Healthcare

Dataset

CE Review

Instruction:

- Download the CEP 1_ Dataset.csv using the link given in the Healthcare project problem statement and upload it in the lab using the up arrow shown below View Tab.
- In this dataset, there are 14 attributes with more than 4000 data points.

Data Dictionary

Variable
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target

Steps to Perform

- **Preliminary analysis:**
  - Perform preliminary data inspection and report the findings as structure of the data, missing values, duplicates, etc.
  - Based on the findings from the previous question, remove duplicates (if any), treat missing values using an appropriate strategy.

- **Prepare an informative report about the data, explaining the distribution of the disease and related factors. You can use the below approach to achieve this objective**
  - Get a preliminary statistical summary of the data. Explore the measures of central tendencies and the spread of the data overall.
  - Identify the data variables that might be categorical in nature. Describe and explore these variables using appropriate tools. For e.g. count plot
  - Study the occurrence of CVD across Age.
  - Study the composition of overall patients with respect to Gender.
  - Can we detect heart attacks based on anomalies in the Resting Blood Pressure of the patient?
  - Describe the relationship between Cholesterol levels and our target variable.
  - What can be concluded from the relationship between peak exercise and the occurrence of heart attacks?
  - Is thalassemia a major cause of CVD?
  - How do other factors determine the occurrence of CVD?
  - Use a pair plot to understand the relationship between all the given variables.
- Build a baseline model to predict using a Logistic Regression and Random Forest, and explore the results. While using correlation analysis and logistic regression (leveraging standard error and p-values from statsmodels) for feature selection.

Solution

Import Libraries

CE Review

There are a few libraries required to build a machine learning model:

- pandas - It helps to retrieve datasets, handle missing data and do data wrangling.
- Numpy - It helps to perform numerical operations in the dataset.
- warnings - It helps to neglect the unwanted popups or exceptions.
- matplotlib - It helps in data visualization.
- seaborn - It also helps in data visualization and exploratory data analysis.
- matplotlib inline - It is used to plot the charts or graphs in the notebook itself.

```
#Select the cell and click on run icon
import pandas as pd
import numpy as np
import warnings
import matplotlib.pyplot as plt
import seaborn as sns
warnings.filterwarnings('ignore')
```

```
plt.style.use('ggplot')
%matplotlib inline
```

CE Review

Note:

- The **`data`** is a dataframe to store the data imported from the csv as rows and columns table format.
- The **head()** function helps to view the first few data present in the **`data`** dataframe.

Import Dataset

```
#Select the cell and click on run icon
data = pd.read_excel('CEP 1_ Dataset.xlsx')
data.head()
```

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

   ca  thal  target
0   0     1       1
1   0     2       1
2   0     2       1
3   0     2       1
4   0     2       1
```

1. Preliminary Analysis

- Perform preliminary data inspection and report the findings as the structure of the data, missing values, duplicates, etc.
- Based on the findings from the previous question remove duplicates (if any), treat missing values using the appropriate strategy.

Understanding the Data

```
#Select the cell and click on run icon
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Identifying the duplicated data present in the dataset

```
#Select the cell and click on run icon
data.duplicated().sum()
```

```
1
```

Removing the duplicate data present in the dataset

```
#Select the cell and click on run icon
data.drop_duplicates(inplace = True)
data.reset_index(drop = True, inplace = True)
```

Identify the total number of duplicates after performing duplication operation

Exactly 1 duplicate row may be removed

```
#Select the cell and click on run icon
#checking duplicate again
data.duplicated().sum()
```

```
0
```

2. Statistical Analysis

Primary Statistical Summary

Explore the measures of central tendencies and the spread of the data overall.

```
#Select the cell and click on run icon
data.describe()
```

```
             age         sex          cp    trestbps        chol
fbs  \
count  302.00000  302.000000  302.000000  302.000000  302.000000
302.000000
mean    54.42053    0.682119    0.963576  131.602649  246.500000
0.149007
std      9.04797    0.466426    1.032044   17.563394   51.753489
0.356686
min     29.00000    0.000000    0.000000   94.000000  126.000000
0.000000
25%     48.00000    0.000000    0.000000  120.000000  211.000000
0.000000
50%     55.50000    1.000000    1.000000  130.000000  240.500000
0.000000
75%     61.00000    1.000000    2.000000  140.000000  274.750000
0.000000
max     77.00000    1.000000    3.000000  200.000000  564.000000
1.000000

          restecg     thalach       exang     oldpeak       slope
ca  \
count  302.000000  302.000000  302.000000  302.000000  302.000000
302.000000
mean     0.526490  149.569536    0.327815    1.043046    1.397351
0.718543
std      0.526027   22.903527    0.470196    1.161452    0.616274
1.006748
min      0.000000   71.000000    0.000000    0.000000    0.000000
0.000000
25%      0.000000  133.250000    0.000000    0.000000    1.000000
0.000000
50%      1.000000  152.500000    0.000000    0.800000    1.000000
0.000000
75%      1.000000  166.000000    1.000000    1.600000    2.000000
1.000000
max      2.000000  202.000000    1.000000    6.200000    2.000000
4.000000

             thal      target
count  302.000000  302.000000
mean     2.314570    0.543046
std      0.613026    0.498970
min      0.000000    0.000000
25%      2.000000    0.000000
50%      2.000000    1.000000
75%      3.000000    1.000000
max      3.000000    1.000000
```

Changing Variable Names

Making variable names to be more representative names

```
#Select the cell and click on run icon
data.rename({'cp' :'chest_pain_type',
             'trestbps':'resting_blood_pressure',
             'chol':'cholesterol',
             'fbs' :'fasting_blood_sugar',
             'restecg':'resting_ecg',
             'thalach' : 'max_heart_rate',
             'exang':'exercise_induced_angina',
             'oldpeak':'st_depression',
             'slope':'st_slope',
             'ca' : 'major_vessels',
             'thal' :'thalessimia' },axis = 1, inplace = True)
```

```
#Select the cell and click on run icon
data.columns
```

```
Index(['age', 'sex', 'chest_pain_type', 'resting_blood_pressure',
       'cholesterol', 'fasting_blood_sugar', 'resting_ecg',
'max_heart_rate',
       'exercise_induced_angina', 'st_depression', 'st_slope',
'major_vessels',
       'thalessimia', 'target'],
      dtype='object')
```

Creating a Separate List of Categorical Variables

Note: Identify the data variables which might be categorical in nature.

```
#Select the cell and click on run icon
cat =
['sex','chest_pain_type','fasting_blood_sugar','exercise_induced_angin
a','st_slope','thalessimia']
```

Secondary Statistical Summary

Note: Categorical variables only

```
#Select the cell and click on run icon
data.loc[ : , ~data.columns.isin(cat)].describe()
```

```
              age  resting_blood_pressure  cholesterol  resting_ecg  \
count  302.00000              302.000000   302.000000   302.000000
mean    54.42053              131.602649   246.500000     0.526490
std      9.04797               17.563394    51.753489     0.526027
min     29.00000               94.000000   126.000000     0.000000
25%     48.00000              120.000000   211.000000     0.000000
50%     55.50000              130.000000   240.500000     1.000000
75%     61.00000              140.000000   274.750000     1.000000
```

```
max     77.00000              200.000000   564.000000     2.000000
```

```
       max_heart_rate  st_depression  major_vessels      target
count      302.000000     302.000000     302.000000  302.000000
mean       149.569536       1.043046       0.718543    0.543046
std         22.903527       1.161452       1.006748    0.498970
min         71.000000       0.000000       0.000000    0.000000
25%        133.250000       0.000000       0.000000    0.000000
50%        152.500000       0.800000       0.000000    1.000000
75%        166.000000       1.600000       1.000000    1.000000
max        202.000000       6.200000       4.000000    1.000000
```

```
#Select the cell and click on run icon
desc= pd.DataFrame(index = cat)
desc['nuinque'] = data[cat].apply(lambda x : x.nunique(), axis = 0)
desc['unique'] = 0
for i in cat :
    desc.loc[i,'unique'] = str(list(data[i].value_counts().index))
desc.T
```

```
            sex chest_pain_type fasting_blood_sugar
exercise_induced_angina  \
nuinque       2               4                   2
2
unique   [1, 0]    [0, 2, 1, 3]              [0, 1]
[0, 1]
```

```
          st_slope   thalessimia
nuinque          3             4
unique   [2, 1, 0]  [2, 3, 1, 0]
```

```
#Select the cell and click on run icon
data.thalessimia.value_counts()
```

```
2    165
3    117
1     18
0      2
Name: thalessimia, dtype: int64
```

```
#Select the cell and click on run icon
data.loc[data.thalessimia==0 , 'thalessimia'] = 2
```

