Name-Chinmaya Nayak(M.Sc Physics)

Admission_No=22ms0045

EDA(EXPLORATORY DATA ANALYSIS)

In [79]:
```
## Import all the library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [80]:
```
df=pd.read_csv("diabetes_dataset.csv")
df.head()
```

Out[80]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
2	183	64	0	0	23.3	0.672	32	1
3	89	66	23	94	28.1	0.167	21	0
4	137	40	35	168	43.1	2.288	33	1

In [81]:
```
##Find weather dataset have missing value or not
df.isnull().sum()
## so there  is no missing value atall
```

Out[81]:
```
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [82]:
```
df.describe()
```

Out[82]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [83]:
```
## Here in the above it shows the min value of Glucose ,BMI,BloodPressure,insulin,skinthickness is zero,i thick
## for resolving this we replace the zero value with the mean of the column
df["Glucose"]=df["Glucose"].replace(0,df["Glucose"].mean())
df["BloodPressure"]=df["BloodPressure"].replace(0,df["BloodPressure"].mean())
df["SkinThickness"]=df["SkinThickness"].replace(0,df["SkinThickness"].mean())
df["Insulin"]=df["Insulin"].replace(0,df["Insulin"].mean())
df["BMI"]=df["BMI"].replace(0,df["BMI"].mean())
```

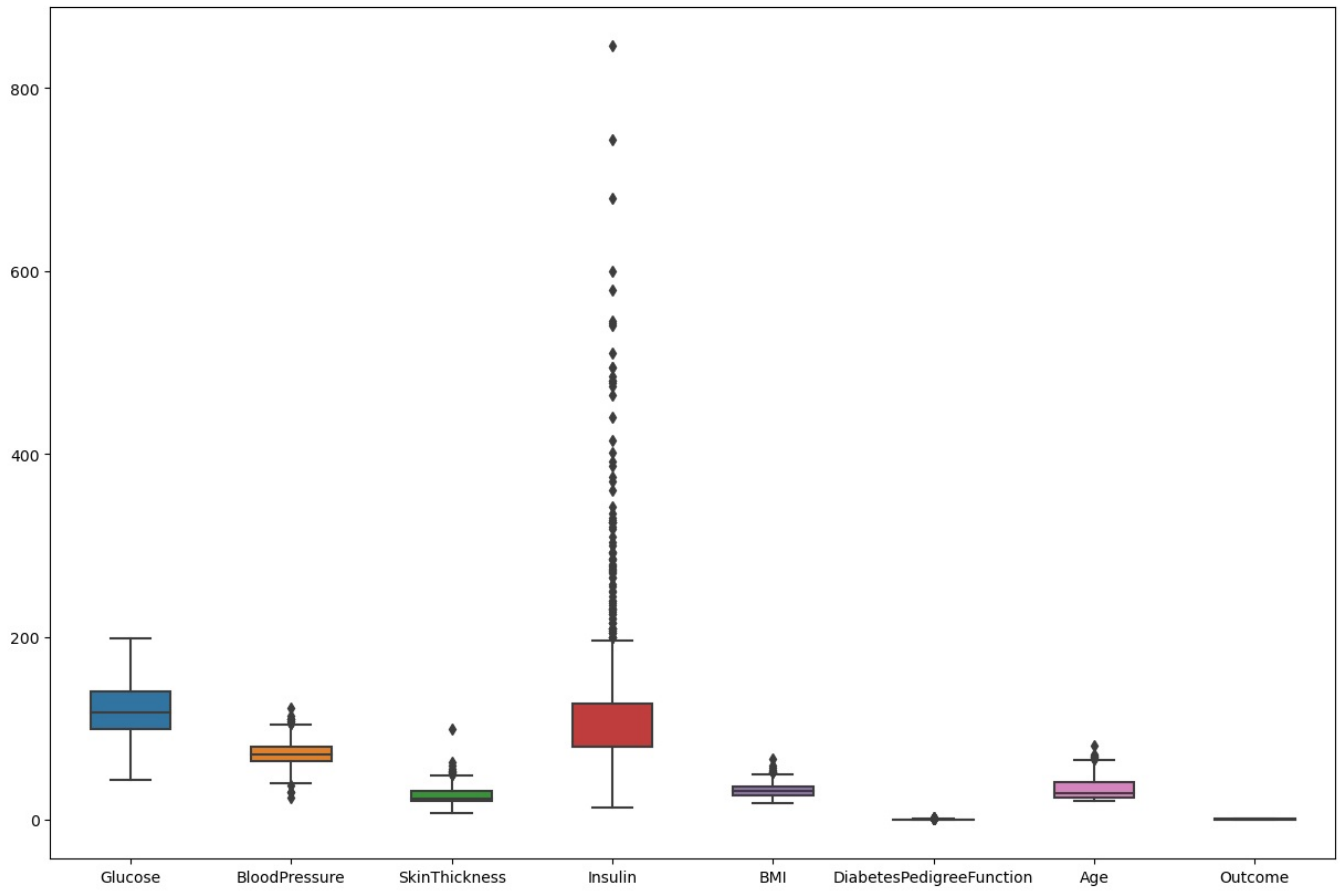
In [85]:
```
##Check after did some manipulation
#The minimum value of all the features(those minimum value contain 0) were replaced by the mean
df.describe()
```

Out[85]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	121.681605	72.254807	26.606479	118.660163	32.450805	0.471876	33.240885	0.348958
std	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329	11.760232	0.476951
min	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750	24.000000	0.000000
50%	117.000000	72.000000	23.000000	79.799479	32.000000	0.372500	29.000000	0.000000
75%	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

In [86]:
```
## Basically for visualize the outlier to that of corresponding feature( we use boxplot)
fig,ax=plt.subplots(figsize=(15,10))
sns.boxplot(data=df,width=0.5)
## More no of outliers are present on the insulin feature
```

Out[86]:
```
<AxesSubplot: >
```



In [87]:
```
df.head()
```

Out[87]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	148.0	72.0	35.000000	79.799479	33.6	0.627	50	1
1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	0
2	183.0	64.0	20.536458	79.799479	23.3	0.672	32	1
3	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1

