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

```
url="https://raw.githubusercontent.com/karansamani/dataset/main/heart.csv"
df= pd.read_csv(url)
```

```
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2
301	57	1	0	130	131	0	1	115	1	1.2	1	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1

303 rows × 14 columns

```
#printing first few lines of the dataset
df.head()
```

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

```
#checking for the null values

df.isnull()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slo
0	False	False	False	False	False	False	False	False	False	False	Fal
1	False	False	False	False	False	False	False	False	False	False	Fal
2	False	False	False	False	False	False	False	False	False	False	Fal
3	False	False	False	False	False	False	False	False	False	False	Fal
4	False	False	False	False	False	False	False	False	False	False	Fal
...	...	...	...	...	...	...	...	...	...	...	
298	False	False	False	False	False	False	False	False	False	False	Fal
299	False	False	False	False	False	False	False	False	False	False	Fal
300	False	False	False	False	False	False	False	False	False	False	Fal
301	False	False	False	False	False	False	False	False	False	False	Fal
302	False	False	False	False	False	False	False	False	False	False	Fal

303 rows × 14 columns

```
df.isnull().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
```

The description about the columns in the dataset:-

1. Age:- The age of the people from the dataset.
2. Sex:- The sex of a person and it has has only 2 possible values in this dataset: 1 - Male and 0 - Female.
3. cp(Chest pain type):- This column defines the chest pain severity in scale of 0-4.

— Value 0: asymptomatic

— Value 1: atypical angina

— Value 2: non-anginal pain

— Value 3: typical angina

4. trestbps(Resting blood pressure):- An healthy has has a blood pressure of 80/120 mmHg , person is at high risk if it is above 180mmHg and lesser than 50mmHg.It has continuous values.
5. chol(Serum cholestrol):- The normal cholestrol level of a healthy person is between 125-200 mm/dl and it is considerable till 240 mm/dl and anything greater than this value will cause higher risk of heart attack and this feature has a continuous value.
6. fbs(Fasting blood sugar):- This feature has only two unique values - 1 if FBS is > 120 mg/dl otherwise 0.If the blood sugar is higher than the value mentioned the person is at high risk of getting heart attack.
7. restecg:- resting electrocardiographic results

— Value 0: normal

— Value 1: showing probable or definite left ventricular hypertrophy by Estes' criteria

— Value 2: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

8. thalach:- Maximum heart rate achieved and the normal heart rate of a person should be below 100 bpm anything above that is risky and person might tend to have a heart attack.
9. exang(exercise induced angina):- This feature has two values (1 = yes; 0 = no) means if a person faces angina due to exercise than the value is 1 else it is 0.
10. oldpeak(ST depression induced by exercise):- If the range is lesser than 1.5 its a high risk and if it is greter than 1.5 the person is at low risk of getting a heart attack.
11. slope:- the slope of the peak exercise ST segment

0: downsloping;

1: flat;

2: upsloping

when the value is 0 and 2 the possibility of heart attack is high and when it is 1 the chances of getting heart attack is less.

12. ca(number of major vessels colored by flouropsy):- This column has a discrete value from [0-3].
13. thal:- A blood disorder called thalassemia, it has discrete values Value 1: normal blood flow Value 2: fixed defect (no blood flow in some part of the heart) Value 3: reversible defect (a blood flow is observed but it is not normal)
14. target:- This column has a discrete values 0 = less chance of heart attack, 1= more chance of heart attack.

Data_Analysis 1: summary for the dataset by target

```
df.shape
```

```
(303, 14)
```

```
df.describe() #summary of the data
```

	age	sex	cp	trestbps	chol	fbs	reste
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.52805
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.5258
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.0000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.0000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.0000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.0000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.0000

```
df.groupby("target").describe()
```

	age								sex		
	count	mean	std	min	25%	50%	75%	max	count	mean	std
target											
0	138.0	56.601449	7.962082	35.0	52.0	58.0	62.0	77.0	138.0	0.826087	0.380
1	165.0	52.496970	9.550651	29.0	44.0	52.0	59.0	76.0	165.0	0.563636	0.497

2 rows × 104 columns

Data_Analysis 2: the testing criteria(chol, fbs, etc.) versus age and sex

```
import pandas as pd
# df.shape
df_target = df.where(df['target']==1).dropna() # get target dataframe where target == 1
df_target
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63.0	1.0	3.0	145.0	233.0	1.0	0.0	150.0	0.0	2.3	0.0	0.(
1	37.0	1.0	2.0	130.0	250.0	0.0	1.0	187.0	0.0	3.5	0.0	0.(
2	41.0	0.0	1.0	130.0	204.0	0.0	0.0	172.0	0.0	1.4	2.0	0.(
3	56.0	1.0	1.0	120.0	236.0	0.0	1.0	178.0	0.0	0.8	2.0	0.(
4	57.0	0.0	0.0	120.0	354.0	0.0	1.0	163.0	1.0	0.6	2.0	0.(
...	...	...	...	...	...	...	...	...	...	...	...	..
160	56.0	1.0	1.0	120.0	240.0	0.0	1.0	169.0	0.0	0.0	0.0	0.(
161	55.0	0.0	1.0	132.0	342.0	0.0	1.0	166.0	0.0	1.2	2.0	0.(
162	41.0	1.0	1.0	120.0	157.0	0.0	1.0	182.0	0.0	0.0	2.0	0.(
163	38.0	1.0	2.0	138.0	175.0	0.0	1.0	173.0	0.0	0.0	2.0	4.(
164	38.0	1.0	2.0	138.0	175.0	0.0	1.0	173.0	0.0	0.0	2.0	4.(

165 rows × 14 columns

```
pd.pivot_table(df_target, values=['trestbps','chol','fbs','thalach'], index=['age', 'sex']
               aggfunc=np.mean) # get the mean value
```

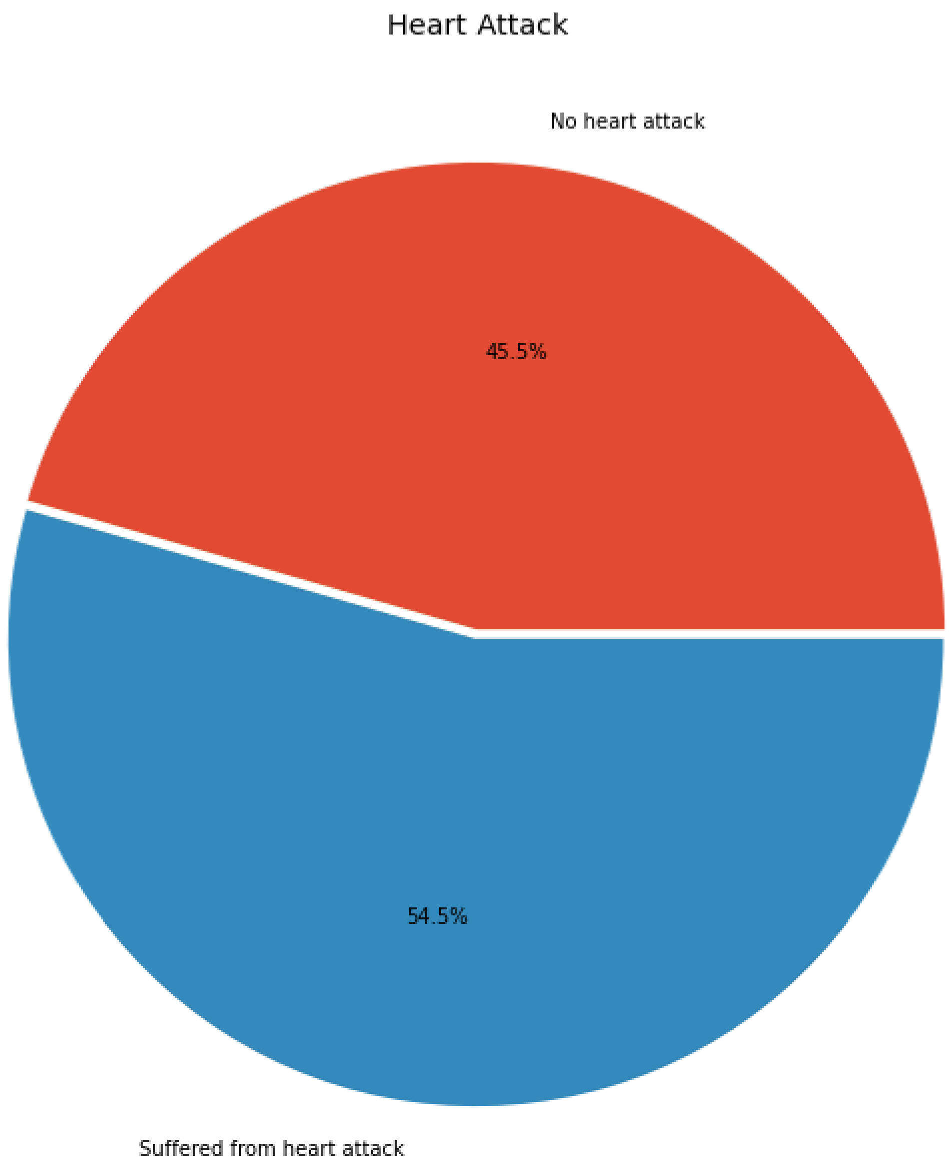
		chol	fbs	thalach	trestbps
age	**sex**				
29.0	**1.0**	204.000000	0.000000	202.0	130.000000
34.0	**0.0**	210.000000	0.000000	192.0	118.000000
	1.0	182.000000	0.000000	174.0	118.000000
35.0	**0.0**	183.000000	0.000000	182.0	138.000000
	1.0	192.000000	0.000000	174.0	122.000000
...	**...**	...	...	...	...
69.0	**1.0**	234.000000	1.000000	131.0	160.000000
70.0	**1.0**	245.000000	0.000000	143.0	156.000000
71.0	**0.0**	238.666667	0.333333	139.0	127.333333
74.0	**0.0**	269.000000	0.000000	121.0	120.000000
76.0	**0.0**	197.000000	0.000000	116.0	140.000000

68 rows × 4 columns

visualization 1: the chances of heart attack