Note :

- Thalessimia has 4 unique categories according to data however, in the description there are only 3 categories
- There are 2 records which are identified as '0'; these can be seen as missing values and hence need to be imputed.
- Imputation can be put in the category with modal value of '2'

Converting the Numeric Categories to Relevent Descriptors

```
#Select the cell and click on run icon
data.loc[data.sex == 0 , 'sex'] = 'female'
data.loc[data.sex == 1, 'sex'] = 'male'

data.loc[data.chest_pain_type == 0,'chest_pain_type'] = 'typical
angina'
data.loc[data.chest_pain_type == 1,'chest_pain_type'] = 'atypical
angina'
data.loc[data.chest_pain_type == 2,'chest_pain_type'] = 'non-anginal
pain'
data.loc[data.chest_pain_type == 3,'chest_pain_type'] = 'asymptomatic'

data.loc[data.fasting_blood_sugar == 0,'fasting_blood_sugar'] = '<
120mg/ml'
data.loc[data.fasting_blood_sugar == 1,'fasting_blood_sugar'] = '>
120mg/ml'

data.loc[data.resting_ecg == 0, 'resting_ecg']  = 'normal'
data.loc[data.resting_ecg == 1 , 'resting_ecg'] = 'abnormal'
data.loc[data.resting_ecg == 2 , 'resting_ecg'] = 'hyper'

data.loc[data.exercise_induced_angina == 0, 'exercise_induced_angina']
= 'no'
data.loc[data.exercise_induced_angina == 1, 'exercise_induced_angina']
= 'yes'

data.loc[data.st_slope == 0, 'st_slope'] = 'upsloping'
data.loc[data.st_slope == 1, 'st_slope'] = 'flat'
data.loc[data.st_slope == 2, 'st_slope'] = 'downsloping'

data.loc[data.thalessimia == 1,'thalessimia'] = 'normal'
data.loc[data.thalessimia == 2,'thalessimia'] = 'fixed defect'
data.loc[data.thalessimia == 3,'thalessimia'] = 'reversable defect'

#data.loc[data.target == 0, 'target']= 'Disease -'
#data.loc[data.target == 1, 'target']= 'Disease +'

#Select the cell and click on run icon
dsprsnt = data[data.target == 1].copy()
dsabsnt = data[data.target == 0].copy()
```

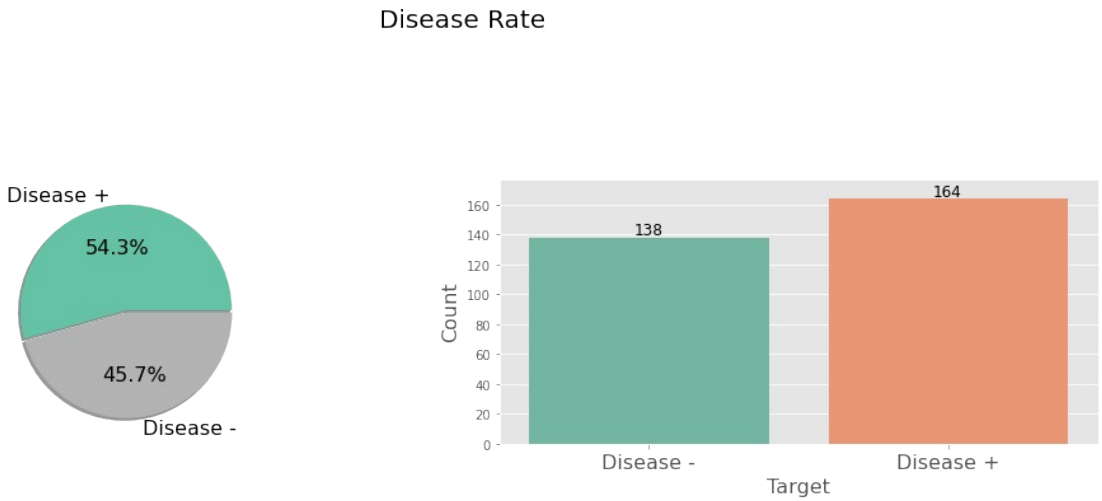
Target Distribution

```
#Select the cell and click on run icon
vc = data.target.value_counts()
vc
```

```
1    164
0    138
Name: target, dtype: int64
```

Checking the Bias

```
#Select the cell and click on run icon
f,axes = plt.subplots(1,2, figsize = (15,6))
# plot no. 1
vc.plot.pie(ax = axes[0], radius = 1, cmap = 'Set2' , explode =
[0.01,0.01], shadow = True, autopct = '%1.1f%%',
           textprops = {'family': 'DejaVu Sans','color':
'black','size': 16}, labels = ['Disease +','Disease -'])
axes[0].set_ylabel('')

# plot no. 2
sns.countplot(data.target,ax = axes[1],palette= 'Set2')
for i in range(len(vc)):
    axes[1].annotate(vc[i], (i-0.05,vc[i]+2), fontsize = 12)
axes[1].set_ylim(0,axes[1].set_ylim()[1]+5)
axes[1].set_xlabel('Target',fontsize = 16, family = 'DejaVu Sans')
axes[1].set_ylabel('Count',fontsize = 16, family = 'DejaVu Sans')
axes[1].set_xticklabels( ['Disease -','Disease +'], fontsize = 16,
family = 'DejaVu Sans')
f.suptitle('Disease Rate\n\n', fontsize = 20, family = 'DejaVu Sans')
plt.tight_layout(pad = 4)
plt.show();
```



Observation :

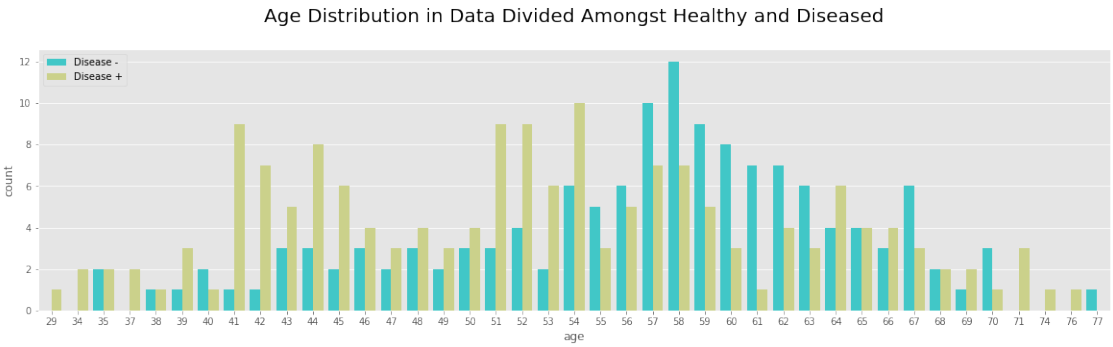
- To understand the data better, it is necessary to understand our target variable.
- The data collected has almost equal representation of Diseased and Healthy samples.