In [88]:
```
df.columns
```

Out[88]:
```
Index(['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
       'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Split the dataset into dependent and independent variable

In [89]:
```
x=df[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI",
      "DiabetesPedigreeFunction", "Age"]]
y=df["Outcome"]
```

In [90]:
```
x
```

Out[90]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	148.0	72.0	35.000000	79.799479	33.6	0.627	50
1	85.0	66.0	29.000000	79.799479	26.6	0.351	31
2	183.0	64.0	20.536458	79.799479	23.3	0.672	32
3	89.0	66.0	23.000000	94.000000	28.1	0.167	21
4	137.0	40.0	35.000000	168.000000	43.1	2.288	33
...	...	...	...	...	...	...	...
763	101.0	76.0	48.000000	180.000000	32.9	0.171	63
764	122.0	70.0	27.000000	79.799479	36.8	0.340	27
765	121.0	72.0	23.000000	112.000000	26.2	0.245	30
766	126.0	60.0	20.536458	79.799479	30.1	0.349	47
767	93.0	70.0	31.000000	79.799479	30.4	0.315	23

768 rows × 7 columns

In [91]:
```
y
```

Out[91]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [92]:
```
df.shape,x.shape,y.shape
```

Out[92]:
```
((768, 8), (768, 7), (768,))
```

In [93]:
```
from sklearn.model_selection import  train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

In [94]:
```
x_train.shape,x_test.shape
```

Out[94]:
```
((537, 7), (231, 7))
```

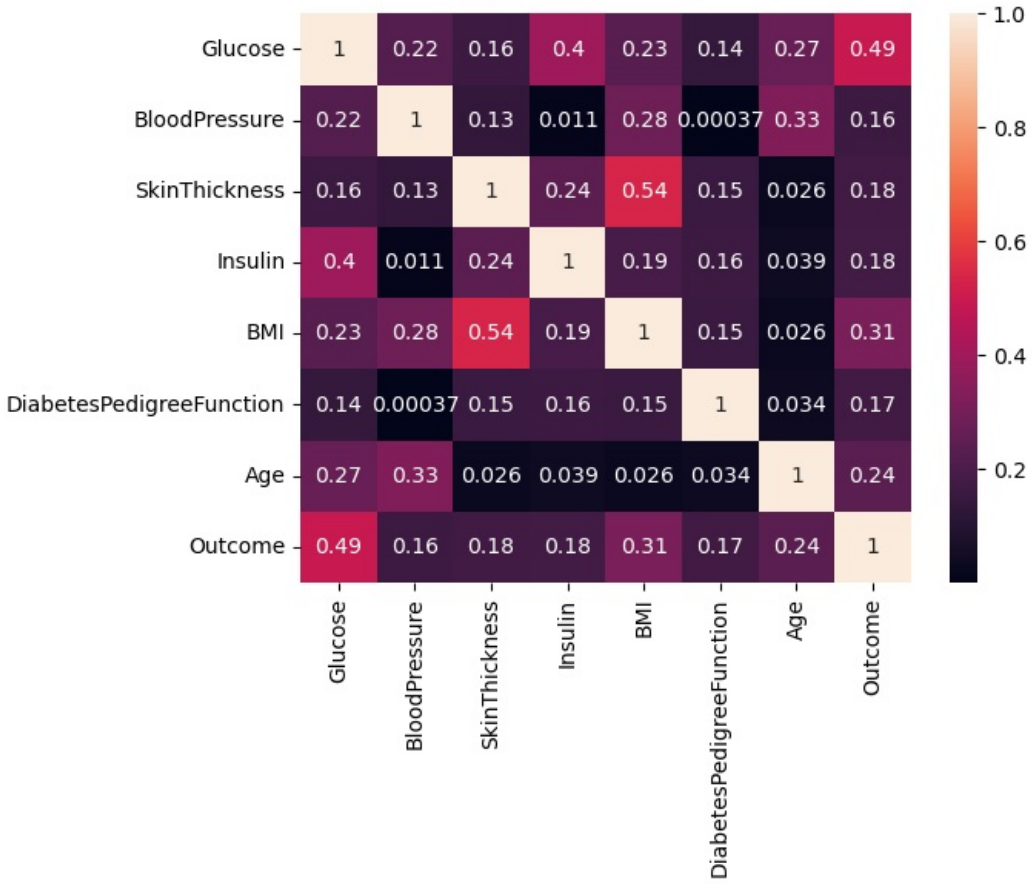
In [95]:
```
##Corealtion among the features
df.corr()
```

Out[95]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Glucose	1.000000	0.219666	0.160766	0.396597	0.231478	0.137106	0.266600	0.492908
BloodPressure	0.219666	1.000000	0.134155	0.010926	0.281231	0.000371	0.326740	0.162986
SkinThickness	0.160766	0.134155	1.000000	0.240361	0.535703	0.154961	0.026423	0.175026
Insulin	0.396597	0.010926	0.240361	1.000000	0.189856	0.157806	0.038652	0.179185
BMI	0.231478	0.281231	0.535703	0.189856	1.000000	0.153508	0.025748	0.312254
DiabetesPedigreeFunction	0.137106	0.000371	0.154961	0.157806	0.153508	1.000000	0.033561	0.173844
Age	0.266600	0.326740	0.026423	0.038652	0.025748	0.033561	1.000000	0.238356
Outcome	0.492908	0.162986	0.175026	0.179185	0.312254	0.173844	0.238356	1.000000

In [96]:
```
##Visualize the corelation by using the seaborn library
import seaborn as sns
sns.heatmap(df.corr(),annot=True)
```

Out[96]:
```
<AxesSubplot: >
```



In [97]:
```
## We want to scale down the feature that's why we Standardize all the featuress by using the Standardscalar
from sklearn.preprocessing import StandardScaler
scalar=StandardScaler()
x_train_scaled=scalar.fit_transform(x_train)
x_test_scaled=scalar.transform(x_test)
```

In [98]:
```
x_train_scaled
```

Out[98]:
```
array([[-0.89585789, -0.99370784, -0.90198999, ..., -1.20340503,
        -0.61421636, -0.94861028],
       [-0.56374882, -0.01067906,  0.29033814, ...,  0.66490433,
        -0.90973787, -0.43466673],
       [ 0.43257839, -0.33835532,  1.69945321, ...,  1.44097129,
        -0.30699103, -0.77729576],
       ...,
       [-0.69659245,  1.13618785,  1.15748588, ...,  1.91523444,
         1.94892066,  0.42190587],
       [ 0.63184384, -0.24779635, -0.62705448, ...,  1.4553429 ,
        -0.77514391, -0.34900947],
       [ 0.10046932,  1.9553785 , -0.62705448, ..., -1.40460758,
        -0.60836445, -1.03426754]])
```

