HEART DISEASE DETECTION

NAME: NITIN SINGH TATRARI

BATCH: 1825

Problem statement:

Heart disease describes a range of conditions that affect your heart. The term “heart disease” is often used interchangeably with the term “cardiovascular disease”. Cardiovascular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart’s muscle, valves or rhythm, also are considered forms of heart disease.

Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of clinical data analysis. The amount of data in the healthcare industry is huge. Data mining turns the large collection of raw healthcare data into information that can help to make informed decisions and predictions. It is difficult to identify heart disease because of several contributory risk factors such as diabetes, high blood pressure, high cholesterol, abnormal pulse rate, and many other factors. Due to such constraints, scientists have turned towards modern approaches like Data Mining and Machine Learning for predicting the disease.

Machine learning (ML) proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry.

In this article, I’ll discuss a project where I worked on predicting potential Heart Diseases in people using Machine Learning algorithms. The algorithms included K Neighbors Classifier, Support Vector Classifier, Decision Tree Classifier, and Logistic Regression & Random Forest Classifier.

DATA ANALYSIC:

The data provided have 12 attributes and 1 target variable. All the variables are as follows:-

Sl. no.	Attribute	Description
0	Age	Age in years
1	sex	(1 = male; 0 = female)
2	cp	chest pain type - (1,2,3,4)
3	trestbps	resting blood pressure (in mm Hg on admission to the hospital)
4	chol	serum cholesterol in mg/dl
5	fbs	(fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
6	restecg	resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)
7	thalach	maximum heart rate achieved
8	exang	exercise induced angina (1 = yes; 0 = no)
9	oldpeak	ST depression induced by exercise relative to rest
10	slope	the slope of the peak exercise ST segment
11	ca	number of major vessels colored by fluoroscopy
12	thal	Thalassemia (3 = normal; 6 = fixed defect; 7 = reversible defect)
13	Target	1 or 0

1. Importing libraries and dataset

Firstly, we import libraries- pandas, numpy, seaborn & matplotlib.Through pandas we import data.

Through pandas we import data “heart_disease.csv”.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```python
data=pd.read_csv('heart_disease.csv',header=None)
data.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	63	1	4	140	260	0	1	112	1	3	2	?	?	2
1	44	1	4	130	209	0	1	127	0	0	?	?	?	0
2	60	1	4	132	218	0	1	140	1	1.5	3	?	?	2
3	55	1	4	142	228	0	1	149	1	2.5	1	?	?	1
4	66	1	3	110	213	1	2	99	1	1.3	2	?	?	0

2. Loading data into Data Frame and renaming columns' name.

```
df=pd.DataFrame(data)
```

```
colmn=['age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak','slope','ca','thal','target']
df.columns=colmn
```

3. Understanding Data

```
for i in df.columns:
    a=df[i].value_counts()
    print('Column',i)
    print(a)
```

```
Column age
63      15
62      15
55      14
60      12
58      12
61      12
57      10
64       9
56       9
69       8
59       8
54       7
51       7
65       6
74       5
67       5
53       5
68       4
48       4
72       3
66       3
75       3
40       2
46       2
50       2
49       2
41       2
52       2
71       2
42       1
37       1
38       1
77       1
43       1
44       1
45       1
76       1
70       1
35       1
Name: age, dtype: int64
Column sex
1    194
0      6
Name: sex, dtype: int64
```

```
Column cp
4    131
3     47
2     14
1      8
Name: cp, dtype: int64
Column trestbps
120    24
130    19
150    16
140    15
160    15
110    12
122    10
128     6
134     6
144     6
125     5
142     5
158     4
126     4
136     4
170     4
124     4
154     4
112     3
156     3
132     3
104     3
155     2
180     2
178     2
116     2
138     2
152     2
127     2
0       1
172     1
102     1
106     1
146     1
118     1
190     1
135     1
96      1
100     1
114     1
Name: trestbps, dtype: int64
```

```
Column chol
0      49
203     4
220     4
223     4
289     4
       ..
297     1
160     1
384     1
333     1
142     1
Name: chol, Length: 99, dtype: int64
Column fbs
0    129
1     71
Name: fbs, dtype: int64
Column restecg
1    93
0    80
2    27
Name: restecg, dtype: int64
```

```
Column exang
1    127
0     73
Name: exang, dtype: int64
Column oldpeak
0       62
2       35
1.5     31
1       21
3       18
2.5     12
0.5      8
4        6
1.3      2
1.6      1
3.5      1
0.8      1
-0.5     1
1.7      1
Name: oldpeak, dtype: int64
Column target
1    56
0    51
3    42
2    41
4    10
Name: target, dtype: int64
```

```
Column thalach
120    18
140    16
150    11
110     9
112     9
130     8
100     7
84      5
135     5
86      5
102     5
119     5
160     5
106     5
128     4
118     4
122     4
111     4
105     4
125     4
154     4
117     3
148     3
96      3
108     3
129     3
138     2
98      2
80      2
144     2
88      2
180     2
133     2
145     2
127     2
94      2
165     2
93      1
161     1
97      1
141     1
159     1
149     1
73      1
155     1
113     1
124     1
121     1
107     1
132     1
69      1
131     1
123     1
99      1
115     1
162     1
157     1
151     1
126     1
```