3. Exploratory Data Analysis

1. Occurrence of CVD across Age

```
#Select the cell and click on run icon
plt.figure(figsize = (20,5))
sns.countplot(data.age, hue = data.target, palette='rainbow')
```

```
plt.legend(['Disease -','Disease +'], loc = 'upper left')
plt.title('\nAge Distribution in Data Divided Amongst Healthy and
Diseased\n', fontsize = 20)
plt.show()
```

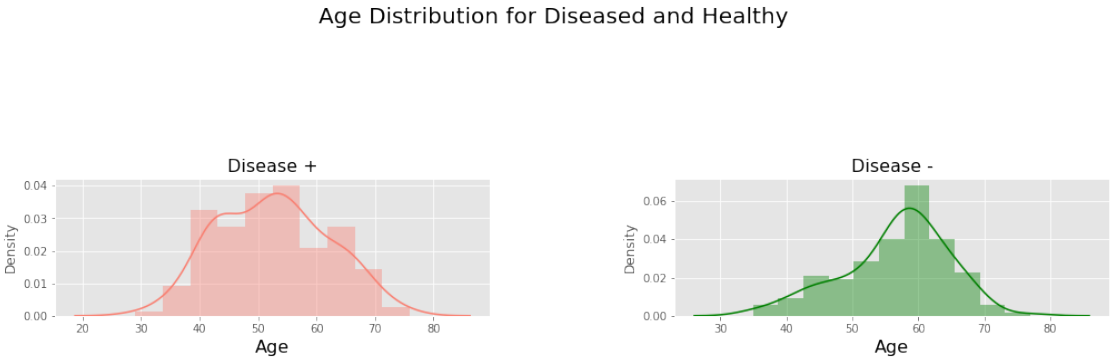


Observation:

- Based on the graph, we can see that the number of people in the various age group who are healthy and diseased.
- Healthy is represented as disease -.
- Disase is represented as disease +.

2. Age Distribution for Diseased and Healthy

```
#Select the cell and click on run icon
f,axes = plt.subplots(1,2, figsize = (15,5))
sns.distplot(dsprsnt.age,ax = axes[0], color = 'salmon')
sns.distplot(dsabsnt.age, ax = axes[1], color = 'green')
axes[0].set_title('Disease +',fontdict = {'family': 'DejaVu
Sans','size': 16})
axes[1].set_title('Disease -',fontdict = {'family': 'DejaVu
Sans','size': 16})
axes[0].set_xlabel('Age', fontdict = {'family': 'DejaVu Sans','color':
'black','weight': 'normal','size': 16})
axes[1].set_xlabel('Age',fontdict = {'family': 'DejaVu Sans','color':
'black','weight': 'normal','size': 16})
f.suptitle('Age Distribution for Diseased and Healthy\n\n ',fontsize=
20)
plt.tight_layout(w_pad= 12, pad = 4 )
plt.show()
```

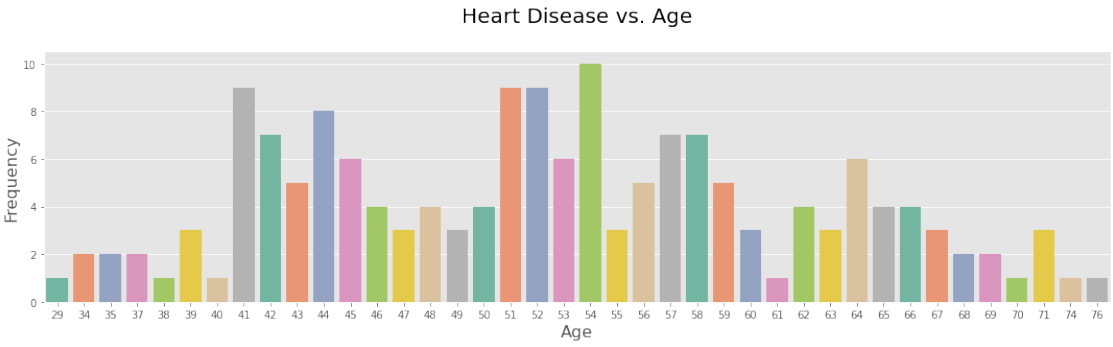


Observation:

- Based on the graph, we can see that the accumulation of total number of people present in particular age group for healthy and disease.
- Healthy is represented as disease -.
- Disase is represented as disease +.

3. Heart Disease vs. Age

```
#Select the cell and click on run icon
plt.figure(figsize = (15,5))
sns.countplot(dsprsnt.age, palette='Set2')
plt.title('\nHeart Disease vs. Age\n',family='DejaVu Sans',fontsize=
20)
plt.tight_layout( )
plt.xlabel('Age',family='DejaVu Sans',fontsize= 16)
plt.ylabel('Frequency',family='DejaVu Sans',fontsize= 16)
plt.show()
```



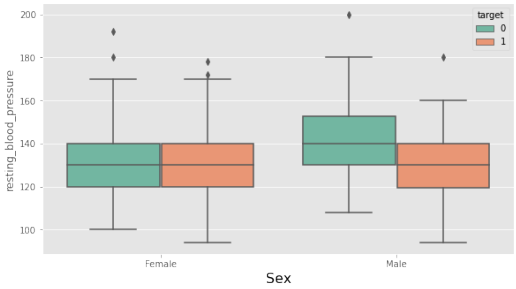
Observation:

- The chances of heart attack across age has intermittent peaks
- Tendency of disease increases after 40
- The age groups 41 to 45 and 51 to 54 have the highest chances of heart attack

4. Resting Blood Pressure of Gender vs. Target

```
#Select the cell and click on run icon
f,axes = plt.subplots(1,2,figsize = (18,5))
```

```
sns.violinplot(y = 'resting_blood_pressure', x = 'sex',hue =
'target',data = data, split = True, palette= 'Accent', ax = axes[0])
axes[0].set_xlabel('Sex', fontdict = {'family': 'DejaVu Sans','color':
'black','weight': 'normal','size': 16})
axes[0].set_ylabel('Resting Blood Pressure', fontdict = {'family':
'DejaVu Sans','color': 'black','weight': 'normal','size': 16})
axes[0].set_xticklabels(['Female','Male'])
sns.boxplot(x = data.sex,  y = data.resting_blood_pressure, hue =
data.target, ax = axes[1], palette='Set2')
axes[1].set_xticklabels(['Female','Male'])
axes[1].set_xlabel('Sex',fontdict = {'family': 'DejaVu Sans','color':
'black','weight': 'normal','size': 16})
plt.tight_layout(w_pad = 10, pad = 2)
plt.show()
```



Observation:

- Based on the graph, we can see the target who has cardiovascular diseases have resting blood pressure.
- It shows category wise data of male and female.

Defining Plots

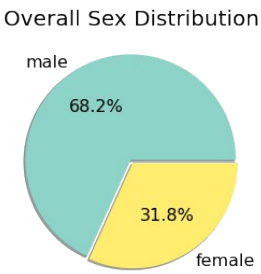
```
#Select the cell and click on run icon
#1. cat_plot
def cat_plot(var):
    f,axes = plt.subplots(1,2, figsize = (18,5))
    vc = data[var].value_counts()
    nouniq = data[var].nunique()
    # overall pie
    vc.plot.pie(radius = 1.25,ax = axes[0], cmap = 'Set3', autopct =
'%0.1f%%',
                                        textprops = {'family':
'times','color': 'black','size': 16},
                                        explode = [0.02]*nouniq,shadow
= True,)
    axes[0].set_ylabel('')
    axes[0].set_title('Overall {} Distribution\
n'.format(var.capitalize()),family='DejaVu Sans',fontsize= 20)

    # count plot
    #pd.crosstab(data[var], data.target).plot.bar(cmap = 'Set2', ax =
```

