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

```
data=pd.read_csv('C:\Rohith\Backup\Desktop\SEM 6\Machine Learning Lab\Practices\Diabetes\diabetes.csv')
```

```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	

```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000

```
data.shape
```

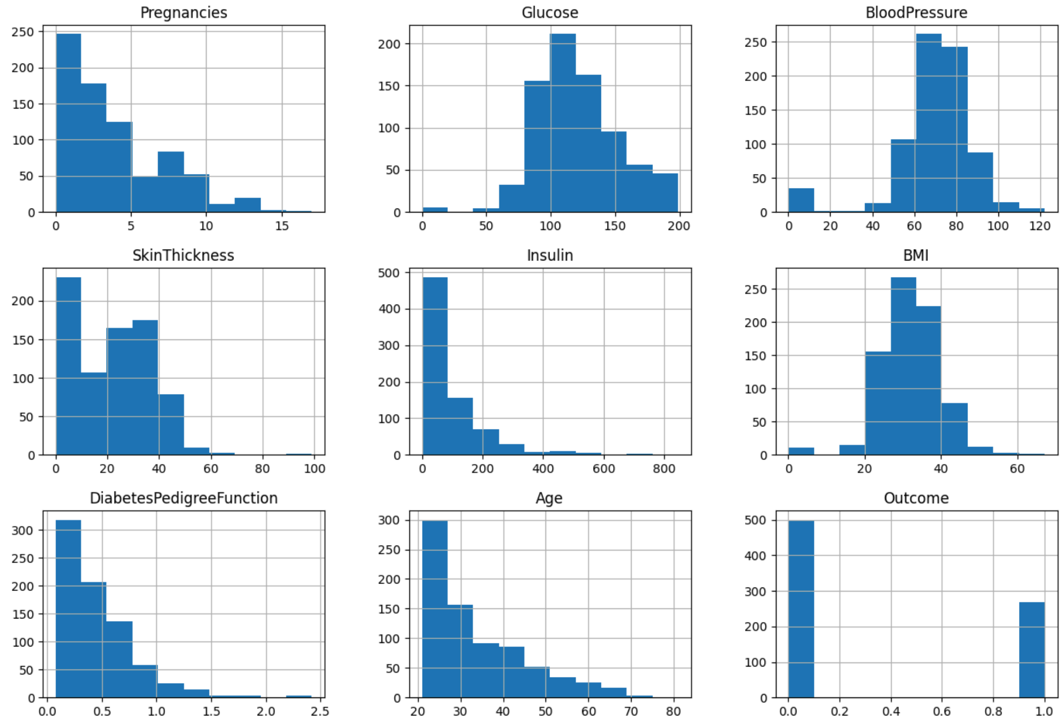
```
(768, 9)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
data.hist(figsize=(15,10))
```

```
array([[<Axes: title={'center': 'Pregnancies'}>,
        <Axes: title={'center': 'Glucose'}>,
        <Axes: title={'center': 'BloodPressure'}>],
       [<Axes: title={'center': 'SkinThickness'}>,
        <Axes: title={'center': 'Insulin'}>,
        <Axes: title={'center': 'BMI'}>],
       [<Axes: title={'center': 'DiabetesPedigreeFunction'}>,
        <Axes: title={'center': 'Age'}>,
        <Axes: title={'center': 'Outcome'}>]], dtype=object)
```




```
#Checking the null values
data.isnull().sum()
```

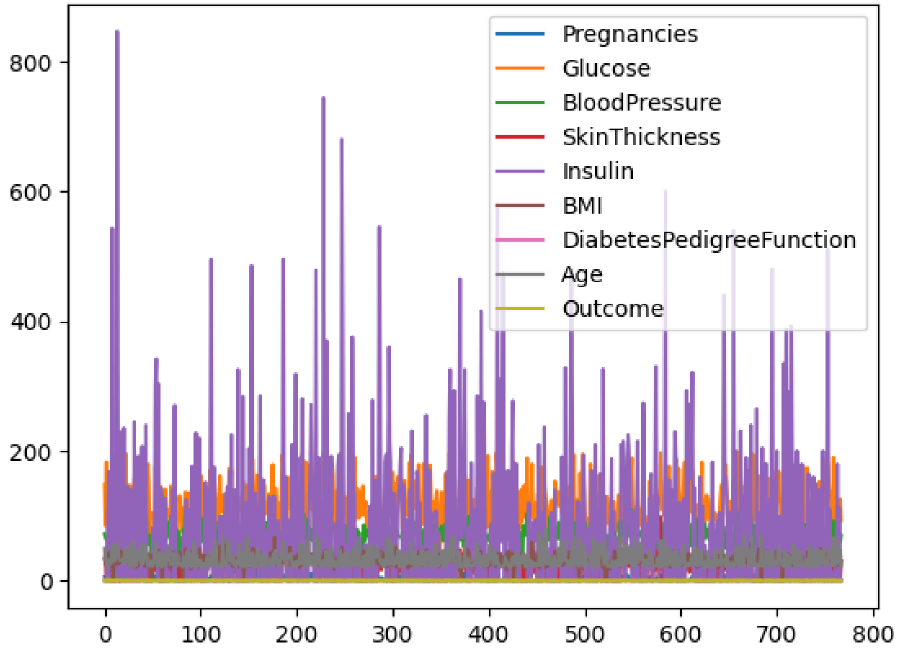
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
data.duplicated().sum()
```

```
0
```

```
data.plot()
```

```
<Axes: >
```




```
data['Insulin'].unique()
```

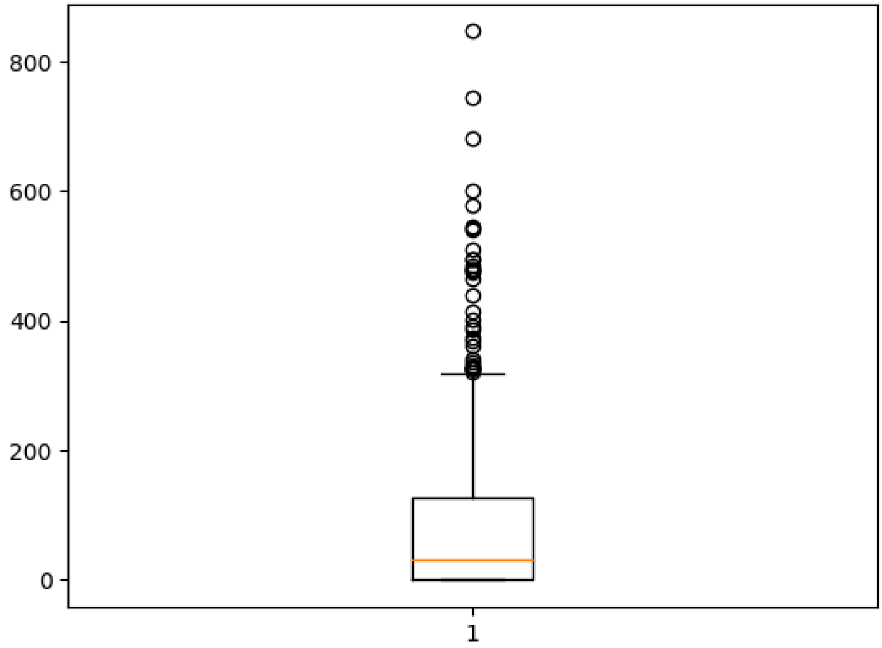
```
array([  0,  94, 168,  88, 543, 846, 175, 230,  83,  96, 235, 146, 115,
       140, 110, 245,  54, 192, 207,  70, 240,  82,  36,  23, 300, 342,
       304, 142, 128,  38, 100,  90, 270,  71, 125, 176,  48,  64, 228,
        76, 220,  40, 152,  18, 135, 495,  37,  51,  99, 145, 225,  49,
        50,  92, 325,  63, 284, 119, 204, 155, 485,  53, 114, 105, 285,
       156,  78, 130,  55,  58, 160, 210, 318,  44, 190, 280,  87, 271,
       129, 120, 478,  56,  32, 744, 370,  45, 194, 680, 402, 258, 375,
       150,  67,  57, 116, 278, 122, 545,  75,  74, 182, 360, 215, 184,
        42, 132, 148, 180, 205,  85, 231,  29,  68,  52, 255, 171,  73,
       108,  43, 167, 249, 293,  66, 465,  89, 158,  84,  72,  59,  81,
       196, 415, 275, 165, 579, 310,  61, 474, 170, 277,  60,  14,  95,
       237, 191, 328, 250, 480, 265, 193,  79,  86, 326, 188, 106,  65,
       166, 274,  77, 126, 330, 600, 185,  25,  41, 272, 321, 144,  15,
       183,  91,  46, 440, 159, 540, 200, 335, 387,  22, 291, 392, 178,
       127, 510,  16, 112], dtype=int64)
```