```
plt.style.use("ggplot")
plt.figure(figsize=(10, 10))
target_values = [len(df[df['target'] == 0]), len(df[df['target'] == 1])]
labels = ["No heart attack", "Suffered from heart attack"]
plt.pie(x=target_values, labels=labels, autopct='%1.1f%%', explode=[0, 0.02])
plt.title("Heart Attack")
```

```
    Text(0.5, 1.0, 'Heart Attack')
```



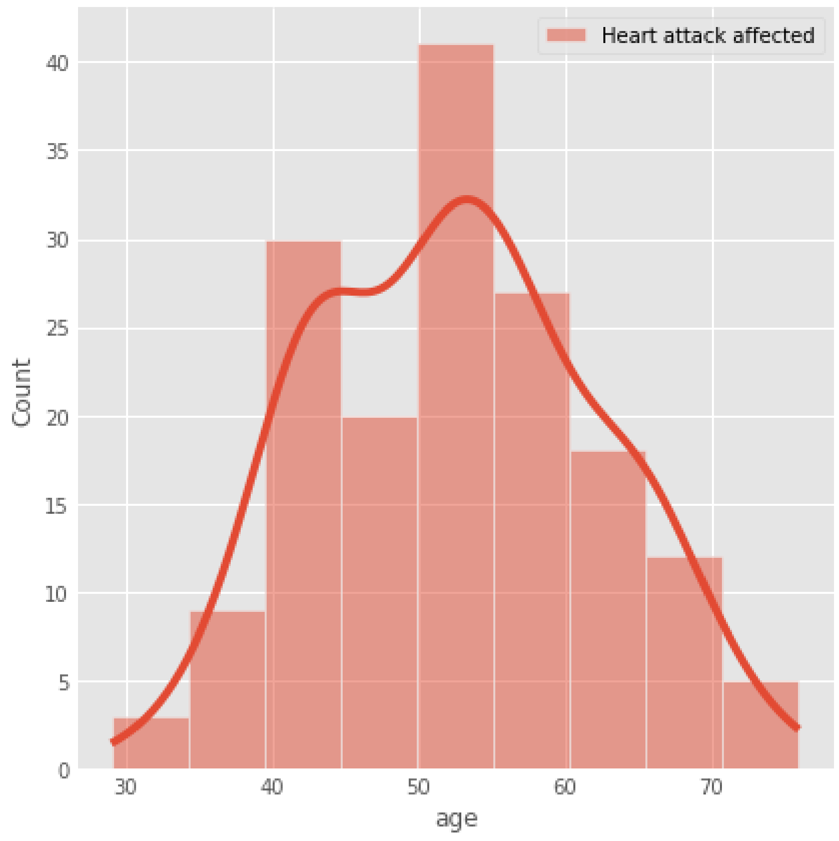
visualization 2: the heart attack

```
df1= df[df['target'] == 1] #new dataframe where the target=1 which means people suffering
df1
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
160	56	1	1	120	240	0	1	169	0	0.0	0	0
161	55	0	1	132	342	0	1	166	0	1.2	2	0
162	41	1	1	120	157	0	1	182	0	0.0	2	0

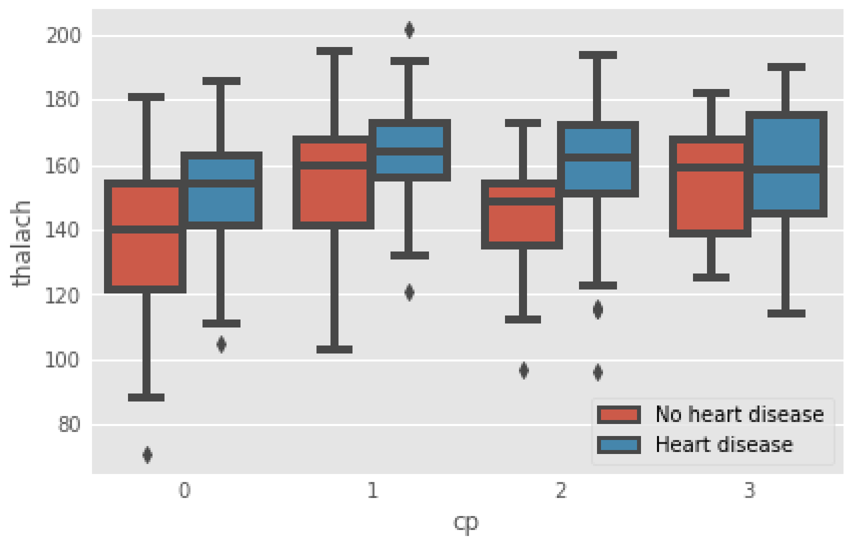
```
sns.displot(data=df1, x="age", kde=True, label= 'Heart attack affected', height=6)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f0dab884a50>
```



visualization 3

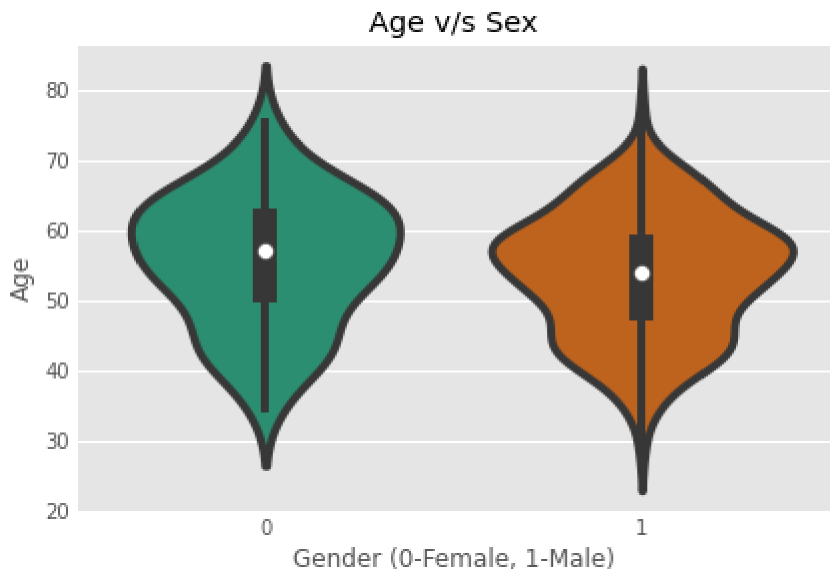
```
sns.boxplot(data=df, x="cp", y="thalach", hue="target")
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
```



violin plots visualization 4

```
sns.violinplot(x=df['sex'], y=df['age'], palette='Dark2')
plt.title("Age v/s Sex")
plt.xlabel("Gender (0-Female, 1-Male)")
plt.ylabel("Age")
```

```
    Text(0, 0.5, 'Age')
```



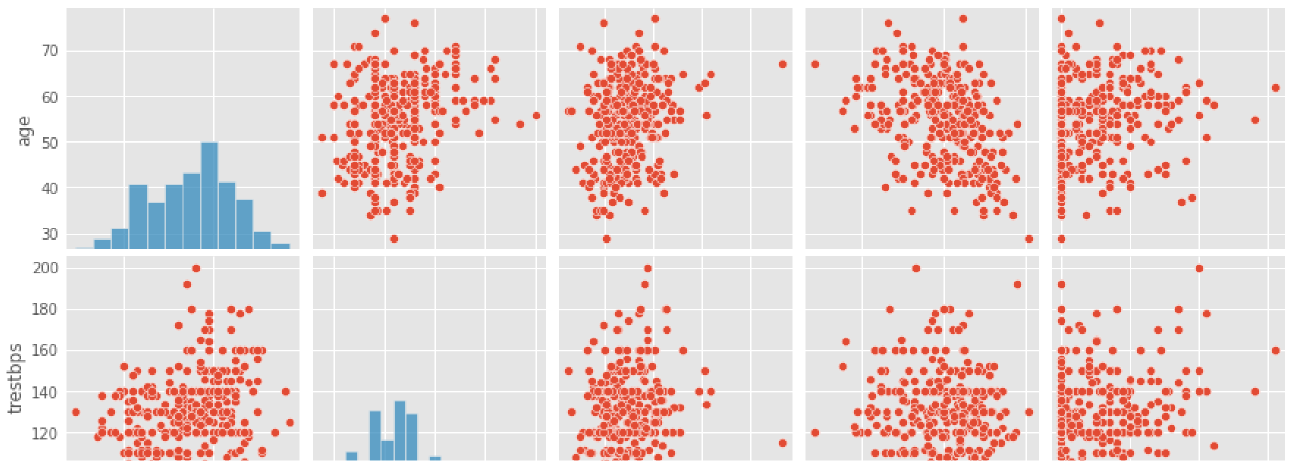
visualization 5: pair plot and heatmap for some variables

```
df_matrix=df[["age","trestbps","chol","thalach","oldpeak"]]
df_matrix
```

	age	trestbps	chol	thalach	oldpeak
0	63	145	233	150	2.3
1	37	130	250	187	3.5
2	41	130	204	172	1.4
3	56	120	236	178	0.8
4	57	120	354	163	0.6
...	...	...	...	...	...
298	57	140	241	123	0.2
299	45	110	264	132	1.2
300	68	144	193	141	3.4

```
sns.pairplot(df_matrix) #for all the continuous variables
```

```
<seaborn.axisgrid.PairGrid at 0x7f0daaf24150>
```