The following points we understand:-

a) Columns 'Slope',' ca' & 'thal' have 102, 198 & 166 missing value '?'. The columns Slope, ca & thal have high percentage of missing value i.e 51%, 99% & 83%, we can drop the columns.
b) Columns 'trestbps', 'chol', 'fbs', 'thalach', 'exang' & 'oldpeak' have 56,7,7,53,53 & 56 missing values'?'. The missing value is considerable, thus we can treat the column for missing values.
c) Column 'sex' has 194 value count of '1 '& 6 value count of '0'. Thus we can ignore value '0'.
d) Column 'Chol' have 49 value count of '0'. But Cholesterol cannot be zero. Thus data is wrong. Thus we have to treat it.

A) Dropping columns- 'slope', 'ca', 'thal':

```
df.drop({'slope','ca','thal'},axis=1,inplace=True)
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
0	63.0	1.0	4.0	140	260	0	1	112	1	3	2
1	44.0	1.0	4.0	130	209	0	1	127	0	0	0
2	60.0	1.0	4.0	132	218	0	1	140	1	1.5	2
3	55.0	1.0	4.0	142	228	0	1	149	1	2.5	1
4	66.0	1.0	3.0	110	213	1	2	99	1	1.3	0

B) Treating '?'

```
series=['trestbps','chol','fbs','thalach','exang','oldpeak']
for i in series:
    df[i]=df[i].replace('?',method ='ffill')
```

We have replace '?' in the columns- 'testbps', 'chol', 'fbs', 'thalach', 'exang' & 'oldpeak' with forward fill method

C) Treating Data type

There are 5 integer type and 6 string type columns in the data frame.
Since all columns have numerical values, we have to change the data type to float.

```
df.dtypes
```

```
age            int64
sex            int64
cp             int64
trestbps      object
chol          object
fbs           object
restecg        int64
thalach       object
exang         object
oldpeak       object
target         int64
dtype: object
```

There 5 integer type columns and 6 string type columns

Since all coulmns have numerical value, we can change the data type to float

```
for i in df.columns:
    df[i]=df[i].astype(float)
```

```
df.dtypes
```

```
age           float64
sex           float64
cp            float64
trestbps      float64
chol          float64
fbs           float64
restecg       float64
thalach       float64
exang         float64
oldpeak       float64
target        float64
dtype: object
```

D) Treating 'Chol' column

We will replace value '0' in the column 'chol' with the mean of column 'Chol'.

```
df['chol'].mean()
```

```
181.545
```

```
df['chol']=df['chol'].replace(0,182)
```

```
df['chol'].value_counts()
```

```
0       49
203      4
220      4
223      4
289      4
        ..
297      1
160      1
384      1
333      1
142      1
Name: chol, Length: 99, dtype: int64
```

E) There are outliers in Age, trestbps, chol & thalach.

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
count	200.000000	200.000000	200.000000	200.00000	200.000000	200.000000	200.000000	200.000000	200.000000	200.00000	200.000000
mean	59.350000	0.970000	3.505000	134.58000	226.135000	0.355000	0.735000	122.680000	0.635000	1.29600	1.520000
std	7.811697	0.171015	0.795701	20.44022	52.378568	0.479714	0.683455	21.749316	0.482638	1.12486	1.219441
min	35.000000	0.000000	1.000000	0.00000	100.000000	0.000000	0.000000	69.000000	0.000000	-0.50000	0.000000
25%	55.000000	1.000000	3.000000	120.00000	182.000000	0.000000	0.000000	108.000000	0.000000	0.00000	0.000000
50%	60.000000	1.000000	4.000000	130.00000	216.500000	0.000000	1.000000	120.000000	1.000000	1.50000	1.000000
75%	64.000000	1.000000	4.000000	150.00000	258.000000	1.000000	1.000000	140.000000	1.000000	2.00000	3.000000
max	77.000000	1.000000	4.000000	190.00000	458.000000	1.000000	2.000000	180.000000	1.000000	4.00000	4.000000

F) Treating outliers and removing skewness