```
max(data['Insulin'].unique())
```

```
846
```

```
plt.boxplot(data['Insulin'])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x13008342980>,
  <matplotlib.lines.Line2D at 0x13008342c20>],
 'caps': [<matplotlib.lines.Line2D at 0x13008342e00>,
  <matplotlib.lines.Line2D at 0x130083430a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x130083426e0>],
 'medians': [<matplotlib.lines.Line2D at 0x13008343340>],
 'fliers': [<matplotlib.lines.Line2D at 0x130083435e0>],
 'means': []}
```

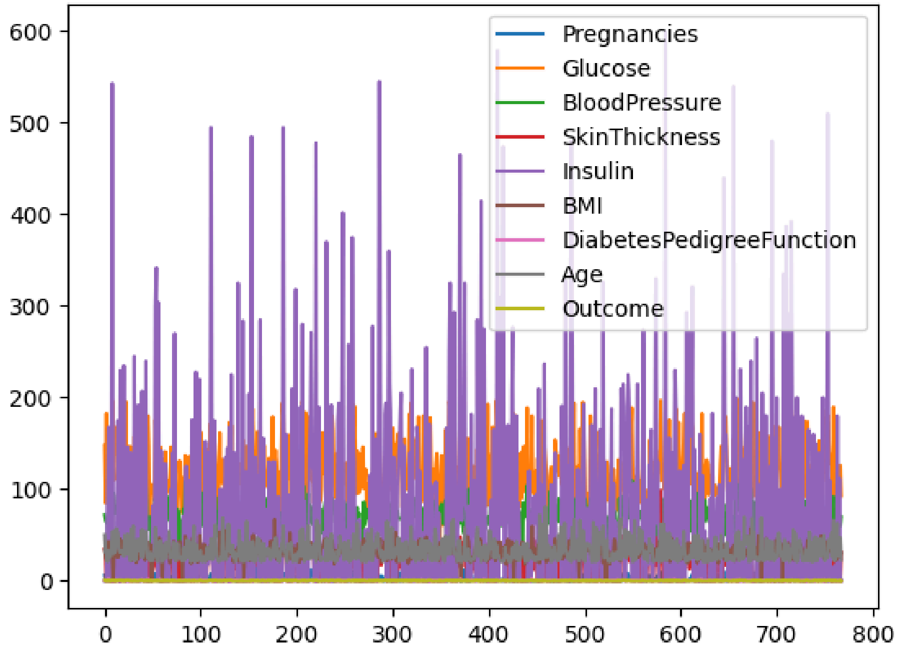



```
df=data[data['Insulin']<=600]
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigı
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

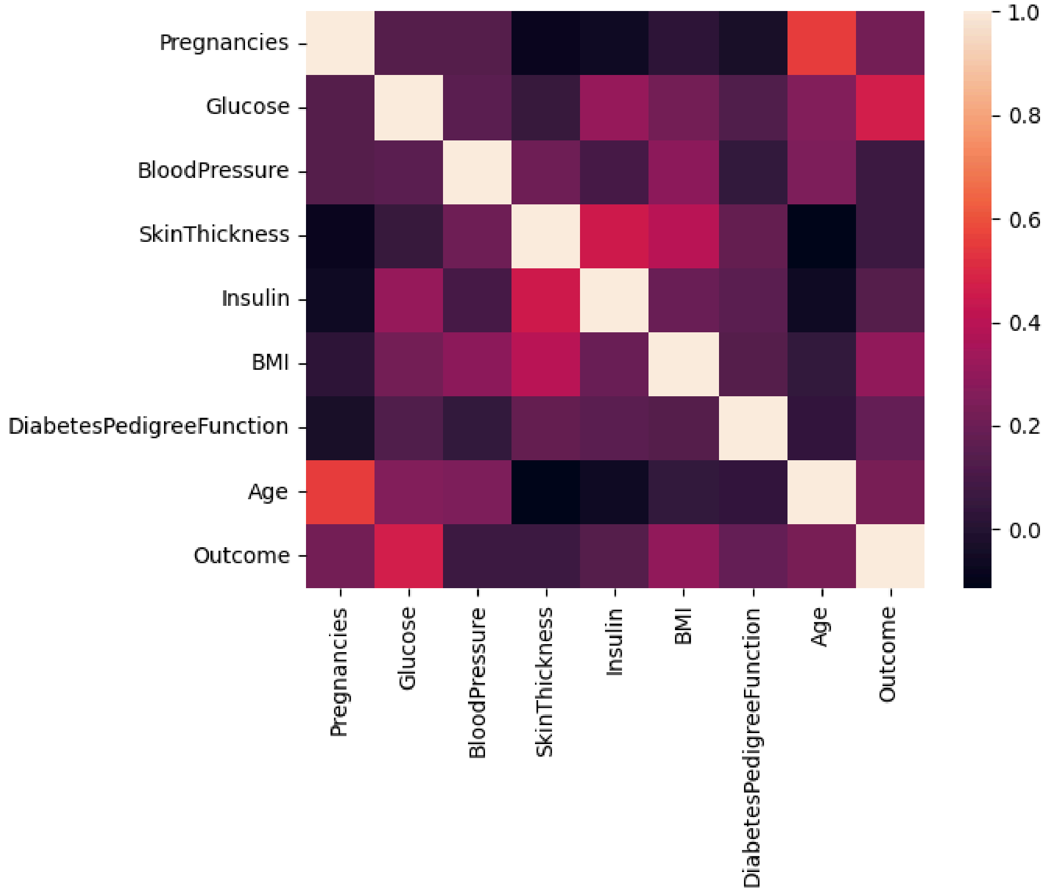
```
df.plot()
```

<Axes: >




```
sns.heatmap(df.corr())
```

<Axes: >



Training and Testing

```
X=df.drop('Outcome',axis=1)
y=df['Outcome']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

```
print("X_train = ",X_train.shape)
print("X_test = ",X_test.shape)
print("y_train = ",y_train.shape)
print("y_test = ",y_test.shape)
```

```
X_train =  (612, 8)
X_test =  (153, 8)
y_train =  (612,)
y_test =  (153,)
```

Model 1 Linear Regression

```
from sklearn.linear_model import LinearRegression
li=LinearRegression()
li
```

```
▾ LinearRegression ⓘ ?
LinearRegression()
```

```
li.fit(X_train,y_train)
y_pred=li.predict(X_test)
accuracy=li.score(X_test,y_test)
accuracy
```

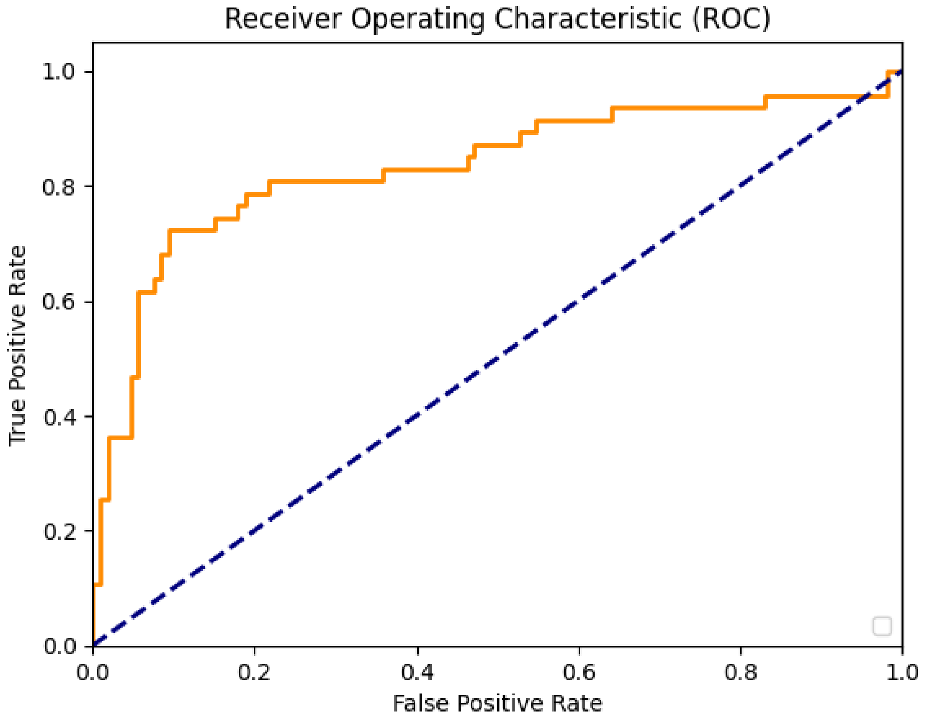
```
0.2775862185270629
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Absolute Error (MAE): 0.3199159992882768
Mean Squared Error (MSE): 0.15374708271597134
Root Mean Squared Error (RMSE): 0.3921059585315828
```