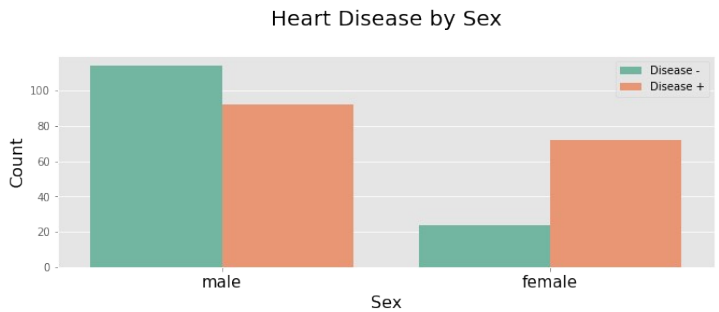
```
axes[1])
    sns.countplot(x = data[var], hue = data.target, ax = axes[1],
palette='Set2')
    plt.xticks( fontsize = 15, color = 'black' , family =  'DejaVu
Sans', rotation = 0)
    axes[1].set_xlabel(var.capitalize(),fontsize = 16, color = 'black'
, family =  'DejaVu Sans', rotation = 0)
    axes[1].set_ylabel('Count',fontsize = 16, color = 'black' , family
=  'DejaVu Sans')
    axes[1].legend(['Disease -','Disease +'])
    axes[1].set_title('Heart Disease by {}\
n'.format( var.capitalize()) ,family='DejaVu Sans',fontsize= 20)
    plt.tight_layout(pad = 4 )
    plt.show()

#Select the cell and click on run icon
#2. real_distribution
def real_distribution(var):
    f,axes = plt.subplots(1,2, figsize = (15,5))
    sns.distplot(dsprsnt[var],ax = axes[0], color = 'salmon')
    sns.distplot(dsabsnt[var], ax = axes[0], color = 'green')
    sns.boxplot(y = data[var], x = data.target, ax = axes[1],
palette='Set2')
    axes[0].set_xlabel(var, fontdict = {'family': 'DejaVu
Sans','color': 'black','weight': 'normal','size': 16})
    axes[1].set_ylabel(var, fontdict = {'family': 'DejaVu
Sans','color': 'black','weight': 'normal','size': 16})
    axes[1].set_xlabel('Target', fontdict = {'family': 'DejaVu
Sans','color': 'black','weight': 'normal','size': 16})
    axes[1].set_xticklabels(['Disease -','Disease +'])
    f.suptitle('{} vs Disease\n\n '.format(var.capitalize()),fontsize=
20, family = 'DejaVu Sans')
    plt.tight_layout(w_pad= 12, pad = 4 )
    plt.show()
```

5. Composition of Overall Patients with respect to Gender

```
#Select the cell and click on run icon
cat_plot('sex')
```

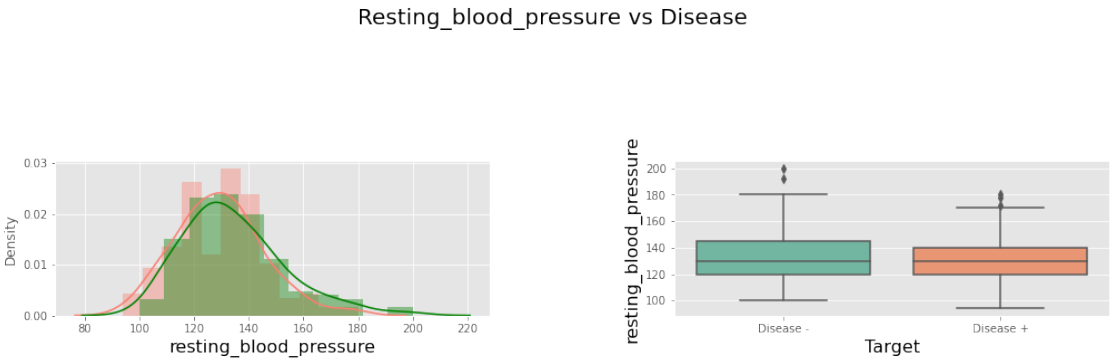
```
findfont: Font family ['times'] not found. Falling back to DejaVu
Sans.
```





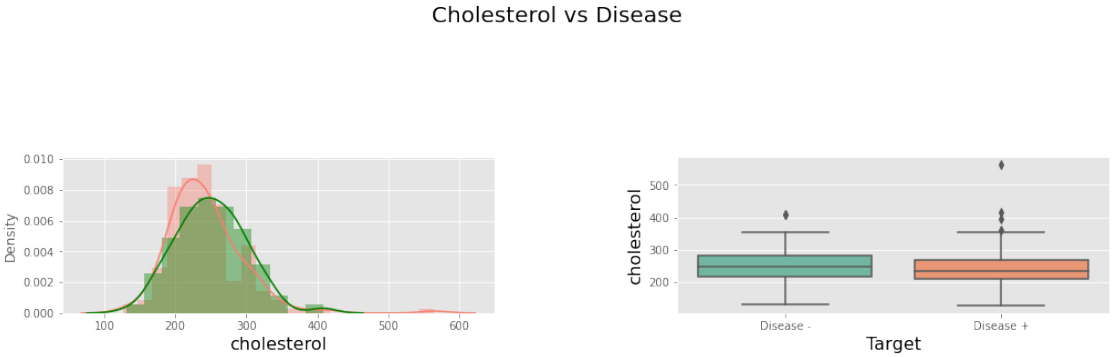
6. Detecting Heart Attack Based on Anomalies in Resting Blood Pressure

```
#Select the cell and click on run icon
real_distribution('resting_blood_pressure')
```



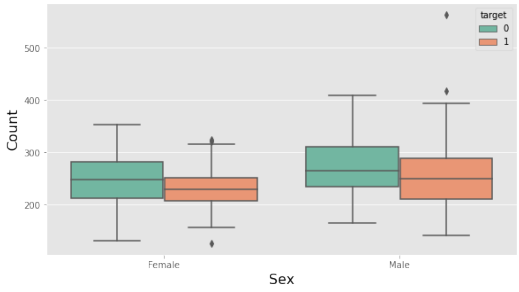
7. Relationship between Cholesterol Levels and Target

```
#Select the cell and click on run icon
real_distribution('cholesterol')
```



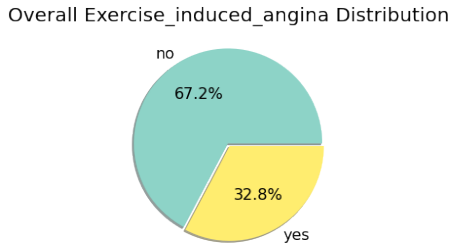

```
#Select the cell and click on run icon
f,axes = plt.subplots(1,2,figsize = (18,5))
sns.violinplot(y = 'cholesterol', x = 'sex',hue = 'target',data =
data, split = True, palette= 'Accent', ax = axes[0])
axes[0].set_xlabel('Sex', fontdict = {'family': 'DejaVu Sans','color':
'black','weight': 'normal','size': 16})
axes[0].set_ylabel('Cholesterol', fontdict = {'family':
'georgia','color': 'black','weight': 'normal','size': 16})
axes[0].set_xticklabels(['Female','Male'])
sns.boxplot(x = data.sex,  y = data.cholesterol, hue = data.target, ax
= axes[1], palette='Set2')
axes[1].set_xticklabels(['Female','Male'])
axes[1].set_xlabel('Sex',fontdict = {'family': 'DejaVu Sans','color':
'black','weight': 'normal','size': 16} )
axes[1].set_ylabel('Count',fontdict = {'family': 'DejaVu
Sans','color': 'black','weight': 'normal','size': 16} )
plt.tight_layout(w_pad = 10, pad = 2)
plt.show()
```

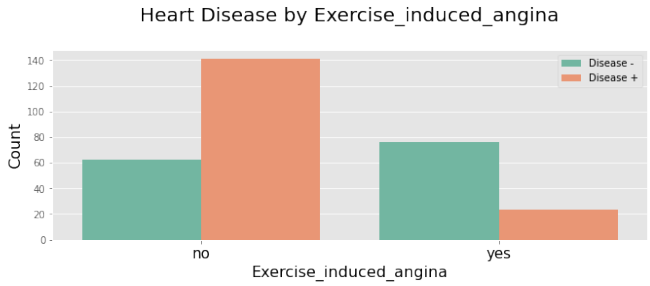
```
findfont: Font family ['georgia'] not found. Falling back to DejaVu Sans.
```