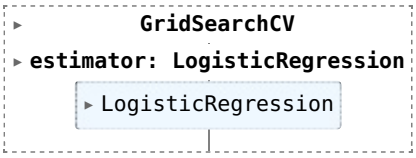
In [99]:
```
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

FIT THE MODEL THROUGH LOGISTIC REGRESSION

In [100…
```
## Doing hyperparameter tuning by using the Grid search CV
##Grid Search CV
from sklearn.model_selection import GridSearchCV
##parameter grid---different parameters such as penalty,c ,solver
parameters = {
    'penalty' : ['l1','l2'],
    'C'       : np.logspace(-3,3,7),
    'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],
```

```
}
```

In [101…
```
#Import Logistic regression from the sklearn module
from sklearn.linear_model import LogisticRegression
lonreg=LogisticRegression()
## we put all the parameters inside the Grid Search CV
clf=GridSearchCV(lonreg,
                 param_grid=parameters,
                 scoring="accuracy",
                 cv=10)
## here we fit the training data into our model
clf.fit(x_train_scaled,y_train)
```

Out[101]:


In [102…
```
## what is the best parameters
clf.best_params_
```

Out[102]:
```
{'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
```

In [103…
```
## let's see how our model deals with the test dataset
y_pred=clf.predict(x_test_scaled)
```

In [104…
```
## Import confusion matrix,accuarcy score for knowing what is the accuracy of the given model
from sklearn.metrics import accuracy_score, confusion_matrix
conf_mat=confusion_matrix(y_test,y_pred)
conf_mat
```

Out[104]:
```
array([[123,  28],
       [ 34,  46]])
```

In [105…
```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][1]
```

In [106…
```
Accuracy = (true_positive + true_negative) / (true_positive +false_positive + false_negative + true_negative)
Accuracy
```

Out[106]:
```
0.7316017316017316
```

In [107…
```
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[107]:
```
0.8145695364238411
```

In [108…
```
Recall = true_positive/(true_positive+false_negative)
Recall
```

Out[108]:
```
0.7834394904458599
```

In [109…
```
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

Out[109]:
```
0.7987012987012987
```

In [111…
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_pred,y_test))
print(accuracy_score(y_test,y_pred))
```

```
[[123  28]
 [ 34  46]]
              precision    recall  f1-score   support

           0       0.81      0.78      0.80       157
           1       0.57      0.62      0.60        74

    accuracy                           0.73       231
   macro avg       0.69      0.70      0.70       231
weighted avg       0.74      0.73      0.73       231

0.7316017316017316
```

FIT THE MODEL THROUGH DECISSION TREE

In [32]:
```
###This is basically a classifiacation Problem so here use Decission Tree Classifier
```

In [112…
```
x_train_scaled.shape,x_test_scaled.shape
```

Out[112]:
```
((537, 7), (231, 7))
```

In [113…
```
from sklearn.tree import DecisionTreeClassifier
```

In [114…
```
##remove all the warning statement
import warnings
warnings.filterwarnings("ignore")
```

In [115…
```
parameter={
 'criterion':['gini','entropy','log_loss'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']

}
```

In [116…
```
### Decission Tree model traing with Hyperparameter tuning
from sklearn.model_selection import GridSearchCV
classifier=DecisionTreeClassifier()
clf=GridSearchCV(classifier,param_grid=parameter,cv=3,scoring="accuracy",verbose=3)
clf.fit(x_train_scaled,y_train)
```

```
Fitting 3 folds for each of 90 candidates, totalling 270 fits
[CV 1/3] END criterion=gini, max_depth=1, max_features=auto, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=1, max_features=auto, splitter=best;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=1, max_features=auto, splitter=best;, score=0.603 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=1, max_features=auto, splitter=random;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=1, max_features=auto, splitter=random;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=1, max_features=auto, splitter=random;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=1, max_features=sqrt, splitter=best;, score=0.765 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=1, max_features=sqrt, splitter=best;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=1, max_features=sqrt, splitter=best;, score=0.670 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=1, max_features=sqrt, splitter=random;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=1, max_features=sqrt, splitter=random;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=1, max_features=sqrt, splitter=random;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=1, max_features=log2, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=1, max_features=log2, splitter=best;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=1, max_features=log2, splitter=best;, score=0.687 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=1, max_features=log2, splitter=random;, score=0.637 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=1, max_features=log2, splitter=random;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=1, max_features=log2, splitter=random;, score=0.603 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=2, max_features=auto, splitter=best;, score=0.765 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=2, max_features=auto, splitter=best;, score=0.737 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=2, max_features=auto, splitter=best;, score=0.698 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=2, max_features=auto, splitter=random;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=2, max_features=auto, splitter=random;, score=0.626 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=2, max_features=auto, splitter=random;, score=0.682 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=2, max_features=sqrt, splitter=best;, score=0.765 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=2, max_features=sqrt, splitter=best;, score=0.777 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=2, max_features=sqrt, splitter=best;, score=0.732 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=2, max_features=sqrt, splitter=random;, score=0.659 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=2, max_features=sqrt, splitter=random;, score=0.637 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=2, max_features=sqrt, splitter=random;, score=0.754 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=2, max_features=log2, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=2, max_features=log2, splitter=best;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=2, max_features=log2, splitter=best;, score=0.749 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=2, max_features=log2, splitter=random;, score=0.687 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=2, max_features=log2, splitter=random;, score=0.693 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=2, max_features=log2, splitter=random;, score=0.732 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=3, max_features=auto, splitter=best;, score=0.642 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=3, max_features=auto, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=3, max_features=auto, splitter=best;, score=0.659 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=3, max_features=auto, splitter=random;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=3, max_features=auto, splitter=random;, score=0.659 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=3, max_features=auto, splitter=random;, score=0.704 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=3, max_features=sqrt, splitter=best;, score=0.732 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=3, max_features=sqrt, splitter=best;, score=0.743 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=3, max_features=sqrt, splitter=best;, score=0.693 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=3, max_features=sqrt, splitter=random;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=3, max_features=sqrt, splitter=random;, score=0.682 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=3, max_features=sqrt, splitter=random;, score=0.659 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=3, max_features=log2, splitter=best;, score=0.754 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=3, max_features=log2, splitter=best;, score=0.687 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=3, max_features=log2, splitter=best;, score=0.665 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=3, max_features=log2, splitter=random;, score=0.760 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=3, max_features=log2, splitter=random;, score=0.743 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=3, max_features=log2, splitter=random;, score=0.698 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=4, max_features=auto, splitter=best;, score=0.698 total time=   0.0s
```