```
#Plotting ROC curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```

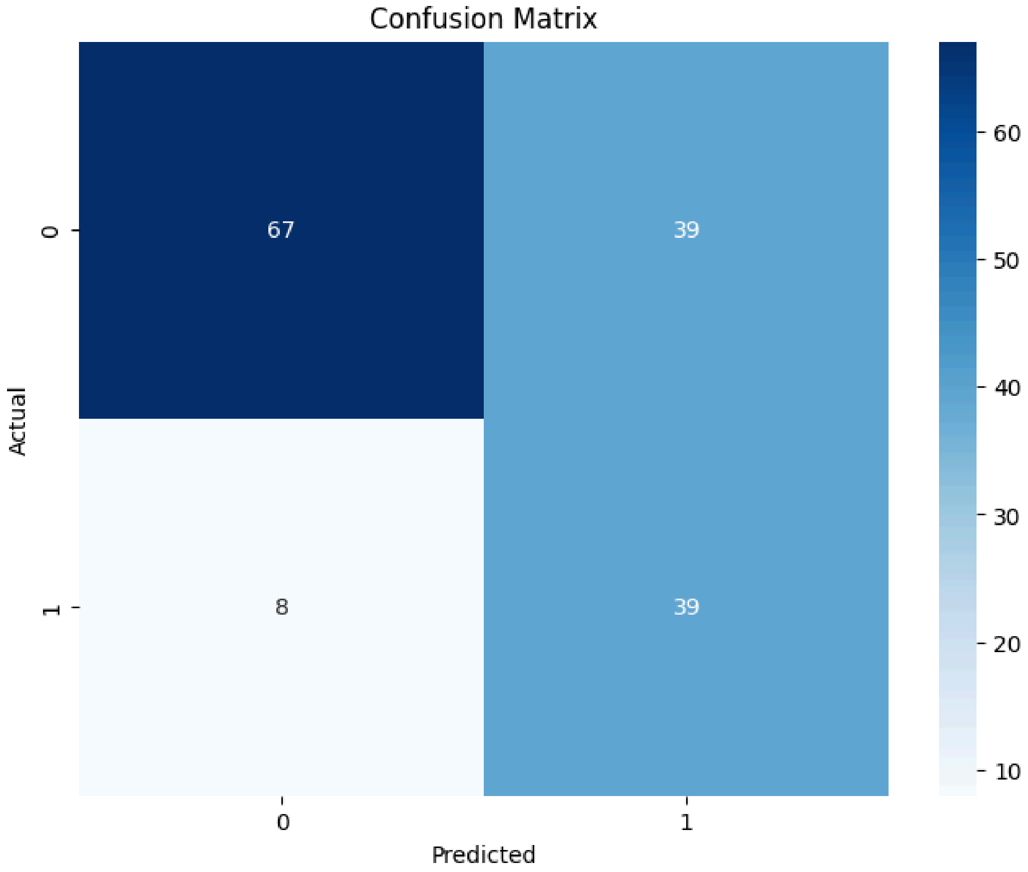
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
from sklearn.metrics import confusion_matrix

# Convert regression output to binary classification
threshold = y_test.mean()  # Example threshold, you can set your own
y_test_class = (y_test > threshold).astype(int)
y_pred_class = (y_pred > threshold).astype(int)

# Compute confusion matrix
cm = confusion_matrix(y_test_class, y_pred_class)
accuracy = accuracy_score(y_test_class, y_pred_class)
print("Linear Regression Accuracy: \n ",accuracy)
print("Linear Regression Confusion Matrix: \n",cm)
```

```
Linear Regression Accuracy: 
  0.6928104575163399
Linear Regression Confusion Matrix: 
 [[67 39]
 [ 8 39]]
```

```
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
from sklearn.metrics import classification_report
print(classification_report(y_test_class,y_pred_class))
```

```
              precision    recall  f1-score   support

           0       0.89      0.63      0.74       106
           1       0.50      0.83      0.62        47

    accuracy                           0.69       153
   macro avg       0.70      0.73      0.68       153
weighted avg       0.77      0.69      0.70       153
```

Model 2 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr
```

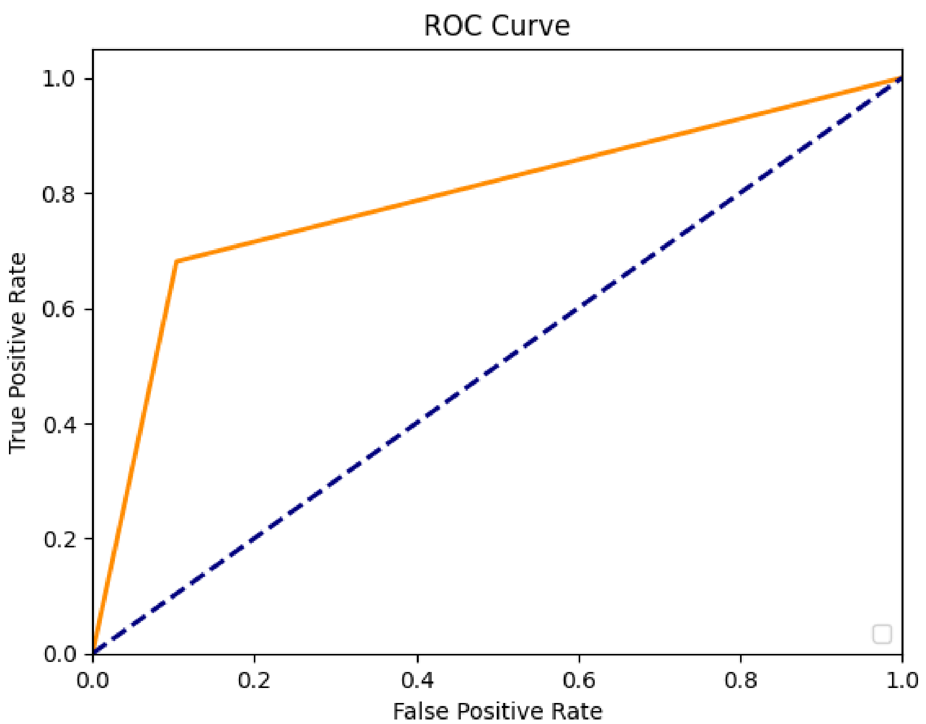
```
LogisticRegression
LogisticRegression()
```

```
lr.fit(X_train,y_train)
y_pred=lr.predict(X_test)
accuracy=lr.score(X_test,y_test)
accuracy
```

```
c:\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
0.8300653594771242
```

```
#Plotting ROC curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color="darkorange",lw=2)
plt.plot([0,1],[0,1],color="navy",lw=2,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

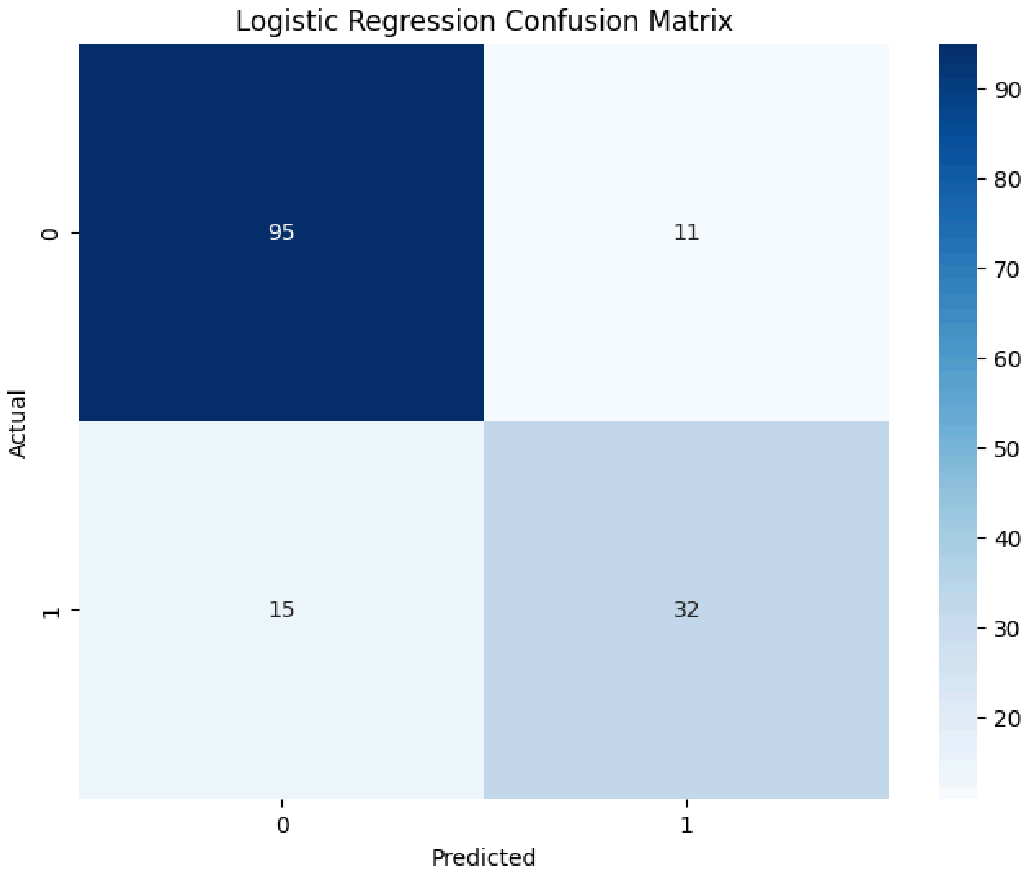
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("Logistic REgression Confusion matrix: \n",cm)
```

```
Logistic REgression Confusion matrix:
 [[95 11]
 [15 32]]
```

```
#Plot Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       106
           1       0.74      0.68      0.71        47

    accuracy                           0.83       153
   macro avg       0.80      0.79      0.80       153
weighted avg       0.83      0.83      0.83       153
```