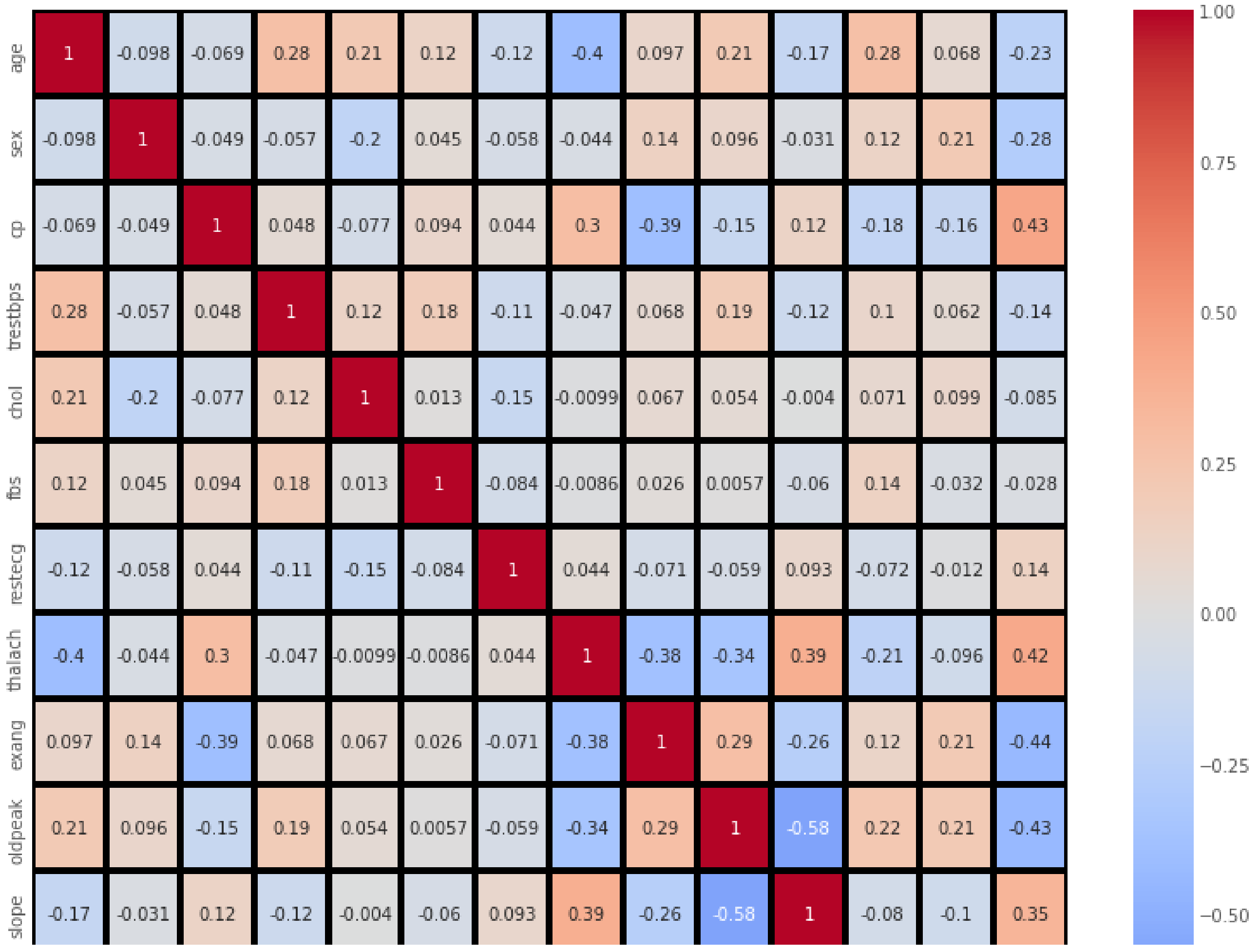


```
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2
301	57	1	0	130	131	0	1	115	1	1.2	1	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1

303 rows × 14 columns

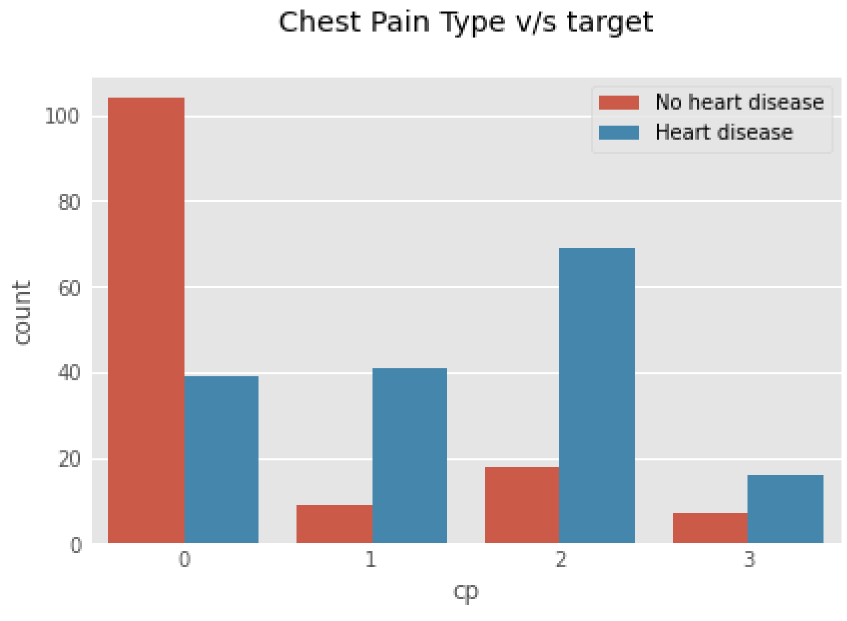
```
plt.figure(figsize=(12,·12))
sns.heatmap(df.corr(),·annot·=·True,·vmin=-1,·vmax=1,·center=·0,·cmap=·'coolwarm',·linewid
<matplotlib.axes._subplots.AxesSubplot at 0x7f0daa8f8710>
```



Double-click (or enter) to edit

```
sns.countplot(data= df, x='cp',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('Chest Pain Type v/s target\n')
```

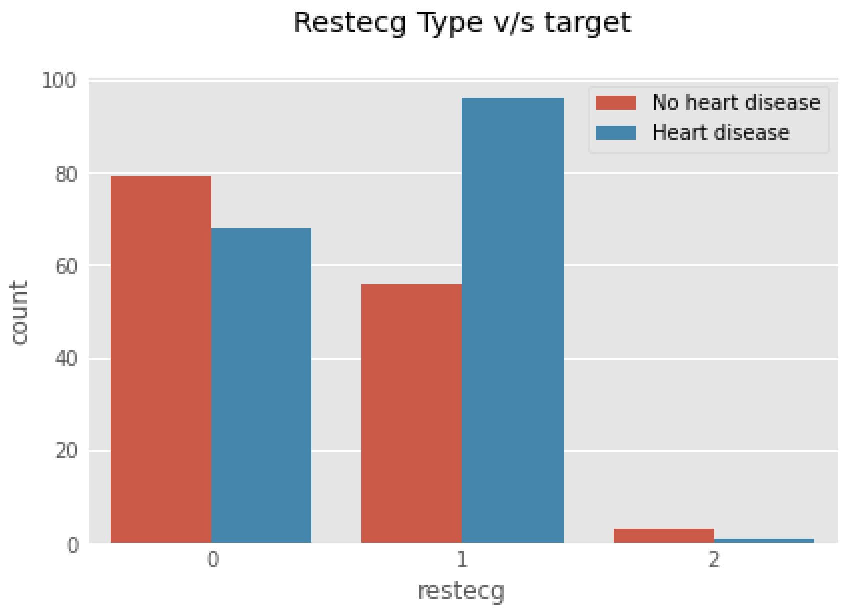
```
Text(0.5, 1.0, 'Chest Pain Type v/s target\n')
```



There are four types of chest pain, asymptomatic, atypical angina, non-anginal pain and typical angina. Most of the Heart Disease patients are found to have atypical anginal and non anginal chest pain and very few have typical angina. As the severity increases the number of people who are suffering from heart disease are more compared to those who are not suffering from.These group of people might show symptoms like indigestion, flu or a strained chest muscle.Heart attack, involves, blockage of blood flow to your heart and possible damage to the heart muscle

```
sns.countplot(data= df, x='restecg',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('Restecg Type v/s target\n')
```

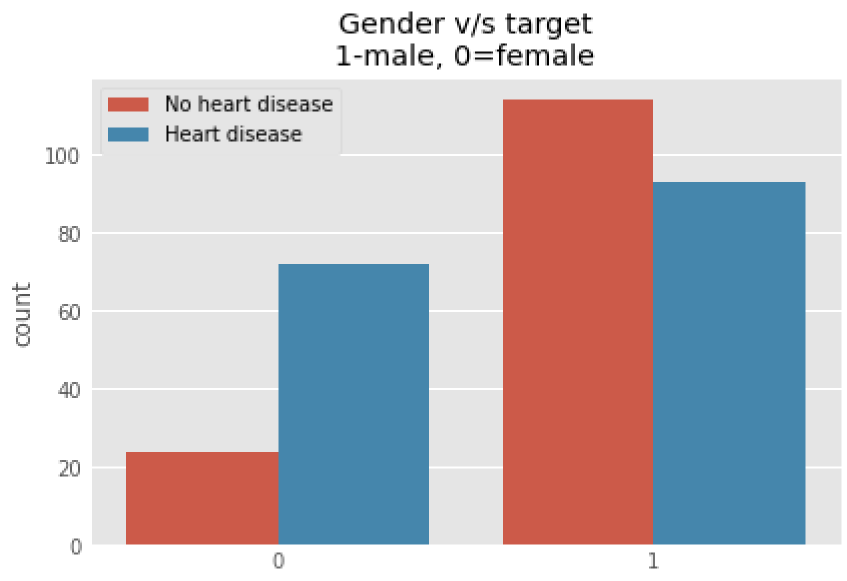
```
Text(0.5, 1.0, 'Restecg Type v/s target\n')
```



There are three types of restecg states Value 0: normal, Value 1: showing probable or definite left ventricular hypertrophy by Estes' criteria, Value 2: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV).As the severity increases the number of people who are suffering from heart disease are more compared to those who are not suffering from heart disease.The result shows that people who are suffering from the heart disease have higher ecg value and maximum people has 'value'=1 and very few who are suffering from the heart disease show value 'value'=2.