8. Relationship between Peak Exercising and Occurrence of Heart Attack

```
#Select the cell and click on run icon
cat_plot('exercise_induced_angina')
```

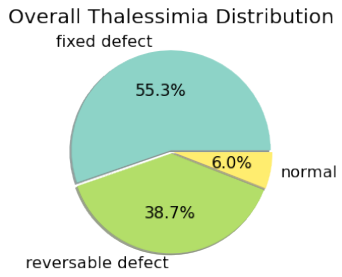


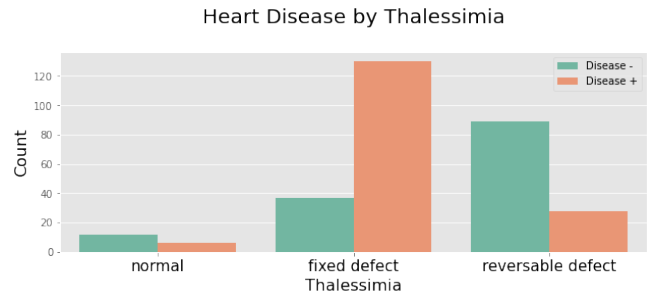


9. Thalassemia vs. CVD

Is thalassemia a major cause of CVD?

```
#Select the cell and click on run icon
cat_plot('thalessimia')
```

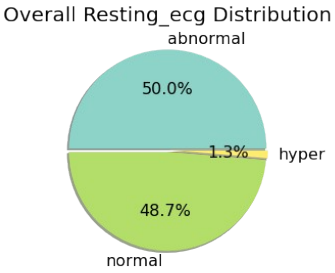


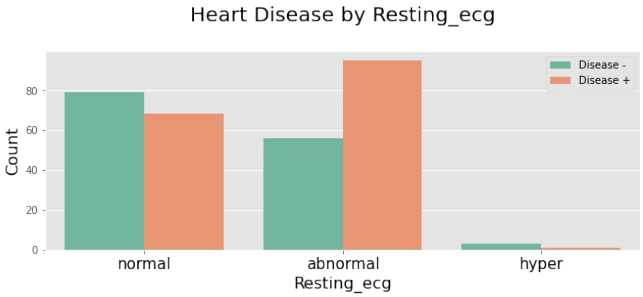


10. Factors Affecting Occurrence of CVD

A. Resting_ecg

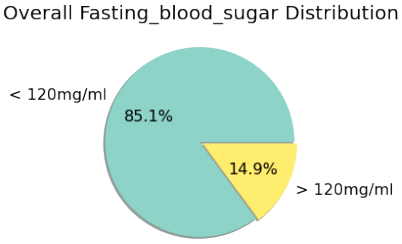
```
#Select the cell and click on run icon
cat_plot('resting_ecg')
```

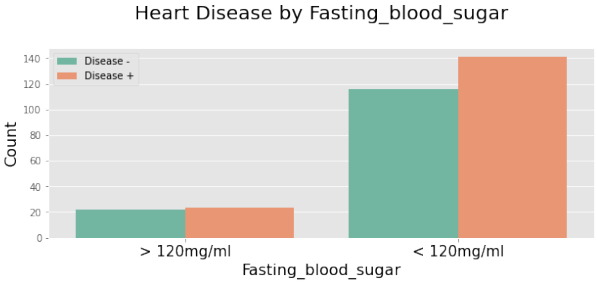




B. Fasting Blood Sugar

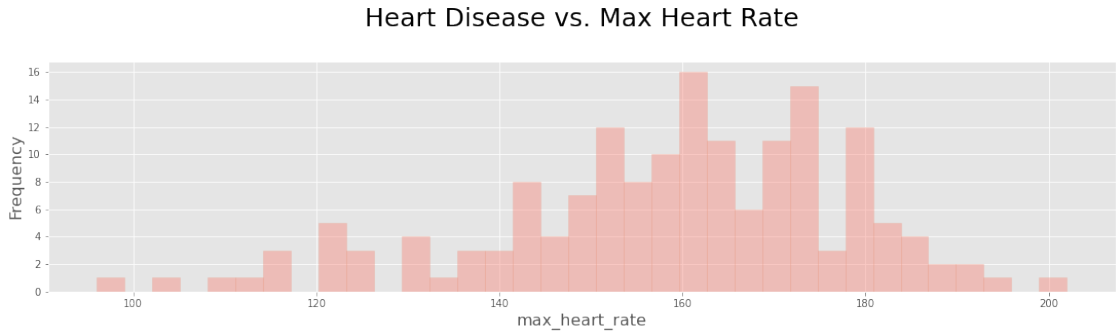
```
#Select the cell and click on run icon
cat_plot('fasting_blood_sugar')
```





C. Max Heart Rate Achieved

```
#Select the cell and click on run icon
#real_distribution('max_heart_rate')
plt.figure(figsize = (15,5))
sns.distplot(dsprsnt.max_heart_rate, kde = False, bins = 35, hist_kws
= {'edgecolor':'darksalmon', 'color' :'salmon'})
plt.title('\nHeart Disease vs. Max Heart Rate\n',family='DejaVu
Sans',fontsize= 25)
plt.tight_layout( )
plt.xlabel('max_heart_rate',family='DejaVu Sans',fontsize= 16)
plt.ylabel('Frequency',family='DejaVu Sans',fontsize= 16)
plt.show()
```



D. major_vessels

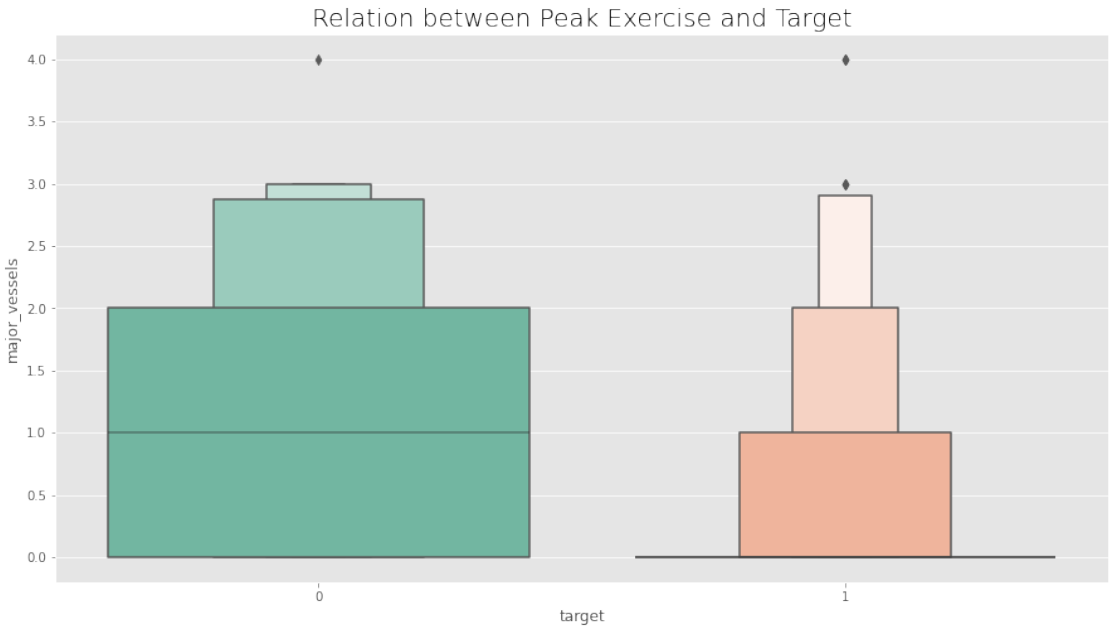
```
#Select the cell and click on run icon
data.columns
```

```
Index(['age', 'sex', 'chest_pain_type', 'resting_blood_pressure',
       'cholesterol', 'fasting_blood_sugar', 'resting_ecg',
'max_heart_rate',
       'exercise_induced_angina', 'st_depression', 'st_slope',
'major_vessels',
       'thalessimia', 'target'],
      dtype='object')
```