```
from scipy.stats import zscore
```

```
z=np.abs(zscore(df))
df2=df[(z<3).all(axis=1)]
```

```
df2.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
0	63.0	1.0	4.0	140.0	260.0	0.0	1.0	112.0	1.0	3.0	2.0
1	44.0	1.0	4.0	130.0	209.0	0.0	1.0	127.0	0.0	0.0	0.0
2	60.0	1.0	4.0	132.0	218.0	0.0	1.0	140.0	1.0	1.5	2.0
3	55.0	1.0	4.0	142.0	228.0	0.0	1.0	149.0	1.0	2.5	1.0
4	66.0	1.0	3.0	110.0	213.0	1.0	2.0	99.0	1.0	1.3	0.0

```
df2.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
count	181.000000	181.0	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000	181.000000
mean	59.348066	1.0	3.618785	135.558011	222.861878	0.337017	0.745856	122.707182	0.657459	1.304972	1.569061
std	7.746351	0.0	0.608715	18.433159	46.762494	0.474002	0.684387	21.248984	0.475876	1.140140	1.211947
min	37.000000	1.0	2.000000	96.000000	100.000000	0.000000	0.000000	73.000000	0.000000	-0.500000	0.000000
25%	55.000000	1.0	3.000000	122.000000	182.000000	0.000000	0.000000	110.000000	0.000000	0.000000	1.000000
50%	60.000000	1.0	4.000000	130.000000	216.000000	0.000000	1.000000	120.000000	1.000000	1.500000	1.000000
75%	63.000000	1.0	4.000000	150.000000	254.000000	1.000000	1.000000	140.000000	1.000000	2.000000	3.000000
max	77.000000	1.0	4.000000	190.000000	369.000000	1.000000	2.000000	180.000000	1.000000	4.000000	4.000000

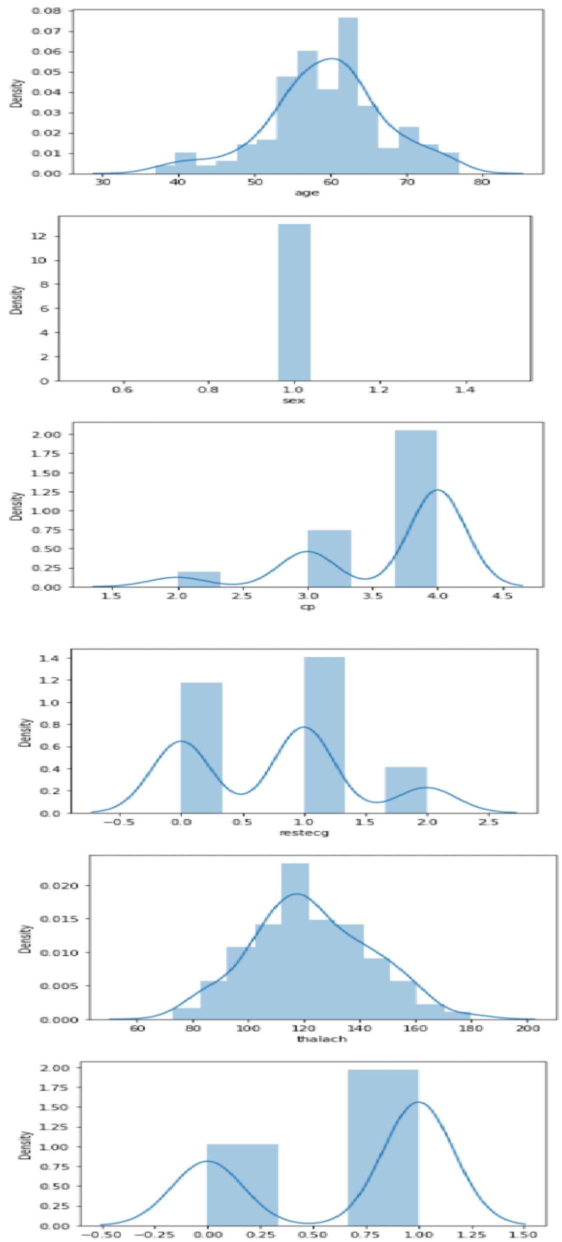
We remove all the value less than or more than threshold =3. The shape of new Data Frame df2 is (181,11).

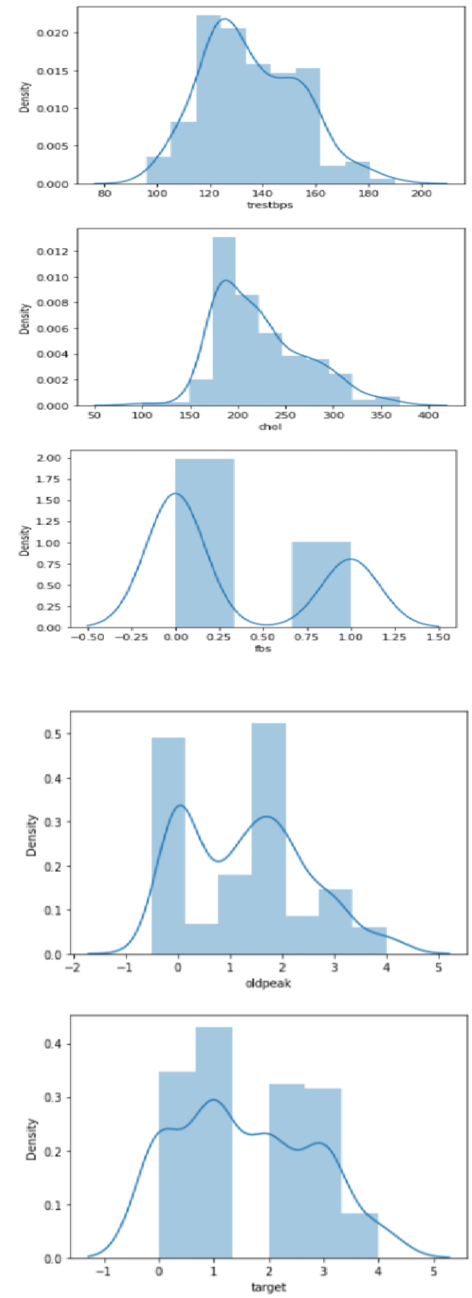
Outliers have reduced in Age, trestbps, chol & thalach.

4. <u>EDA</u>

A) Now, we will graphically study the variables.