```
sns.countplot(data= df, x='sex',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('Gender v/s target\n'"1-male, 0=female")
```

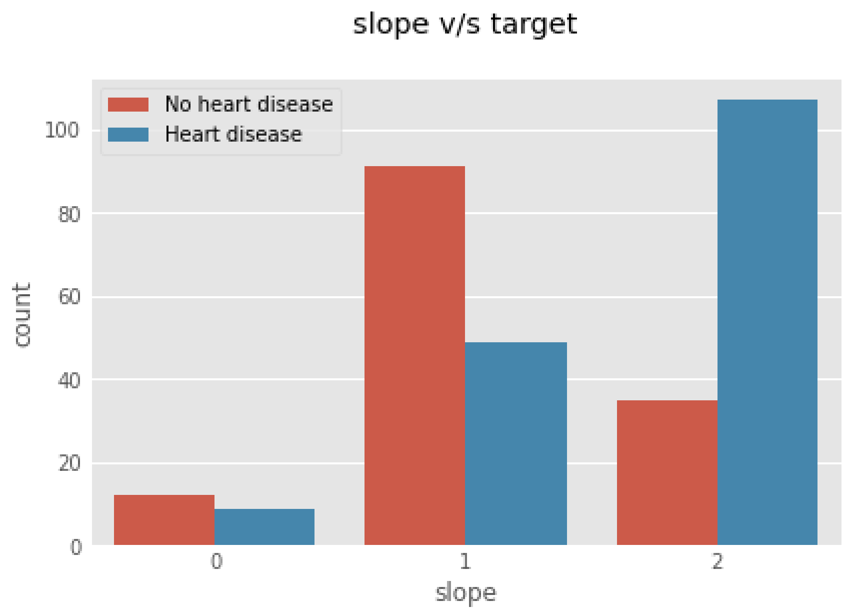
```
Text(0.5, 1.0, 'Gender v/s target\n1-male, 0=female')
```



According to this dataset males are more susceptible to get Heart Disease than females. Men experience heart attacks more than women. Sudden Heart Attacks are experienced by men between 70% — 89%. Woman may experience a heart attack with no chest pressure at all, they usually experience nausea or vomiting which are often confused with acid reflux or the flu.

```
sns.countplot(data= df, x='slope',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('slope v/s target\n')
```

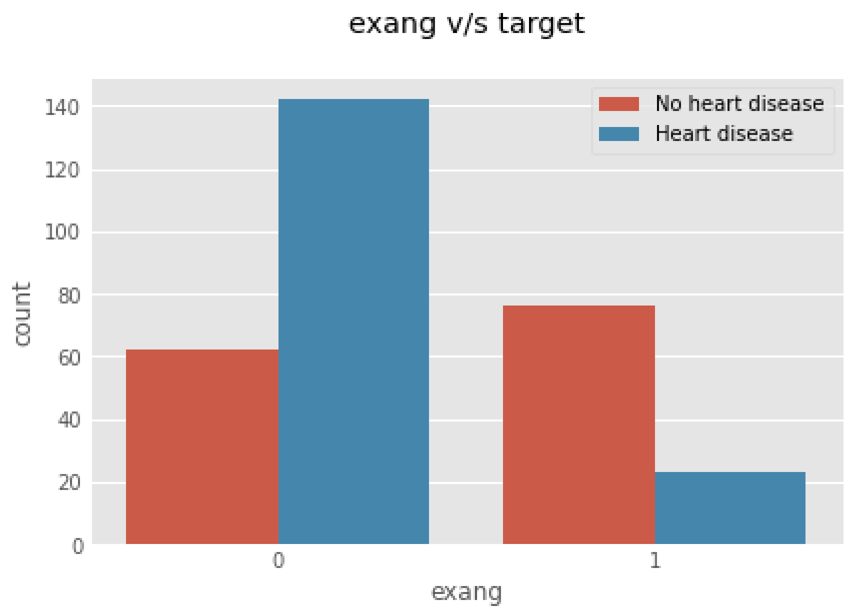
```
Text(0.5, 1.0, 'slope v/s target\n')
```



The slope of the peak exercise ST segment has three values 0: downsloping; 1: flat; 2: upsloping, when the value is 0 and 2 the possibility of heart attack is high and when it is 1 the chances of getting heart attack is less. The graph shows the people who are suffering from the heart disease has the slope value of either 0 or 2 and maximum people who do not suffer from the heart disease has the slope value 1.

```
sns.countplot(data= df, x='exang',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('exang v/s target\n')
```

```
Text(0.5, 1.0, 'exang v/s target\n')
```



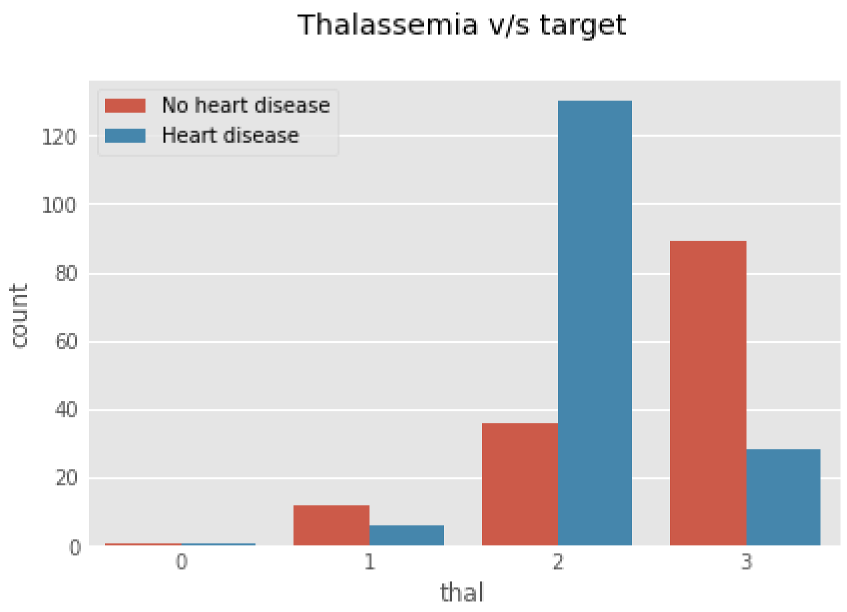
```
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2
301	57	1	0	130	131	0	1	115	1	1.2	1	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1

303 rows × 14 columns

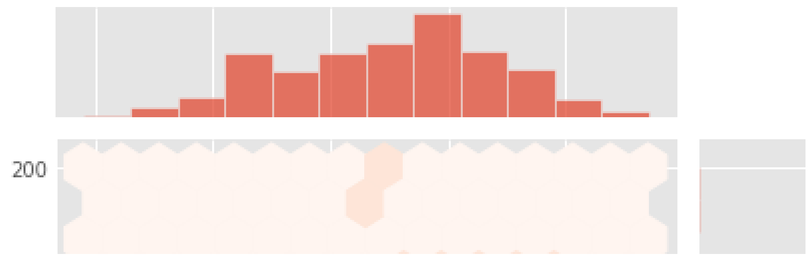
```
sns.countplot(data= df, x='thal',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('Thalassemia v/s target\n')
```

```
    Text(0.5, 1.0, 'Thalassemia v/s target\n')
```



A blood disorder called thalassemia, it has discrete values Value 1: normal blood flow Value 2: fixed defect (no blood flow in some part of the heart) Value 3: reversible defect (a blood flow is observed but it is not normal).The graph shows that people who are suffering from heart disease has a defect value 2 and 3 and those arent suffering from heart disease has a value 1 and 3. The correlation with the target is also positive , which shows that as the value of thalach increases the person has a chance of getting heart attack, target=1.

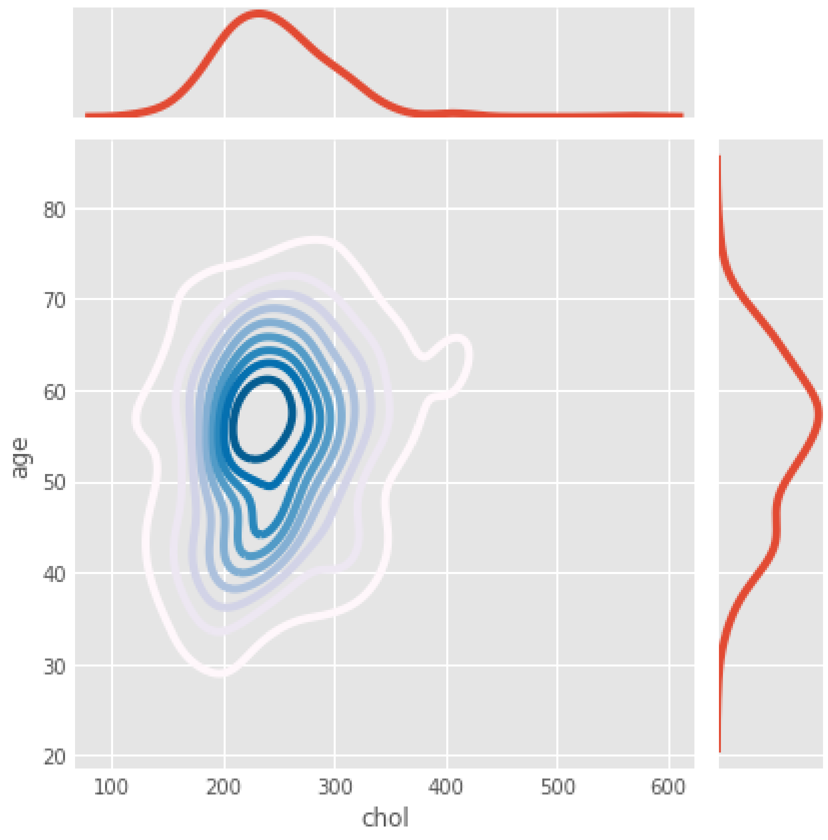
```
p=sns.jointplot(data=df,x='age',y='trestbps',kind='hex',cmap='Reds')
```