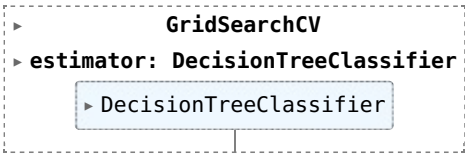
```
[CV 2/3] END criterion=gini, max_depth=4, max_features=auto, splitter=best;, score=0.698 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=4, max_features=auto, splitter=best;, score=0.765 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=4, max_features=auto, splitter=random;, score=0.709 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=4, max_features=auto, splitter=random;, score=0.698 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=4, max_features=auto, splitter=random;, score=0.682 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=4, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=4, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=4, max_features=sqrt, splitter=best;, score=0.709 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=4, max_features=sqrt, splitter=random;, score=0.782 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=4, max_features=sqrt, splitter=random;, score=0.687 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=4, max_features=sqrt, splitter=random;, score=0.615 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=4, max_features=log2, splitter=best;, score=0.732 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=4, max_features=log2, splitter=best;, score=0.743 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=4, max_features=log2, splitter=best;, score=0.698 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=4, max_features=log2, splitter=random;, score=0.760 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=4, max_features=log2, splitter=random;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=4, max_features=log2, splitter=random;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=5, max_features=auto, splitter=best;, score=0.721 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=5, max_features=auto, splitter=best;, score=0.743 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=5, max_features=auto, splitter=best;, score=0.709 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=5, max_features=auto, splitter=random;, score=0.771 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=5, max_features=auto, splitter=random;, score=0.654 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=5, max_features=auto, splitter=random;, score=0.659 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=5, max_features=sqrt, splitter=best;, score=0.687 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=5, max_features=sqrt, splitter=best;, score=0.721 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=5, max_features=sqrt, splitter=best;, score=0.709 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=5, max_features=sqrt, splitter=random;, score=0.620 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=5, max_features=sqrt, splitter=random;, score=0.654 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=5, max_features=sqrt, splitter=random;, score=0.670 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=5, max_features=log2, splitter=best;, score=0.665 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=5, max_features=log2, splitter=best;, score=0.709 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=5, max_features=log2, splitter=best;, score=0.693 total time=   0.0s
[CV 1/3] END criterion=gini, max_depth=5, max_features=log2, splitter=random;, score=0.698 total time=   0.0s
[CV 2/3] END criterion=gini, max_depth=5, max_features=log2, splitter=random;, score=0.709 total time=   0.0s
[CV 3/3] END criterion=gini, max_depth=5, max_features=log2, splitter=random;, score=0.715 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=1, max_features=auto, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=1, max_features=auto, splitter=best;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=1, max_features=auto, splitter=best;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=1, max_features=auto, splitter=random;, score=0.654 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=1, max_features=auto, splitter=random;, score=0.648 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=1, max_features=auto, splitter=random;, score=0.687 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=1, max_features=sqrt, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=1, max_features=sqrt, splitter=best;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=1, max_features=sqrt, splitter=best;, score=0.687 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=1, max_features=sqrt, splitter=random;, score=0.648 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=1, max_features=sqrt, splitter=random;, score=0.648 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=1, max_features=sqrt, splitter=random;, score=0.648 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=1, max_features=log2, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=1, max_features=log2, splitter=best;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=1, max_features=log2, splitter=best;, score=0.670 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=1, max_features=log2, splitter=random;, score=0.721 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=1, max_features=log2, splitter=random;, score=0.654 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=1, max_features=log2, splitter=random;, score=0.648 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=2, max_features=auto, splitter=best;, score=0.670 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=2, max_features=auto, splitter=best;, score=0.631 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=2, max_features=auto, splitter=best;, score=0.670 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=2, max_features=auto, splitter=random;, score=0.665 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=2, max_features=auto, splitter=random;, score=0.659 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=2, max_features=auto, splitter=random;, score=0.659 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=2, max_features=sqrt, splitter=best;, score=0.637 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=2, max_features=sqrt, splitter=best;, score=0.693 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=2, max_features=sqrt, splitter=best;, score=0.732 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=2, max_features=sqrt, splitter=random;, score=0.687 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=2, max_features=sqrt, splitter=random;, score=0.615 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=2, max_features=sqrt, splitter=random;, score=0.732 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=2, max_features=log2, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=2, max_features=log2, splitter=best;, score=0.626 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=2, max_features=log2, splitter=best;, score=0.670 total time=   0.0s
```

```
[CV 1/3] END criterion=entropy, max_depth=2, max_features=log2, splitter=random;, score=0.749 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=2, max_features=log2, splitter=random;, score=0.648 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=2, max_features=log2, splitter=random;, score=0.648 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=3, max_features=auto, splitter=best;, score=0.715 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=3, max_features=auto, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=3, max_features=auto, splitter=best;, score=0.721 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=3, max_features=auto, splitter=random;, score=0.676 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=3, max_features=auto, splitter=random;, score=0.693 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=3, max_features=auto, splitter=random;, score=0.732 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=3, max_features=sqrt, splitter=best;, score=0.754 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=3, max_features=sqrt, splitter=best;, score=0.698 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=3, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=3, max_features=sqrt, splitter=random;, score=0.765 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=3, max_features=sqrt, splitter=random;, score=0.670 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=3, max_features=sqrt, splitter=random;, score=0.665 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=3, max_features=log2, splitter=best;, score=0.754 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=3, max_features=log2, splitter=best;, score=0.709 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=3, max_features=log2, splitter=best;, score=0.749 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=3, max_features=log2, splitter=random;, score=0.682 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=3, max_features=log2, splitter=random;, score=0.754 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=3, max_features=log2, splitter=random;, score=0.721 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=4, max_features=auto, splitter=best;, score=0.760 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=4, max_features=auto, splitter=best;, score=0.642 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=4, max_features=auto, splitter=best;, score=0.743 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=4, max_features=auto, splitter=random;, score=0.782 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=4, max_features=auto, splitter=random;, score=0.709 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=4, max_features=auto, splitter=random;, score=0.676 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=4, max_features=sqrt, splitter=best;, score=0.754 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=4, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=4, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=4, max_features=sqrt, splitter=random;, score=0.654 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=4, max_features=sqrt, splitter=random;, score=0.648 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=4, max_features=sqrt, splitter=random;, score=0.693 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=4, max_features=log2, splitter=best;, score=0.687 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=4, max_features=log2, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=4, max_features=log2, splitter=best;, score=0.749 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=4, max_features=log2, splitter=random;, score=0.726 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=4, max_features=log2, splitter=random;, score=0.726 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=4, max_features=log2, splitter=random;, score=0.743 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=5, max_features=auto, splitter=best;, score=0.754 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=5, max_features=auto, splitter=best;, score=0.749 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=5, max_features=auto, splitter=best;, score=0.765 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=5, max_features=auto, splitter=random;, score=0.732 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=5, max_features=auto, splitter=random;, score=0.704 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=5, max_features=auto, splitter=random;, score=0.726 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=5, max_features=sqrt, splitter=best;, score=0.704 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=5, max_features=sqrt, splitter=best;, score=0.749 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=5, max_features=sqrt, splitter=best;, score=0.771 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=5, max_features=sqrt, splitter=random;, score=0.732 total time=   0.0