Model 3 SVM

```
from sklearn.svm import SVC
svm_classifier=SVC(probability=True)
svm_classifier
```

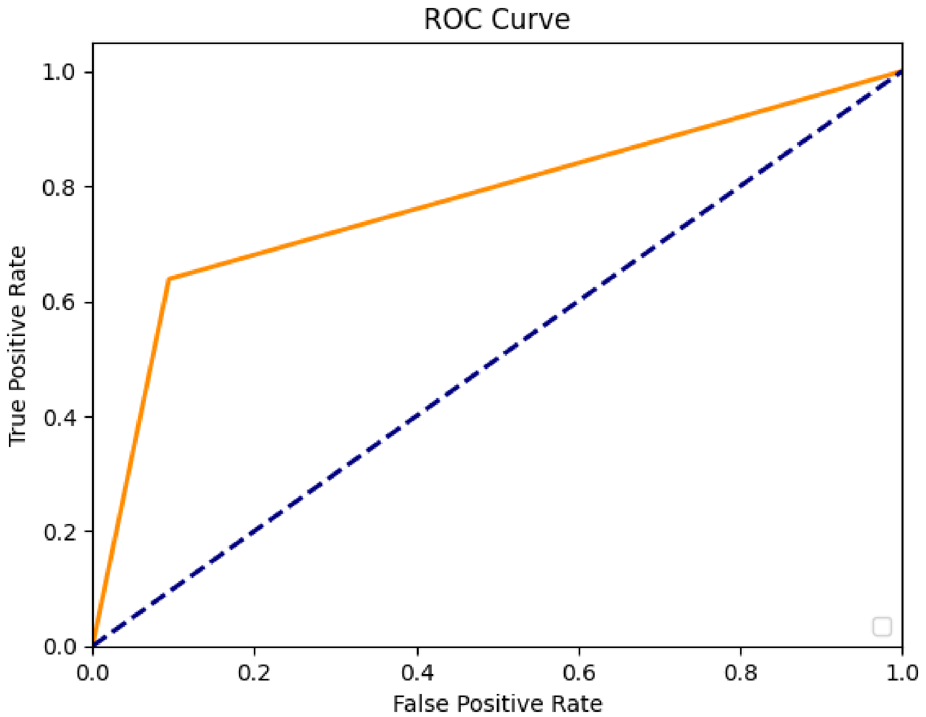
```
SVC
SVC(probability=True)
```

```
svm_classifier.fit(X_train,y_train)
y_pred=svm_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.8235294117647058
```

```
#Plotting ROC_curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1], [0,1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

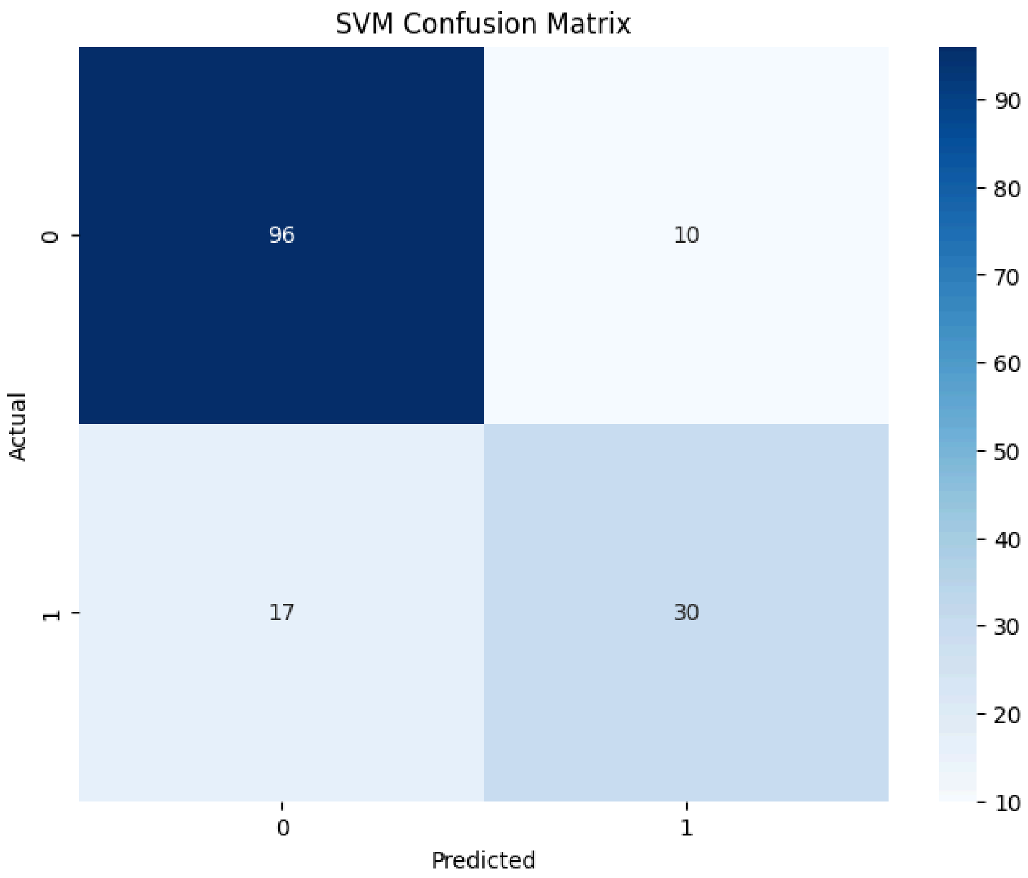
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("SVM Confusion Matrix: \n",cm)
```

```
SVM Confusion Matrix:
 [[96 10]
 [17 30]]
```

```
#Plot Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('SVM Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       106
           1       0.75      0.64      0.69        47

    accuracy                           0.82       153
   macro avg       0.80      0.77      0.78       153
weighted avg       0.82      0.82      0.82       153
```

Model 4 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dt_classifier=DecisionTreeClassifier()
dt_classifier
```

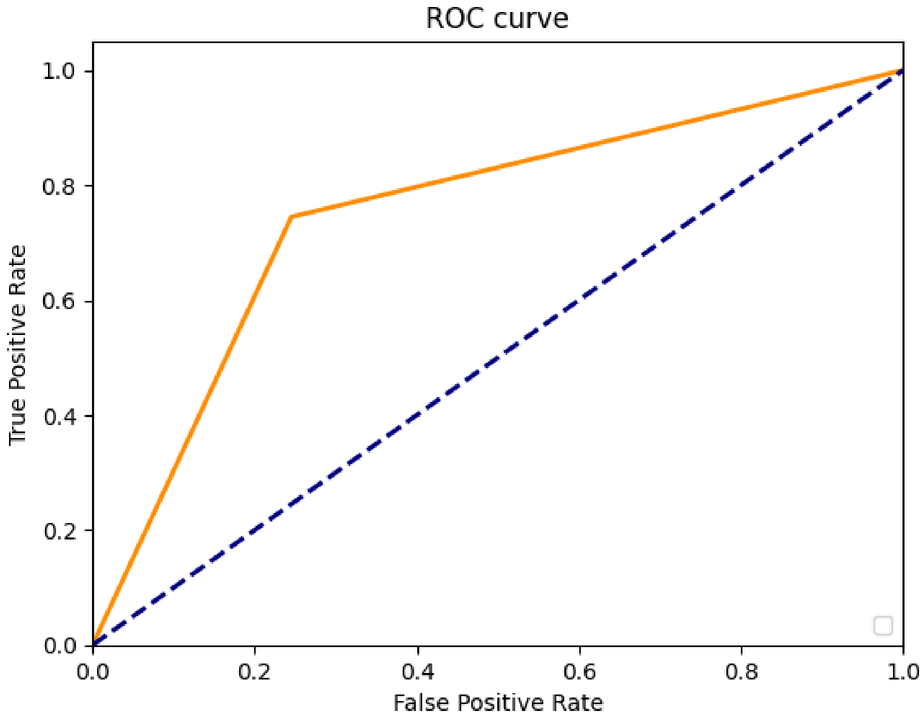
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
dt_classifier.fit(X_train,y_train)
y_pred=dt_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.7516339869281046
```

```
#Plotting ROC Curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1], [0,1], color='navy',linestyle='--',lw=2)
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc='lower right')
plt.show()
```

