Feature: 6, Score: 0.01982
     Feature: 7, Score: 0.12768
     Feature: 8, Score: 0.06877
     Feature: 9, Score: 0.09907
     Feature: 10, Score: 0.04727
     Feature: 11, Score: 0.11704
     Feature: 12, Score: 0.07464


index= data.columns[:-1]
importance = pd.Series(model6.feature_importances_, index=index)
importance.nlargest(13).plot(kind='barh', colormap='winter')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbda3cb2490>
```


```
print(model6.predict(sc.transform([[20,1,2,110,230,1,1,140,1,2.2,2,0,2]])))
```

```
[1]
```

```
y_pred = model6.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[0 0]
[0 0]
[0 0]
[1 1]
[0 0]
[1 1]
[1 1]
[0 0]
[1 0]
[0 0]
[0 0]
[1 0]
[1 1]
[0 0]
[1 1]
[1 0]
[1 1]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[0 1]
[0 0]
[1 0]
[0 1]
[1 1]

[0 0]
[0 1]
[0 0]
[1 0]
```

```
[1 0]
[0 0]
[1 1]
[1 0]
[1 1]
[1 1]
[1 0]
[0 0]
[1 1]
[1 1]
[1 1]
[1 1]
[0 0]
[1 0]
[0 0]
[1 1]]
```