```
for i in df2.columns:
    plt.figure
    sns.distplot(df2[i])
    plt.show()
```





While removing outliers, value ‘0’ having only 6 counts in column ‘age ‘was removed. Thus it has only value ‘1’. Thus we can remove the column.

We will also covert replace values 2,3,4 with 1 in Target variable, as we have to find out whether the person is having heart diseases or not. We does not have to find the severity of the disease.

```
df2.drop('sex',axis=1,inplace=True)
```

```
df2.head()
```

	age	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	target
0	63.0	4.0	140.0	260.0	0.0	1.0	112.0	1.0	3.0	2.0
1	44.0	4.0	130.0	209.0	0.0	1.0	127.0	0.0	0.0	0.0
2	60.0	4.0	132.0	218.0	0.0	1.0	140.0	1.0	1.5	2.0
3	55.0	4.0	142.0	228.0	0.0	1.0	149.0	1.0	2.5	1.0
4	66.0	3.0	110.0	213.0	1.0	2.0	99.0	1.0	1.3	0.0

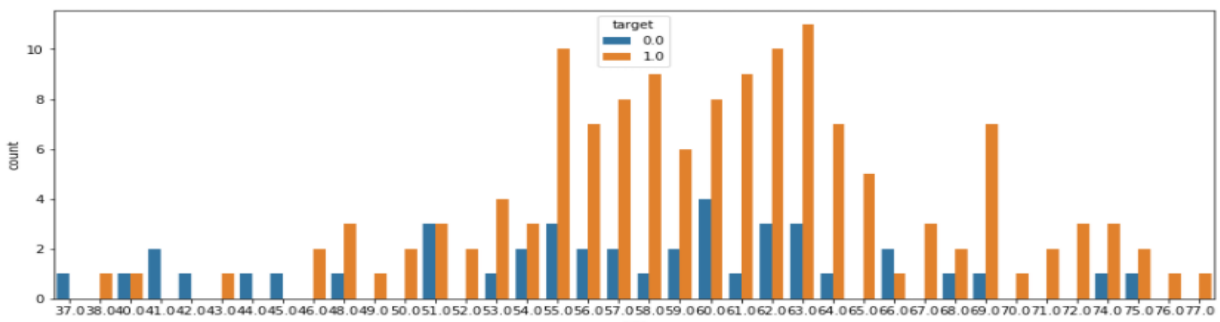
```
df2['target'].replace([2,3,4],1, inplace=True)
```

```
df2['target'].value_counts()
```

```
1.0    139
0.0     42
Name: target, dtype: int64
```

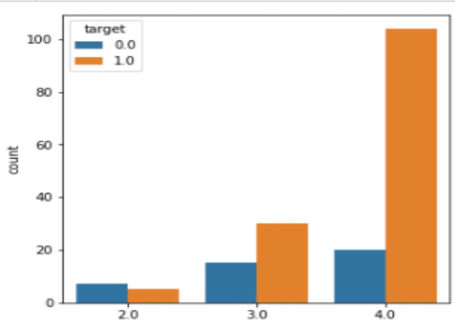
B) *Graphically, we will try to interpret the relationship of target variable with other attributes.*