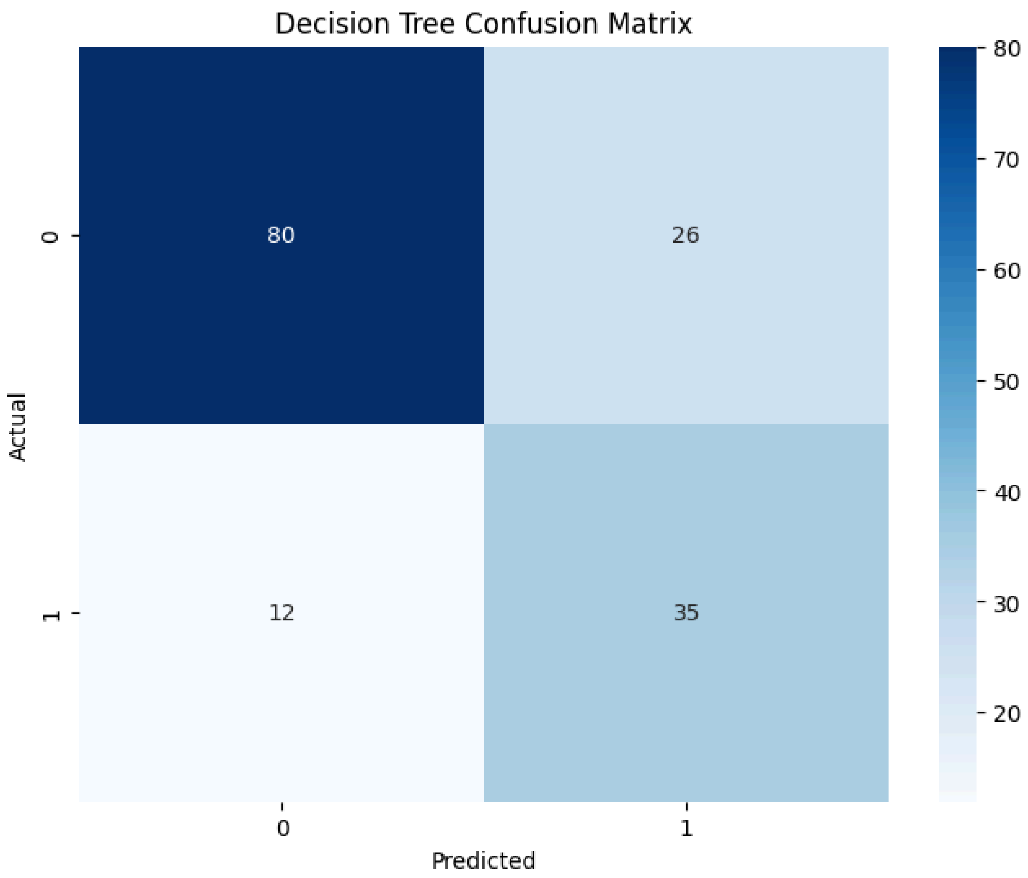
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion Matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("Decision Tree Confusion Matrix: \n",cm)
```

```
Decision Tree Confusion Matrix:
 [[80 26]
 [12 35]]
```

```
#Plot Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.87      0.75      0.81       106
           1       0.57      0.74      0.65        47

    accuracy                           0.75       153
   macro avg       0.72      0.75      0.73       153
weighted avg       0.78      0.75      0.76       153
```

Model 5 Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rf_classifier=RandomForestClassifier()
rf_classifier
```

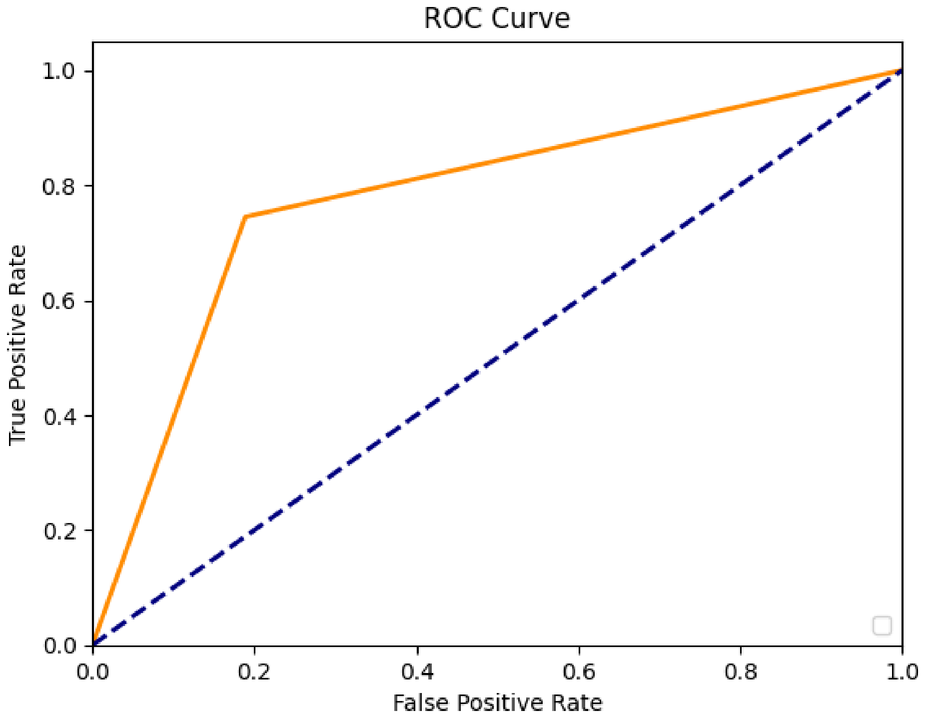
```
RandomForestClassifier
RandomForestClassifier()
```

```
rf_classifier.fit(X_train,y_train)
y_pred=rf_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.7908496732026143
```

```
#Plotting ROC curve

from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1], [0,1], color='navy',lw=2,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

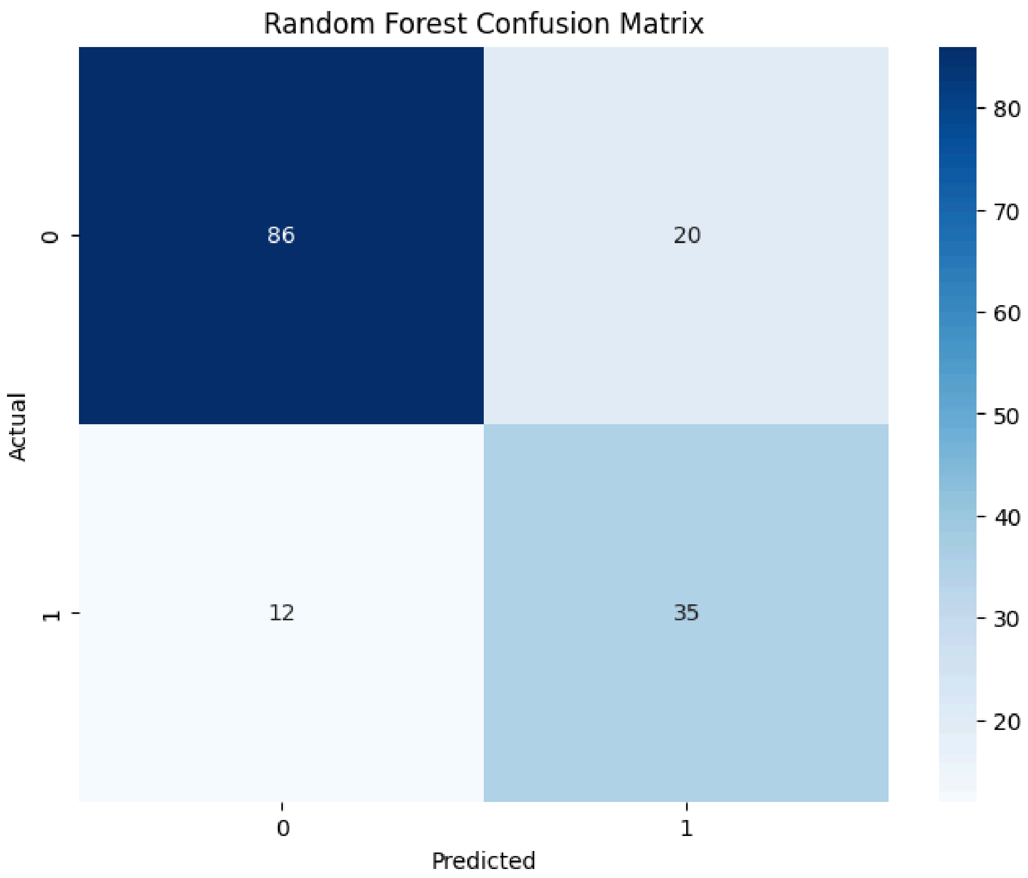
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion Matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("Random Forest Confusion Matrix: \n",cm)
```

```
Random Forest Confusion Matrix:
 [[86 20]
 [12 35]]
```

```
#Plot Confusion Matrix
plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Random Forest Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.88      0.81      0.84       106
           1       0.64      0.74      0.69        47

    accuracy                           0.79       153
   macro avg       0.76      0.78      0.76       153
weighted avg       0.80      0.79      0.79       153
```

Model 6 K Neighbors

```
from sklearn.neighbors import KNeighborsClassifier

knn_classifier=KNeighborsClassifier()
knn_classifier
```