s
[CV 2/3] END criterion=entropy, max_depth=5, max_features=sqrt, splitter=random;, score=0.631 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=5, max_features=sqrt, splitter=random;, score=0.715 total time=   0.0
s
[CV 1/3] END criterion=entropy, max_depth=5, max_features=log2, splitter=best;, score=0.777 total time=   0.0s
[CV 2/3] END criterion=entropy, max_depth=5, max_features=log2, splitter=best;, score=0.715 total time=   0.0s
[CV 3/3] END criterion=entropy, max_depth=5, max_features=log2, splitter=best;, score=0.743 total time=   0.0s
[CV 1/3] END criterion=entropy, max_depth=5, max_features=log2, splitter=random;, score=0.793 total time=   0.0
s
```

```
[CV 2/3] END criterion=entropy, max_depth=5, max_features=log2, splitter=random;, score=0.743 total time=   0.0
s
[CV 3/3] END criterion=entropy, max_depth=5, max_features=log2, splitter=random;, score=0.749 total time=   0.0
s
[CV 1/3] END criterion=log_loss, max_depth=1, max_features=auto, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=1, max_features=auto, splitter=best;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=1, max_features=auto, splitter=best;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=1, max_features=auto, splitter=random;, score=0.654 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=1, max_features=auto, splitter=random;, score=0.648 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=1, max_features=auto, splitter=random;, score=0.670 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=1, max_features=sqrt, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=1, max_features=sqrt, splitter=best;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=1, max_features=sqrt, splitter=best;, score=0.670 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=1, max_features=sqrt, splitter=random;, score=0.698 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=1, max_features=sqrt, splitter=random;, score=0.637 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=1, max_features=sqrt, splitter=random;, score=0.648 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=1, max_features=log2, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=1, max_features=log2, splitter=best;, score=0.648 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=1, max_features=log2, splitter=best;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=1, max_features=log2, splitter=random;, score=0.749 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=1, max_features=log2, splitter=random;, score=0.648 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=1, max_features=log2, splitter=random;, score=0.648 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=2, max_features=auto, splitter=best;, score=0.670 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=2, max_features=auto, splitter=best;, score=0.670 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=2, max_features=auto, splitter=best;, score=0.648 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=2, max_features=auto, splitter=random;, score=0.637 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=2, max_features=auto, splitter=random;, score=0.648 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=2, max_features=auto, splitter=random;, score=0.749 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=2, max_features=sqrt, splitter=best;, score=0.765 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=2, max_features=sqrt, splitter=best;, score=0.771 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=2, max_features=sqrt, splitter=best;, score=0.732 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=2, max_features=sqrt, splitter=random;, score=0.654 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=2, max_features=sqrt, splitter=random;, score=0.615 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=2, max_features=sqrt, splitter=random;, score=0.648 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=2, max_features=log2, splitter=best;, score=0.654 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=2, max_features=log2, splitter=best;, score=0.687 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=2, max_features=log2, splitter=best;, score=0.732 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=2, max_features=log2, splitter=random;, score=0.665 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=2, max_features=log2, splitter=random;, score=0.648 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=2, max_features=log2, splitter=random;, score=0.743 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=3, max_features=auto, splitter=best;, score=0.682 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=3, max_features=auto, splitter=best;, score=0.749 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=3, max_features=auto, splitter=best;, score=0.654 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=3, max_features=auto, splitter=random;, score=0.665 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=3, max_features=auto, splitter=random;, score=0.670 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=3, max_features=auto, splitter=random;, score=0.670 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=3, max_features=sqrt, splitter=best;, score=0.682 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=3, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=3, max_features=sqrt, splitter=best;, score=0.721 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=3, max_features=sqrt, splitter=random;, score=0.760 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=3, max_features=sqrt, splitter=random;, score=0.693 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=3, max_features=sqrt, splitter=random;, score=0.709 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=3, max_features=log2, splitter=best;, score=0.765 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=3, max_features=log2, splitter=best;, score=0.687 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=3, max_features=log2, splitter=best;, score=0.620 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=3, max_features=log2, splitter=random;, score=0.687 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=3, max_features=log2, splitter=random;, score=0.665 total time=   0.
0s
```

```
[CV 3/3] END criterion=log_loss, max_depth=3, max_features=log2, splitter=random;, score=0.682 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=4, max_features=auto, splitter=best;, score=0.737 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=4, max_features=auto, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=4, max_features=auto, splitter=best;, score=0.726 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=4, max_features=auto, splitter=random;, score=0.771 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=4, max_features=auto, splitter=random;, score=0.631 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=4, max_features=auto, splitter=random;, score=0.676 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=4, max_features=sqrt, splitter=best;, score=0.726 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=4, max_features=sqrt, splitter=best;, score=0.721 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=4, max_features=sqrt, splitter=best;, score=0.743 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=4, max_features=sqrt, splitter=random;, score=0.754 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=4, max_features=sqrt, splitter=random;, score=0.754 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=4, max_features=sqrt, splitter=random;, score=0.771 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=4, max_features=log2, splitter=best;, score=0.765 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=4, max_features=log2, splitter=best;, score=0.726 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=4, max_features=log2, splitter=best;, score=0.737 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=4, max_features=log2, splitter=random;, score=0.682 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=4, max_features=log2, splitter=random;, score=0.676 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=4, max_features=log2, splitter=random;, score=0.749 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=5, max_features=auto, splitter=best;, score=0.659 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=5, max_features=auto, splitter=best;, score=0.749 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=5, max_features=auto, splitter=best;, score=0.743 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=5, max_features=auto, splitter=random;, score=0.659 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=5, max_features=auto, splitter=random;, score=0.682 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=5, max_features=auto, splitter=random;, score=0.698 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=5, max_features=sqrt, splitter=best;, score=0.732 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=5, max_features=sqrt, splitter=best;, score=0.715 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=5, max_features=sqrt, splitter=best;, score=0.771 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=5, max_features=sqrt, splitter=random;, score=0.726 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=5, max_features=sqrt, splitter=random;, score=0.704 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=5, max_features=sqrt, splitter=random;, score=0.765 total time=   0.
0s
[CV 1/3] END criterion=log_loss, max_depth=5, max_features=log2, splitter=best;, score=0.670 total time=   0.0s
[CV 2/3] END criterion=log_loss, max_depth=5, max_features=log2, splitter=best;, score=0.721 total time=   0.0s
[CV 3/3] END criterion=log_loss, max_depth=5, max_features=log2, splitter=best;, score=0.721 total time=   0.0s
[CV 1/3] END criterion=log_loss, max_depth=5, max_features=log2, splitter=random;, score=0.631 total time=   0.
0s
[CV 2/3] END criterion=log_loss, max_depth=5, max_features=log2, splitter=random;, score=0.721 total time=   0.
0s
[CV 3/3] END criterion=log_loss, max_depth=5, max_features=log2, splitter=random;, score=0.670 total time=   0.
0s
```