Joint plots in seaborn helps us to understand the trend seen among two features. As observed from the above plot we can see that most of the Heart diseased patients in their age of upper 50s or lower 60s tend to have trestbps higher than 120mmHg.

```
sns.jointplot(data=df,x='chol',y='age',kind='kde',cmap='PuBu')
```

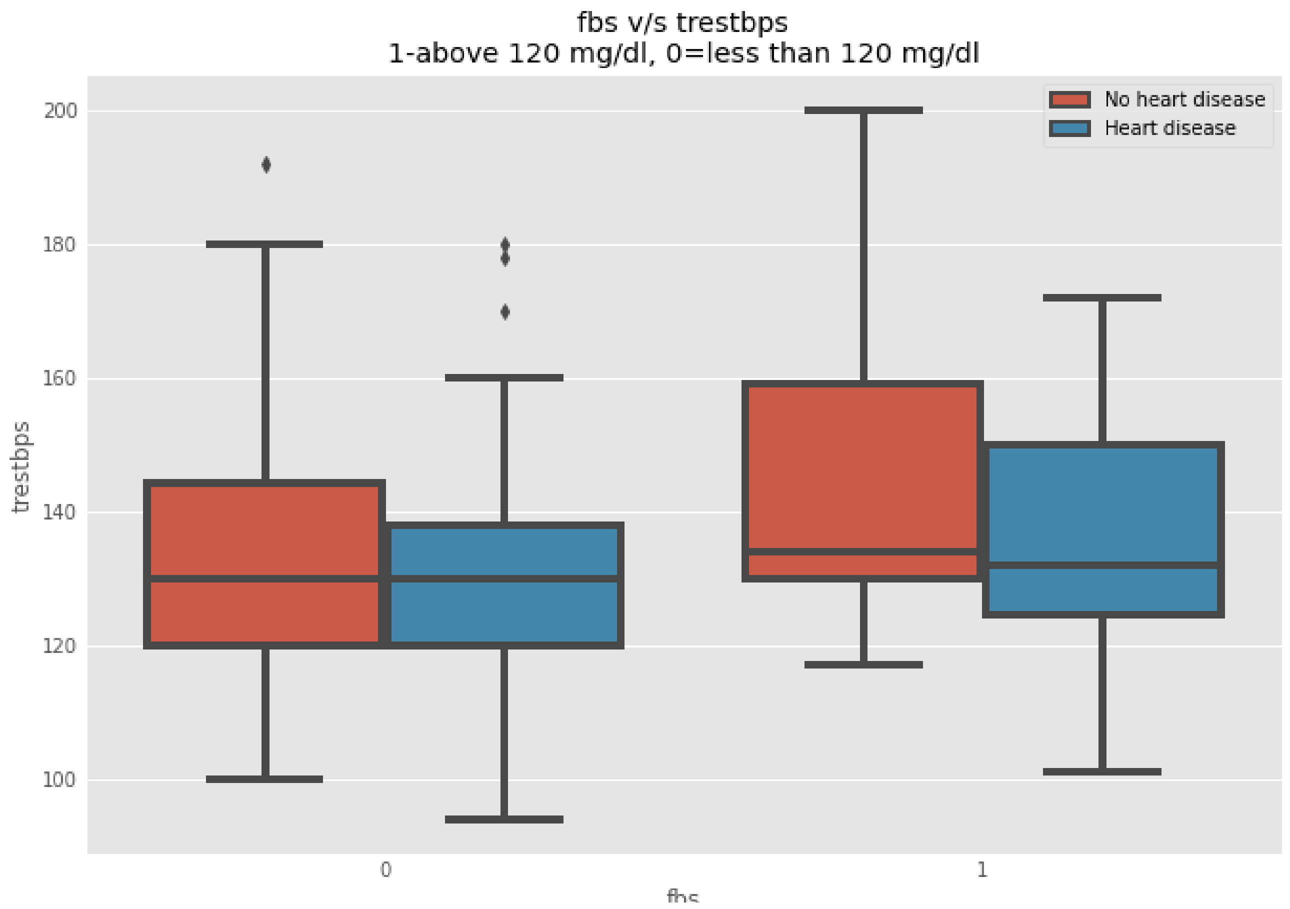
```
<seaborn.axisgrid.JointGrid at 0x7f0daa3f17d0>
```



Joint plots in seaborn helps us to understand the trend seen among two features. As observed from the above plot we can see that most of the Heart diseased patients in their age of upper 50s or lower 60s tend to have Cholesterol between 200mg/dl to 300mg/dl.

```
plt.figure(figsize=(10,7))
sns.boxplot(data=df,x='fbs',y='trestbps',hue='target')
L=plt.legend()
L.get_texts()[0].set_text('No heart disease') #Changing the legend texts from 0 and 1 to t
L.get_texts()[1].set_text('Heart disease')
plt.title('fbs v/s trestbps\n'"1-above 120 mg/dl, 0=less than 120 mg/dl")
```

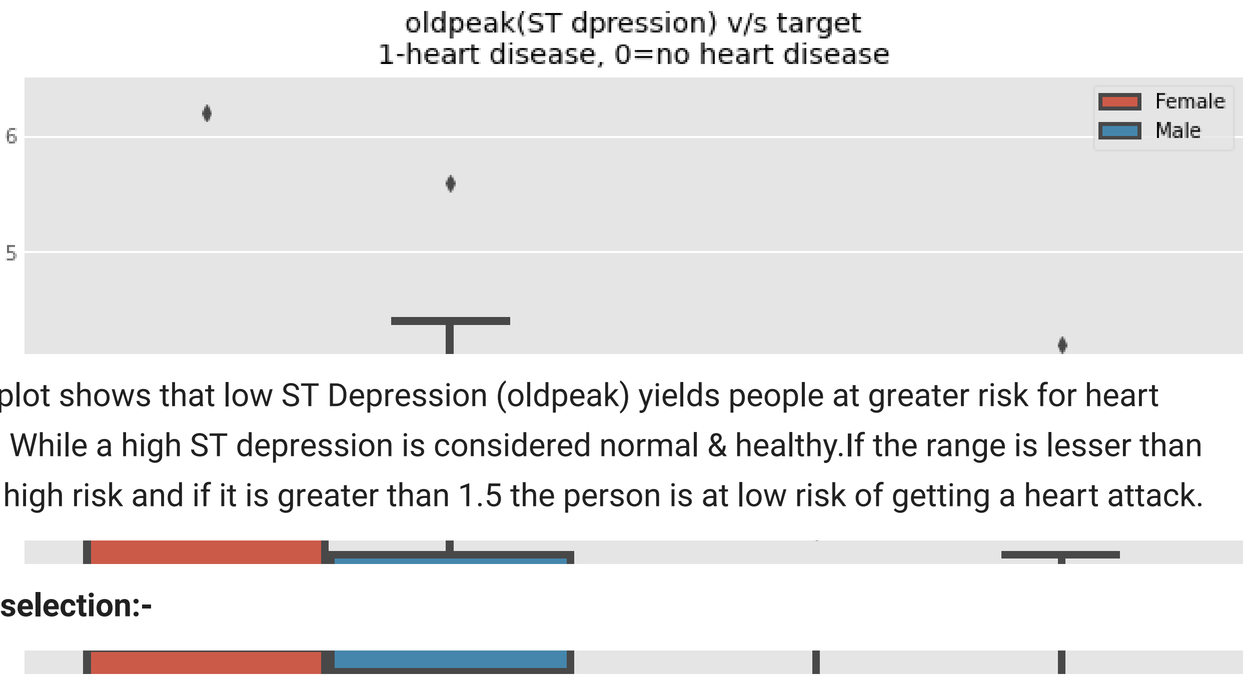
```
Text(0.5, 1.0, 'fbs v/s trestbps\n1-above 120 mg/dl, 0=less than 120 mg/dl')
```



The boxplot shows that the people who are diabetic (fasting blood sugar) has higher resting blood pressure which concludes that these group of people tend to have higher chances of having heart disease.

```
plt.figure(figsize=(10,7))
sns.boxplot(data=df,x='target',y='oldpeak',hue='sex')
L=plt.legend()
L.get_texts()[0].set_text('Female') #Changing the legend texts from 0 and 1 to these.
L.get_texts()[1].set_text('Male')
plt.title('oldpeak(ST dpression) v/s target\n'"1-heart disease, 0=no heart disease")
```

```
Text(0.5, 1.0, 'oldpeak(ST dpression) v/s target\n1-heart disease, 0=no heart diseas
```



The boxplot shows that low ST Depression (oldpeak) yields people at greater risk for heart disease. While a high ST depression is considered normal & healthy.If the range is lesser than 1.5 its a high risk and if it is greater than 1.5 the person is at low risk of getting a heart attack.