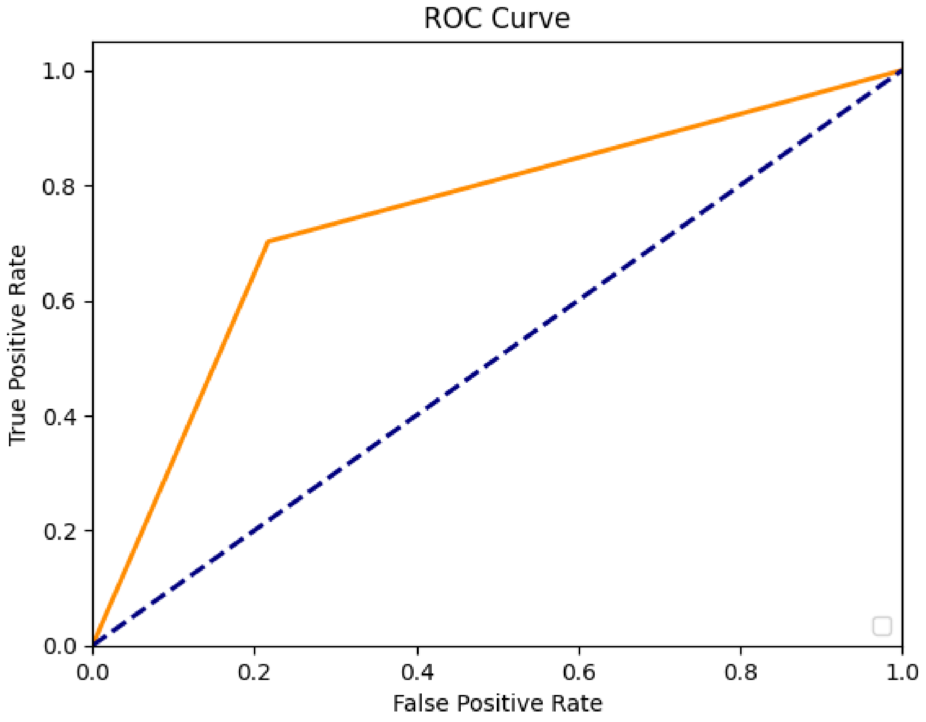
```
KNeighborsClassifier
KNeighborsClassifier()
```

```
knn_classifier.fit(X_train,y_train)
y_pred=knn_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.7581699346405228
```

```
#Plotting ROC_curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1],[0,1],color='navy',lw=2,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

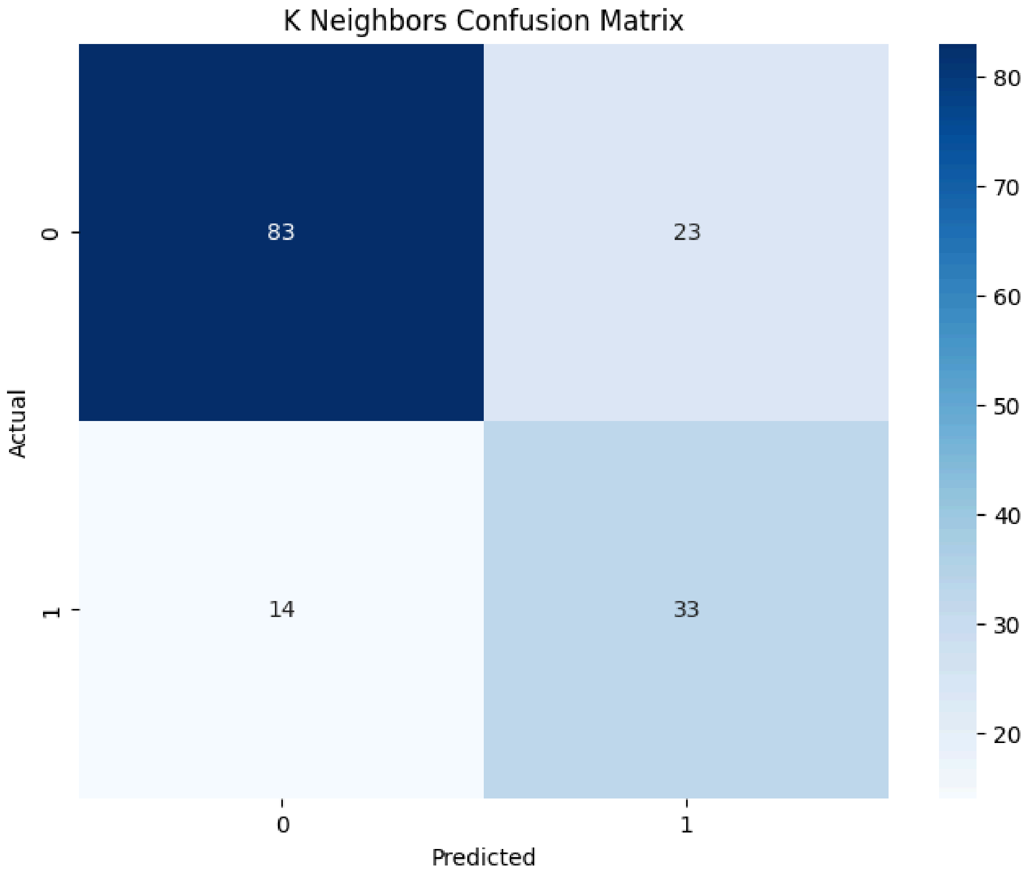
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion Matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("K Neighbors Confusion Matrix: \n",cm)
```

```
K Neighbors Confusion Matrix:
 [[83 23]
 [14 33]]
```

```
#Plotting Confusion Matrix

plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('K Neighbors Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.86      0.78      0.82       106
           1       0.59      0.70      0.64        47

    accuracy                           0.76       153
   macro avg       0.72      0.74      0.73       153
weighted avg       0.77      0.76      0.76       153
```

Model 7 PLA

```
from sklearn.linear_model import Perceptron

pla_classifier=Perceptron(max_iter=1000,tol=1e-3,random_state=42)
pla_classifier
```

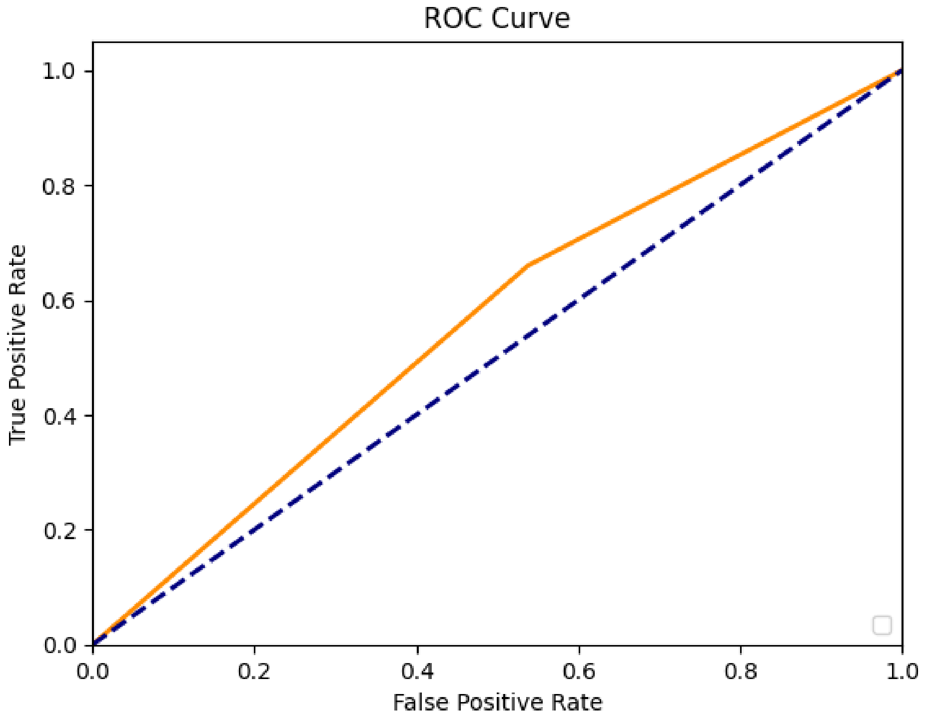
```
Perceptron
Perceptron(random_state=42)
```

```
pla_classifier.fit(X_train,y_train)
y_pred=pla_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.5228758169934641
```

```
#Plotting ROC_curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1],[0,1],color='navy',lw=2,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

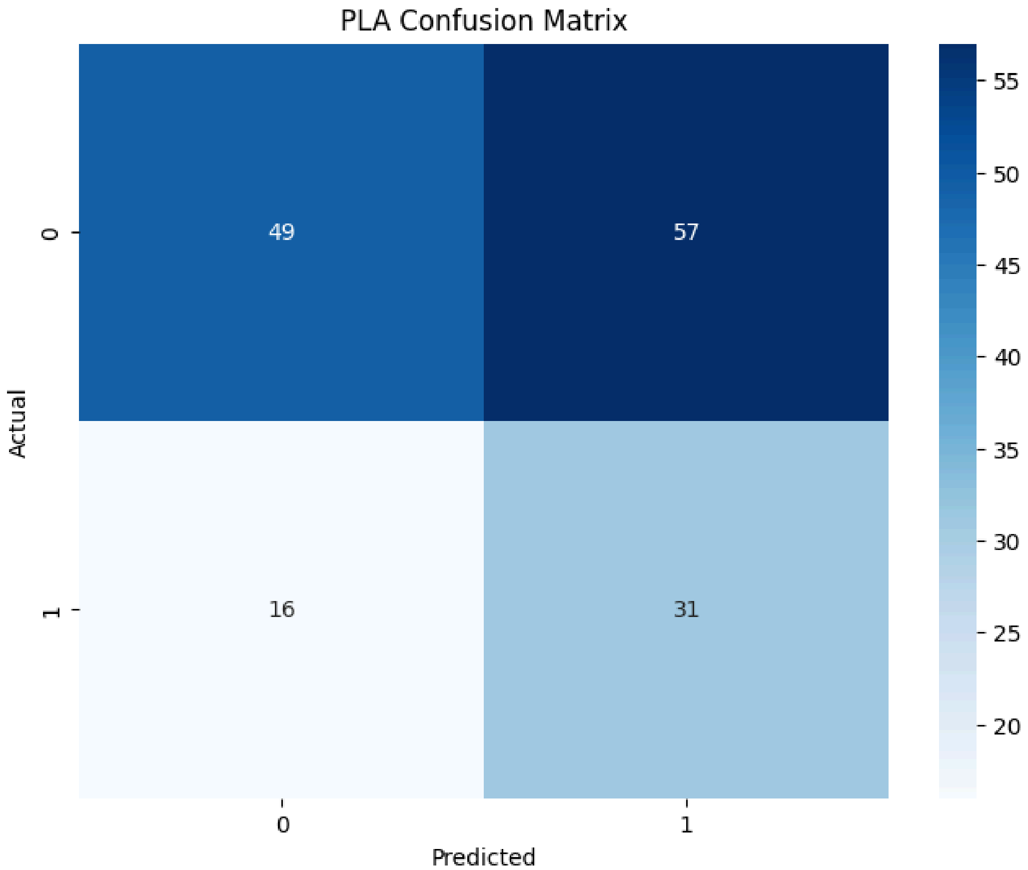
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion Matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("PLA Matrix: \n",cm)
```

```
PLA Matrix:
 [[49 57]
 [16 31]]
```

```
#Plotting Confusion Matrix

plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('PLA Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.75      0.46      0.57       106
           1       0.35      0.66      0.46        47

    accuracy                           0.52       153
   macro avg       0.55      0.56      0.52       153
weighted avg       0.63      0.52      0.54       153
```