Out[116]:



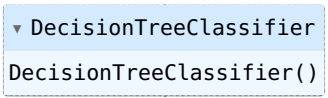
In [117…
```
clf.best_params_
```

Out[117]:
```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 'log2',
 'splitter': 'random'}
```

In [118…
```
##We again fit the model with the best parameter
lassifier=DecisionTreeClassifier(criterion='entropy',max_depth=5,max_features='log2',splitter='random')
```

In [119…
```
classifier.fit(x_train_scaled,y_train)
##Decission Tree initialize
```

Out[119]:



In [120…
```
y_pred=clf.predict(x_test_scaled)
```

In [121…
```
conf_mat = confusion_matrix(y_pred,y_test)
conf_mat
```

Out[121]:
```
array([[97, 24],
       [54, 56]])
```

In [122…
```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][1]
```

In [123…
```
Accuracy = (true_positive + true_negative) / (true_positive +false_positive + false_negative + true_negative)
Accuracy
```

Out[123]:
```
0.6623376623376623
```

In [124…
```
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[124]:
```
0.8016528925619835
```

In [125…
```
Recall = true_positive/(true_positive+false_negative)
Recall
```

Out[125]:
```
0.6423841059602649
```

In [126…
```
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

Out[126]:
```
0.7132352941176471
```

In [127…
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print(confusion_matrix(y_pred,y_test))
print(classification_report(y_pred,y_test))
print(accuracy_score(y_test,y_pred))
```

```
[[97 24]
 [54 56]]
              precision    recall  f1-score   support

           0       0.64      0.80      0.71       121
           1       0.70      0.51      0.59       110

    accuracy                           0.66       231
   macro avg       0.67      0.66      0.65       231
weighted avg       0.67      0.66      0.65       231

0.6623376623376623
```

FIT THE MODEL THROUGH SVC

In [128…
```
##though it is a classification problem we use Support vector classifier
```

In [129…
```
##Support Vector classifier with hyparameter Tubing

##Define the parameter first
param_grid = {'C': [0.1, 1, 10],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel':['linear','rbf','polynomial']
              }
```

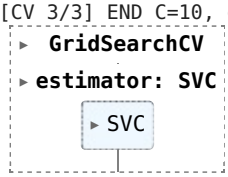
In [130…
```
## import SVC from the sklearn module
from sklearn.svm import SVC
svc=SVC()
from sklearn.model_selection import GridSearchCV
grid=GridSearchCV(svc,param_grid=param_grid,refit=True,cv=3,verbose=3,scoring="accuracy")
grid.fit(x_train_scaled,y_train)
```

```
Fitting 3 folds for each of 45 candidates, totalling 135 fits
[CV 1/3] END .....C=0.1, gamma=1, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END .....C=0.1, gamma=1, kernel=linear;, score=0.777 total time=   0.0s
[CV 3/3] END .....C=0.1, gamma=1, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END ........C=0.1, gamma=1, kernel=rbf;, score=0.654 total time=   0.0s
[CV 2/3] END ........C=0.1, gamma=1, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END ........C=0.1, gamma=1, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END ...C=0.1, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END ...C=0.1, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END ...C=0.1, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.777 total time=   0.0s
[CV 3/3] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.765 total time=   0.0s
```