```
plt.figure(figsize=(15,5))
sns.countplot(x='age',hue='target',data=df2)
plt.show()
```

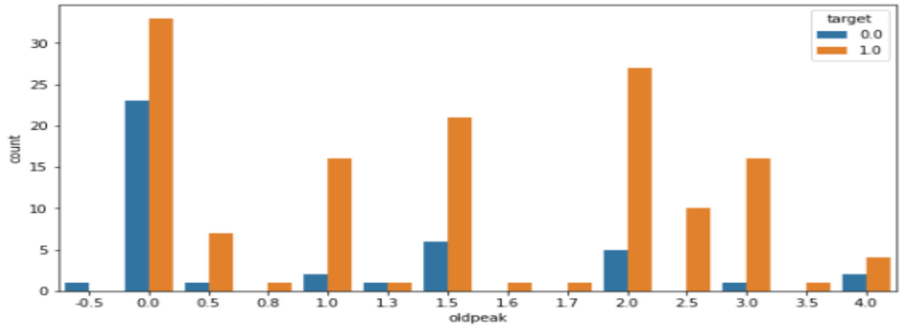


he can interpret that at younger age changes for heart disease is low.

```
plt.figure(figsize=(5,5))
sns.countplot(x='cp',hue='target',data=df2)
plt.show()
```

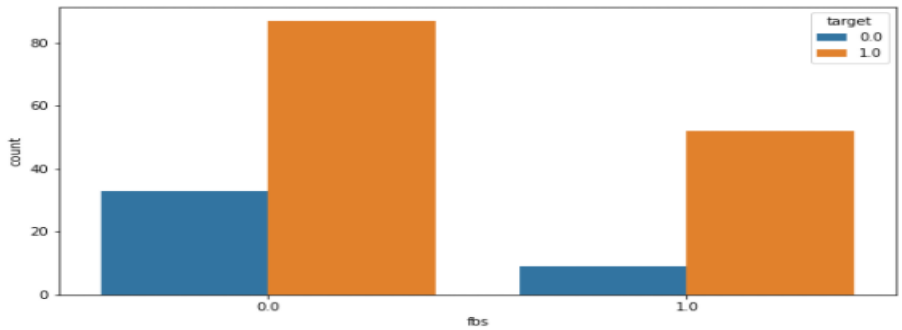


```python
plt.figure(figsize=(10,5))
sns.countplot(x='oldpeak',hue='target',data=df2)
plt.show()
```

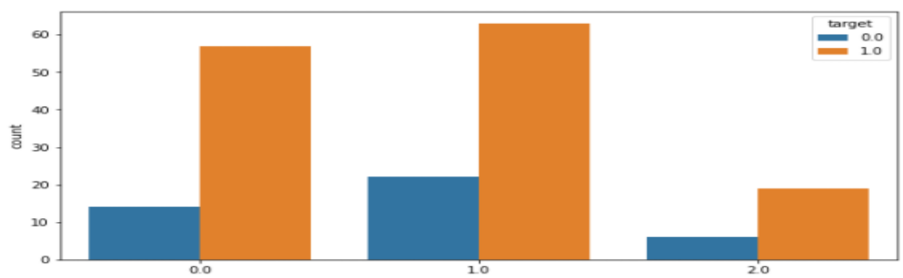


with the increase of oldpeak value , probability of heart diseases increases.

```python
plt.figure(figsize=(10,5))
sns.countplot(x='fbs',hue='target',data=df2)
plt.show()
```

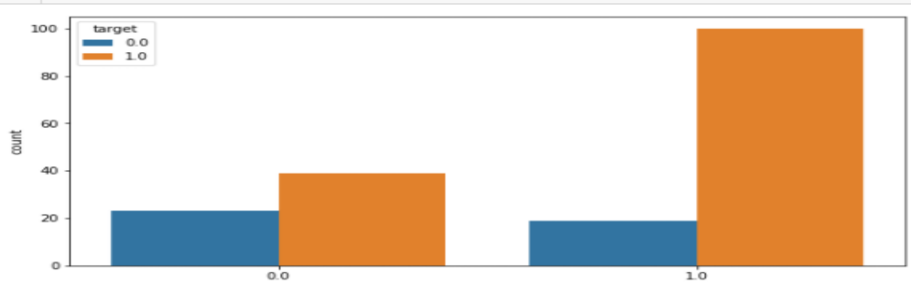