Model 8 MLP

```
from sklearn.neural_network import MLPClassifier

mlp_classifier=MLPClassifier(hidden_layer_sizes=(100,),max_iter=300,random_state=42)
mlp_classifier
```

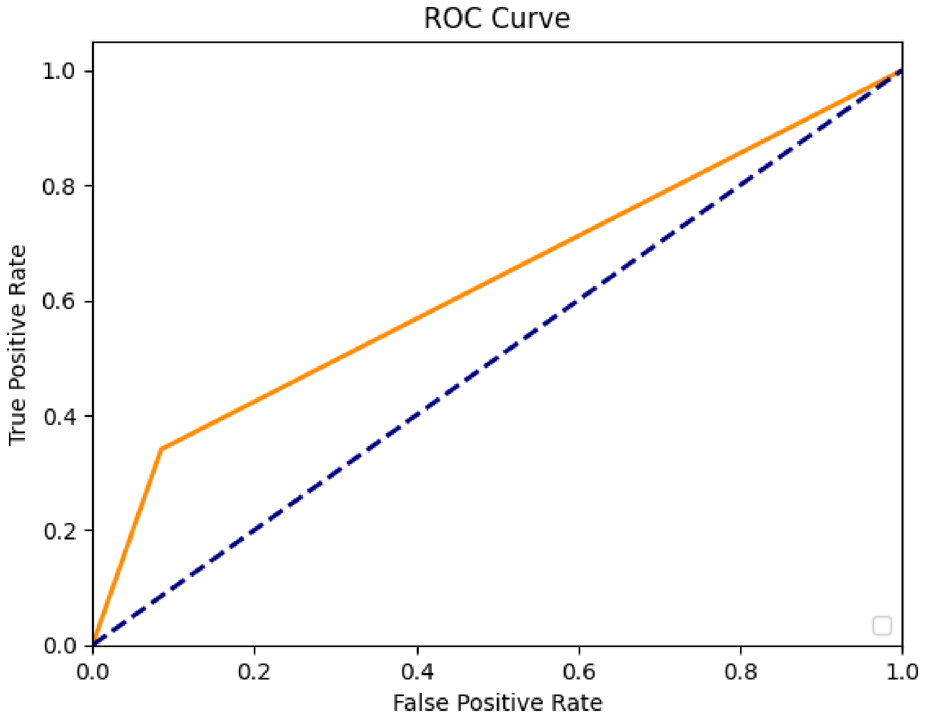
```
MLPClassifier
MLPClassifier(max_iter=300, random_state=42)
```

```
mlp_classifier.fit(X_train,y_train)
y_pred=mlp_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.738562091503268
```

```
#Plotting ROC_curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1],[0,1],color='navy',lw=2,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

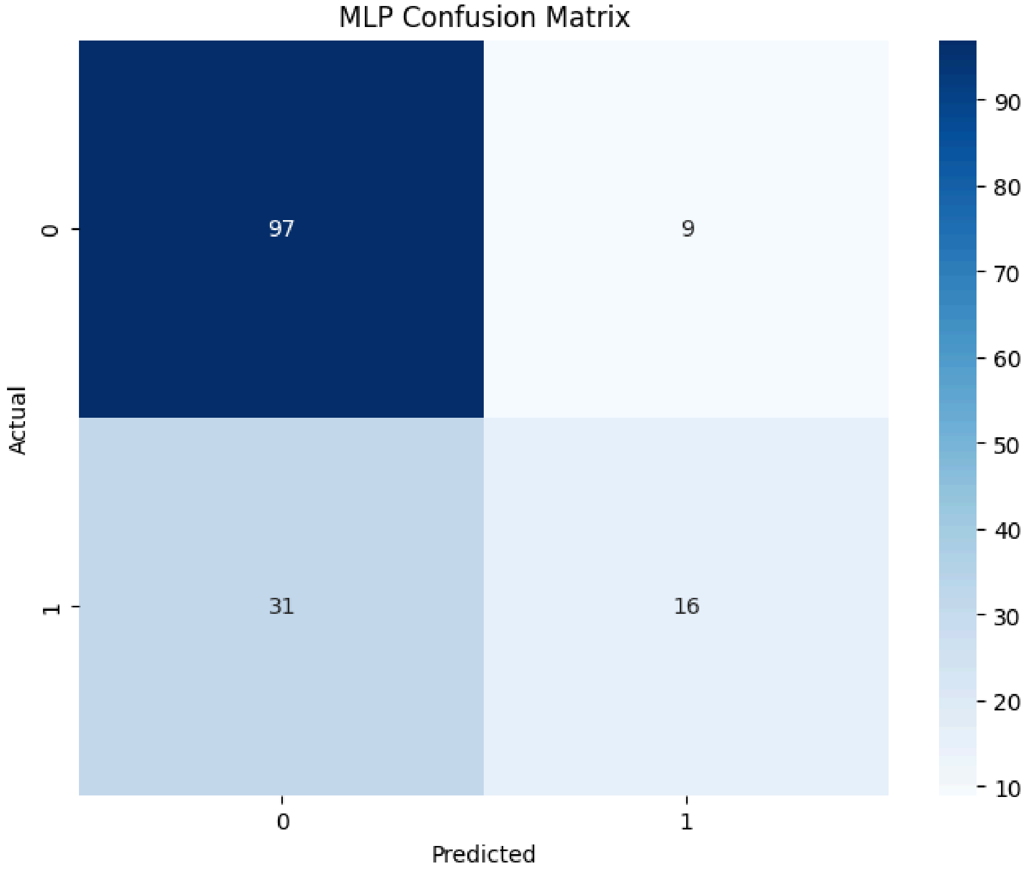
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion Matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("MLP Matrix: \n",cm)
```

```
MLP Matrix: 
 [[97  9]
 [31 16]]
```

```
#Plotting Confusion Matrix

plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('MLP Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.76      0.92      0.83       106
           1       0.64      0.34      0.44        47

    accuracy                           0.74       153
   macro avg       0.70      0.63      0.64       153
weighted avg       0.72      0.74      0.71       153
```