Feature selection:-

```
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import preprocessing #for normalization of data
X = df.iloc[:,0:13] # Features
y = df.iloc[:,13] # Target variable

X = preprocessing.normalize(X) #normalizing the features
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=253
```

The feature and the target label is assigned and the predictor variables are normalized using the preprocessing API to have a similar range across all the independent features. The train,test split is done where the test size allocated is 15% on the data with a random state.

Lasso (L1) penalty feature selection:

```
from sklearn.linear_model import Lasso, LogisticRegression
from sklearn.feature_selection import SelectFromModel

sel = SelectFromModel(LogisticRegression(C=1, penalty='l1',solver='liblinear'))
sel.fit(X_train, y_train)
selected=sel.get_support()
selected

    array([False, False, False,  True,  True, False, False,  True, False,
           False, False, False, False])
```

The L1 penalty feature selection is used to determine the features that are needed that is the once that influence the target variable the most are selected.The features whose coefficients shrank to zero are removed and the once which didn't are selected.

```
for i in range(len(selected)):
    if(selected[i]==True):
       print("selected feature",df.columns[i])
    else:
       print("Feature not selected",df.columns[i])
```

```
Feature not selected age
Feature not selected sex
Feature not selected cp
selected feature trestbps
selected feature chol
Feature not selected fbs
Feature not selected restecg
selected feature thalach
Feature not selected exang
Feature not selected oldpeak
Feature not selected slope
Feature not selected ca
Feature not selected thal
```

The features which are selcted by the L1 penalty are "trestbps", "chol" and "thalach".

Backward Feature selection:

```
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2
301	57	1	0	130	131	0	1	115	1	1.2	1	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1

303 rows × 14 columns

```
import statsmodels.api as sm
X = sm.add_constant(X)
logit_mod = sm.Logit(y,X)
logit_res = logit_mod.fit()
print(logit_res.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.352262
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  303
Model:                          Logit   Df Residuals:                      289
Method:                           MLE   Df Model:                           13
Date:                Tue, 16 Nov 2021   Pseudo R-squ.:                  0.4889
Time:                        01:26:37   Log-Likelihood:                -106.74
converged:                       True   LL-Null:                       -208.82
Covariance Type:            nonrobust   LLR p-value:                 1.903e-36
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         15.6826     24.841      0.631      0.528     -33.004      64.370
x1             1.3951      7.016      0.199      0.842     -12.356      15.146
x2          -501.9737    145.189     -3.457      0.001    -786.540    -217.408
x3           276.2157     58.950      4.686      0.000     160.676     391.756
x4           -11.3190     11.061     -1.023      0.306     -32.999      10.361
x5           -13.0555     19.207     -0.680      0.497     -50.700      24.589
x6            -3.1053    170.119     -0.018      0.985    -336.533     330.323
x7           159.5062    109.469      1.457      0.145     -55.050     374.062
x8             2.5333     11.700      0.217      0.829     -20.398      25.464
x9          -268.3064    127.799     -2.099      0.036    -518.788     -17.825
x10         -173.6546     67.649     -2.567      0.010    -306.244     -41.065
x11          200.8780    111.192      1.807      0.071     -17.053     418.809
x12         -249.4740     59.509     -4.192      0.000    -366.110    -132.838
x13         -274.3364     91.331     -3.004      0.003    -453.343     -95.330
==============================================================================
```

We use the other method known as backward feature selection , the selection of a feature here depends on its p-value.In this method at first all the features are selected and the p-value for every feature is noted, if the p-value of any variable greater than that of "0.05" that feature removed from the group and features with p-value lesser than "0.05" are kept and tested again and we continue until we reach a point where all the feature has p-value lesser than "0.05", once we achieve this we stop and select those features.

```
X=df[['sex','exang','oldpeak','ca','thal']]
X=preprocessing.normalize(X)
X
```

```
array([[0.37037037, 0.        , 0.85185185, 0.        , 0.37037037],
       [0.24077171, 0.        , 0.84270097, 0.        , 0.48154341],
       [0.        , 0.        , 0.57346234, 0.        , 0.81923192],
       ...,
       [0.19779694, 0.        , 0.67250961, 0.39559389, 0.59339083],
       [0.27277236, 0.27277236, 0.32732684, 0.27277236, 0.81831709],
       [0.        , 0.        , 0.        , 0.4472136 , 0.89442719]])
```

```
X = sm.add_constant(X)
logit_mod = sm.Logit(y,X)
logit_res = logit_mod.fit()
print(logit_res.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.476454
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                 target   No. Observations:                  303
Model:                          Logit   Df Residuals:                      297
Method:                           MLE   Df Model:                            5
Date:                Tue, 16 Nov 2021   Pseudo R-squ.:                  0.3087
Time:                        01:26:37   Log-Likelihood:                -144.37
converged:                       True   LL-Null:                       -208.82
Covariance Type:            nonrobust   LLR p-value:                 4.060e-26
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.1999      1.892      4.862      0.000       5.491      12.909
x1            -4.3979      1.086     -4.048      0.000      -6.527      -2.269
x2            -6.3986      1.094     -5.851      0.000      -8.542      -4.255
x3            -5.6949      0.984     -5.788      0.000      -7.623      -3.766
x4            -6.2186      0.951     -6.540      0.000      -8.082      -4.355
x5            -5.7756      1.671     -3.457      0.001      -9.050      -2.501
==============================================================================
```

The Selected features are 'sex','exang','oldpeak','ca','thal'. There is a difference when we use a different method for feature selection. There is no particular method it all depends on the model and the data, so we need to try different methods. We can also use "ridge regression", "forward selection" and etc.

Modeling:

We are using several classification models to see which model gives us the highest accuracy score in predicting the heart attack if a new record enrolls.

```
#updating the values

X = df.iloc[:,0:13] # Features
y = df.iloc[:,13] # Target variable

X = preprocessing.normalize(X) #normalizing the features
```

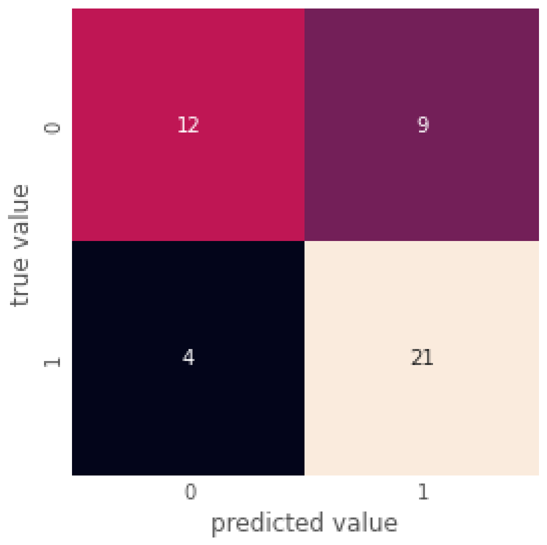
The first model which we are using is the logistic regression model.

Logistic regression

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(C=1, penalty='l2',solver='newton-cg')
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

y_pred
```

```
    array([0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
           1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
           1, 1])
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix


mat = confusion_matrix(y_test, y_pred) #we dont do this because we dont get the whole numb
#plot_confusion_matrix(lr, X, y, values_format = '') #this is to fet the whole number anno
sns.heatmap(mat, square=True, annot=True, cbar=False)
plt.xlabel('predicted value')
plt.ylabel('true value');
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.57      0.65        21
           1       0.70      0.84      0.76        25

    accuracy                           0.72        46
   macro avg       0.72      0.71      0.71        46
weighted avg       0.72      0.72      0.71        46
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


print('Accuracy: %.3f' % accuracy_score(y_true=y_test, y_pred=y_pred))
print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

```
Accuracy: 0.717
Precision: 0.700
Recall: 0.840
F1: 0.764
```

Using the metrics we get the accuracy,precision,Recall and F1-score .

```
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?

logistic_reg_accuracy=metrics.accuracy_score(y_test, y_pred)



print("Accuracy:",logistic_reg_accuracy)
```

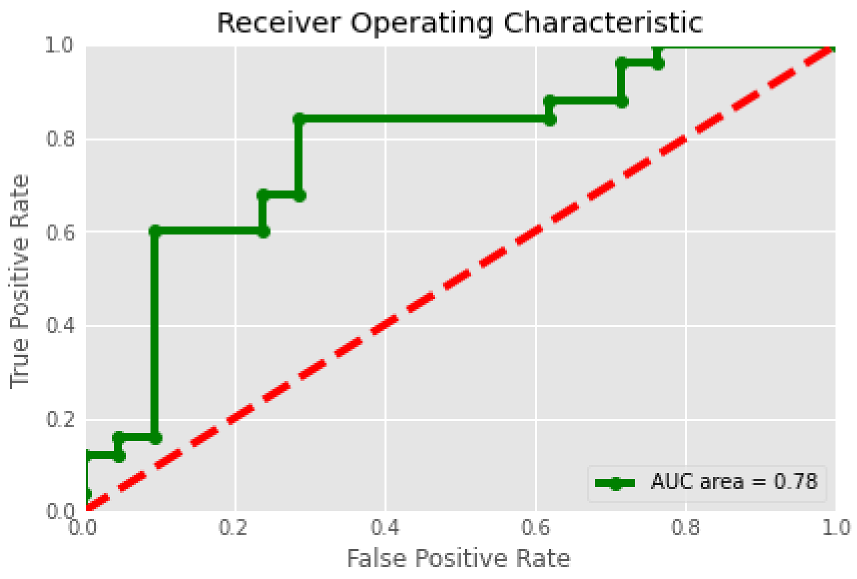
```
Accuracy: 0.717391304347826
```

The accuracy score for the model is 71.7%.

```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = lr.predict_proba(X_test)





preds = probs[:,1] # we need the 1st column as we need prob of being positive
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, color='green',marker='o', label = 'AUC area = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--') #diagonal line
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
```

```
lr_roc=roc_auc

print('Roc_score = %.2f' %(lr_roc)) #printing the auc score


plt.show()
```

```
Roc_score = 0.78
```



The Roc_auc score for the model is 78%.

2.The second classification model which we are using is Support Vector Machine

Support vector machine

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear',probability=True) # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


print('Accuracy: %.3f' % accuracy_score(y_true=y_test, y_pred=y_pred))
print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

```
     Accuracy: 0.630
     Precision: 0.611
     Recall: 0.880
     F1: 0.721
```

Using the metrics we get the accuracy,precision,Recall and F1-score .

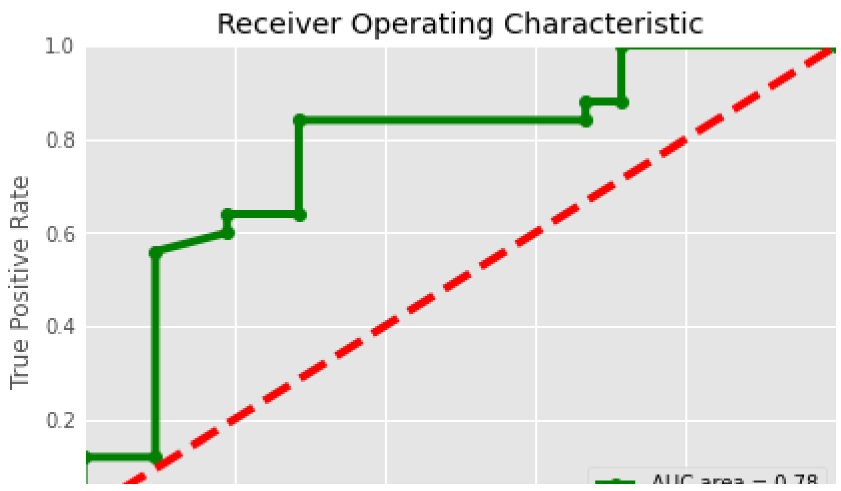
```
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?

svm_accuracy=metrics.accuracy_score(y_test, y_pred)


print("Accuracy:",svm_accuracy)
```

```
     Accuracy: 0.6304347826086957
```

The accuracy score for the model is 63%.

```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = clf.predict_proba(X_test)





preds = probs[:,1] # we need the 1st column as we need prob of being positive
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, color='green',marker='o', label = 'AUC area = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--') #diagonal line
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

svm_roc=roc_auc

print('Roc_score = %.2f' %(svm_roc)) #printing the auc score

plt.show()
```

```
Roc_score = 0.78
```



The Roc_auc score for the model is 77%.

3.The third model which we are using is Gaussian Naive Bayes.

Gaussian Naive bayes

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)
```

Confusion matrix for GNB model.

```
#confusion matrix for GNB
mat = confusion_matrix(y_test, y_pred) #we dont do this because we dont get the whole numb
#plot_confusion_matrix(lr, X, y, values_format = '') #this is to fet the whole number anno
sns.heatmap(mat, square=True, annot=True, cbar=False)
plt.xlabel('predicted value')
plt.ylabel('true value');
```

```
print('Accuracy: %.3f' % accuracy_score(y_true=y_test, y_pred=y_pred))
print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

```
Accuracy: 0.826
Precision: 0.774
Recall: 0.960
F1: 0.857
```

Using the metrics we get the accuracy,precision,Recall and F1-score .

```
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?
gaussian_nb_accuracy=metrics.accuracy_score(y_test, y_pred)

print("Accuracy:",gaussian_nb_accuracy)
```

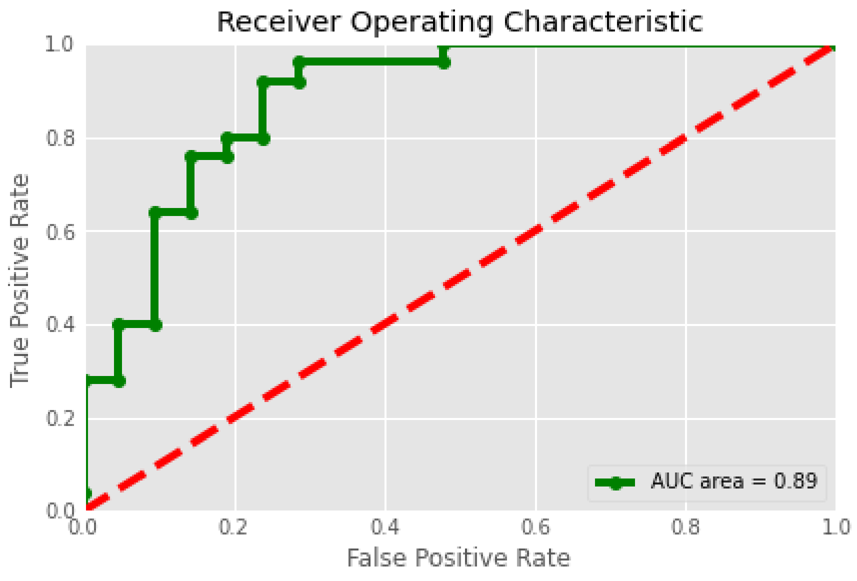
```
Accuracy: 0.8260869565217391
```

We use the gaussian naive bayes model to classify the class whether the patient will suffer from "Heart attack" or not.We use this model because every feature in this data is independent of each other and its relation to the target variable is also independent.The model is able to predict the results with 82.6% accuracy which is relatively decent.

```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = gnb.predict_proba(X_test)
```

```
preds = probs[:,1] # we need the 1st column as we need prob of being positive
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, color='green',marker='o', label = 'AUC area = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--') #diagonal line
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

GNB_roc=roc_auc

print('Roc_score = %.2f' %(GNB_roc)) #printing the auc score


plt.show()
```

```
    Roc_score = 0.89
```



The Roc_auc score for the model is 89%.

KNN

The third classification model we use is KNN.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

```
    KNeighborsClassifier()
```

```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[15  6]
 [10 15]]
              precision    recall  f1-score   support

           0       0.60      0.71      0.65        21
           1       0.71      0.60      0.65        25

    accuracy                           0.65        46
   macro avg       0.66      0.66      0.65        46
weighted avg       0.66      0.65      0.65        46
```

```
print('Accuracy: %.3f' % accuracy_score(y_true=y_test, y_pred=y_pred))
print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

```
Accuracy: 0.652
Precision: 0.714
Recall: 0.600
F1: 0.652
```

Using the metrics we get the accuracy,precision,Recall and F1-score .

```
from sklearn import metrics

# Model Accuracy, how often is the classifier correct?

KNN_accuracy=metrics.accuracy_score(y_test, y_pred)


print("Accuracy:",KNN_accuracy)
```

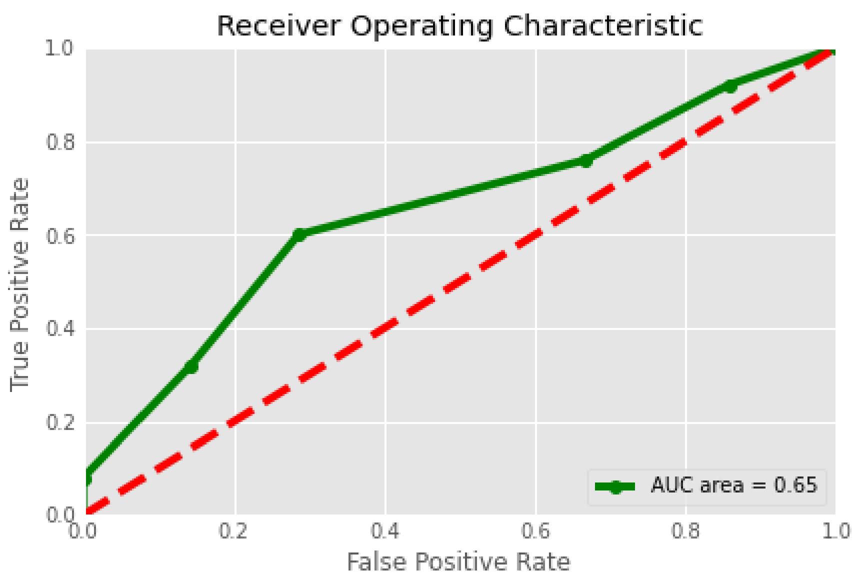
```
Accuracy: 0.6521739130434783
```

The accuracy score for the model is 65.2%.

```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = classifier.predict_proba(X_test)
```

```
preds = probs[:,1] # we need the 1st column as we need prob of being positive
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, color='green',marker='o', label = 'AUC area = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--') #diagonal line
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

KNN_roc=roc_auc

print('Roc_score = %.2f' %(KNN_roc)) #printing the auc score


plt.show()
```

```
Roc_score = 0.65
```



The Roc_auc score for the model is 65%.

Decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)



print('Accuracy: %.3f' % accuracy_score(y_true=y_test, y_pred=y_pred))
print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

```
Accuracy: 0.717
Precision: 0.773
Recall: 0.680
F1: 0.723
```

Using the metrics we get the accuracy,precision,Recall and F1-score .