```python
plt.figure(figsize=(10,5))
sns.countplot(x='restecg',hue='target',data=df2)
plt.show()
```

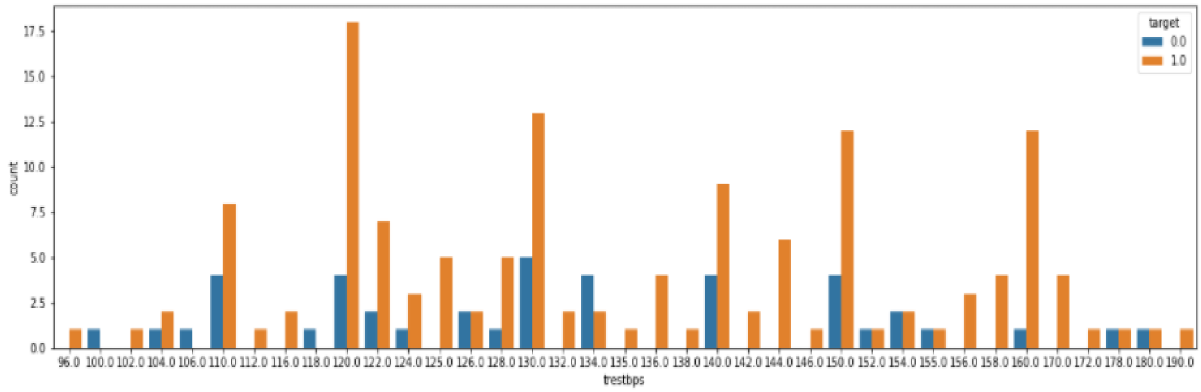


Patience having restecg value (1 = having ST-T) & (2 = hypertrophy), chances for heart diseases is less than 0 =norr

```python
plt.figure(figsize=(10,5))
sns.countplot(x='exang',hue='target',data=df2)
plt.show()
```

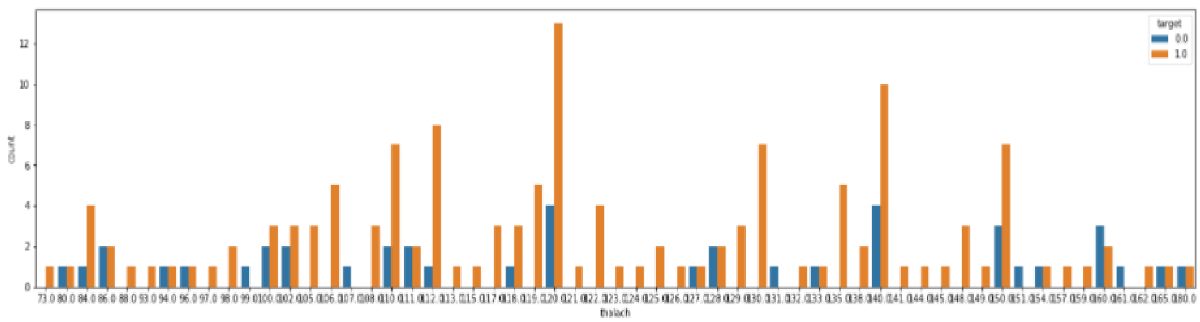


```python
plt.figure(figsize=(20,5))
sns.countplot(x='trestbps',hue='target',data=df2)
plt.show()
```



Patience having high trestbps(Resting blood pressure) tends to have more chances of Heart diseases

```python
plt.figure(figsize=(25,5))
sns.countplot(x='thalach',hue='target',data=df2)
plt.show()
```



Following points observed from the graphical representation:-

- The chances of Heart diseases in younger age group are less.
- Patience have type 4 of CP have more chances to diagnosed with heart disease.
- Higher value of oldpeak value shows higher percentage of heart diseases.
- We have observed patience having higher fasting sugar level, tend to have more chances to diagnose with heart disease.
- Patience having restecg value (1 = having ST-T) & (2 = hypertrophy) have lesser ratio of diagnosed with heart diseases is less than 0 =normal.
- Exercise induced angina is seen more in patience having heart problem.
- Patience having high trestbps(Resting blood pressure) tends to have higher diagnosed ratio of Heart diseases
- Heart disease is high between 100-150 value of thalach(maximum heart rate achieved).

C) Analyzing Correlation of target variable with other variables

```
plt.figure(figsize=(9,7))
sns.heatmap(df2.corr(), annot=True)
plt.show()
```


```
df2.drop({'thalach','restecg'},axis=1,inplace=True)
df2.head()
```

	age	cp	trestbps	chol	fbs	exang	oldpeak	target
0	63.0	4.0	140.0	260.0	0.0	1.0	3.0	1.0
1	44.0	4.0	130.0	209.0	0.0	0.0	0.0	0.0
2	60.0	4.0	132.0	218.0	0.0	1.0	1.5	1.0
3	55.0	4.0	142.0	228.0	0.0	1.0	2.5	1.0
4	66.0	3.0	110.0	213.0	1.0	1.0	1.3	0.0

'age', 'cp', 'Chol', 'fbs', 'exang', 'oldpeak shows correlation with target.

'restecg' and 'thalach shows less than 6% correlation with target. Thus, we have drop them.

5. Model Development and Evaluation

We drop ‘Target’ column from df2 and store the remaining variables in array ‘x’.

We store the target variable in array ‘y’

```
x=df2.drop(['target'],axis=1)
x.head(5)
```

	age	cp	trestbps	chol	fbs	exang	oldpeak
0	63.0	4.0	140.0	260.0	0.0	1.0	3.0
1	44.0	4.0	130.0	209.0	0.0	0.0	0.0
2	60.0	4.0	132.0	218.0	0.0	1.0	1.5
3	55.0	4.0	142.0	228.0	0.0	1.0	2.5
4	66.0	3.0	110.0	213.0	1.0	1.0	1.3