Model 9 Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

nb_classifier=GaussianNB()
nb_classifier
```

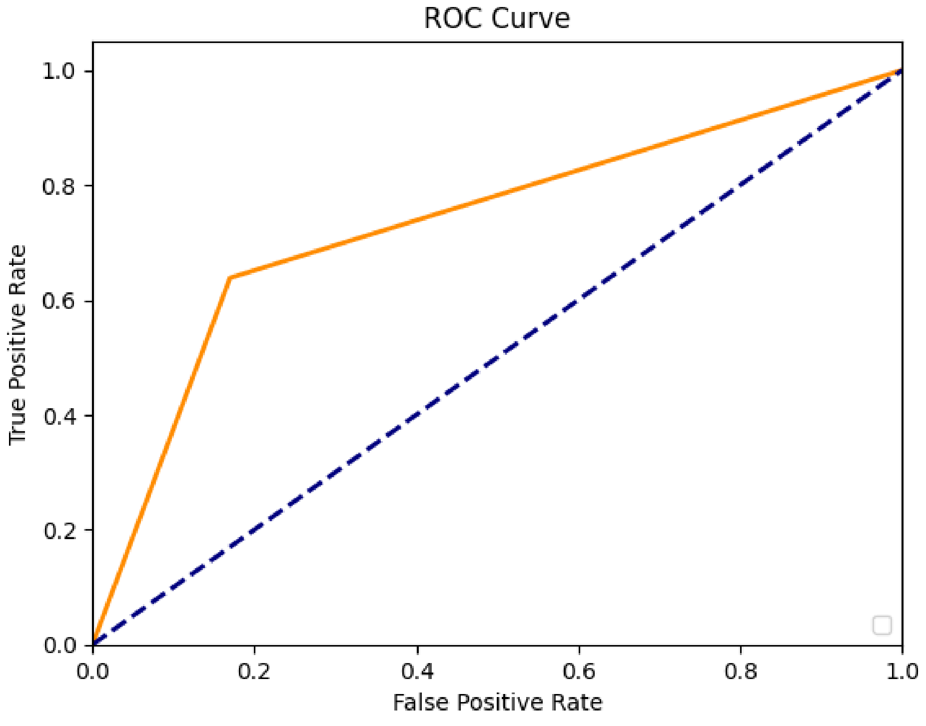
```
GaussianNB
GaussianNB()
```

```
nb_classifier.fit(X_train,y_train)
y_pred=nb_classifier.predict(X_test)
accuracy=accuracy_score(y_test,y_pred)
accuracy
```

```
0.7712418300653595
```

```
#Plotting ROC_curve
from sklearn.metrics import roc_curve,auc,accuracy_score

fpr,tpr,threshold=roc_curve(y_test,y_pred)
roc_auc=auc(fpr,tpr)

plt.figure()
plt.plot(fpr,tpr,color='darkorange',lw=2)
plt.plot([0,1],[0,1],color='navy',lw=2,linestyle='--')
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

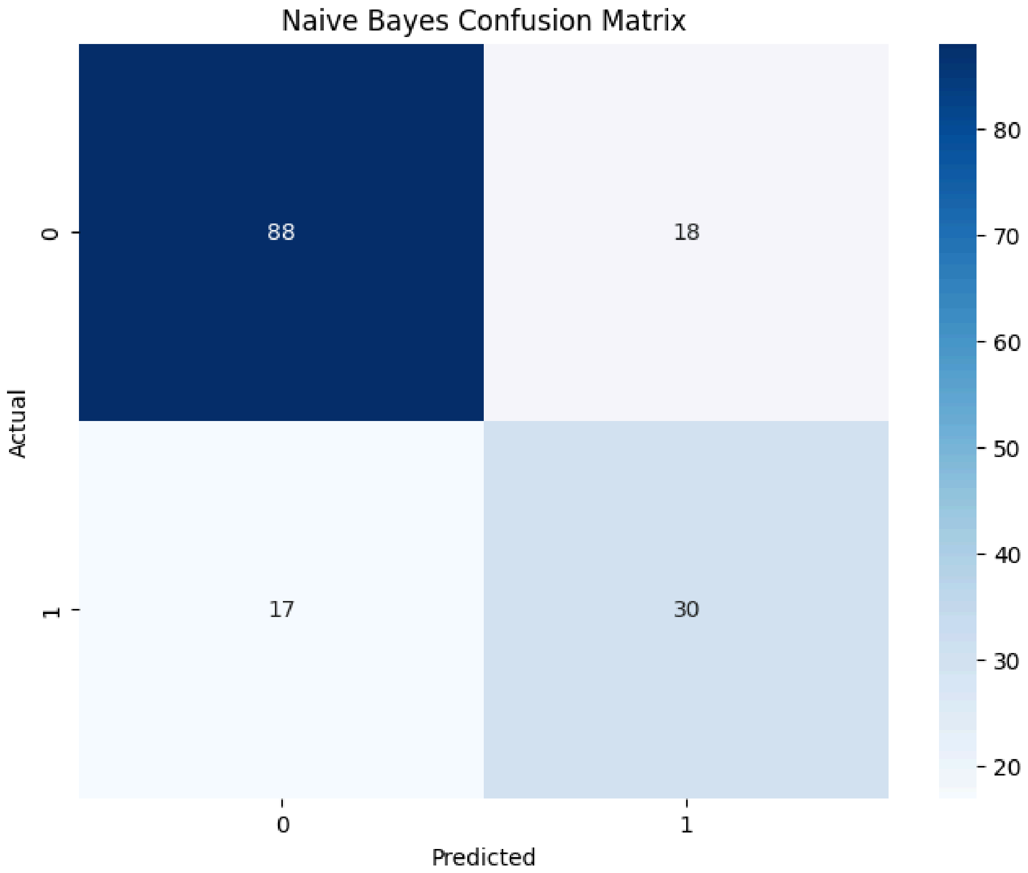
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#Confusion Matrix
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print("Naive Bayes Matrix: \n",cm)
```

```
Naive Bayes Matrix:
 [[88 18]
 [17 30]]
```

```
#Plotting Confusion Matrix

plt.figure(figsize=(8,6))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Naive Bayes Confusion Matrix')
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.84      0.83      0.83       106
           1       0.62      0.64      0.63        47

    accuracy                           0.77       153
   macro avg       0.73      0.73      0.73       153
weighted avg       0.77      0.77      0.77       153
```

Model 10 KMeans Clustering

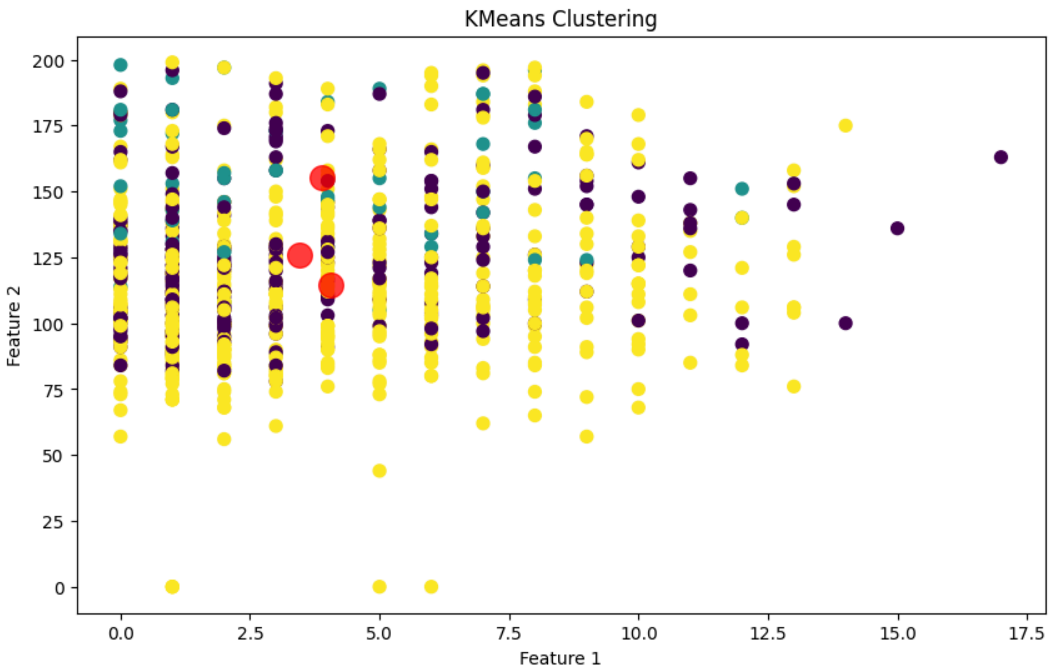
```
from sklearn.cluster import KMeans

kmeans=KMeans(n_clusters=3,random_state=42)
kmeans
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

```
kmeans.fit(X)
y_pred=kmeans.predict(X)

from sklearn.metrics import silhouette_score
silhouette_avg=silhouette_score(X,y_pred)
print("silhouette_score: ",silhouette_avg)
```

```
silhouette_score:  0.4989562777213992
```

```
plt.figure(figsize=(10,6))
plt.scatter(X.iloc[:,0],X.iloc[:,1],c=y_pred,s=50,cmap='viridis')
centers=kmeans.cluster_centers_
plt.scatter(centers[:,0],centers[:,1],c='red',s=200,alpha=0.75)
plt.title('KMeans Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```



Model 11 Dimensionality Reduction using PCA

```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca
```

PCA

PCA(n_components=2)

```
X_pca=pca.fit_transform(X)
```

```
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0],X_pca[:,1],cmap='viridis')
plt.title('Dimensionality Reduction using PCA')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.show()
```

```
C:\Users\rohit\AppData\Local\Temp\ipykernel_17604\2795715780.py:2: UserWarning: No da
  plt.scatter(X_pca[:,0],X_pca[:,1],cmap='viridis')
```