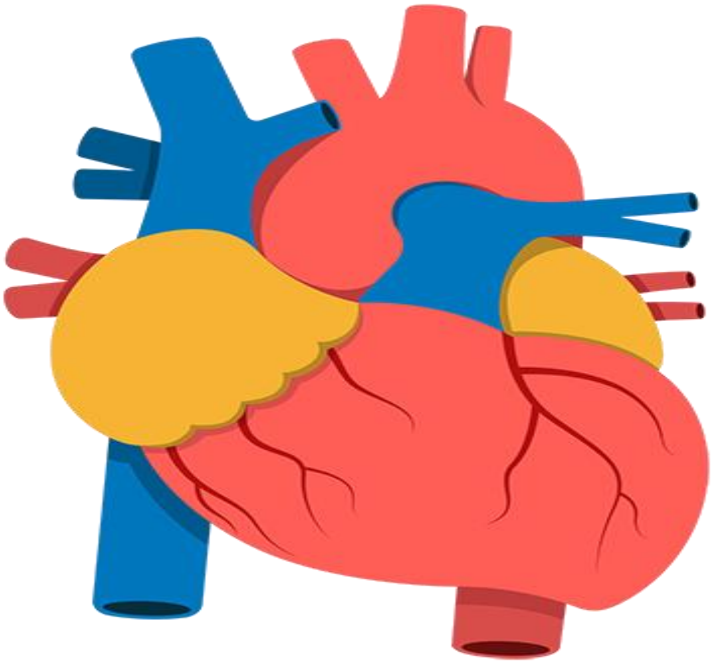
Predicting the Presence of Heart Disease in Patients



Heart disease is one of the biggest cause of Death among the population of the world.

Prediction of cardiovascular disease is considered as one of the most important subjects in the section of clinical data analysis.

In this blog-post, I will be applying Machine Learning approaches for ***classifying whether a person is suffering from Heart Disease or not*** by using Cleveland Heart Disease Dataset from the UCI Repository.

Dataset link:
https://archive.ics.uci.edu/ml/datasets/heart+disease

Importing necessary library

```
import pandas as pd
import numpy as np

#Data Visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

from sklearn.model_selection import train_test_split

#Importing warnings
import warnings
warnings.filterwarnings('ignore')
```

Getting the Data

```
df=pd.read_csv('heart_diseaseucimachine.csv')
df.head()
```

:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0

In this dataset 303 rows & 14 columns are present,which is describes below:

1.***Age***:displays the age of individual in years

2.***Sex***: displays the gender of individual

1 = male

0 = female

3.***cp***(*chest pain type*) :displays the type of chest pain experienced by individual;

1: typical angina

2: atypical angina

3: non-anginal pain

4: asymptomatic

4.***trestbps:***displays ***resting blood pressure*** of an individual (in mm Hg on admission to the hospital)

5.***chol***:displays ***serum cholestoral*** in mg/dl

6***.fbs***:compares the ***fasting blood sugar***

If fbs > 120 mg/dl,then

1 = true , 0 = false

7.***restecg***:displays ***resting electrocardiographic*** results

0: normal

1: having ST-T wave abnormality

2: showing probable or definite left ventricular hypertrophy

8.***thalach***:displays ***maximum heart rate*** achieved

9.***exang***:(exercise induced angina)

1 = yes , 0 = no

10.***oldpeak***-ST depression induced by exercise relative to rest

11.***slope***- the slope of the peak exercise ST segment

1: upsloping

2: flat

3: downsloping

12.***ca***-number of ***major vessels*** (0-3) colored by flourosopy

13.***thal:***displays the ***thalassemia***

3 = normal ,

6 = fixed defect ,

7 = reversable defect

14.***num***:displays whether the individual is suffering from heart disease or not(***target column)***

0=absence

1,2,3,4=present

Data Exploration/Analysis

```
df.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    float64
 1   sex       303 non-null    float64
 2   cp        303 non-null    float64
 3   trestbps  303 non-null    float64
 4   chol      303 non-null    float64
 5   fbs       303 non-null    float64
 6   restecg   303 non-null    float64
 7   thalach   303 non-null    float64
 8   exang     303 non-null    float64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    float64
 11  ca        303 non-null    object
 12  thal      303 non-null    object
 13  num       303 non-null    int64
dtypes: float64(11), int64(1), object(2)
memory usage: 33.3+ KB
```

Observation:

1.All columns are of float64 type except ca & thal

2.ca & thal are of object type

* First I will find the **missing data** present in dataset.

```
df.isnull().sum().any()
```

```
False
```

It shows that no null values are present in dataset,but we notice that some '?' are present in our datasert

```
# in the given dataset some '?' are present,replacing
'?' with nan value in ca & thal column

df.replace({'ca':{'?':np.nan}},regex=False,inplace=True)
df.replace({'thal':{'?':np.nan}},regex=False,inplace=True)
```

we **need to convert 2 categorical features into numeric** ones so that the machine learning algorithms can process them.

```
#changing object type to float type

col=['ca','thal']
for c in col:
    df[c]=df[c].astype(float)
```

Checking null values

```
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          4
thal        2
num         0
dtype: int64
```

we see that there are only 6 cells with null value,with 4 belonging to attribute ca & 2 to thal, that wee need to deal with.

As null values are very less,we can either drop them or impute them.I have imputed most frequent/mode in place of null vlaue

Imputing null values

```
from sklearn.impute import SimpleImputer
imp=SimpleImputer(strategy='most_frequent')
df['ca']=imp.fit_transform(df['ca'].values.reshape(-1,1))
df['thal']=imp.fit_transform(df['thal'].values.reshape(-1,1))

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

false

- We see that null value have been removed.

Summary Statistics

Now we will check information about all the numerical column

```
df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.990099	149.607261	0.326733	1.039604	1.600660	0.663366	4.722772	0.937294
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.994971	22.875003	0.469794	1.161075	0.616226	0.934375	1.938383	1.228536
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.000000	0.800000	2.000000	0.000000	3.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.000000	1.600000	2.000000	1.000000	7.000000	2.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	3.000000	3.000000	7.000000	4.000000

From above summary statistics ,we observe that:

1.Minimum age is 29 & maximum age is 77.