```
[CV 1/3] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.754 total time=   0.0s
[CV 2/3] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.715 total time=   0.0s
[CV 3/3] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.754 total time=   0.0s
[CV 1/3] END .C=0.1, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END .C=0.1, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END .C=0.1, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.777 total time=   0.0s
[CV 3/3] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.654 total time=   0.0s
[CV 2/3] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END C=0.1, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END C=0.1, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END C=0.1, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END .C=0.1, gamma=0.001, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END .C=0.1, gamma=0.001, kernel=linear;, score=0.777 total time=   0.0s
[CV 3/3] END .C=0.1, gamma=0.001, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.654 total time=   0.0s
[CV 2/3] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END C=0.1, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END C=0.1, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END C=0.1, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END C=0.1, gamma=0.0001, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END C=0.1, gamma=0.0001, kernel=linear;, score=0.777 total time=   0.0s
[CV 3/3] END C=0.1, gamma=0.0001, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.654 total time=   0.0s
[CV 2/3] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END C=0.1, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END C=0.1, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END C=0.1, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END .......C=1, gamma=1, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END .......C=1, gamma=1, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END .......C=1, gamma=1, kernel=linear;, score=0.771 total time=   0.0s
[CV 1/3] END ..........C=1, gamma=1, kernel=rbf;, score=0.709 total time=   0.0s
[CV 2/3] END ..........C=1, gamma=1, kernel=rbf;, score=0.687 total time=   0.0s
[CV 3/3] END ..........C=1, gamma=1, kernel=rbf;, score=0.704 total time=   0.0s
[CV 1/3] END .....C=1, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END .....C=1, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END .....C=1, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END .....C=1, gamma=0.1, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END .....C=1, gamma=0.1, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END .....C=1, gamma=0.1, kernel=linear;, score=0.771 total time=   0.0s
[CV 1/3] END ........C=1, gamma=0.1, kernel=rbf;, score=0.804 total time=   0.0s
[CV 2/3] END ........C=1, gamma=0.1, kernel=rbf;, score=0.749 total time=   0.0s
[CV 3/3] END ........C=1, gamma=0.1, kernel=rbf;, score=0.777 total time=   0.0s
[CV 1/3] END ...C=1, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END ...C=1, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END ...C=1, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ....C=1, gamma=0.01, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ....C=1, gamma=0.01, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ....C=1, gamma=0.01, kernel=linear;, score=0.771 total time=   0.0s
[CV 1/3] END .......C=1, gamma=0.01, kernel=rbf;, score=0.788 total time=   0.0s
[CV 2/3] END .......C=1, gamma=0.01, kernel=rbf;, score=0.782 total time=   0.0s
[CV 3/3] END .......C=1, gamma=0.01, kernel=rbf;, score=0.765 total time=   0.0s
[CV 1/3] END ..C=1, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END ..C=1, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END ..C=1, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ...C=1, gamma=0.001, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ...C=1, gamma=0.001, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ...C=1, gamma=0.001, kernel=linear;, score=0.771 total time=   0.0s
[CV 1/3] END ......C=1, gamma=0.001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 2/3] END ......C=1, gamma=0.001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END ......C=1, gamma=0.001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END .C=1, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END .C=1, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END .C=1, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ..C=1, gamma=0.0001, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ..C=1, gamma=0.0001, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ..C=1, gamma=0.0001, kernel=linear;, score=0.771 total time=   0.0s
[CV 1/3] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.654 total time=   0.0s
[CV 2/3] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END C=1, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END C=1, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END C=1, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ......C=10, gamma=1, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ......C=10, gamma=1, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ......C=10, gamma=1, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END .........C=10, gamma=1, kernel=rbf;, score=0.654 total time=   0.0s
[CV 2/3] END .........C=10, gamma=1, kernel=rbf;, score=0.654 total time=   0.0s
```

```
[CV 3/3] END .........C=10, gamma=1, kernel=rbf;, score=0.715 total time=   0.0s
[CV 1/3] END ....C=10, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END ....C=10, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END ....C=10, gamma=1, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ....C=10, gamma=0.1, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ....C=10, gamma=0.1, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ....C=10, gamma=0.1, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END .......C=10, gamma=0.1, kernel=rbf;, score=0.782 total time=   0.0s
[CV 2/3] END .......C=10, gamma=0.1, kernel=rbf;, score=0.754 total time=   0.0s
[CV 3/3] END .......C=10, gamma=0.1, kernel=rbf;, score=0.760 total time=   0.0s
[CV 1/3] END ..C=10, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END ..C=10, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END ..C=10, gamma=0.1, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ...C=10, gamma=0.01, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ...C=10, gamma=0.01, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ...C=10, gamma=0.01, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END ......C=10, gamma=0.01, kernel=rbf;, score=0.810 total time=   0.0s
[CV 2/3] END ......C=10, gamma=0.01, kernel=rbf;, score=0.743 total time=   0.0s
[CV 3/3] END ......C=10, gamma=0.01, kernel=rbf;, score=0.771 total time=   0.0s
[CV 1/3] END .C=10, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END .C=10, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END .C=10, gamma=0.01, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END ..C=10, gamma=0.001, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END ..C=10, gamma=0.001, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END ..C=10, gamma=0.001, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END .....C=10, gamma=0.001, kernel=rbf;, score=0.788 total time=   0.0s
[CV 2/3] END .....C=10, gamma=0.001, kernel=rbf;, score=0.782 total time=   0.0s
[CV 3/3] END .....C=10, gamma=0.001, kernel=rbf;, score=0.754 total time=   0.0s
[CV 1/3] END C=10, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END C=10, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END C=10, gamma=0.001, kernel=polynomial;, score=nan total time=   0.0s
[CV 1/3] END .C=10, gamma=0.0001, kernel=linear;, score=0.804 total time=   0.0s
[CV 2/3] END .C=10, gamma=0.0001, kernel=linear;, score=0.765 total time=   0.0s
[CV 3/3] END .C=10, gamma=0.0001, kernel=linear;, score=0.765 total time=   0.0s
[CV 1/3] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 2/3] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 3/3] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 1/3] END C=10, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 2/3] END C=10, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
[CV 3/3] END C=10, gamma=0.0001, kernel=polynomial;, score=nan total time=   0.0s
```

Out[130]:



In [133…
```
grid.best_params_
```

Out[133]:
```
{'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
```

In [134…
```
grid.best_score_
```

Out[134]:
```
0.7821229050279329
```

In [135…
```
## after training we do prediction of the data
y_pred=grid.predict(x_test_scaled)
```

In [136…
```
##Accuaracy of the model found through the confusion matrix because it is a classification problem
```

In [137…
```
##CONFUSION MATRIX
conf_mat = confusion_matrix(y_test,y_pred)
conf_mat
```

Out[137]:
```
array([[127,  24],
       [ 34,  46]])
```

In [138…
```
true_positive = conf_mat[0][0]
false_positive = conf_mat[0][1]
false_negative = conf_mat[1][0]
true_negative = conf_mat[1][1]
```

In [139…
```
Accuracy = (true_positive + true_negative) / (true_positive +false_positive + false_negative + true_negative)
Accuracy
```

Out[139]:
```
0.7489177489177489
```

In [140…
```
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[140]:
```
0.8410596026490066
```