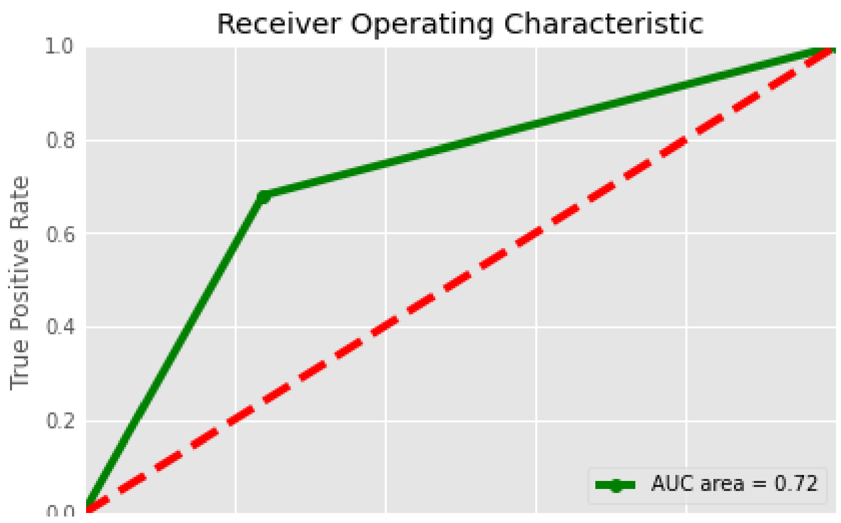
```
# Model Accuracy, how often is the classifier correct?

DecisionTree_accuracy=metrics.accuracy_score(y_test, y_pred)


print("Accuracy:",DecisionTree_accuracy)
```

```
Accuracy: 0.717391304347826
```

The accuracy score for the model is 76%.

```
import sklearn.metrics as metrics
# calculate the fpr and tpr for all thresholds of the classification
probs = clf.predict_proba(X_test)





preds = probs[:,1] # we need the 1st column as we need prob of being positive
fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr, tpr)

# method I: plt
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, color='green',marker='o', label = 'AUC area = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--') #diagonal line
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

DecissionTree_roc=roc_auc

print('Roc_score = %.2f' %(DecissionTree_roc)) #printing the auc score


plt.show()
```

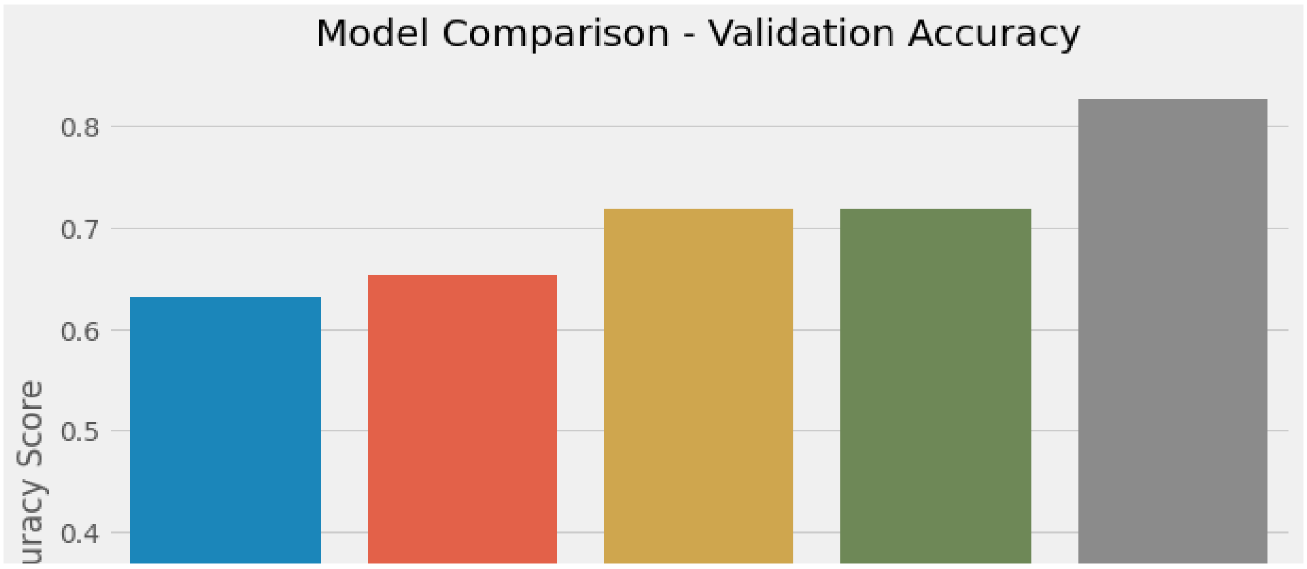
```
Roc_score = 0.72
```



The Roc_auc score for the model is 76%.

Comparing the models using accuracy score

```
accuracy_list=[logistic_reg_accuracy,svm_accuracy,gaussian_nb_accuracy,KNN_accuracy,Decisi
accuracy_list.sort()
# print(accuracy_list)
classifier_names_list=['SVM','KNN','logistic_regression','DecisionTree','GaussianNB']


plt.style.use("fivethirtyeight")
plt.figure(figsize=(10, 8))
sns.barplot(x=classifier_names_list, y=accuracy_list)
plt.xlabel("Models")
plt.ylabel("Accuracy Score")
plt.xticks(rotation=45)
plt.title("Model Comparison - Validation Accuracy")
plt.show()
```



Results:-

The barplot shows the Gaussian Naive bayes model performs the best amongst all the classification models that we used.

The accuracy of the models follows the order as listed:-

1.Gaussian naive bayes (Best model)

2.Decission Tree

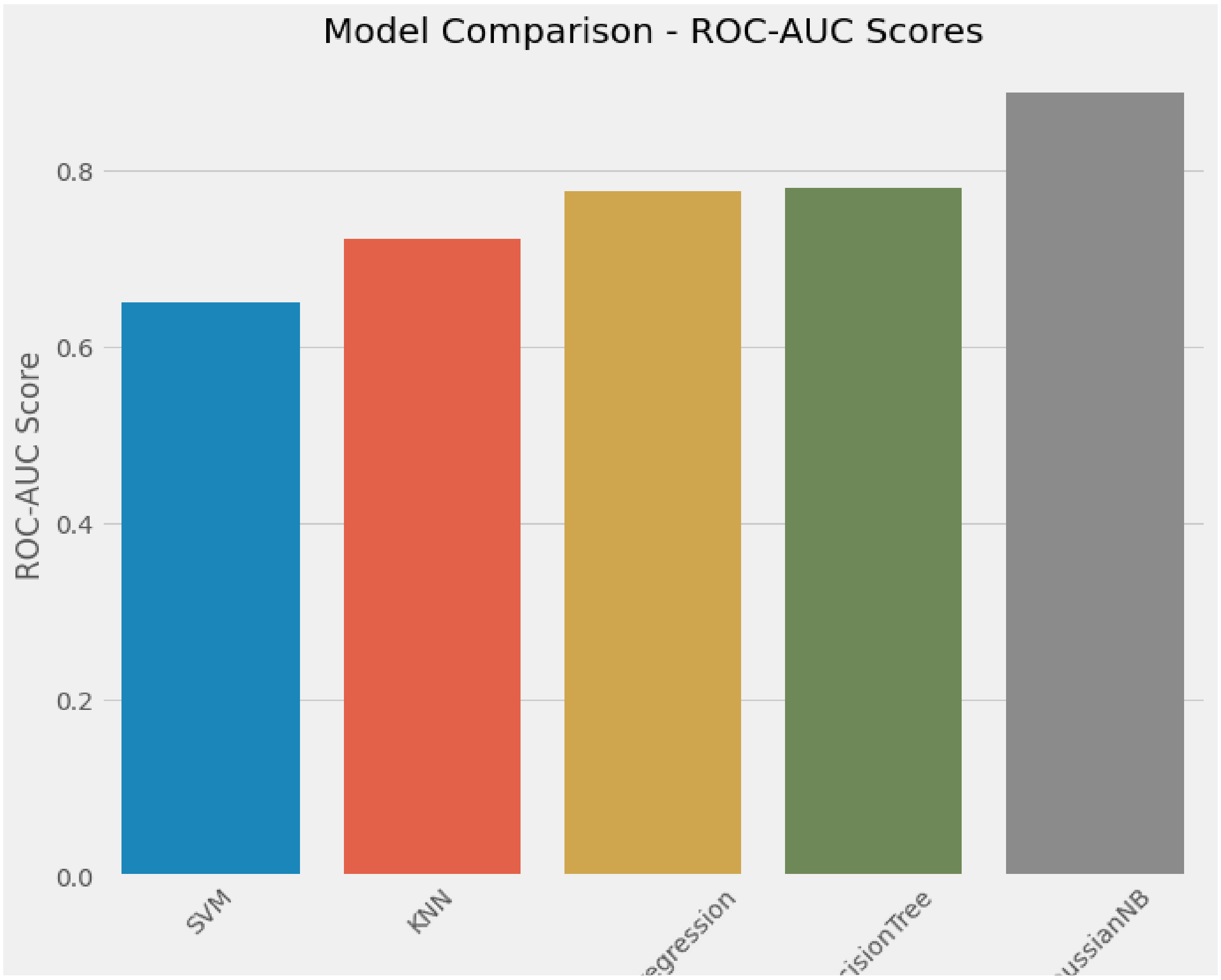
3.Logistic Regression

4.K-Nearest-Neighbors

5.Support vector Machine

Comparing models using Roc_auc score

```
Roc_auc_list=[lr_roc,svm_roc,GNB_roc,KNN_roc,DecissionTree_roc]
Roc_auc_list.sort()
classifier_names_list=['SVM','KNN','logistic_regression','DecisionTree','GaussianNB']


plt.style.use("fivethirtyeight")
plt.figure(figsize=(10, 8))
sns.barplot(x=classifier_names_list, y=Roc_auc_list)
plt.xlabel("Models")
plt.ylabel("ROC-AUC Score")
plt.xticks(rotation=45)
plt.title("Model Comparison - ROC-AUC Scores")
plt.show()
```



Results:-

The barplot shows the Gaussian Naive bayes model performs the best amongst all the classification models that we used.

The Roc_auc score of the models follows the order as listed:-

1.Gaussian naive bayes (Best model)

2.Decission Tree

3.Logistic Regression

4.K-Nearest-Neighbors

5.Support vector Machine

✓ 0s completed at 8:26 PM