2.std. deviation is maximum in chol.

3.In chol columns the difference b\w 75% & maxm is more,so outliers may be present.

```
#checking unique values of target column
df['num'].value_counts()
```

```
0    164
1     55
2     36
3     35
4     13
Name: num, dtype: int64
```

★ Here df['num'] shows whether a person is suffering from heart disease or not;

0=absence(not suffering) ,

(1,2,3,4)=present(suffering)

```
#performing mapping in target column
df['num']=df.num.map({0:0,1:1,2:1,3:1,4:1})
df['num'].value_counts()
```

```
0    164
1    139
Name: num, dtype: int64
```

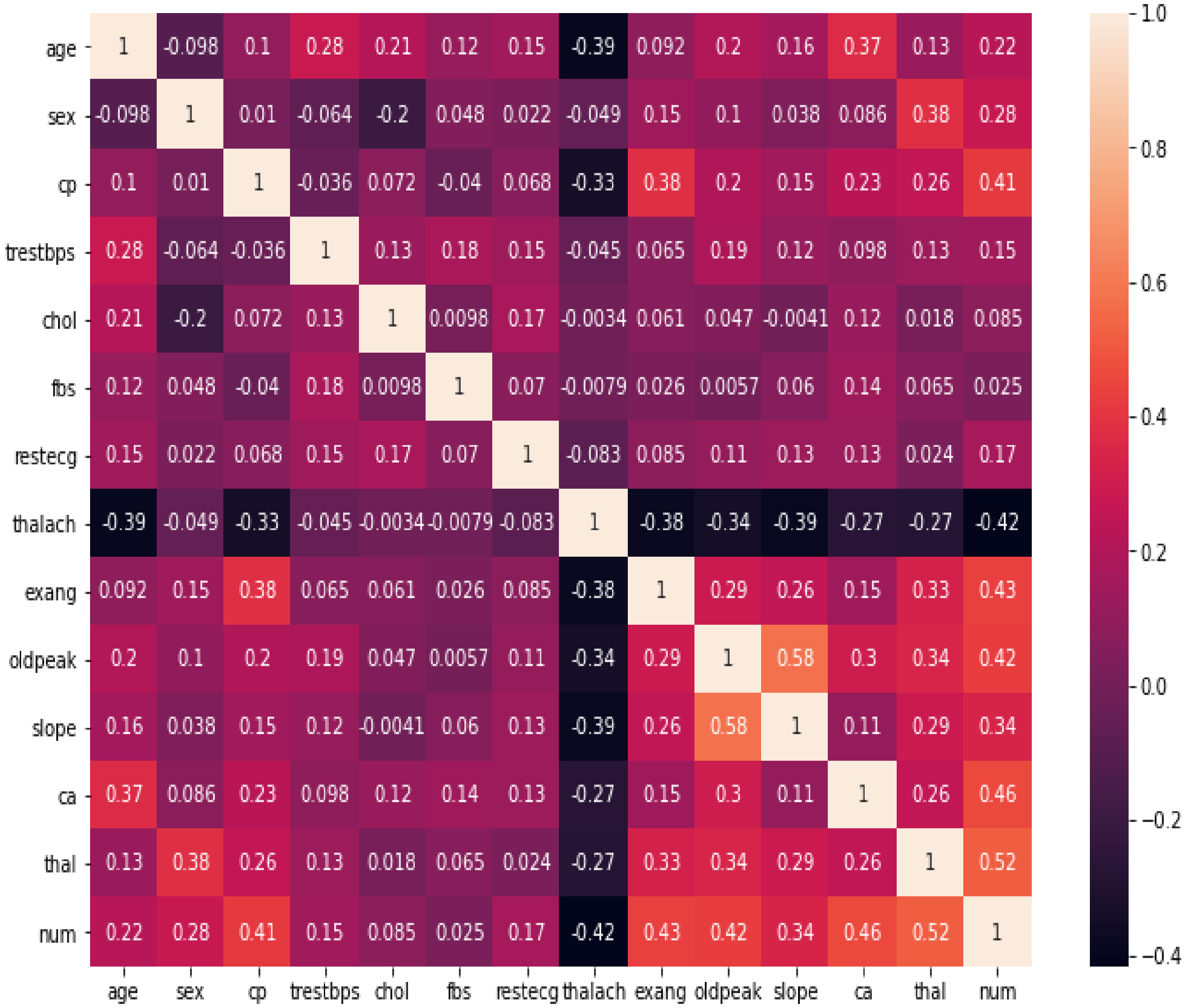
we nocice that out of total 139 people are suffering from heart disease

Now we will try to find out, ***What features could contribute to cardiovascular disease ?***

First we check correlation between the columns.