Observation:

The column names such as age, sex, chest_pain_type, resting_blood_pressure, cholesterol, fasting_blood_sugar, resting_ecg, max_heart_rate, exercise_induced_angina, st_depression, st_slope, major_vessels, thalessimia, and target present in the **data** dataframe.

```
#Select the cell and click on run icon
plt.figure(figsize= (15,8))
sns.boxenplot(data['target'], data['major_vessels'], palette = 'Set2')
plt.title('Relation between Peak Exercise and Target', fontsize = 20,
fontweight = 30)
plt.show()
```



Observation:

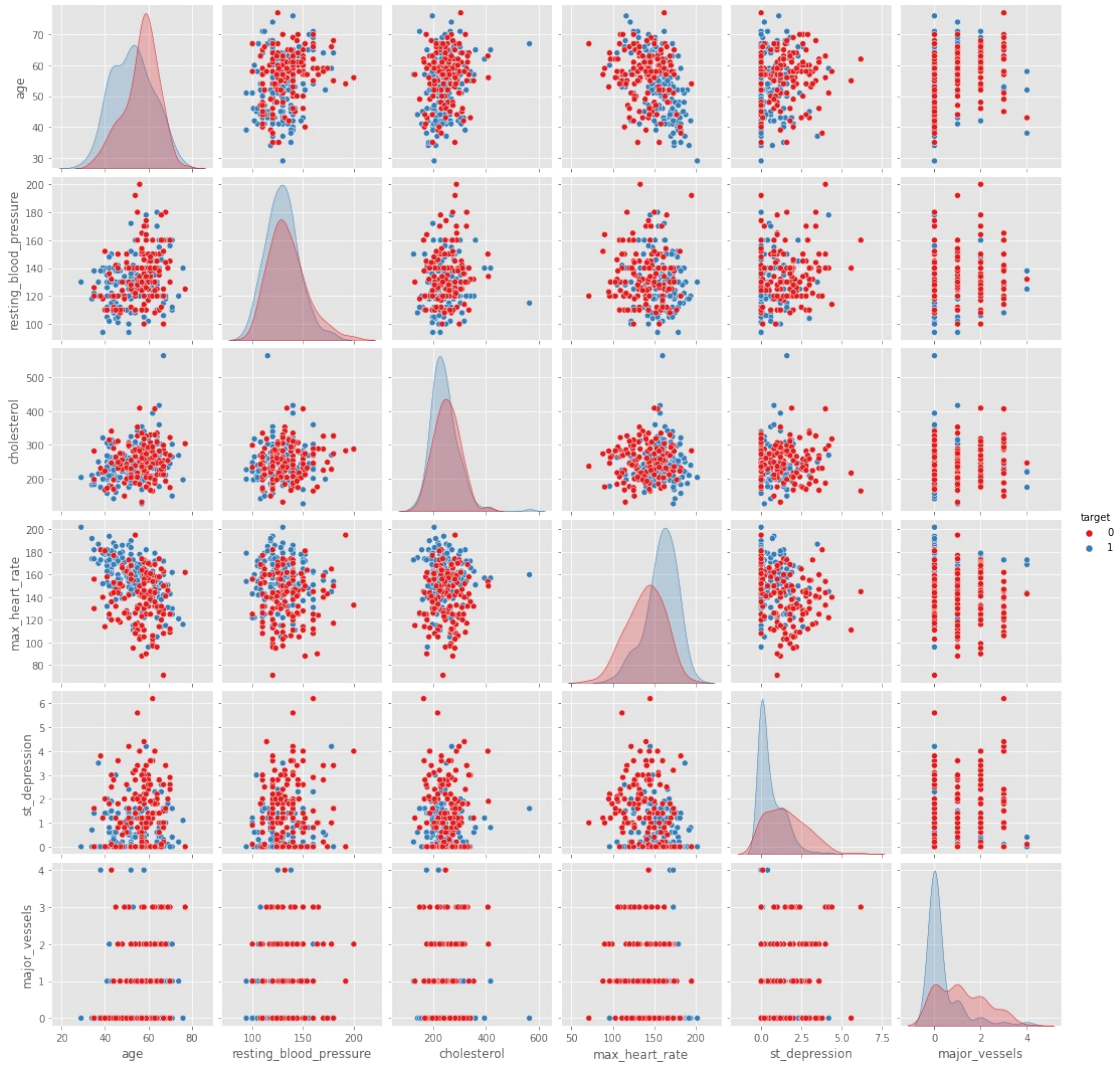
The above Bivariate plot between Target and Number of Major Vessels, shows that the patients who are more likely to suffer from Heart diseases are having high values of Major Vessels wheras the patiets who are very less likely to suffer from any kind of heart diseases have very low values of Major Vessels.

11. Relationship between all the variables

Note: Use a pair plot.

```
#Select the cell and click on run icon
sns.pairplot(data, hue = 'target', palette='Set1')
```

```
<seaborn.axisgrid.PairGrid at 0x7f4901f304d0>
```



4. Algorithms

1. Classification: First Iteration

Note: This iteration involves only the numerical variables that are registered in the correlation matrix.

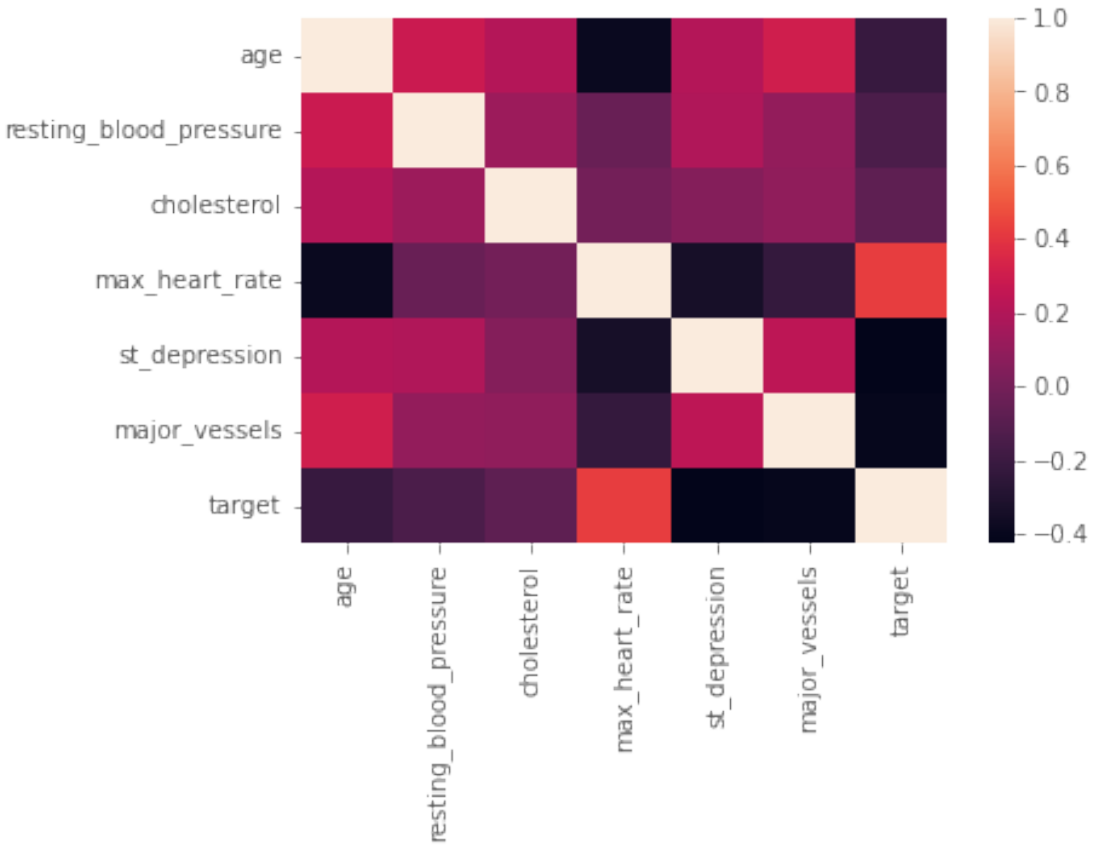
Using Correlation for Feature Selection

```
#Select the cell and click on run icon
data.head()
```

```
   age     sex    chest_pain_type  resting_blood_pressure  cholesterol
\
0   63    male       asymptomatic                    145          233
```

```
1   37    male  non-anginal pain                    130          250

2   41  female   atypical angina                    130          204

3   56    male   atypical angina                    120          236

4   57  female    typical angina                    120          354


  fasting_blood_sugar resting_ecg  max_heart_rate
exercise_induced_angina  \
0          > 120mg/ml      normal             150
no
1          < 120mg/ml    abnormal             187
no
2          < 120mg/ml      normal             172
no
3          < 120mg/ml    abnormal             178
no
4          < 120mg/ml    abnormal             163
yes

   st_depression     st_slope  major_vessels   thalessimia  target
0            2.3    upsloping              0        normal       1
1            3.5    upsloping              0  fixed defect       1
2            1.4  downsloping              0  fixed defect       1
3            0.8  downsloping              0  fixed defect       1
4            0.6  downsloping              0  fixed defect       1
```

```
#Select the cell and click on run icon
corr = data.corr()

#Select the cell and click on run icon
sns.heatmap(corr);
```