```
x.shape
```

```
(181, 7)
```

```
y=df2['target']
y
```

```
0      1.0
1      0.0
2      1.0
3      1.0
4      0.0
      ...
192    1.0
193    1.0
194    1.0
197    1.0
199    1.0
Name: target, Length: 181, dtype: float64
```

```
y.shape
```

```
(181,)
```

Both array x and y have equal rows, which is 181.

By using StandardScaler from sklearn library, we standardizes the value in ‘x’.

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
x
```

```
array([[ 0.47274696,  0.62799976,  0.24164663, ..., -0.71297499,
         0.72180912,  1.49080751],
       [-1.98682459,  0.62799976, -0.30235885, ..., -0.71297499,
        -1.38540782, -1.14774685],
       [ 0.08439356,  0.62799976, -0.19355775, ..., -0.71297499,
         0.72180912,  0.17153033],
       ...,
       [-1.72792232,  0.62799976, -0.08475665, ..., -0.71297499,
        -1.38540782, -1.14774685],
       [-0.56286211,  0.62799976, -0.73756323, ...,  1.40257375,
        -1.38540782, -1.14774685],
       [ 0.34329583, -2.66672362, -0.84636433, ..., -0.71297499,
         0.72180912, -1.14774685]])
```

A) Importing algorithms and key matrics

The algorithms used for training and testing are LogisticRegression(), DecisionTreeClassifier(), KneighborsClassifier() & SpaceVectorClassifier().

The key metrics used are Accuracy score, Confusion matrix, roc_score and classification report.

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curve,roc_auc_score
from sklearn.model_selection import train_test_split, cross_val_score
```

```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2, random_state=42)
```

B) Model Selection

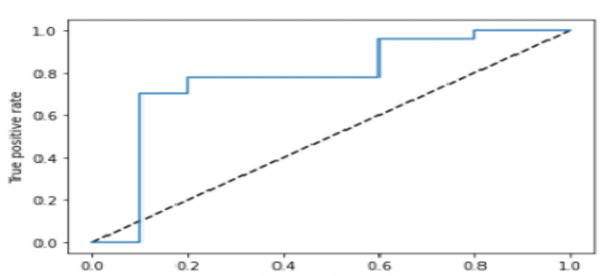
```
model=[LogisticRegression(),DecisionTreeClassifier(), KNeighborsClassifier()]
for i in model:
    print(i)
    i.fit(x_train,y_train)
    pred=i.predict(x_test)
    y_pred_pro=i.predict_proba(x_test)[:,1]
    fpr,tpr,thresholds=roc_curve(y_test,y_pred_pro)
    print('Accuracy score :', accuracy_score(y_test,pred))
    print('Confusion matrix :\n', confusion_matrix(y_test,pred))
    print('Classification report: \n ', classification_report(y_test,pred))
    print('**************************************************')
    print('\n')
    plt.plot([0,1],[0,1],'k--')
    plt.plot(fpr,tpr,label='KNeighborClassifier')
    plt.xlabel('False positive rate')
    plt.ylabel('True positive rate')
    plt.show()
```

```
LogisticRegression()
Accuracy score : 0.7837837837837838
Confusion matrix :
 [[ 4  6]
 [ 2 25]]
Classification report:
               precision    recall  f1-score   support

         0.0       0.67      0.40      0.50        10
         1.0       0.81      0.93      0.86        27

    accuracy                           0.78        37
   macro avg       0.74      0.66      0.68        37
weighted avg       0.77      0.78      0.76        37

**************************************************
```

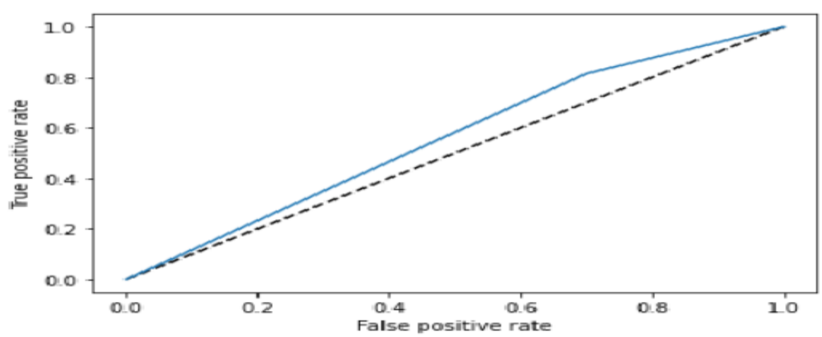


```
DecisionTreeClassifier()
Accuracy score : 0.6756756756756757
Confusion matrix :
 [[ 3  7]
 [ 5 22]]
Classification report:
              precision    recall  f1-score   support