check corelation

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True)
```

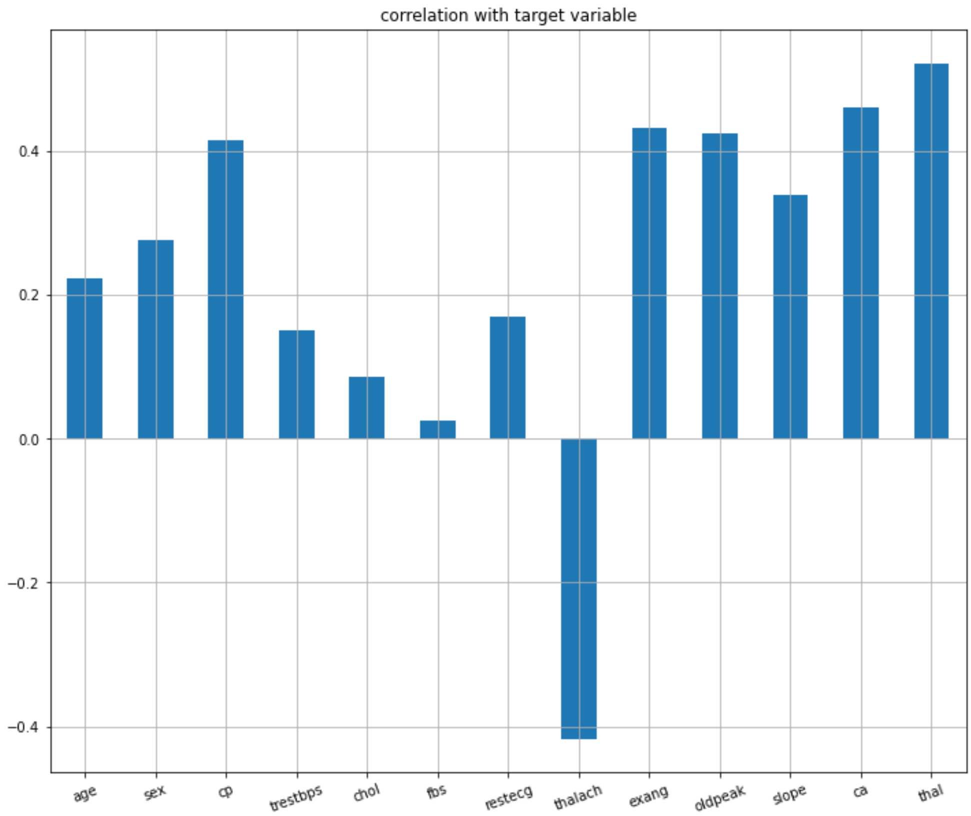


Observation:

1.num is negatively corelated with thalach & posively corelated wiyh thal

2. most columns are moderately correlated with num, but 'fbs' is weakly correlated.

```
#Checking correlation with the traget variable
.ie num
plt.figure(figsize=(8,6))
df.drop('num',axis=1).corrwith(df['num']).plot(
kind='bar',grid=True)
plt.xticks(rotation=20)
plt.title('correlation with target variable'
```



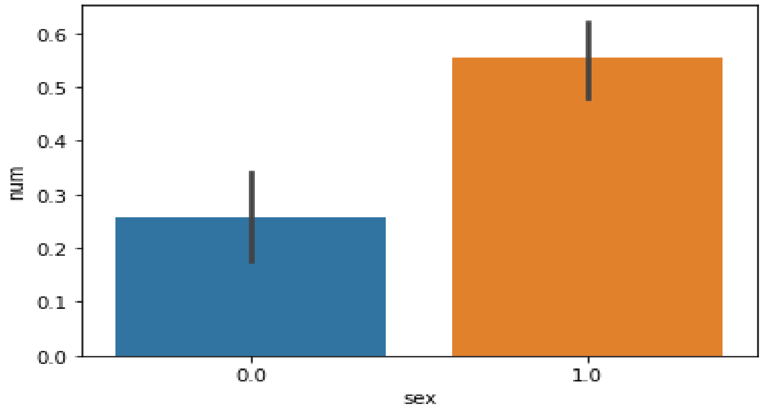
- We see that ***fbs*** is very ***weakly correlated*** with num.So it can be dropped