Logistic Regression

```
#Select the cell and click on run icon
from sklearn.model_selection import train_test_split as split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

#Select the cell and click on run icon
data0 =
data.drop(['sex','chest_pain_type','fasting_blood_sugar','exercise_ind
uced_angina','resting_ecg','st_slope','thalessimia'], axis=1)


train0, test0 = split(data0, test_size = .30, random_state = 12)
print(train0.shape)

train0.head(2)
```

```
(211, 7)

     age  resting_blood_pressure  cholesterol  max_heart_rate
st_depression  \
137   62                     128          208             140
0.0
231   55                     160          289             145
```

```
0.8
```

```
     major_vessels  target
137              0       1
231              1       0
```

```
#Select the cell and click on run icon
X_train0 = train0.drop('target', axis = 1)
Y_train0 = train0.target
X_test0 = test0.drop('target', axis = 1)
Y_test0 = test0.target
```

```
#Select the cell and click on run icon
lr = LogisticRegression()
lr.fit(X_train0,Y_train0)
```

```
pred0 = lr.predict(X_test0)
```

```
#Select the cell and click on run icon
accuracy_score(y_true = Y_test0,y_pred = pred0)
```

```
print(classification_report(y_true=Y_test0,y_pred = pred0))
```

```
              precision    recall  f1-score   support

           0       0.82      0.80      0.81        45
           1       0.81      0.83      0.82        46

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.81      0.81      0.81        91
```

```
#Select the cell and click on run icon
from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_test0, pred0))
```

```
[[36  9]
 [ 8 38]]
```

```
#Select the cell and click on run icon
#Accuracy of confusion matrix
Acc0 = (36+38)/(36+8+9+38)
Acc0
```

```
0.8131868131868132
```

Random Forest: First Iteration

```
#Select the cell and click on run icon
from sklearn.ensemble import RandomForestClassifier
clf0 = RandomForestClassifier(n_estimators=100)
```

```
clf0.fit(X_train0, Y_train0)
y_pred_random_forest0 = clf0.predict(X_test0)
acc_random_forest0 = round(clf0.score(X_train0, Y_train0) * 100, 2)
print (acc_random_forest0)
```

```
100.0
```

```
#Select the cell and click on run icon
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y_test0,
y_pred_random_forest0))
```

```
Accuracy: 0.7472527472527473
```

Observations from the above output:

The accuracy is 100 and this suggests that there is overfitting and issues with the feature selection.

To counter this, let us consider **standard error** and **p-value** for feature selection.

2. Classification: Second Iteration

Converting Categorical Variables to Numerical

```
#Select the cell and click on run icon
#Male=1, Female=0
#Converting sex to numerical variable in data
data['sex'] = data['sex'].map( {'male': 1, 'female': 0} ).astype(int)

#Converting resting_ecg to numerical variable in data
data['resting_ecg'] = data['resting_ecg'].map( {'normal': 0,
'abnormal': 1, 'hyper': 2} ).astype(int)

#Converting thalessimia to numerical variable in data
data['thalessimia'] = data['thalessimia'].map( {'normal': 0, 'fixed
defect': 1, 'reversable defect': 2} ).astype(int)

#Converting chest_pain_type to numerical variable in data
data['chest_pain_type'] =
data['chest_pain_type'].map( {'asymptomatic': 0, 'atypical angina': 1,
'non-anginal pain': 2, 'typical angina': 3} ).astype(int)

#Converting fasting_blood_sugar to numerical variable in data
data['fasting_blood_sugar'] = data['fasting_blood_sugar'].map( {'<
120mg/ml': 0, '> 120mg/ml': 1} ).astype(int)

##Converting st_slope to numerical variable in data
data['st_slope'] = data['st_slope'].map( {'upsloping': 0,
'downsloping': 1, 'flat': 2} ).astype(int)
```

```
##Converting exercise_induced_angina to numerical variable in data
data['exercise_induced_angina'] = data['exercise_induced_angina'].map(
{'no': 0, 'yes': 1} ).astype(int)


#Select the cell and click on run icon
data.head(1)
```

```
   age  sex  chest_pain_type  resting_blood_pressure  cholesterol  \
0   63    1                0                     145          233

   fasting_blood_sugar  resting_ecg  max_heart_rate
exercise_induced_angina  \
0                    1            0             150
0

   st_depression  st_slope  major_vessels  thalessimia  target
0            2.3         0              0            0       1
```

Using Logistic Regression for Feature Selection

```
#Select the cell and click on run icon
#Replicating dataset
data_LR = data
data_LR.head(2)
```

```
   age  sex  chest_pain_type  resting_blood_pressure  cholesterol  \
0   63    1                0                     145          233
1   37    1                2                     130          250

   fasting_blood_sugar  resting_ecg  max_heart_rate
exercise_induced_angina  \
0                    1            0             150
0
1                    0            1             187
0

   st_depression  st_slope  major_vessels  thalessimia  target
0            2.3         0              0            0       1
1            3.5         0              0            1       1
```

```
#Select the cell and click on run icon
#Splitting the dataset
train_LR, test_LR = split(data_LR, test_size = .30, random_state = 12)
print(train_LR.shape)

train_LR.head(2)
```

```
(211, 14)

     age  sex  chest_pain_type  resting_blood_pressure  cholesterol  \
137   62    1                1                     128          208
```

```
231   55    1                3                       160          289

     fasting_blood_sugar  resting_ecg  max_heart_rate  \
137                    1            0             140
231                    0            0             145

     exercise_induced_angina  st_depression  st_slope
major_vessels  \
137                        0            0.0         1              0

231                        1            0.8         2              1


     thalessimia  target
137            1       1
231            2       0
```

```
#Select the cell and click on run icon
X_train_LR = train_LR.drop('target', axis = 1)
Y_train_LR = train_LR.target
X_test_LR = test_LR.drop('target', axis = 1)
Y_test_LR = test_LR.target
```

```
#Select the cell and click on run icon
import statsmodels.api as sm
#Apply logistic regression
model_LR = sm.Logit(Y_train_LR, X_train_LR)
model_LR = model_LR.fit()
#Find the summary
model_LR.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.379944
         Iterations 7

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

==============================================================================
========
Dep. Variable:                 target   No. Observations:
211
Model:                          Logit   Df Residuals:
198
Method:                           MLE   Df Model:
12
Date:                Tue, 25 Jan 2022   Pseudo R-squ.:
0.4462
Time:                        06:58:23   Log-Likelihood:
```

```
-80.168
converged:                       True   LL-Null:
-144.77
Covariance Type:            nonrobust   LLR p-value:
8.921e-22
==============================================================================
=====================
                               coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
---------------------
age                          0.0228      0.022      1.032      0.302
-0.021       0.066
sex                         -1.5938      0.490     -3.251      0.001
-2.555      -0.633
chest_pain_type             -0.4338      0.221     -1.959      0.050
-0.868       0.000
resting_blood_pressure      -0.0174      0.011     -1.560      0.119
-0.039       0.004
cholesterol                 -0.0028      0.005     -0.615      0.539
-0.012       0.006
fasting_blood_sugar          0.6102      0.607      1.005      0.315
-0.580       1.800
resting_ecg                  0.7108      0.391      1.820      0.069
-0.055       1.476
max_heart_rate               0.0411      0.010      4.180      0.000
0.022       0.060
exercise_induced_angina     -0.5806      0.485     -1.196      0.232
-1.532       0.371
st_depression               -0.7242      0.240     -3.019      0.003
-1.194      -0.254
st_slope                     0.1797      0.365      0.492      0.623
-0.536       0.896
major_vessels               -0.8405      0.236     -3.557      0.000
-1.304      -0.377
thalessimia                 -1.1985      0.347     -3.459      0.001
-1.878      -0.519
==============================================================================
=====================
"""
```