         0.0       0.38      0.30      0.33        10
         1.0       0.76      0.81      0.79        27

    accuracy                           0.68        37
   macro avg       0.57      0.56      0.56        37
weighted avg       0.65      0.68      0.66        37

*************************************************
```

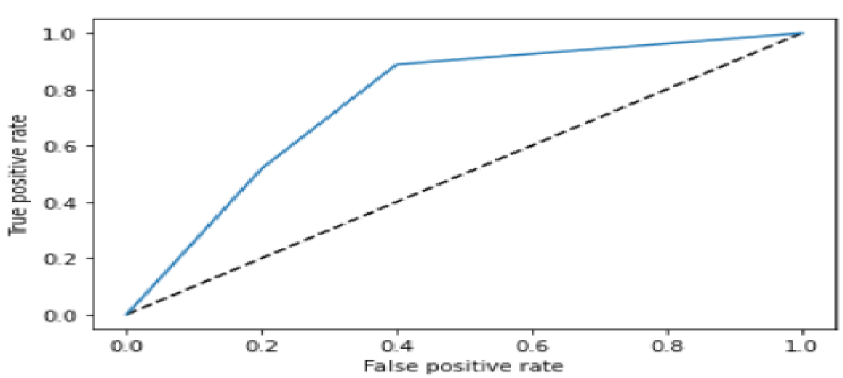



```
KNeighborsClassifier()
Accuracy score : 0.7567567567567568
Confusion matrix :
 [[ 2  8]
 [ 1 26]]
Classification report:
              precision    recall  f1-score   support

         0.0       0.67      0.20      0.31        10
         1.0       0.76      0.96      0.85        27

    accuracy                           0.76        37
   macro avg       0.72      0.58      0.58        37
weighted avg       0.74      0.76      0.71        37

*************************************************
```

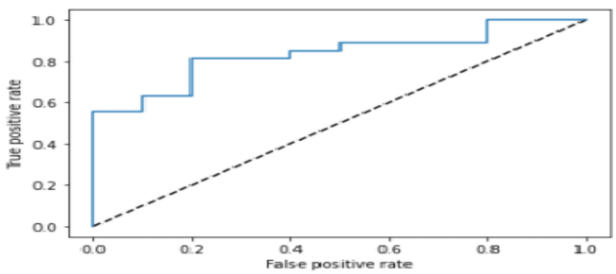


```
svc=SVC(probability=True)
svc.fit(x_train,y_train)
pred=svc.predict(x_test)
y_pred_pro=svc.predict_proba(x_test)[:,1]
fpr,tpr,thresholds=roc_curve(y_test,y_pred_pro)
print('Accuracy score :', accuracy_score(y_test,pred))
print('Confusion matrix :\n', confusion_matrix(y_test,pred))
print('Classification report: \n ', classification_report(y_test,pred))
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr,label='KNeighborClassifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.show()
```

```
Accuracy score : 0.7567567567567568
Confusion matrix :
 [[ 2  8]
 [ 1 26]]
Classification report:
               precision    recall  f1-score   support

         0.0       0.67      0.20      0.31        10
         1.0       0.76      0.96      0.85        27

    accuracy                           0.76        37
   macro avg       0.72      0.58      0.58        37
weighted avg       0.74      0.76      0.71        37
```



LogisticRegression() algorithm is showing highest accuracy score of 78.37% among all other alogorithms.

C) Ensemble Technique

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
pred2=rf.predict(x_test)
print('Accuracy score :', accuracy_score(y_test,pred))
print('Confusion matrix :\n', confusion_matrix(y_test,pred2))
print('Classification report: \n ', classification_report(y_test,pred2))
```

```
Accuracy score : 0.7567567567567568
Confusion matrix :
 [[ 2  8]
 [ 1 26]]
Classification report:
               precision    recall  f1-score   support

         0.0       0.67      0.20      0.31        10
         1.0       0.76      0.96      0.85        27

    accuracy                           0.76        37
   macro avg       0.72      0.58      0.58        37
weighted avg       0.74      0.76      0.71        37
```

D) Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
parameters = {'C':[1,10],'random_state': range(42,100)}
lg=LogisticRegression()
Grid=GridSearchCV(lg,parameters)
Grid.fit(x_train,y_train)
```

```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [1, 10], 'random_state': range(42, 100)})
```

```
Grid.best_params_
```

```
{'C': 1, 'random_state': 42}
```

Since it is the default parameters, we will that only.

```
lg=LogisticRegression()
lg.fit(x_train,y_train)
pred3=lg.predict(x_test)
```

Since LogisticRegression() gives the best accuracy, we apply hyperparameter Tuning on it. The best result is shown in default parameter. They are random_state: 42 and float: 1.

It gives an accuracy score of 78.38%.