```
#drop fbs
df.drop('fbs',axis=1,inplace=True)
```

- Now we perform some **DATA VISUALIZATION**

1.sex vs num

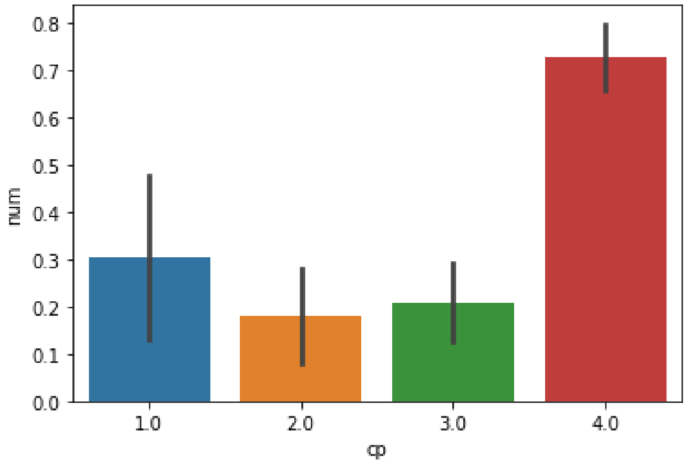
```
sns.barplot(x=df["sex"],y=df['num'])
```



we notice that males are more likely to have heart problems than female.

2.cp vs num

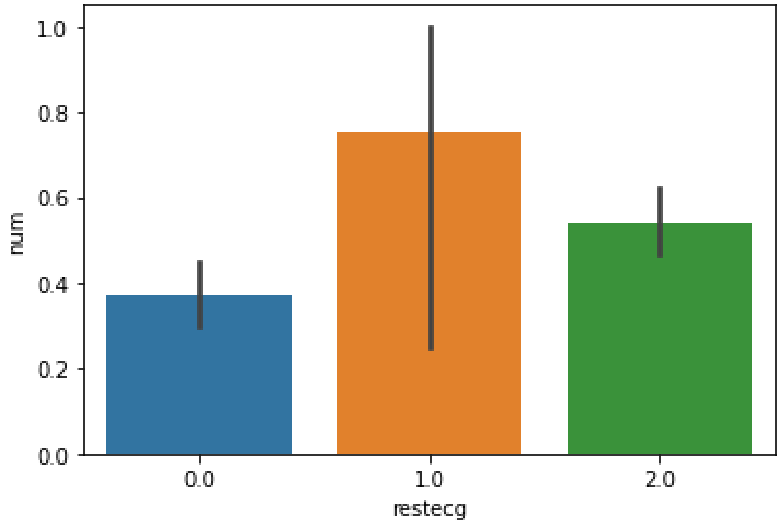
```
sns.barplot(df["cp"],y=df['num'])
```



From graph we notice, that chest pain of type '4.0', i.e. asymptomatic are much more likely to have heart problems

3.restecg vs num

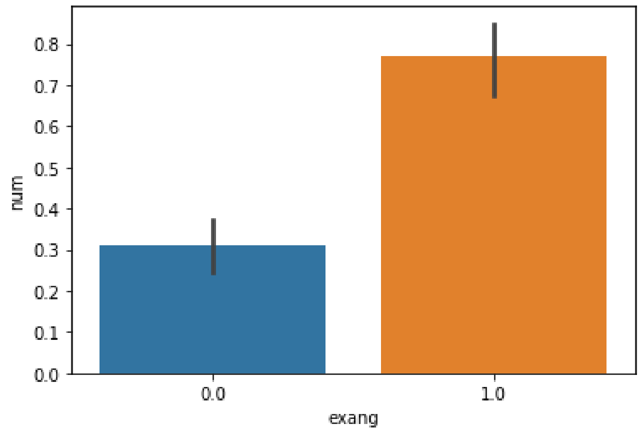
```
sns.barplot(df["restecg"],y=df['num'])
```



We notice that people with restecg '1'(having ST-T wave abnormality) are much more likely to have a heart disease

4.exang vs num

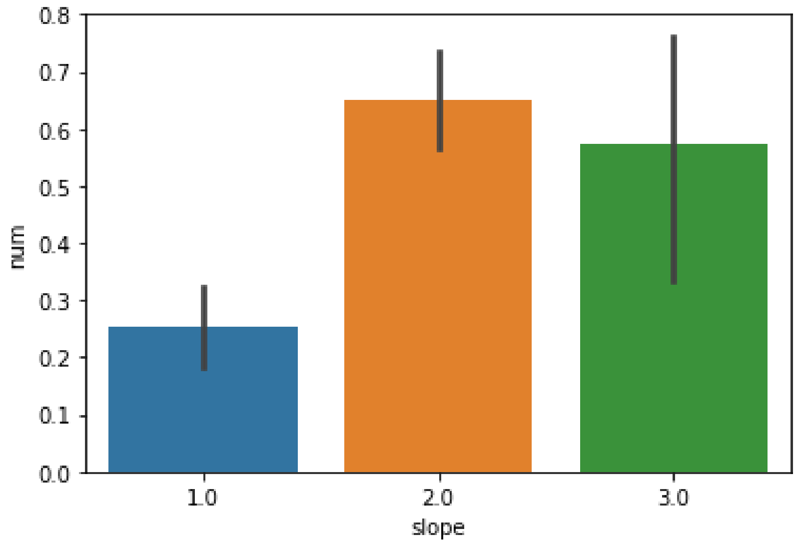
```
sns.barplot(df["exang"],y=df['num'])
```



People with exang=1 i.e. Exercise induced angina are more likely to have heart problems

5.slope vs num

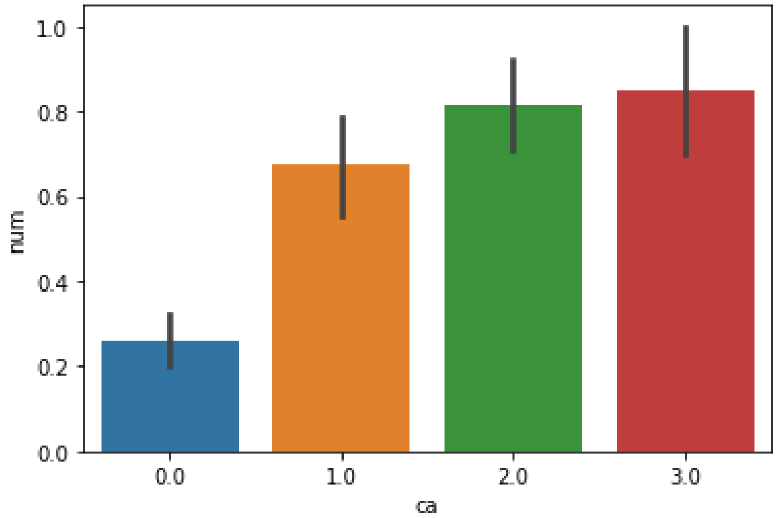
```
sns.barplot(df["slope"],y=df['num'])
```



from graph,it is clear that Slope >= 2 causes more heart problem

6.ca vs num

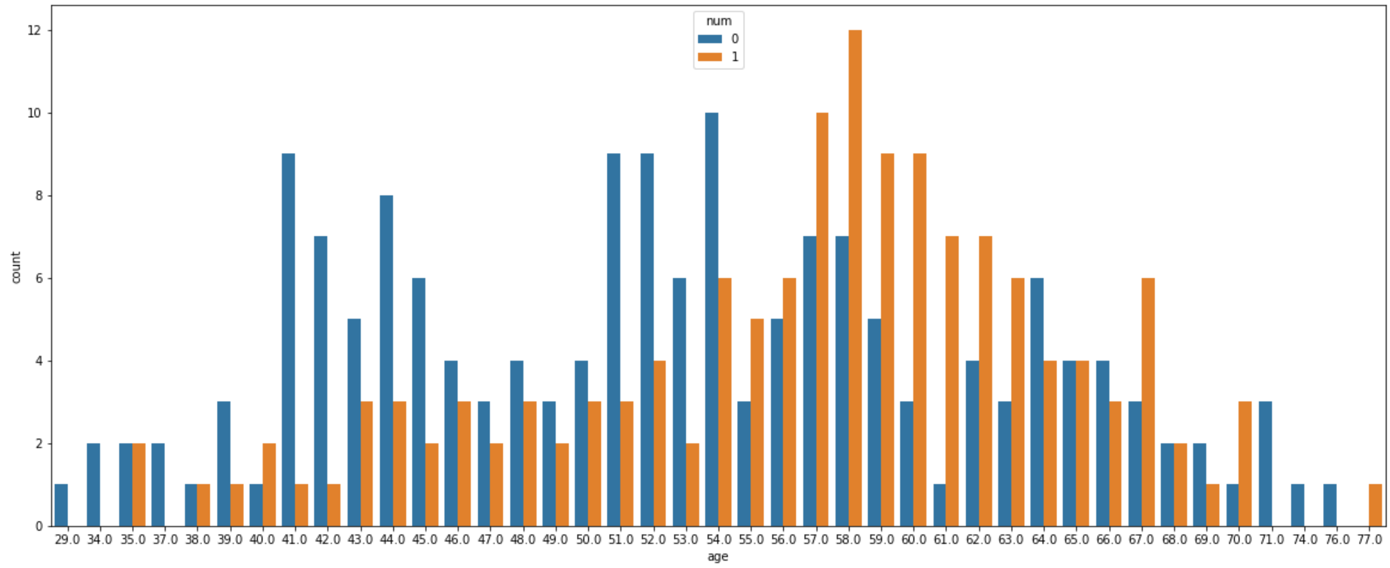
```
sns.barplot(df["ca"],y=df['num'])
```



ca=2.0 & 3.0 has large number of heart patients

7.age vs num

Here we look at the people's age who are suffering from the disease or not.

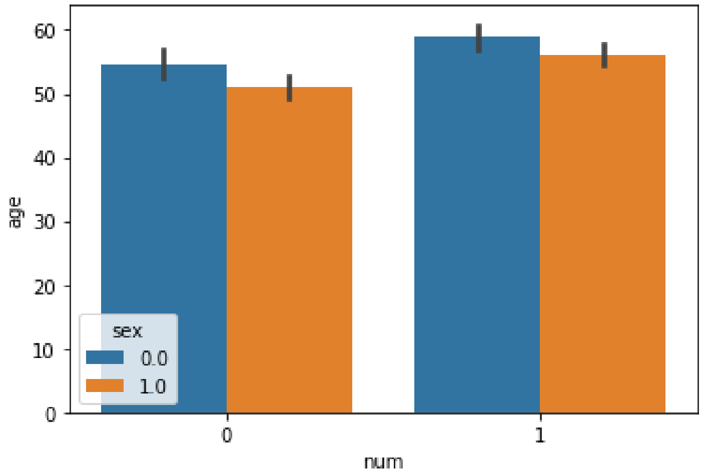


We see that most of the people who are suffering are of the age 58,followed by 57.

Majorly people belonging to the age above 50 are suffering from disease**.**

- **Next ,we look at distribution of age & sex for each target class**

```
sns.barplot(x='num',y='age',hue='sex',data=df)
```



We see that females who are suffering from the disease are older than males.

In this blog,I will be using the following models for classification:

1.Logistic Regression

2.Decison Tree Classifier

3.SVC

4.GaussianNB

5.KNeighborsClassifier

6.RandomForestClassifier

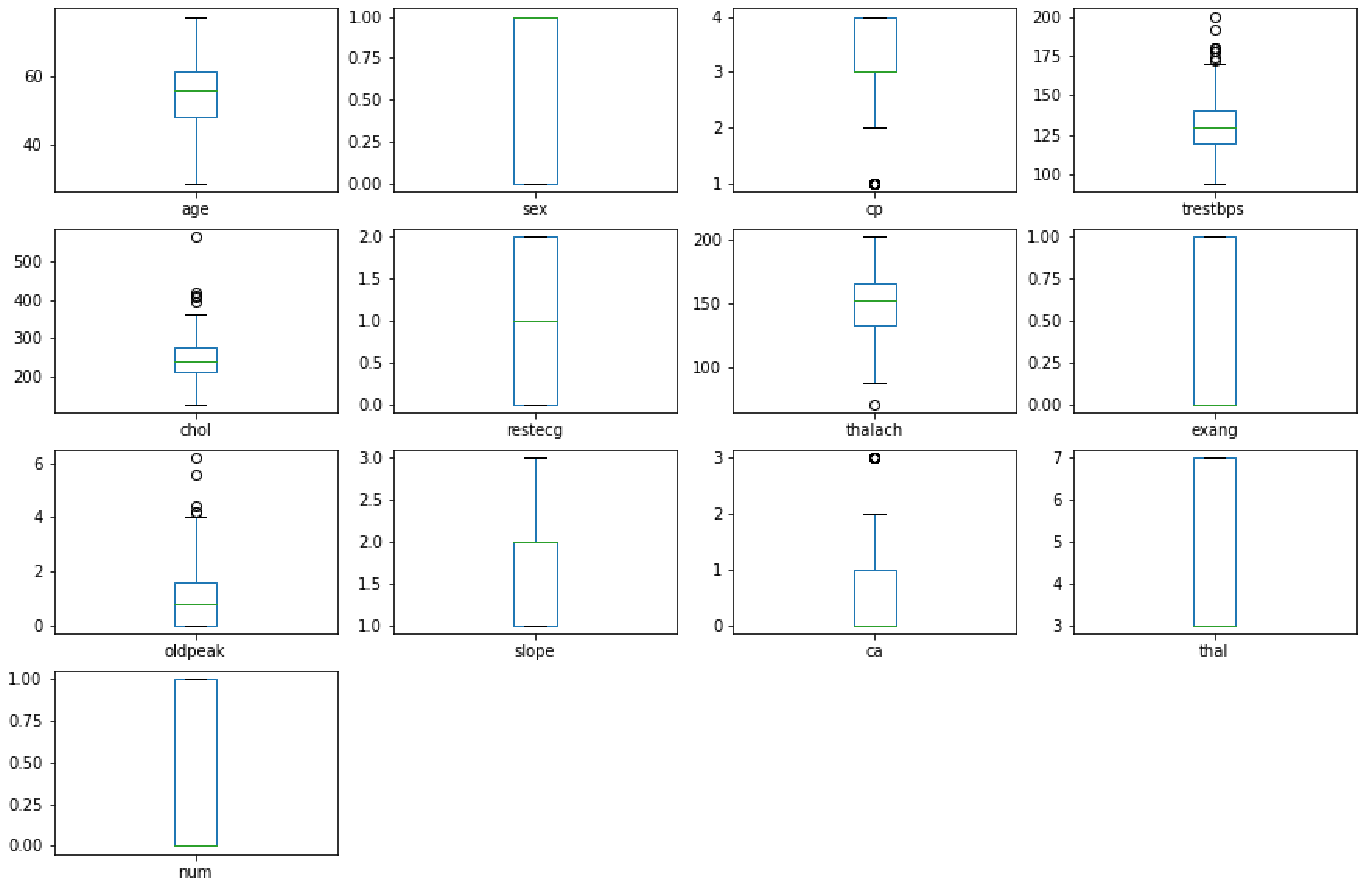
7.AdaboostClassifier

Checking skewness & outliers

Now by using boxplot,we check if outliers are present or not.

Outliers are nothing ,but the abnormal data present in the dataset,that deviates from other observation in dataset.

```
df.plot(kind='box',subplots=True,layout=(4,4),figsize=(15,10))
```



we see that outliers are **present**,which we need to remove.

Checking skewness

The skewness is a measure of how asymmetrical our data is distributed

If distribution is between -0.5 & 0.5,the distribution is approximately symmetric

```
df.skew()
```

```
age         -0.209060
sex         -0.774935
cp          -0.841754
trestbps     0.706035
chol         1.135503
restecg      0.019900
thalach     -0.537449
exang        0.742532
oldpeak      1.269720
slope        0.508316
ca           1.208791
thal         0.256375
num          0.166406
dtype: float64
```

We see that skewness is present in the data,which needs to be removed

```
#remove skewness
for col in df.columns:
    if df[col].skew()>0.55:
        df[col]=np.log1p(df[col])

for col in df.columns:
    if df[col].skew()<-0.55:
        df[col]=np.log1p(df[col])
```

- Again checking skewness

```
df.skew()
```

```
age         -0.209060
sex         -0.774935
cp          -1.266692
trestbps     0.281940
chol         0.081733
restecg      0.019900
thalach     -0.537449
exang        0.742532
oldpeak      0.396825
slope        0.508316
ca           0.770355
thal         0.256375
num          0.166406
```

Skewness has been removed.

Removing Outliers

```
from scipy.stats import zscore
z_score=abs(zscore(df))
print(df.shape)

df_new=df.loc[(z_score<3).all(axis=1)]
print(df_new.shape)
```

```
(303, 13)
(298, 13)
```

We see that outliers have been removed.Before, dataset consist of 303 rows & 13 columns .Dataset after removal of outliers contains 298 rows & 13 columns.

```
#spliting the data into input and output variable
x=df_new.iloc[:,:-1]
x.shape
```

```
(298, 12)
```

```


y=df_new.iloc[:,-1]
y.shape
```

```
(298,)
```

Scaling the input variable

We apply standard scaling to make sure that all features are on same scale so that each feature is equally importa nt & make it easier to process by most ML algorithm.

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

Train_Test_Split

Now let us divide the data into train & test set.
In this project ,I have divided the data into 80:20 ratio .ie training size is 80% & testing size is 20% of the whol e data.

```
from sklearn.model_selection import train
_test_split

x_train,x_test,y_train,y_test=train_test_
split(x,y,test_size=.20,random_state=42)
```

```
print('x_train_shape:',x_train.shape)
print('x_test_shape:',x_test.shape)
print('y_train_shape:',y_train.shape)
print('y_test_shape:',y_test.shape
```

```
x_train_shape: (238, 12)
x_test_shape: (60, 12)
y_train_shape: (238,)
y_test_shape: (60,)
```

```
#importing our models library

from sklearn.linear_model import LogisticRegres
sion
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassif
ier

#importing metrics
from sklearn.metrics import accuracy_score,clas
sification_report,confusion_matrix
```

logistic regression

```
max_accuracy_score=0
for r_state in range(30,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_
state=r_state,test_size=.20)
    lg=LogisticRegression()
    lg.fit(x_train,y_train)
    lg_pred=lg.predict(x_test)
    accuracy_scr=accuracy_score(y_test,lg_pred)
    if accuracy_scr>max_accuracy_score:
        max_accuracy_score=accuracy_scr
        final_r_state=r_state

print('max accuracy score corresponding to ',final_r_state,'is
',max_accuracy_score)
```

max accuracy score corresponding to 55 is 0.91666666666666666

```
x_train,x_test,y_train,y_test=train_test_split(x,y,te
st_size=.20,random_state=55)
```

- lg=LogisticRegression()

```
lg.fit(x_train,y_train)
predlg=lg.predict(x_test)
print('accuracy_score:',accuracy_score(y_test,predlg)
)
print(confusion_matrix(y_test,predlg))
print(classification_report(y_test,predlg))
```

```
accuracy_score: 0.91666666666666666
[[29  3]
 [ 2 26]]
              precision    recall  f1-score   support

           0       0.94      0.91      0.92        32
           1       0.90      0.93      0.91        28

    accuracy                           0.92        60
   macro avg       0.92      0.92      0.92        60
weighted avg       0.92      0.92      0.92        60
```

Decision Tree Classifier

```
from sklearn.model_selection import GridSearchCV

parameters={'criterion':['gini','entropy'],'random_state':rang
e(40,100)}
dtc=DecisionTreeClassifier()
clf=GridSearchCV(dtc,parameters)
clf.fit(x,y)

print(clf.best_params_)
```

```
{'criterion': 'gini', 'random_state': 41}
```

```
#dtc with best parameters
dtc=DecisionTreeClassifier(criterion='gini',random_st
ate=41)
dtc.fit(x_train,y_train)
preddtc=dtc.predict(x_test)
print('accuracy_score:',accuracy_score(y_test,preddtc
))
print('\n')
print(confusion_matrix(y_test,preddtc))
print('\n')
print(classification_report(y_test,preddtc))
```

```
accuracy_score: 0.7833333333333333


[[25  7]
 [ 6 22]]


              precision    recall  f1-score   support

           0       0.81      0.78      0.79        32
           1       0.76      0.79      0.77        28

    accuracy                           0.78        60
   macro avg       0.78      0.78      0.78        60
weighted avg       0.78      0.78      0.78        60
```

SVC

```
#gridsearchcv
parameters={'kernel':['linear','rbf'],'C':[1,10],'ran
dom_state':range(40,100)}
svc=SVC()
clf=GridSearchCV(svc,parameters)
clf.fit(x,y)


print(clf.best_params_)
```

```
{'C': 1, 'kernel': 'rbf', 'random_state': 40}
```

```
#svc with best parameter

svc=SVC(kernel='rbf',C=1,random_state=40)
svc.fit(x_train,y_train)
predsvc=svc.predict(x_test)
print('accuracy_score:',accuracy_score(y_test,predsvc
))
print('\n')
print(confusion_matrix(y_test,predsvc))
print('\n')
print(classification_report(y_test,predsvc))
```

```
accuracy_score: 0.9166666666666666


[[29  3]
 [ 2 26]]


              precision    recall  f1-score   support

           0       0.94      0.91      0.92        32
           1       0.90      0.93      0.91        28

    accuracy                           0.92        60
   macro avg       0.92      0.92      0.92        60
weighted avg       0.92      0.92      0.92        60
```

Naive Bayes

```
gnb=GaussianNB()
gnb.fit(x_train,y_train)
predgnb=gnb.predict(x_test)
print('accuracy_score:',accuracy_score(y_test,predgnb
))
print('\n')
print(confusion_matrix(y_test,predgnb))
print('\n')
print(classification_report(y_test,predgnb))
```

```
accuracy_score: 0.9


[[29  3]
 [ 3 25]]


              precision    recall  f1-score   support

           0       0.91      0.91      0.91        32
           1       0.89      0.89      0.89        28

    accuracy                           0.90        60
   macro avg       0.90      0.90      0.90        60
weighted avg       0.90      0.90      0.90        60
```

KNN

```
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
predknn=knn.predict(x_test)
print('accuracy_score:',accuracy_score(y_test,p
redknn))
print('\n')
print(confusion_matrix(y_test,predknn))
print('\n')
print(classification_report(y_test,predknn))
```

accuracy_score: 0.9166666666666666

```
[[29  3]
 [ 2 26]]
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.92        32
           1       0.90      0.93      0.91        28

    accuracy                           0.92        60
   macro avg       0.92      0.92      0.92        60
weighted avg       0.92      0.92      0.92        60
```

using ensemble technique to boostup our score

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=50,random_stat
e=30)

rf.fit(x_train,y_train)
predrf=rf.predict(x_test)
print(accuracy_score(y_test,predrf))
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf,labels=[0,1
]))
```

```
0.9166666666666666
[[29  3]
 [ 2 26]]
              precision    recall  f1-score   support

           0       0.94      0.91      0.92        32
           1       0.90      0.93      0.91        28

    accuracy                           0.92        60
```

```
   macro avg       0.92      0.92      0.92        60
weighted avg       0.92      0.92      0.92        60
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier

ad=AdaBoostClassifier(n_estimators=50,algorithm='SAMM
E.R')
ad.fit(x_train,y_train)
ad_pred=ad.predict(x_test)
print(accuracy_score(y_test,ad_pred))
print(confusion_matrix(y_test,ad_pred))
print(classification_report(y_test,ad_pred))
0.9166666666666666
[[30  2]
 [ 3 25]]
              precision    recall  f1-score   support

           0       0.91      0.94      0.92        32
           1       0.93      0.89      0.91        28

    accuracy                           0.92        60
   macro avg       0.92      0.92      0.92        60
weighted avg       0.92      0.92      0.92        60
```

Cross Validation

Cross validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross validation the 1st part

(20%) of the 5 parts will be kept out as a hold out set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset. In the similar way further iterations are made for the second 20% of the dataset is held as a hold out set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross validation process to get the remaining estimate of the model quality.

```
##cross validate the models
from sklearn.model_selection import cross_val_score
model=[LogisticRegression(),DecisionTreeClassifier(),
KNeighborsClassifier(),SVC(),GaussianNB(),AdaBoostCla
ssifier(),RandomForestClassifier()]
for m in model:
    score=cross_val_score(m,x,y,cv=5)
    print('score of ',m,'is:')
    print('score:',score)
    print('mean score:',score.mean())
    print('standard deviation:',score.std())
    print('*************************************
************')
    print('\n')
```

```
score of  LogisticRegression() is:
score: [0.85       0.86666667 0.76666667 0.81355932 0
.79661017]
mean score: 0.8187005649717515
standard deviation: 0.036062018992666554
```

****
*

score of **DecisionTreeClassifier()** is:
score: [0.71666667 0.9   0.75   0.79661017 0
.74576271]
mean score: 0.7818079096045197
standard deviation: 0.06440940623051476
****
*

score of **KNeighborsClassifier()** is:
score: [0.83333333 0.88333333 0.8   0.79661017 0
.76271186]
mean score: 0.8151977401129944
standard deviation: 0.04074946635169481
****
*

score of **SVC()** is:
score: [0.85   0.9   0.8   0.83050847 0
.76271186]
mean score: 0.8286440677966102
standard deviation: 0.046300667332783395
****
*

score of **GaussianNB()** is:
score: [0.83333333 0.86666667 0.8   0.83050847 0
.79661017]
mean score: 0.8254237288135593
standard deviation: 0.025557713487226102
****
*

score of **AdaBoostClassifier()** is:
score: [0.85   0.86666667 0.8   0.77966102 0
.76271186]
mean score: 0.8118079096045199

```
standard deviation: 0.040128269551185335
**************************************************
*

score of  RandomForestClassifier() is:
score: [0.86666667 0.9        0.81666667 0.79661017 0
.79661017]
mean score: 0.8353107344632769
standard deviation: 0.041257639976843966
**************************************************
*
```

AUC ROC CURVE

ROC curve is nothing but a graph displaying the perfor mance of classification model.
AUC ROC plot is used to visualise the performace of a binary classifier.
More the aea is under the curve ,better the model is working.

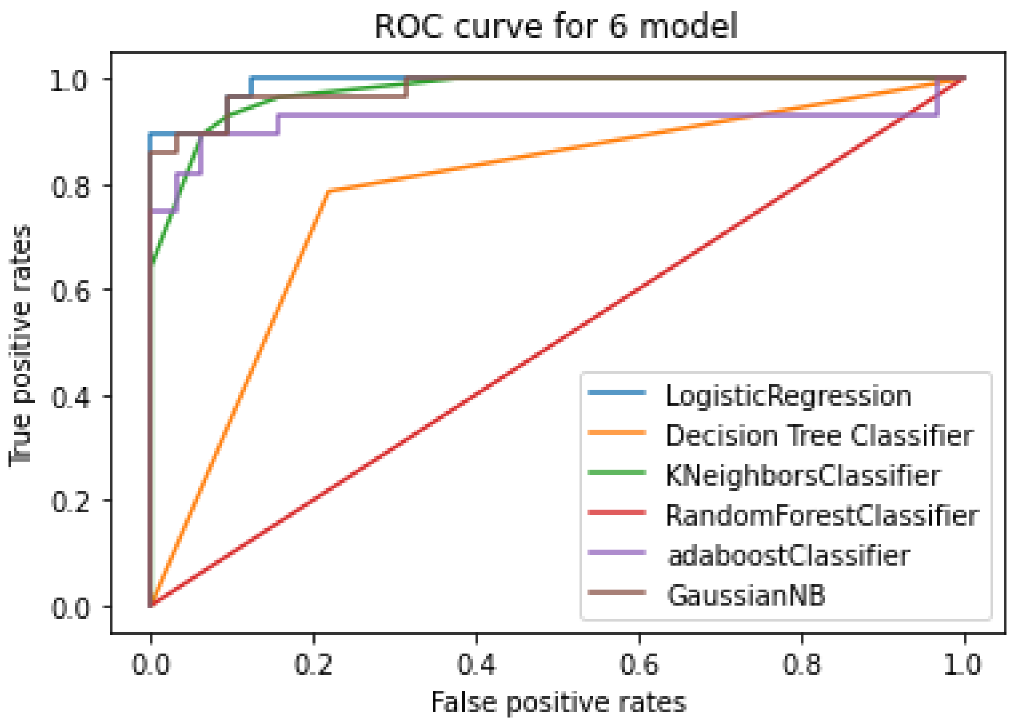
```
lgpred_prob=lg.predict_proba(x_test)[:,1]
dtcpred_prob=dtc.predict_proba(x_test)[:,1]
knnpred_prob=knn.predict_proba(x_test)[:,1]
rfpred_prob=rf.predict_proba(x_test)[:,1]
adpred_prob=ad.predict_proba(x_test)[:,1]
gnbpred_prob=gnb.predict_proba(x_test)[:,1]


from sklearn.metrics import roc_curve
lg_tpr,lg_fpr,lg_thresholds=roc_curve(y_test,lgpred_p
rob)
dtc_tpr,dtc_fpr,dtc_thresholds=roc_curve(y_test,dtcpr
ed_prob)
knn_tpr,knn_fpr,knn_thresholds=roc_curve(y_test,knnpr
ed_prob)
rf_tpr,rf_fpr,rf_threshold=roc_curve(y_test,rfpred_pr
ob)
ad_tpr,ad_fpr,ad_threshold=roc_curve(y_test,adpred_pr
ob)
gnb_tpr,gnb_fpr,gnb_threshold=roc_curve(y_test,gnbpre
d_prob)
```

```
plt.plot(lg_tpr,lg_fpr,label='LogisticRegression')
plt.plot(dtc_tpr,dtc_fpr,label ='Decision Tree Classi
fier')
plt.plot(knn_tpr,knn_fpr,label='KNeighborsClassifier'
)
plt.plot(rf_tpr,rf_tpr,label='RandomForestClassifier'
)
plt.plot(ad_tpr,ad_fpr,label='adaboostClassifier')
plt.plot(gnb_tpr,gnb_fpr,label='GaussianNB')


plt.xlabel('False positive rates')
plt.ylabel('True positive rates')
plt.title('ROC curve for 6 model')
plt.legend(loc='best')
plt.show()

from sklearn.metrics import roc_auc_score
print('LG AUC score',roc_auc_score(y_test,lgpre
d_prob))
print('DTC AUC SCORE',roc_auc_score(y_test,dtcp
red_prob))
print('KNN auc score',roc_auc_score(y_test,knnp
red_prob))
print('Random forest classifier',roc_auc_score(
y_test,rfpred_prob))
print('Adaboost classifier',roc_auc_score(y_tes
t,adpred_prob))
print('Gaussian NB',roc_auc_score(y_test,gnbpre
d_prob))
```




```
LG AUC score 0.9888392857142857
DTC AUC SCORE 0.7834821428571428
KNN auc score 0.9754464285714286
Random forest classifier 0.984375
Adaboost classifier 0.9185267857142858
Gaussian NB 0.9810267857142858
```

Conclusion;

The hightest aacuracy score is achieved by Logistic Regression.

Also we know that,higher the AUC score,better the model is working ..

LOGISTIC REGRESSION with accuracy of 91% & AUC score of 98% is best among all model. So we save it as the best model

```
import joblib

joblib.dump(lg,'heartdisease.pkl')
```