List of variables with less standard error and p-value less than or equal to 0.05:

- sex
- chest_pain_type
- max_heart_rate
- st_depression
- major_vessels
- thalessimia

Dropping Variables

```
#Select the cell and click on run icon
data_LR.head(2)
```

```
   age  sex  chest_pain_type  resting_blood_pressure  cholesterol  \
0   63    1                0                     145          233
1   37    1                2                     130          250

   fasting_blood_sugar  resting_ecg  max_heart_rate
exercise_induced_angina  \
0                    1            0             150
0
1                    0            1             187
0

   st_depression  st_slope  major_vessels  thalessimia  target
0            2.3         0              0            0       1
1            3.5         0              0            1       1
```

```
#Select the cell and click on run icon
#dropping variables
data_Log =
data_LR.drop(['age','resting_blood_pressure','cholesterol','fasting_bl
ood_sugar','resting_ecg','exercise_induced_angina','st_slope'],
axis=1)
```

Dividing Dataset

```
#Select the cell and click on run icon
train_Log, test_Log = split(data_Log, test_size = .30, random_state =
12)

#Select the cell and click on run icon
print(train_Log.shape)

train_Log.head(2)
```

```
(211, 7)

     sex  chest_pain_type  max_heart_rate  st_depression
major_vessels  \
137    1                1             140            0.0
0
231    1                3             145            0.8
1

     thalessimia  target
137            1       1
231            2       0
```

```
#Select the cell and click on run icon
print(test_Log.shape)
```

```
test_Log.head(2)
```

```
(91, 7)
```

```
     sex  chest_pain_type  max_heart_rate  st_depression
major_vessels  \
227    1                0             159            0.2
0
136    0                2              96            0.0
0

     thalessimia  target
227            2       0
136            1       1
```

```
#Select the cell and click on run icon
X_train_Log = train_Log.drop('target', axis = 1)
Y_train_Log = train_Log.target
X_test_Log = test_Log.drop('target', axis = 1)
Y_test_Log = test_Log.target
```

Logistic Regression Using Statsmodels

```
#Select the cell and click on run icon
#import statsmodels.api as sm
#Apply logistic regression
model_Log = sm.Logit(Y_train_Log, X_train_Log)
model_Log = model_Log.fit()
#Find the summary
model_Log.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.400084
         Iterations 7

<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results

==============================================================================
========
Dep. Variable:                 target   No. Observations:
211
Model:                          Logit   Df Residuals:
205
Method:                           MLE   Df Model:
5
Date:                Tue, 25 Jan 2022   Pseudo R-squ.:
0.4169
Time:                        06:58:23   Log-Likelihood:
-84.418
```

```
converged:                       True   LL-Null:
-144.77
Covariance Type:            nonrobust   LLR p-value:
2.227e-24
================================================================================
=============
                      coef    std err          z      P>|z|
[0.025      0.975]
--------------------------------------------------------------------------------
-------------
sex                -1.4688      0.447     -3.284      0.001      -
2.346      -0.592
chest_pain_type    -0.4832      0.200     -2.418      0.016      -
0.875      -0.092
max_heart_rate      0.0333      0.005      7.022      0.000
0.024       0.043
st_depression      -0.7740      0.205     -3.778      0.000      -
1.175      -0.372
major_vessels      -0.8070      0.218     -3.704      0.000      -
1.234      -0.380
thalessimia        -1.1471      0.316     -3.631      0.000      -
1.766      -0.528
================================================================================
=============
"""
```

Prediction

```
#Select the cell and click on run icon
pred_Log = model_Log.predict(X_test_Log)
model_Log.predict(X_train_Log)
```

```
137    0.826704
231    0.140540
68     0.928333
142    0.974590
149    0.804113
         ...
259    0.345733
130    0.924623
241    0.156410
253    0.824364
155    0.785966
Length: 211, dtype: float64
```

Confusion Matrix and Accuracy

```
#Select the cell and click on run icon
#Confusion matrix
(model_Log.predict(X_train_Log) >= 0.5).astype(int)
model_Log.pred_table()
```

```
array([[ 72.,  21.],
       [ 13., 105.]])

#Select the cell and click on run icon
#Accuracy
print((72+105)/(72+21+13+105))

0.8388625592417062
```

Random Forest: Second Iteration

```
#Select the cell and click on run icon
from sklearn.ensemble import RandomForestClassifier
clf_RF = RandomForestClassifier(n_estimators=100)
clf_RF.fit(X_train_Log, Y_train_Log)
y_pred_random_forest_RF = clf_RF.predict(X_test_Log)
acc_random_forest_RF = round(clf_RF.score(X_train_Log, Y_train_Log) *
100, 2)
print (acc_random_forest_RF)

100.0

#Select the cell and click on run icon
#Predict
#Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y_test_Log,
y_pred_random_forest_RF))

Accuracy: 0.7362637362637363
```

3. Classification: Third Iteration

Note: We have identified significant variables and now we will use logistic regression from sklearn to train and predict the desired outcome.

```
#Select the cell and click on run icon
lr_slog = LogisticRegression()
lr_slog.fit(X_train_Log,Y_train_Log)

pred_slog = lr_slog.predict(X_test_Log)

#Select the cell and click on run icon
#Y_test_Log
pred_slog

array([1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
0,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1,
0,
       1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
```

```
0,
       1, 1, 1])

#Select the cell and click on run icon
accuracy_score(y_true = Y_test_Log,y_pred = pred_slog)

print(classification_report(y_true=Y_test_Log,y_pred = pred_slog))

              precision    recall  f1-score   support

           0       0.80      0.82      0.81        45
           1       0.82      0.80      0.81        46

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.81      0.81      0.81        91


#Select the cell and click on run icon
from sklearn.metrics import confusion_matrix
print(confusion_matrix(Y_test_Log, pred_slog))

[[37  8]
 [ 9 37]]

#Select the cell and click on run icon
#Accuracy of confusion matrix
Acc_slog = (37+37)/(37+9+8+37)
Acc_slog

0.8131868131868132
```

Random Forest: Third Iteration

```
#Select the cell and click on run icon
from sklearn.ensemble import RandomForestClassifier
clf0 = RandomForestClassifier(n_estimators=100)
clf0.fit(X_train_Log, Y_train_Log)
y_pred_random_forest_Log = clf0.predict(X_test_Log)
acc_random_forest_Log = round(clf0.score(X_train_Log, Y_train_Log) *
100, 2)
print (acc_random_forest_Log)

100.0

#Select the cell and click on run icon
#Predict
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y_test_Log,
y_pred_random_forest_Log))

Accuracy: 0.7362637362637363
```