In [141…
```
Precision = true_positive/(true_positive+false_positive)
Precision
```

Out[141]: 0.8410596026490066

In [142…
```
F1_Score = 2*(Recall * Precision) / (Recall + Precision)
F1_Score
```

Out[142]: 0.7284176915799432

In [143…
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print(confusion_matrix(y_pred,y_test))
print(classification_report(y_pred,y_test))
print(accuracy_score(y_test,y_pred))
```

```
[[127  34]
 [ 24  46]]
              precision    recall  f1-score   support

           0       0.84      0.79      0.81       161
           1       0.57      0.66      0.61        70

    accuracy                           0.75       231
   macro avg       0.71      0.72      0.71       231
weighted avg       0.76      0.75      0.75       231

0.7489177489177489
```

FIT THE MODEL THROUGH NAIVE BAYES

In [144…
```
df.head()
```

Out[144]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	148.0	72.0	35.000000	79.799479	33.6	0.627	50	1
1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	0
2	183.0	64.0	20.536458	79.799479	23.3	0.672	32	1
3	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1

In [145…
```
## Before fitting the model we want to see the distribution of the features(like weather the numerical feature a
```

In [146…
```
import seaborn as sns
#sns.pairplot(x_train)
```

##Here in the above majority of the features are showing the Gaussian Distribution ,so in this case we can use Gaussian Naivebayes

In [147…
```
from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(x_train_scaled,y_train)
```

Out[147]:
```
▾ GaussianNB
GaussianNB()
```

In [148…
```
## prediction
y_pred=gnb.predict(x_test_scaled)
```

In [149…
```
y_pred
```

Out[149]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
```

In [150…
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print(confusion_matrix(y_pred,y_test))
print(classification_report(y_pred,y_test))
print(accuracy_score(y_test,y_pred))
```

```
[[123  32]
 [ 28  48]]
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       155
           1       0.60      0.63      0.62        76

    accuracy                           0.74       231
   macro avg       0.71      0.71      0.71       231
weighted avg       0.74      0.74      0.74       231

0.7402597402597403
```

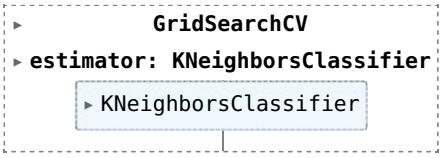
FIT THE MODEL THROUGH KNN(K NEAREST NEIGHBOUR)

In [151…
```
##Though it is a Classification problem we use k_nearest_neighbour_classifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
classifier=KNeighborsClassifier()
parameters={"n_neighbors":[i for i in range(1,10)],"weights":["uniform", "distance"],"algorithm":["auto", "ball
```

In [152…
```
clf=GridSearchCV(classifier,param_grid=parameters)
```

In [153…
```
clf.fit(x_train_scaled,y_train)
```

Out[153]:



In [154…
```
clf.best_params_
```

Out[154]:
```
{'algorithm': 'auto', 'n_neighbors': 9, 'weights': 'uniform'}
```

In [155…
```
##PREDICTION
y_pred=clf.predict(x_test_scaled)
```

In [156…
```
y_pred
```

Out[156]:
```
array([0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
```

In [157…
```
## Import confusion matrix,mean_absolute_error,mean_squared_value
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print(confusion_matrix(y_pred,y_test))
print(classification_report(y_pred,y_test))
print(accuracy_score(y_test,y_pred))
```

```
[[126  33]
 [ 25  47]]
              precision    recall  f1-score   support

           0       0.83      0.79      0.81       159
           1       0.59      0.65      0.62        72

    accuracy                           0.75       231
   macro avg       0.71      0.72      0.72       231
weighted avg       0.76      0.75      0.75       231

0.7489177489177489
```

In [158…
```
## Check the dataset is imbalanced or not means the outcomes of the corresponding dataset is imbalance ,that's u
df["Outcome"].value_counts()
```

Out[158]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

KNN CLASSIFICATION REPORT [[126 33] [25 47]] precision recall f1-score support 0 0.83 0.79 0.81 159 1 0.59 0.65 0.62 72 accuracy 0.75 231 macro avg 0.71 0.72 0.72 231 weighted avg 0.76 0.75 0.75 231 Acuuarcy of the model--0.7489177489177489## SVC CLASSIFICATION REPORT [[127 34] [24 46]] precision recall f1-score support 0 0.84 0.79 0.81 161 1 0.57 0.66 0.61 70 accuracy 0.75 231 macro avg 0.71 0.72 0.71 231 weighted avg 0.76

0.75 0.75 231 0.7489177489177489##Decission Tree Classification Report [[97 24] [54 56]] precision recall f1-score support 0 0.64 0.80 0.71 121 1 0.70 0.51 0.59 110 accuracy 0.66 231 macro avg 0.67 0.66 0.65 231 weighted avg 0.67 0.66 0.65 231 0.6623376623376623## LOGISTIC REGRESSION CLASSIFICATION REPORT [[123 28] [34 46]] precision recall f1-score support 0 0.81 0.78 0.80 157 1 0.57 0.62 0.60 74 accuracy 0.73 231 macro avg 0.69 0.70 0.70 231 weighted avg 0.74 0.73 0.73 231 0.7316017316017316## NAIVE BAYES CLASSIFICATION REPORT [[123 32] [28 48]] precision recall f1-score support 0 0.81 0.79 0.80 155 1 0.60 0.63 0.62 76 accuracy 0.74 231 macro avg 0.71 0.71 0.71 231 weighted avg 0.74 0.74 0.74 231 0.7402597402597403## we know that In the confusion matrix if there is less no of false positive and less no of false negative the model would be a best model ## so now we wants to compare the 5 confusion matrix ,see which have less no of false positive and false negative value ## KNN confusion matrix [[126 33] [25 47]] ##NAIVE BAYES confusion matrix [[123 32] [28 48]] ## LOGISTIC REGRESSION confusion matrix [[123 34] [28 46]] ## DECISSION TREE confusion matrix [[97 24] [54 56]] ## SVC confusion matrix [[127 34] [24 46]]

CONCLUSION

In [78]:
```
## From  the above we saw decission tree gave u.s Really bad result.
## KNN and SVC gave the good result
## SO in this dataset i would prefer KNN and SVC to train the model
```

In [ ]:

In [ ]: