Import necessary Package

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

https://archive.ics.uci.edu/dataset/850/raisin

```
data=pd.read_excel('C:\Rohith\Backup\Desktop\SEM 6\Machine Learning Lab\Practices\Raisin\
```

```
data.head()
```

	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	Extent	Perim
0	87524	442.246011	253.291155	0.819738	90546	0.758651	1184
1	75166	406.690687	243.032436	0.801805	78789	0.684130	1121
2	90856	442.267048	266.328318	0.798354	93717	0.637613	1208
3	45928	286.540559	208.760042	0.684989	47336	0.699599	844

```
data.shape
```

```
(900, 8)
```

```
data.describe()
```

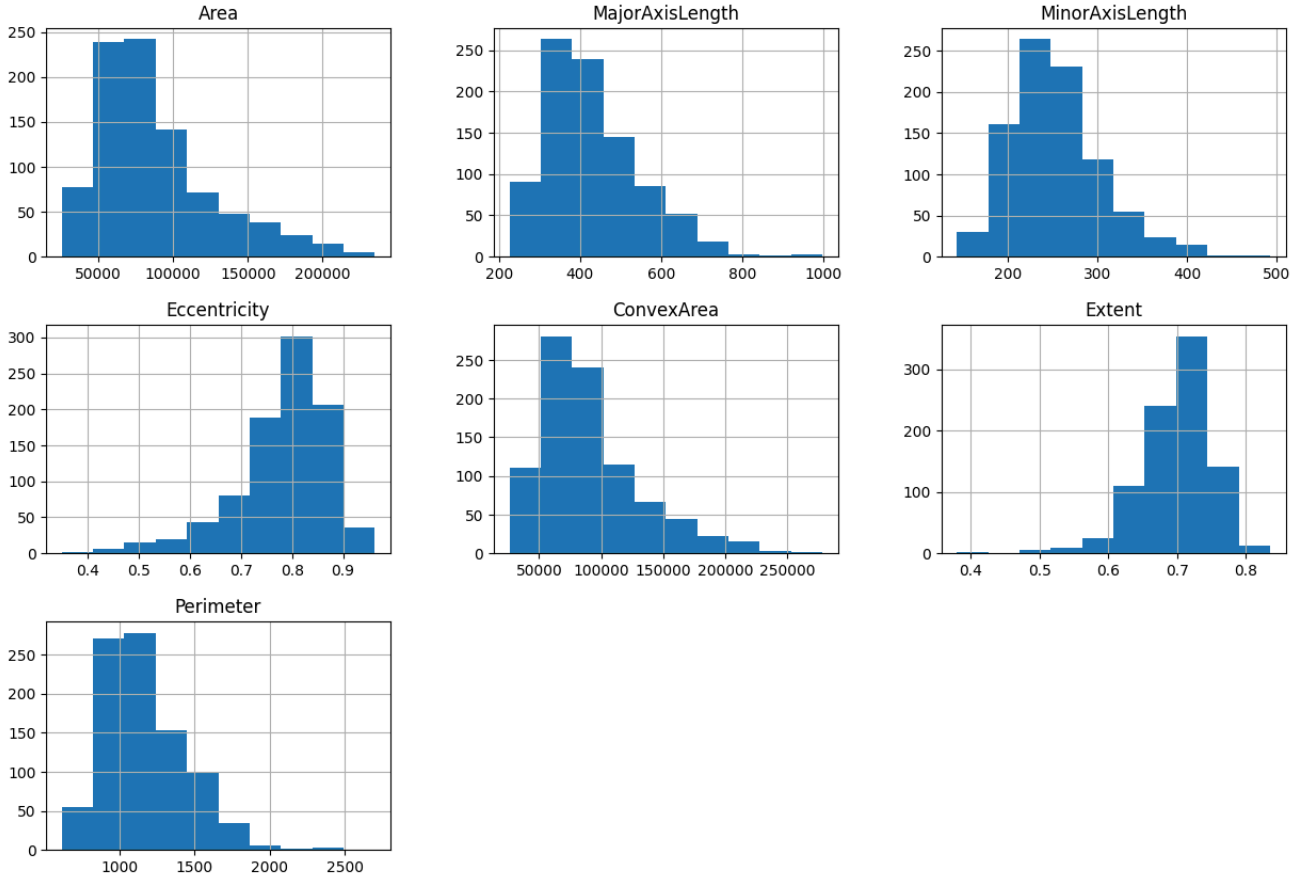
	Area	MajorAxisLength	MinorAxisLength	Eccentricity	ConvexArea	
count	900.000000	900.000000	900.000000	900.000000	900.000000	9
mean	87804.127778	430.929950	254.488133	0.781542	91186.090000	
std	39002.111390	116.035121	49.988902	0.090318	40769.290132	
min	25387.000000	225.629541	143.710872	0.348730	26139.000000	
25%	59348.000000	345.442898	219.111126	0.741766	61513.250000	
50%	78902.000000	407.803951	247.848409	0.798846	81651.000000	
75%	105028.250000	494.187014	279.888575	0.842571	108375.750000	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             900 non-null    int64  
 1   MajorAxisLength  900 non-null    float64
 2   MinorAxisLength  900 non-null    float64
 3   Eccentricity     900 non-null    float64
 4   ConvexArea       900 non-null    int64  
 5   Extent           900 non-null    float64
 6   Perimeter        900 non-null    float64
 7   Class            900 non-null    object 
dtypes: float64(5), int64(2), object(1)
memory usage: 56.4+ KB
```

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

```
array([[<Axes: title={'center': 'Area'}>,
        <Axes: title={'center': 'MajorAxisLength'}>,
        <Axes: title={'center': 'MinorAxisLength'}>],
       [<Axes: title={'center': 'Eccentricity'}>,
        <Axes: title={'center': 'ConvexArea'}>,
        <Axes: title={'center': 'Extent'}>],
       [<Axes: title={'center': 'Perimeter'}>, <Axes: >, <Axes: >]],
      dtype=object)
```




```
data.isnull().sum()
```

```
Area               0
MajorAxisLength    0
MinorAxisLength    0
Eccentricity       0
ConvexArea         0
Extent             0
Perimeter          0
Class              0
dtype: int64
```

```
data.dropna(inplace=True)
```

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

```
0
```

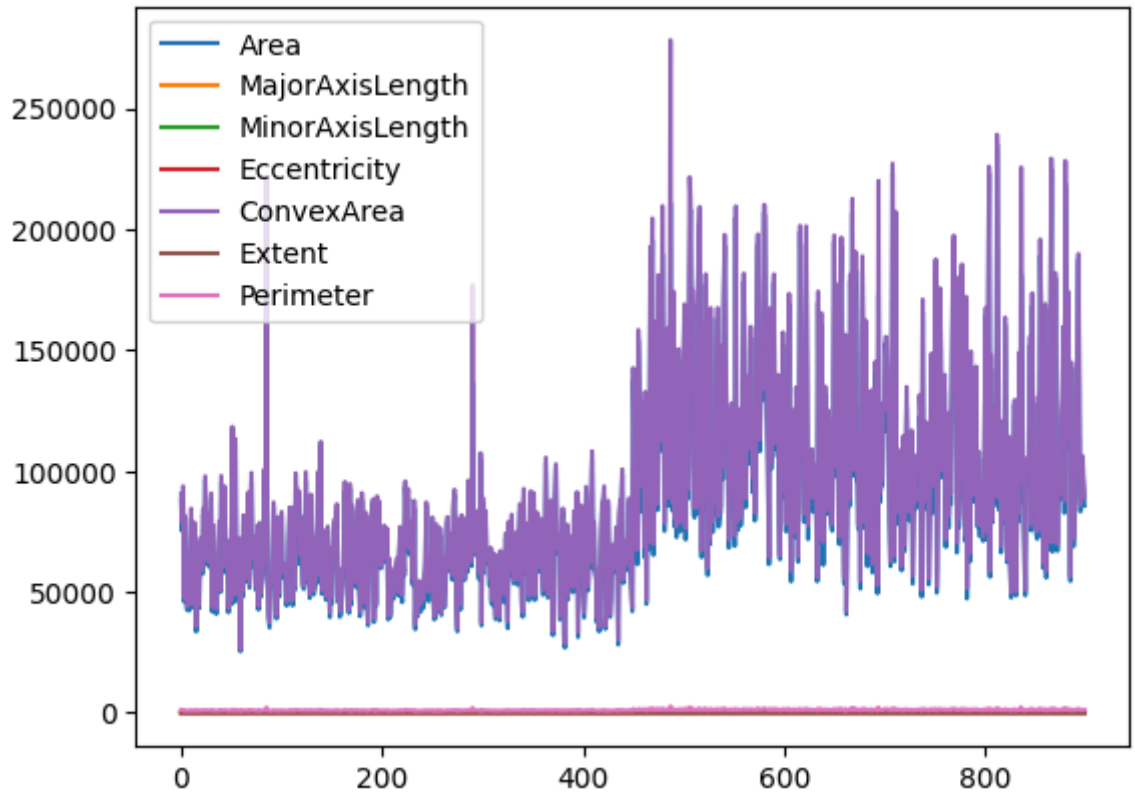
```
data.drop_duplicates(inplace=True)
```

```
data.shape
```

```
(900, 8)
```

```
data.plot()
```

```
<Axes: >
```

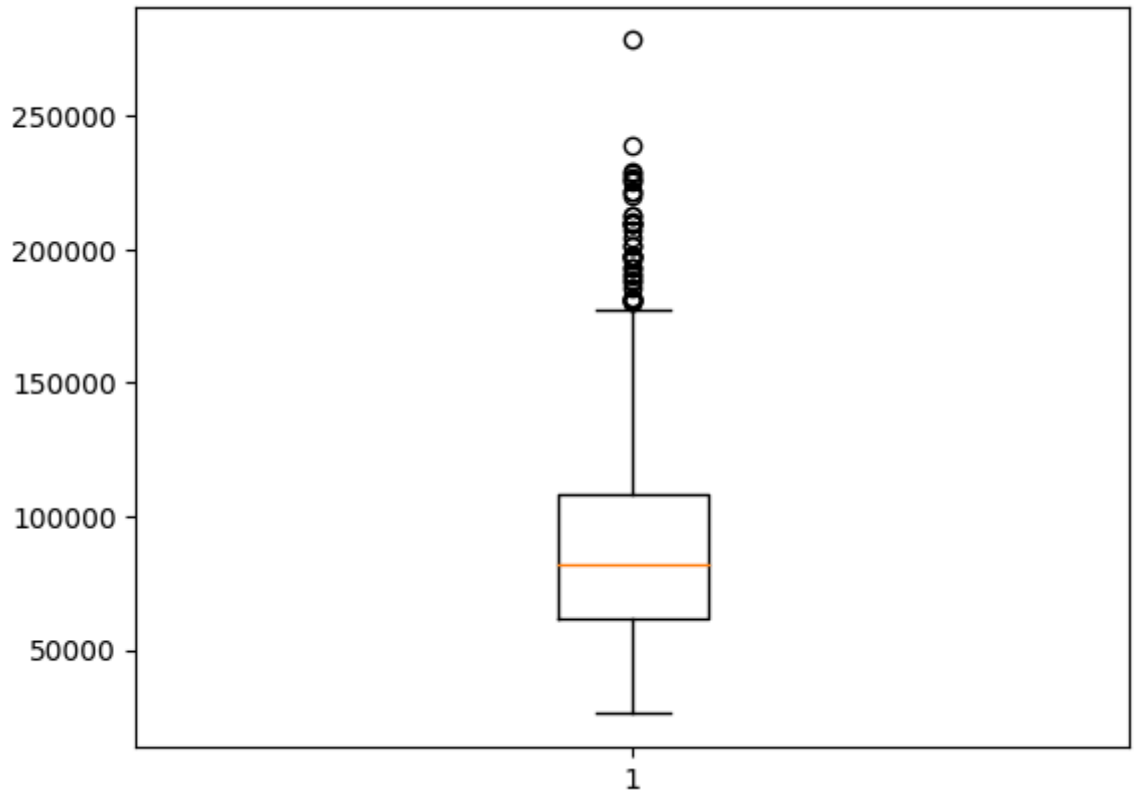



```
max(data['ConvexArea'].unique())
```

```
278217
```

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

```
{'whiskers': [<matplotlib.lines.Line2D at 0x13a1667e920>,
  <matplotlib.lines.Line2D at 0x13a1667ebc0>],
 'caps': [<matplotlib.lines.Line2D at 0x13a1667ee60>,
  <matplotlib.lines.Line2D at 0x13a1667f100>],
 'boxes': [<matplotlib.lines.Line2D at 0x13a1667e680>],
 'medians': [<matplotlib.lines.Line2D at 0x13a1667f3a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x13a1667f640>],
 'means': []}
```

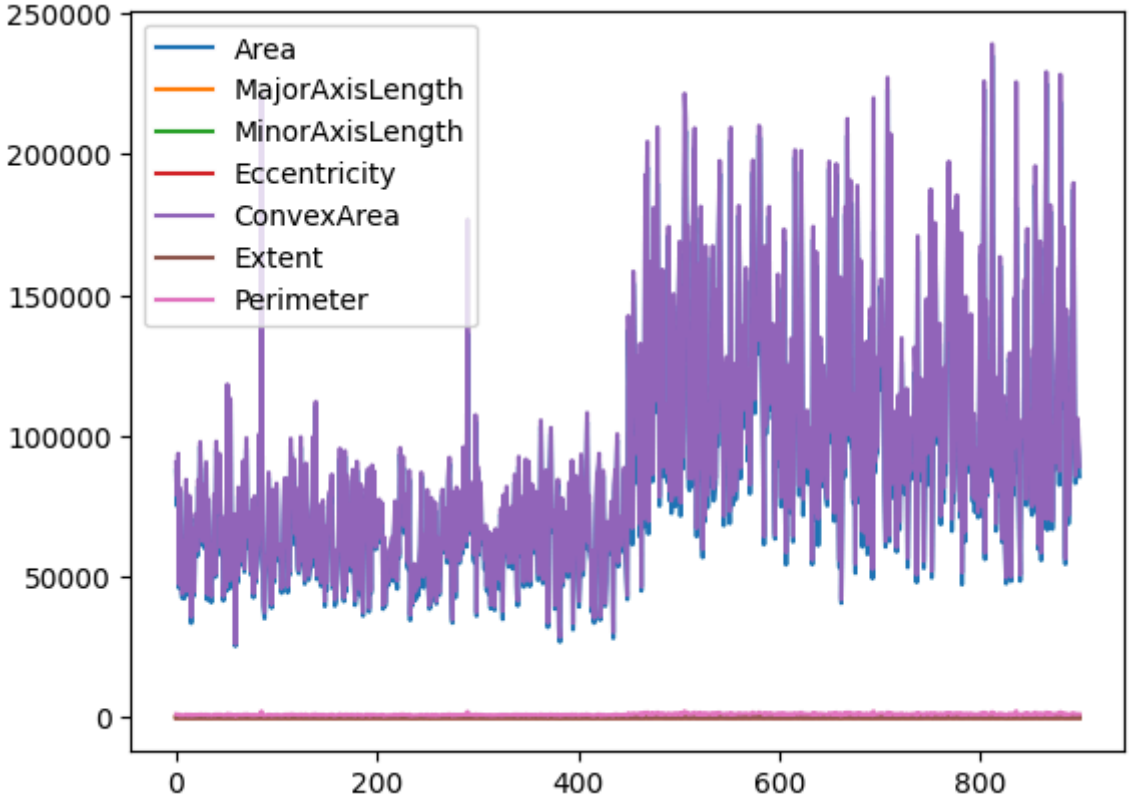



```
df=data[data['ConvexArea']<=240000]
df.shape
```

```
(899, 8)
```

```
df.plot()
```

<Axes: >




```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 899 entries, 0 to 899
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             899 non-null    int64  
 1   MajorAxisLength  899 non-null    float64
 2   MinorAxisLength  899 non-null    float64
 3   Eccentricity     899 non-null    float64
 4   ConvexArea       899 non-null    int64  
 5   Extent           899 non-null    float64
 6   Perimeter        899 non-null    float64
 7   Class            899 non-null    object 
dtypes: float64(5), int64(2), object(1)
memory usage: 63.2+ KB
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
le
```

```
▾ LabelEncoder i ?
LabelEncoder()
```

```
le.fit(df['Class'])
df['Class']=le.transform(df['Class'])
```

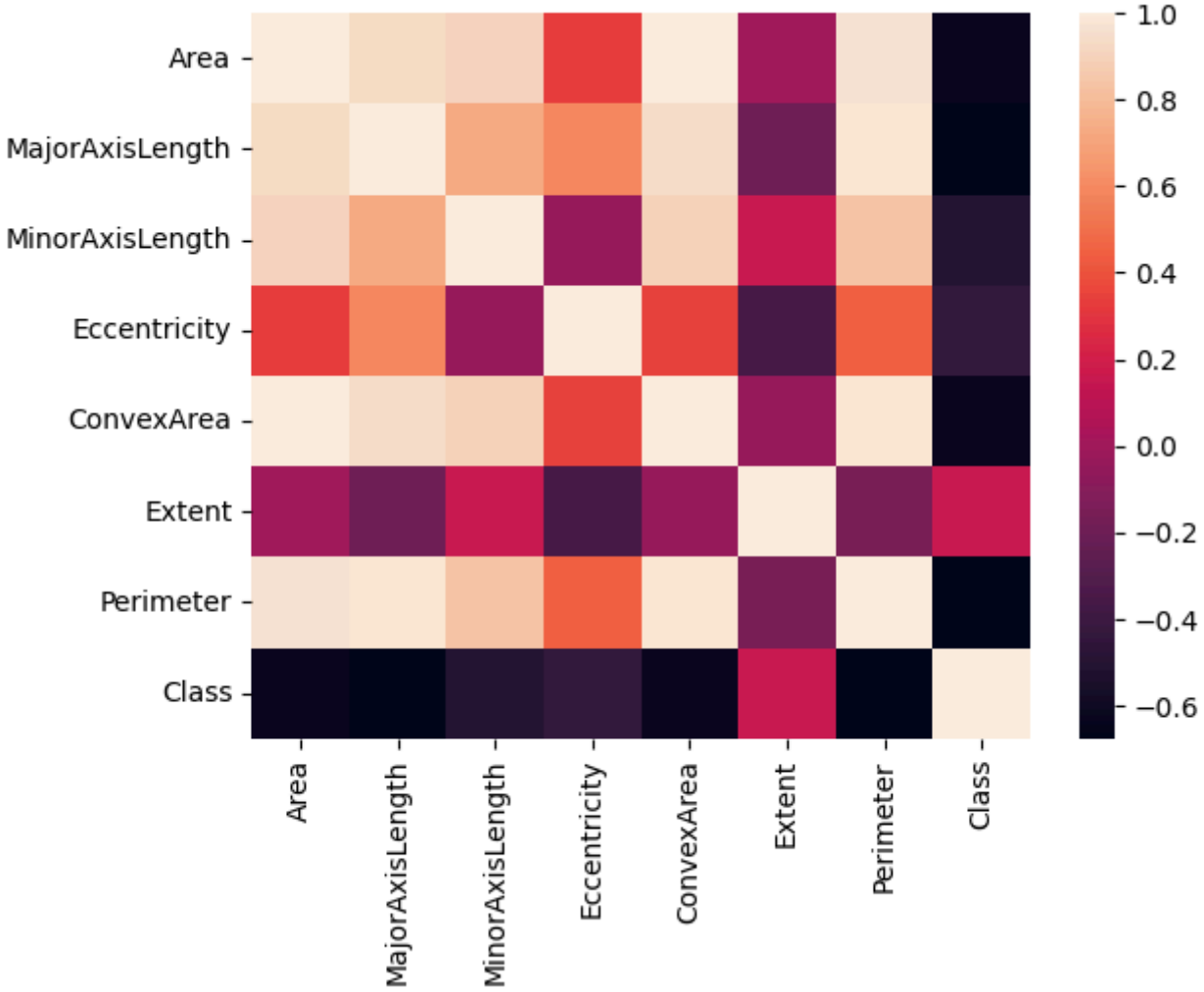
```
C:\Users\rohit\AppData\Local\Temp\ipykernel_2976\1694167980.py:2: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  df['Class']=le.transform(df['Class'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 899 entries, 0 to 899
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area             899 non-null    int64  
 1   MajorAxisLength  899 non-null    float64
 2   MinorAxisLength  899 non-null    float64
 3   Eccentricity     899 non-null    float64
 4   ConvexArea       899 non-null    int64  
 5   Extent           899 non-null    float64
 6   Perimeter        899 non-null    float64
 7   Class            899 non-null    int32  
dtypes: float64(5), int32(1), int64(2)
memory usage: 59.7 KB
```

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

<Axes: >



Training and Testing Data

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

```
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 =  (719, 7)
X_test =  (180, 7)
y_train =  (719,)
y_test =  (180,)
```

Model 1 Linear Regression

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

```
LinearRegression()
```

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

```
0.42138975251373
```

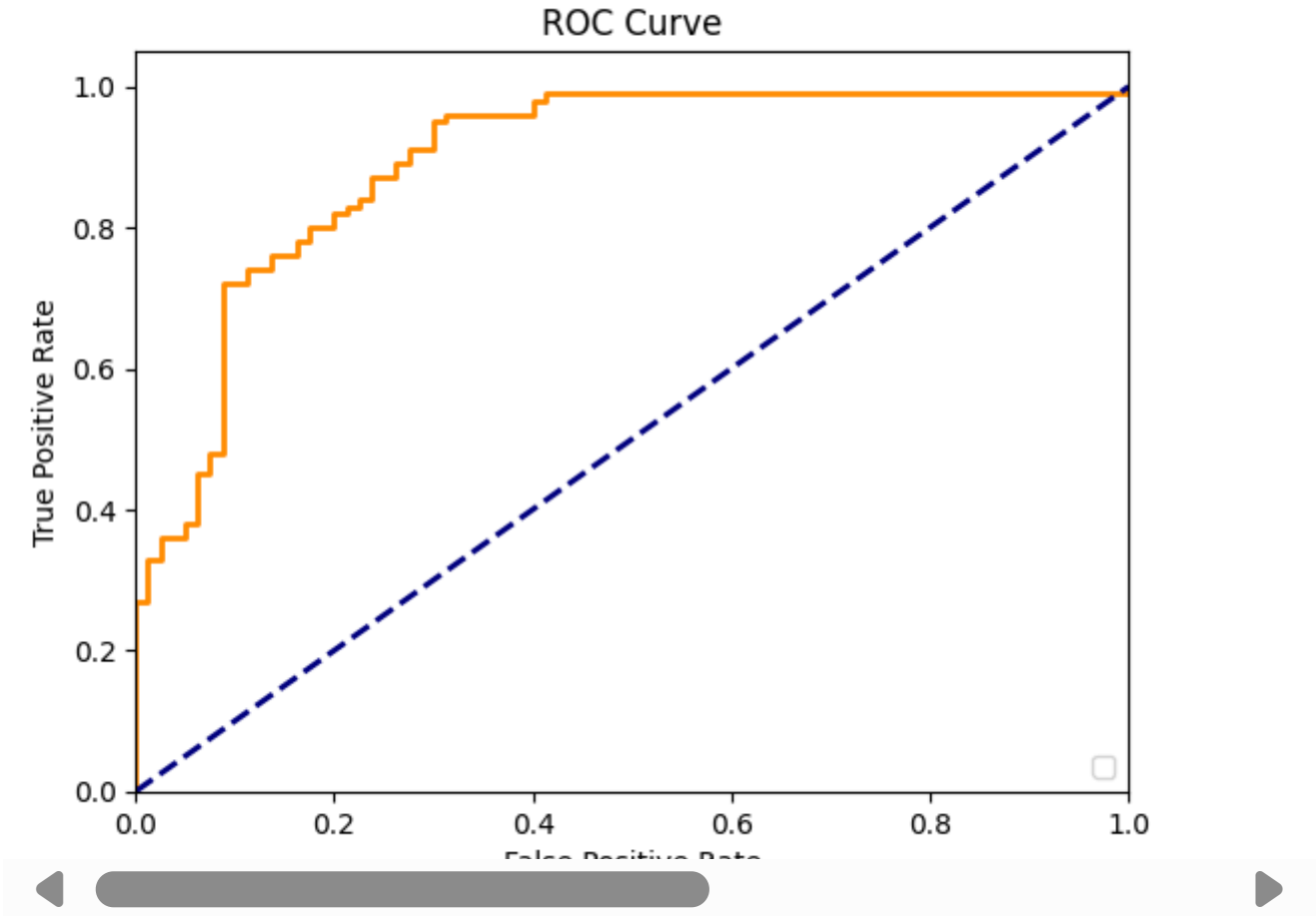
```
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

```
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): 0.30527313252470945
Mean Squared Error (MSE): 0.14286672777438766
Root Mean Squared Error (RMSE): 0.3779771524502343
```

```
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
```



Model 2 Logistic Regression

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

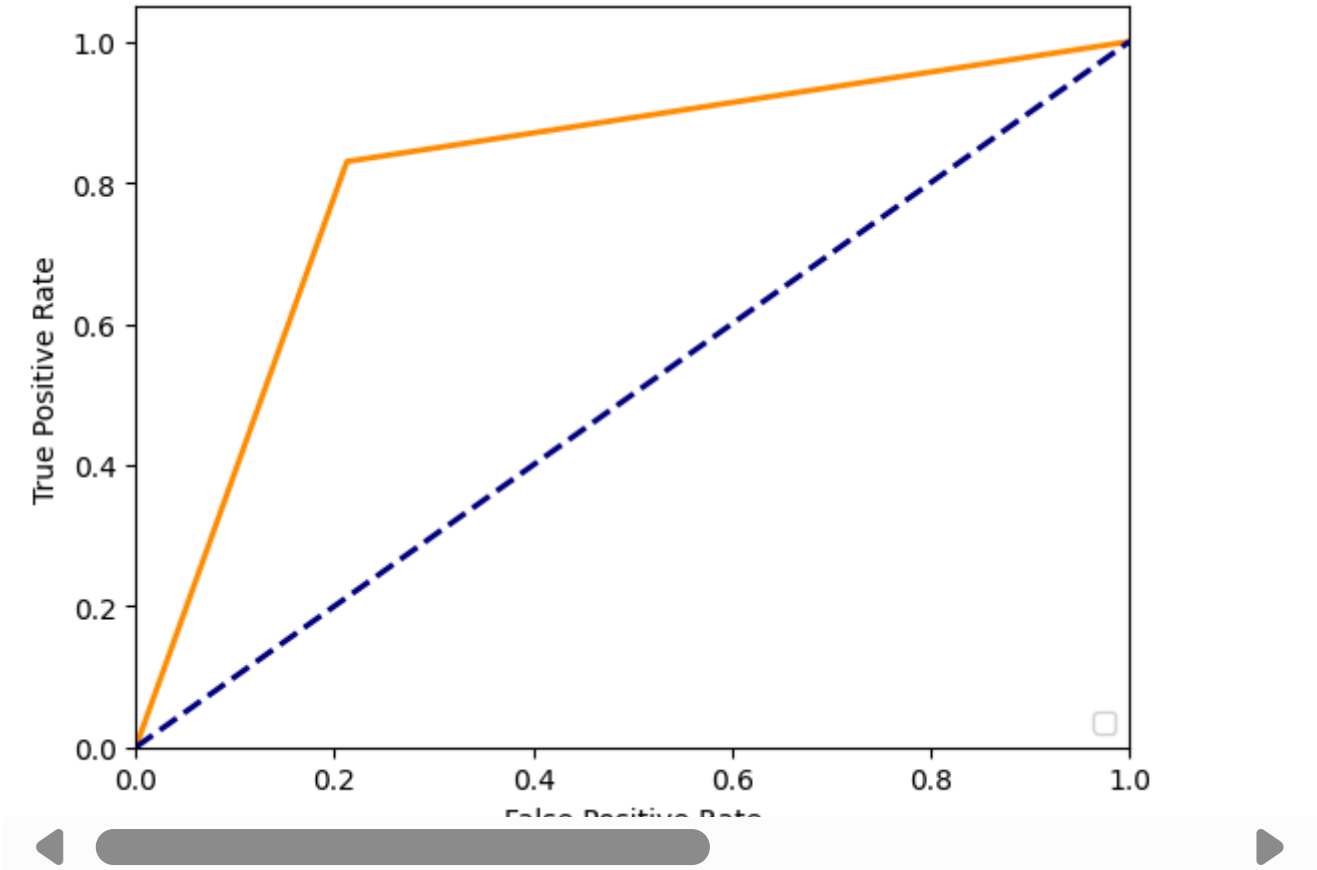
```
LogisticRegression()
```

```
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: ConvergenceWarn
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(
0.8111111111111111
```

```
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.legend(loc='lower right')
plt.show()
```

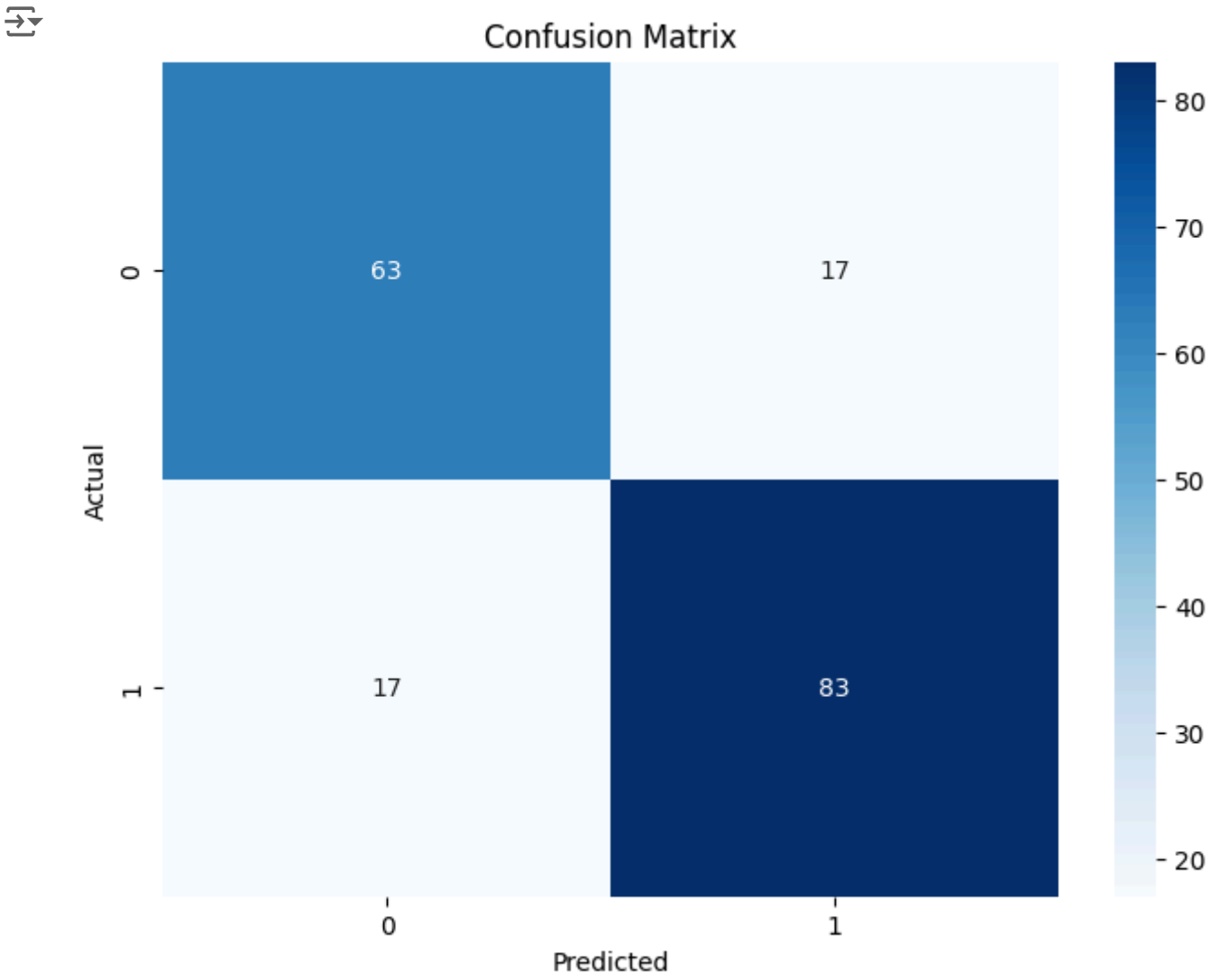
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[63 17]
 [17 83]]
```

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




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

```
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        80
           1       0.83      0.83      0.83       100

    accuracy                           0.81       180
   macro avg       0.81      0.81      0.81       180
weighted avg       0.81      0.81      0.81       180
```

Model 3 SVM

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

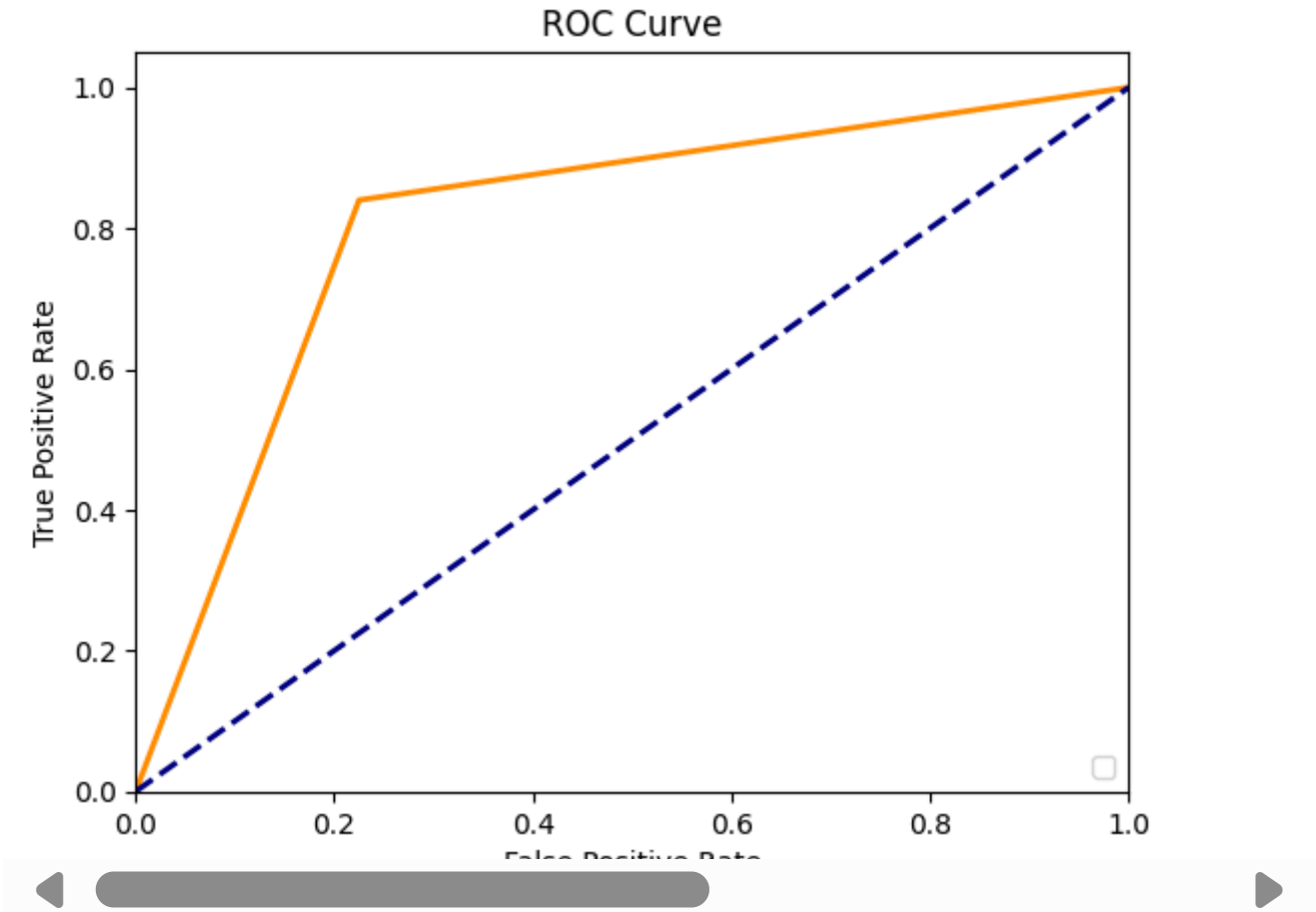
```
SVC
SVC(probability=True)
```

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

```
0.8111111111111111
```

```
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()
```

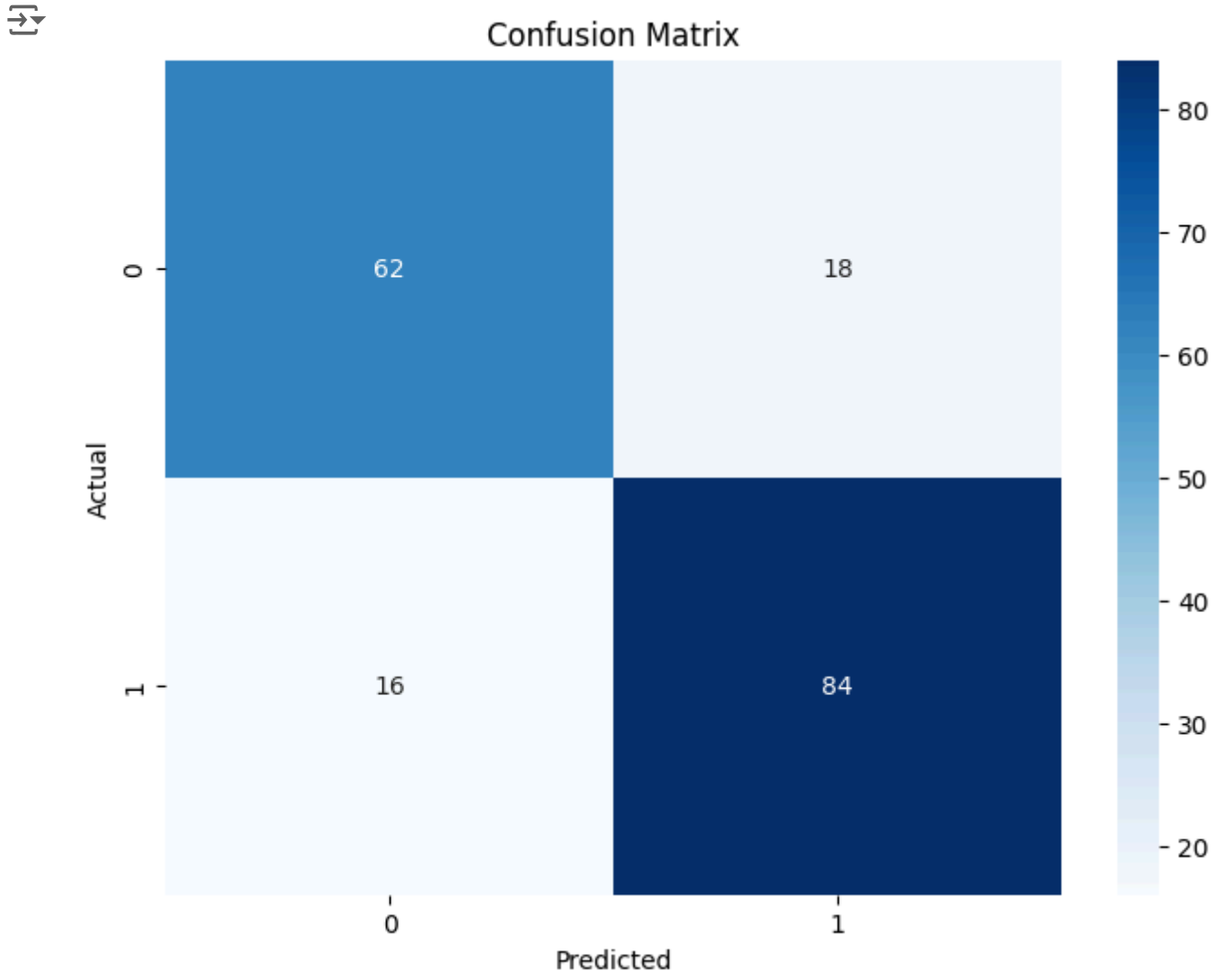
```
No artists with labels found to put in legend.  Note that artists whose label start w
```



```
from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[62 18]
 [16 84]]
```

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




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

```
              precision    recall  f1-score   support

           0       0.79      0.78      0.78        80
           1       0.82      0.84      0.83       100

    accuracy                           0.81       180
```

```
       macro avg       0.81      0.81      0.81       180
    weighted avg       0.81      0.81      0.81       180
```

Model 4 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

dt_classifier=DecisionTreeClassifier()
dt_classifier
```

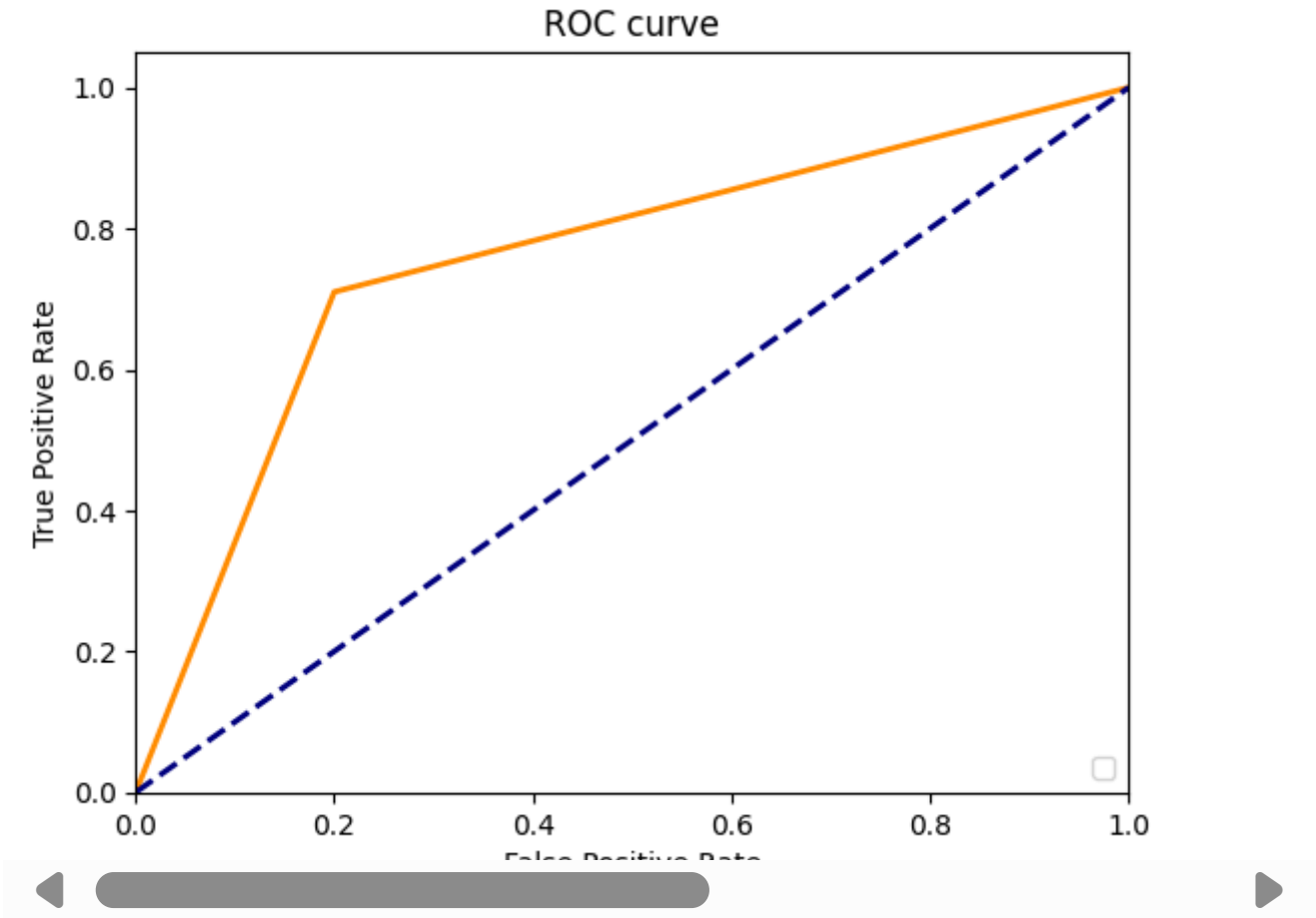
```
DecisionTreeClassifier()
```

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

```
0.75
```

```
#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()
```

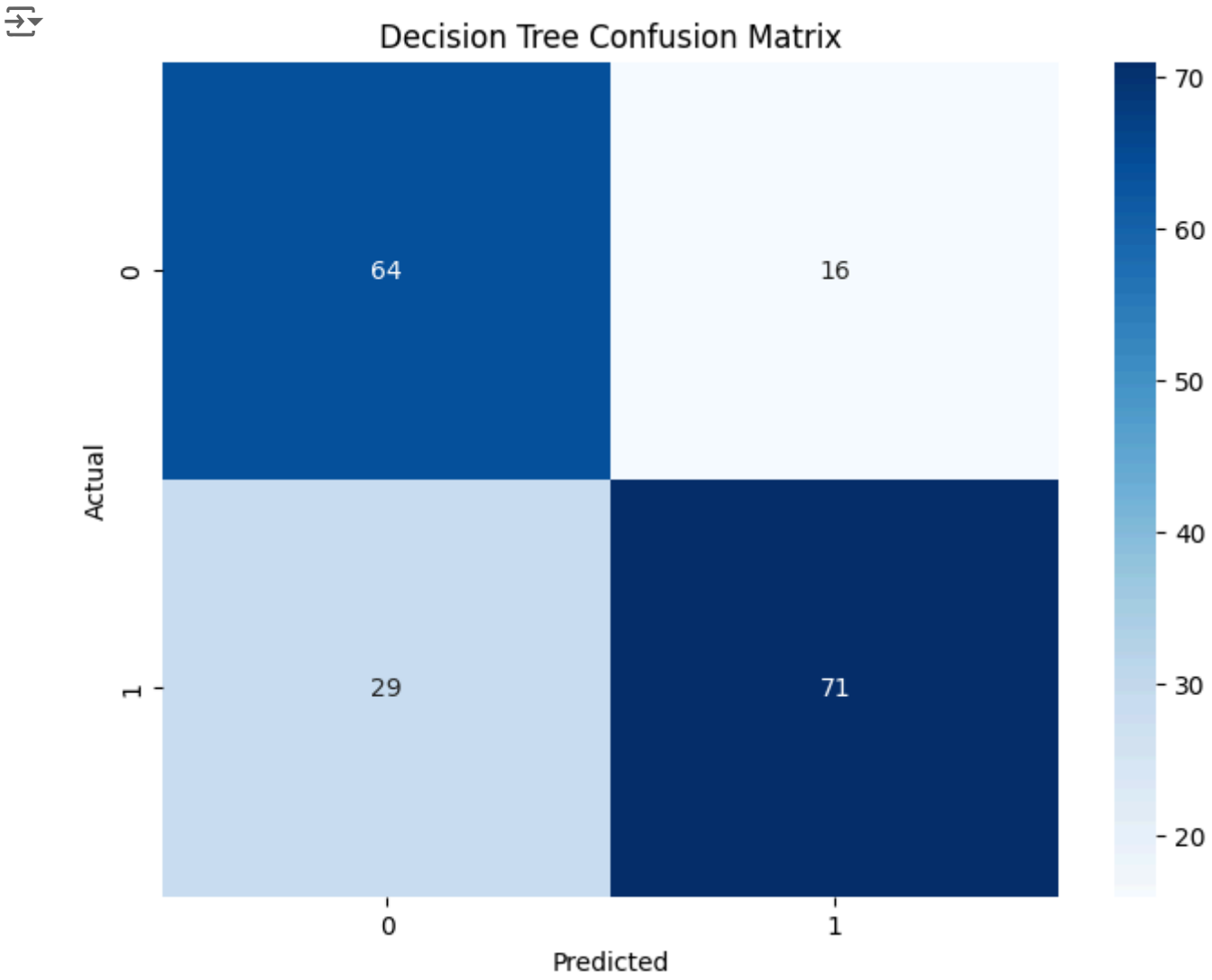
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#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:
 [[64 16]
 [29 71]]
```

```
#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       0.69      0.80      0.74        80
           1       0.82      0.71      0.76       100

    accuracy                           0.75       180
   macro avg       0.75      0.76      0.75       180
weighted avg       0.76      0.75      0.75       180
```

Model 5 RandomForest

```
from sklearn.ensemble import RandomForestClassifier

rf_classifier=RandomForestClassifier()
rf_classifier
```

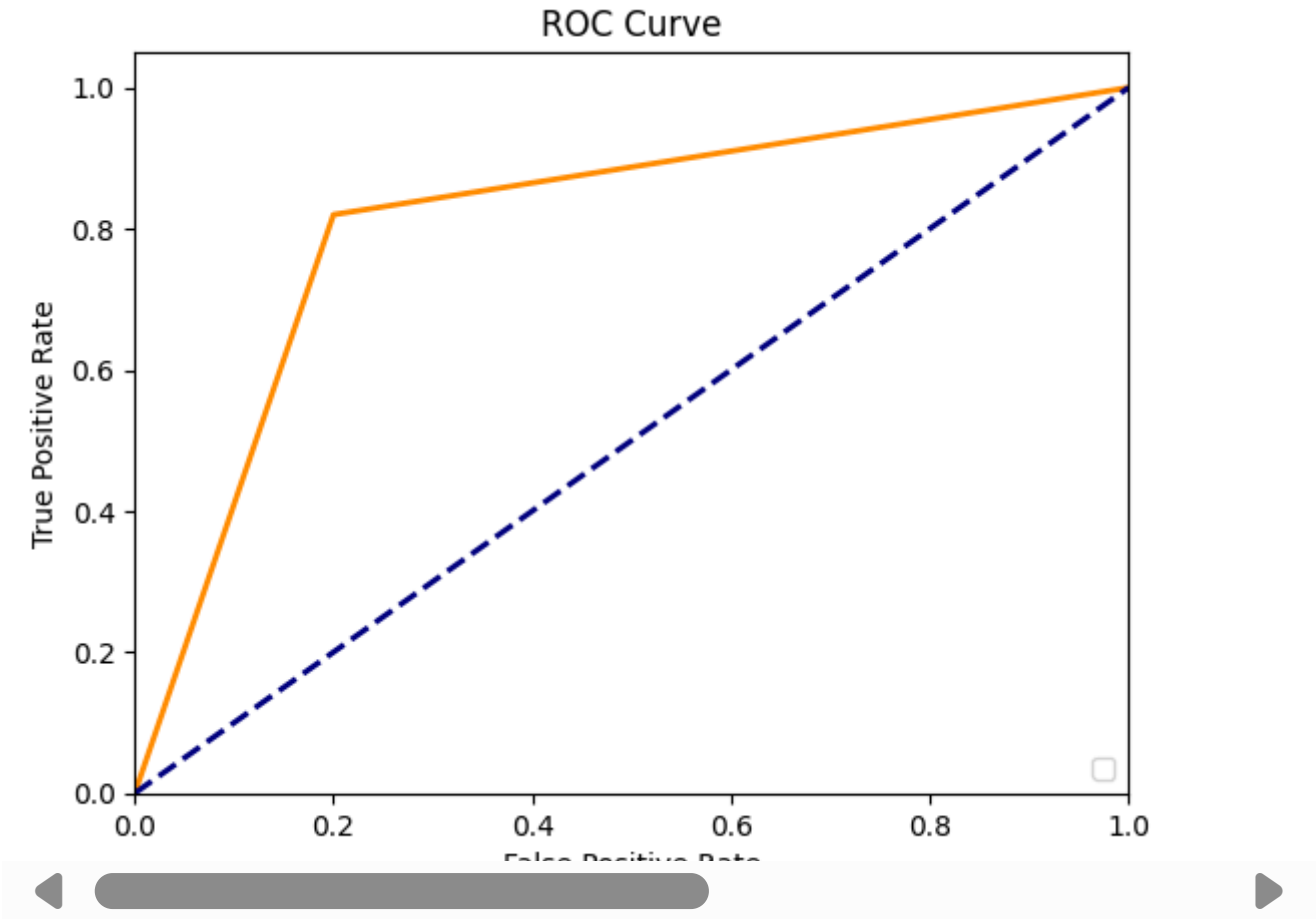
```
RandomForestClassifier
RandomForestClassifier()
```

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

```
0.8111111111111111
```

```
#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()
```

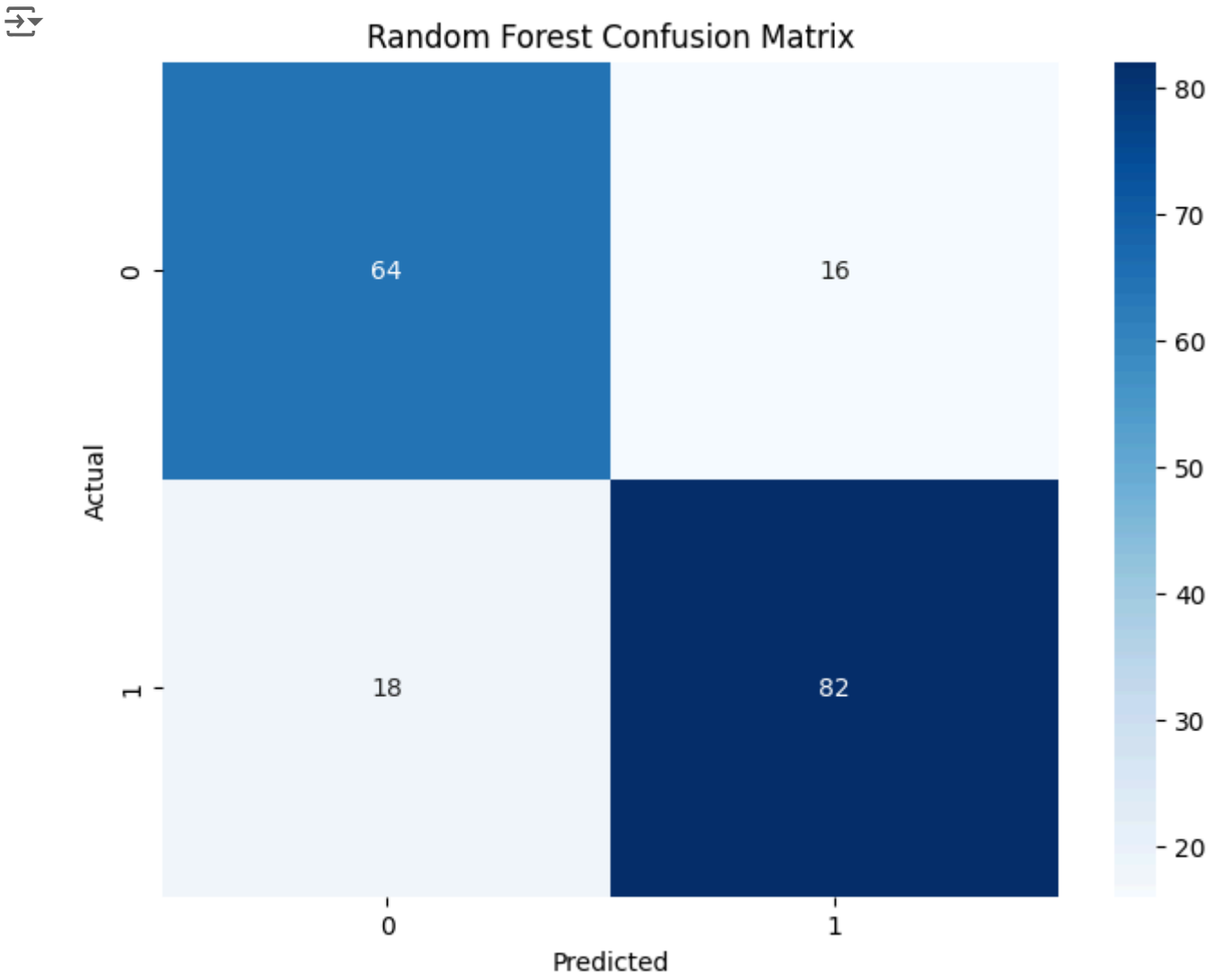
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#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:
 [[64 16]
 [18 82]]
```

```
#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       0.78      0.80      0.79        80
           1       0.84      0.82      0.83       100

    accuracy                           0.81       180
   macro avg       0.81      0.81      0.81       180
weighted avg       0.81      0.81      0.81       180
```

Model 6 KNeighbors

```
from sklearn.neighbors import KNeighborsClassifier

knn_classifier=KNeighborsClassifier()
knn_classifier
```

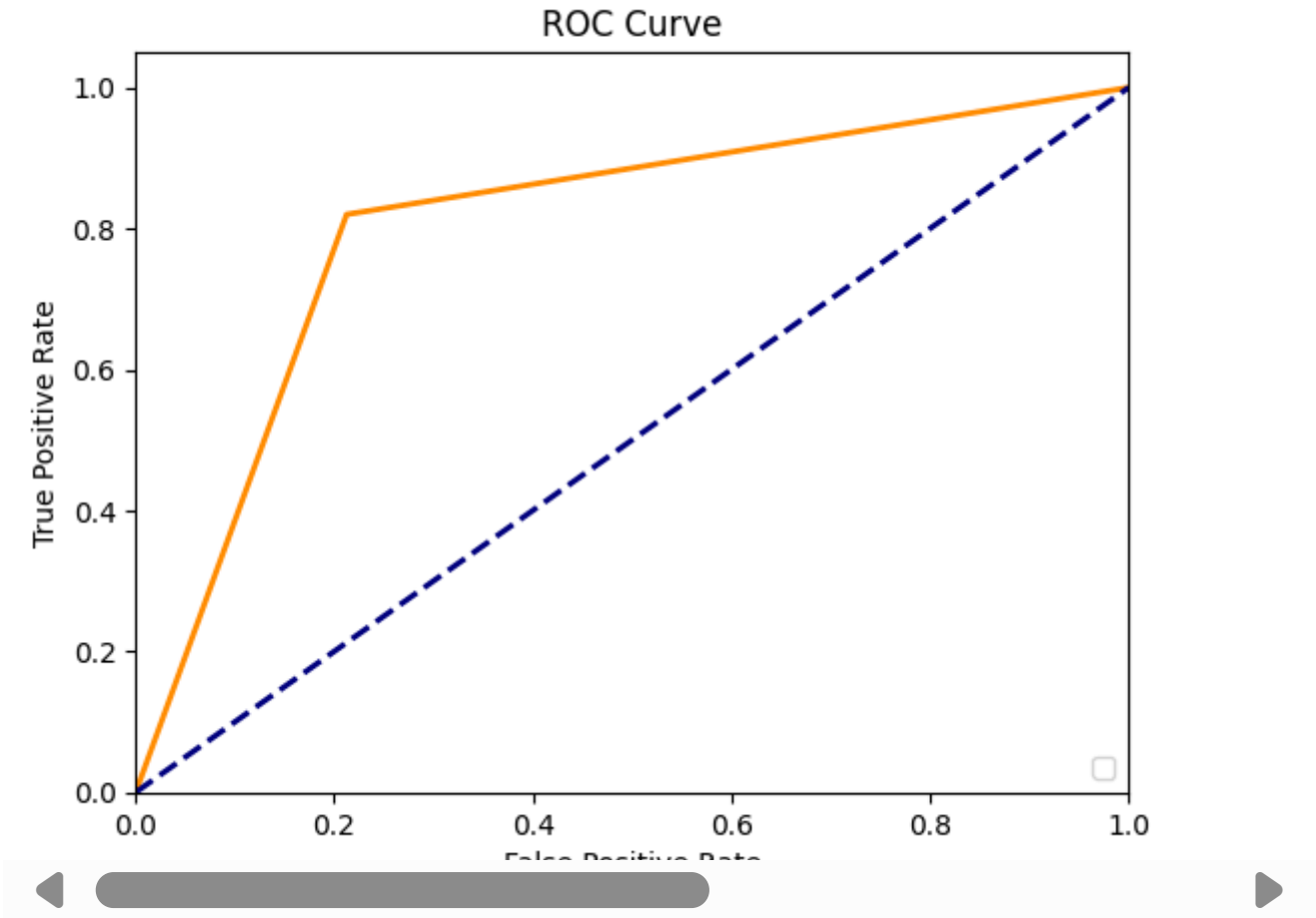
```
KNeighborsClassifier()
```

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

```
0.8055555555555556
```

```
#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()
```

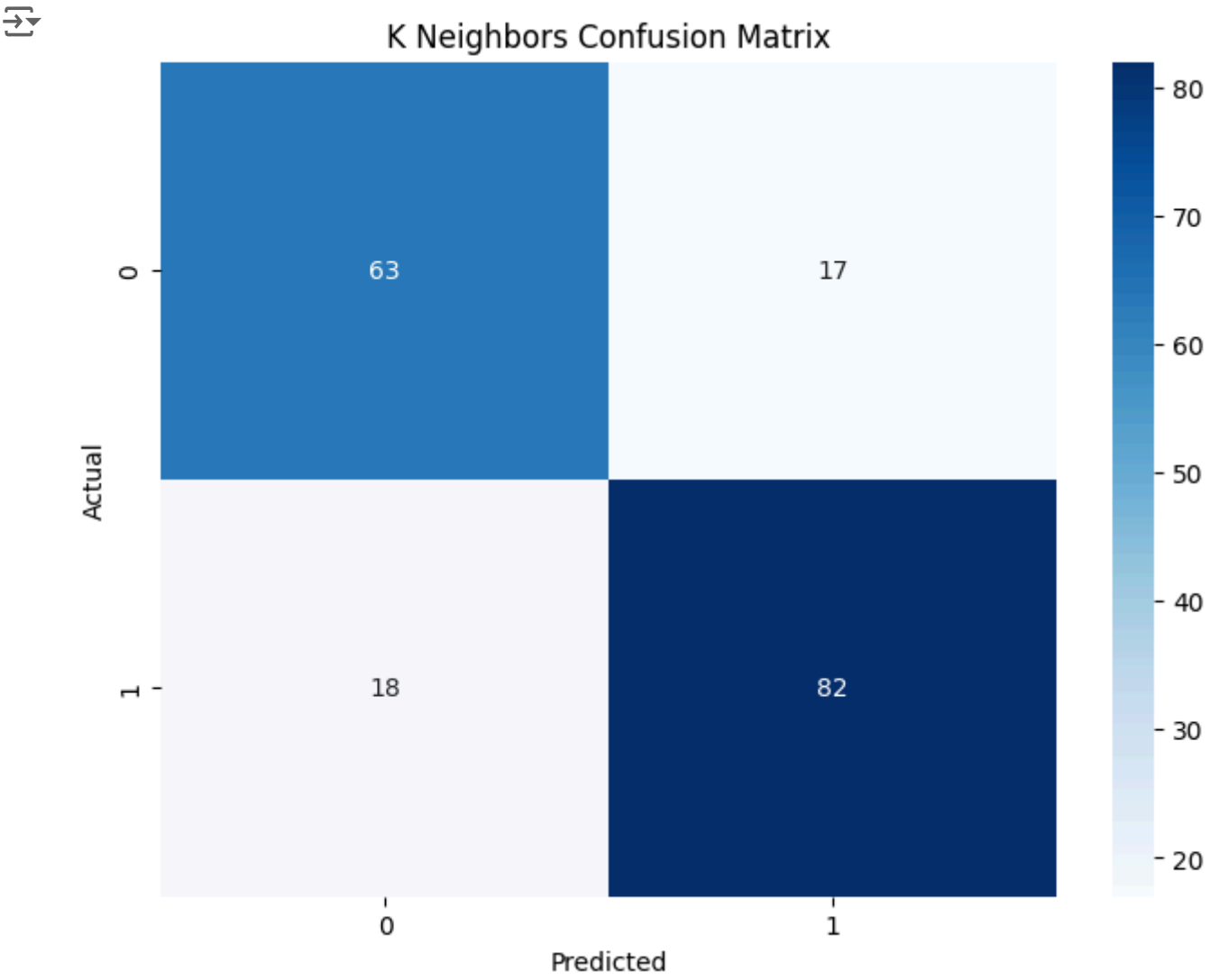
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




```
#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:
 [[63 17]
 [18 82]]
```

```
#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       0.78      0.79      0.78        80
           1       0.83      0.82      0.82       100

    accuracy                           0.81       180
   macro avg       0.80      0.80      0.80       180
weighted avg       0.81      0.81      0.81       180
```

Model 7 PLA

```
from sklearn.linear_model import Perceptron

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

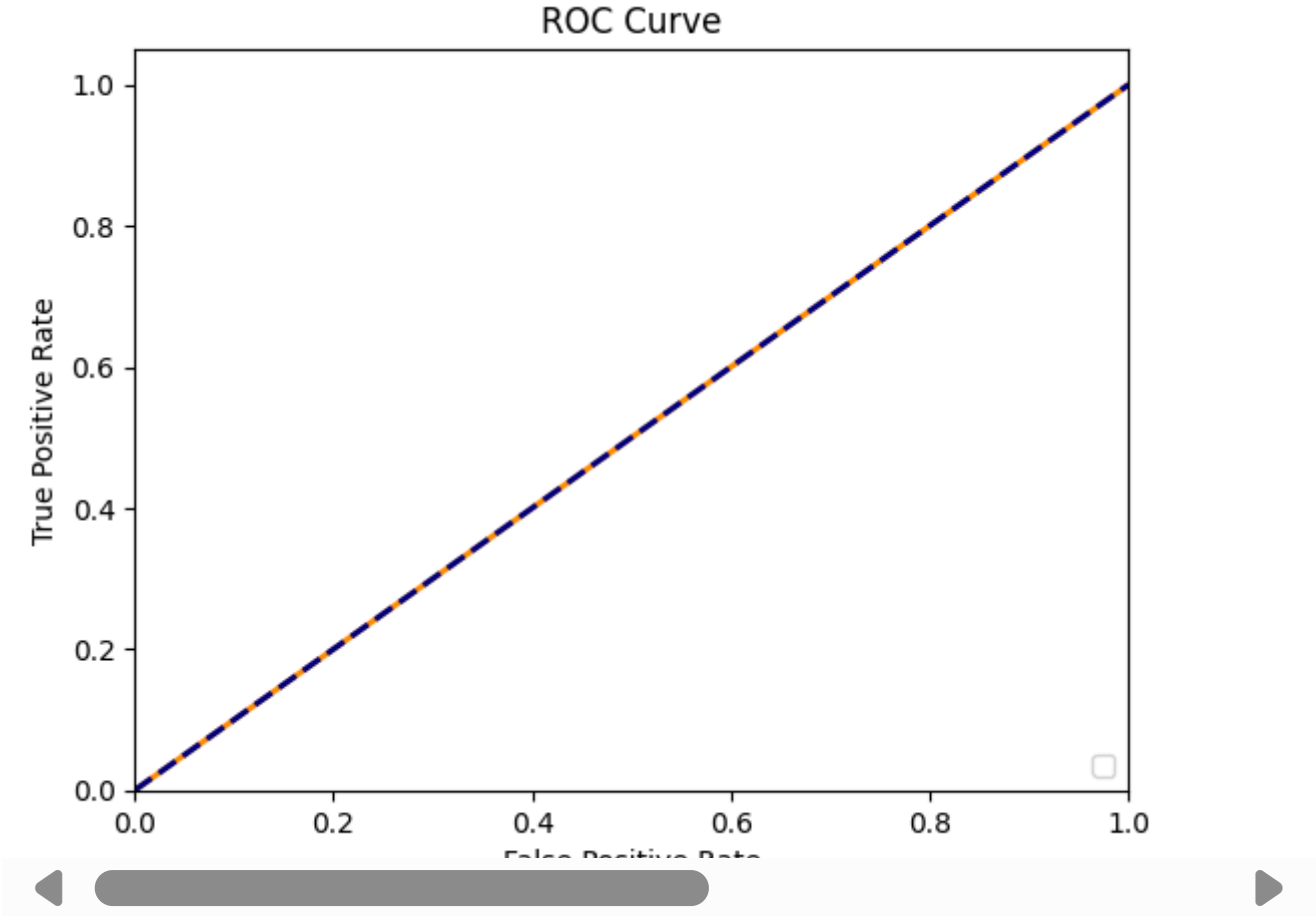
```
Perceptron
Perceptron(random_state=42)
```

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

```
0.4444444444444444
```

```
#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()
```

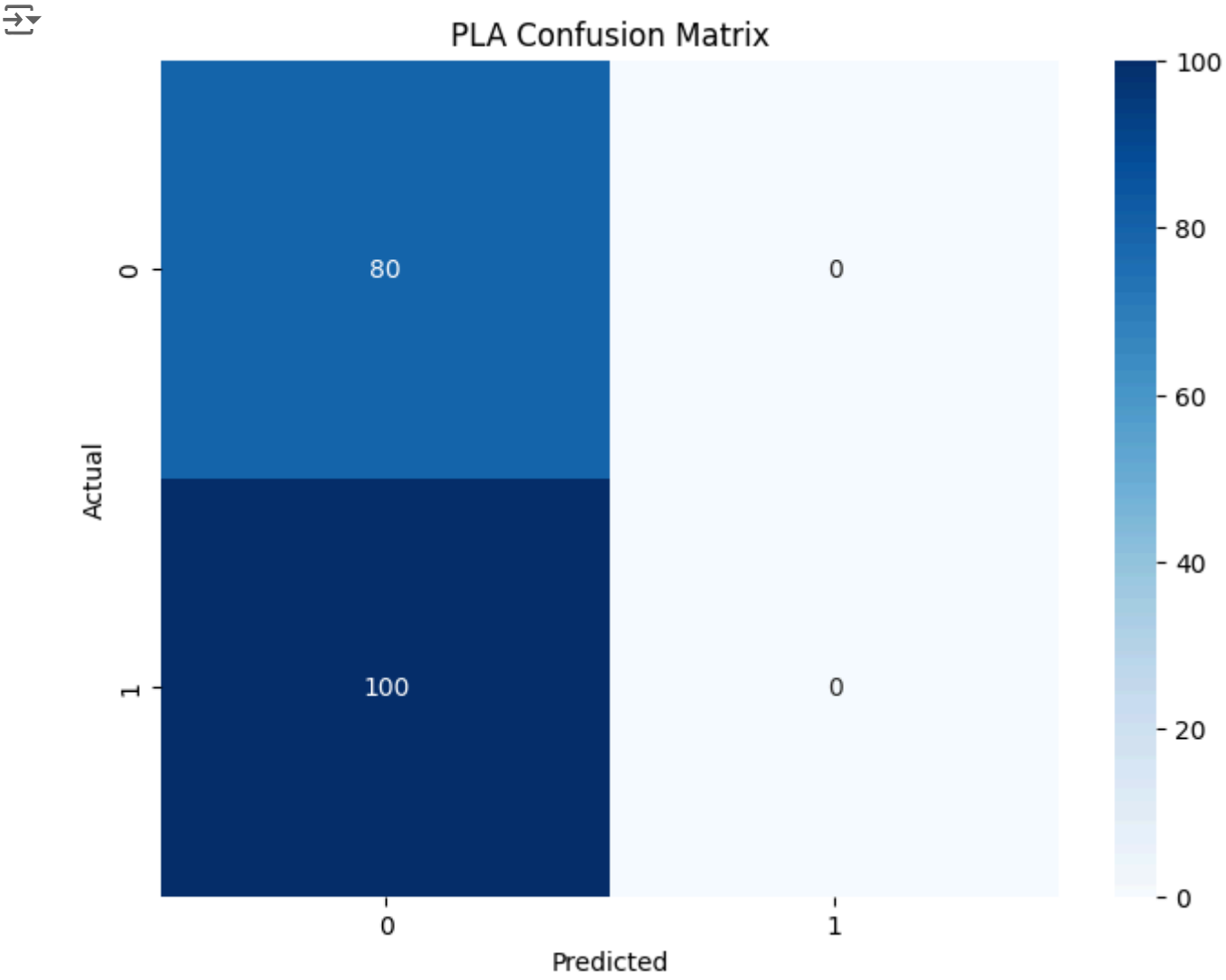
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




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

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

```
PLA Matrix:
 [[ 80   0]
 [100   0]]
```

```
#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       0.44      1.00      0.62        80
           1       0.00      0.00      0.00       100

    accuracy                           0.44       180
   macro avg       0.22      0.50      0.31       180
weighted avg       0.20      0.44      0.27       180

c:\Python310\lib\site-packages\sklearn\metrics\_classification.py:1509: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Python310\lib\site-packages\sklearn\metrics\_classification.py:1509: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Python310\lib\site-packages\sklearn\metrics\_classification.py:1509: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Model 8 MLP

```
from sklearn.neural_network import MLPClassifier

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

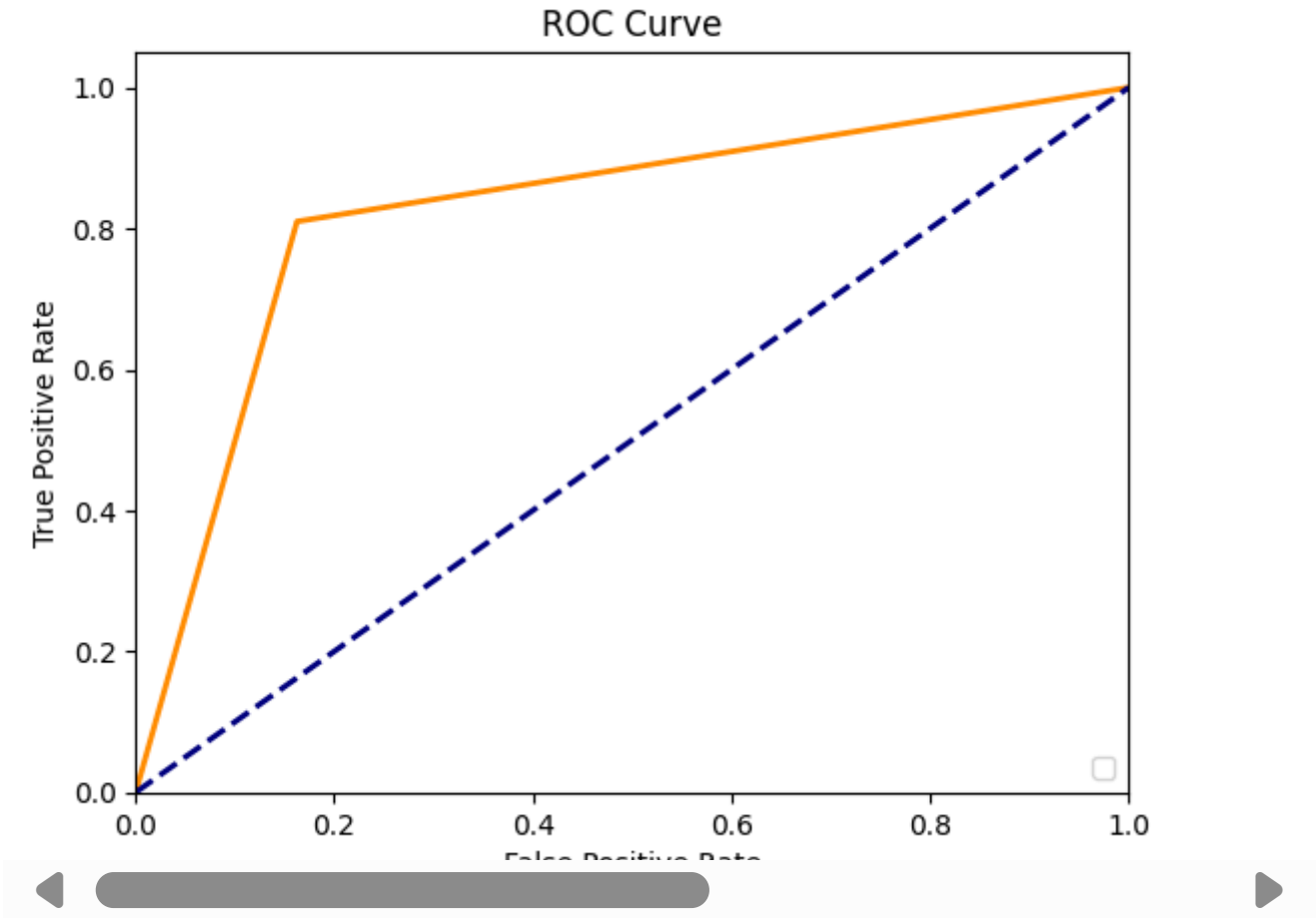
```
MLPClassifier
MLPClassifier(max_iter=300, random_state=42)
```

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

```
0.8222222222222222
```

```
#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()
```

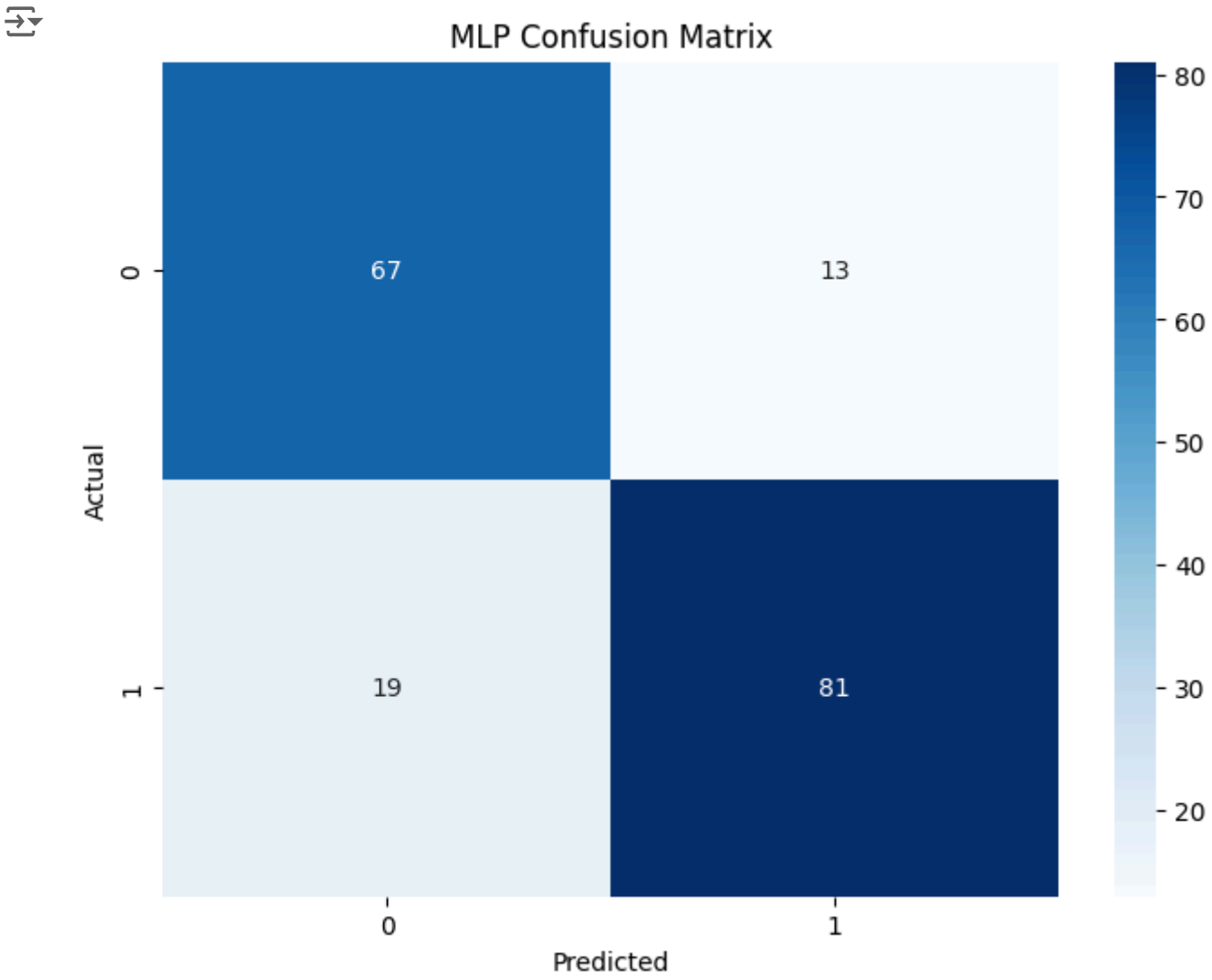
```
No artists with labels found to put in legend.  Note that artists whose label start w
```




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

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

```
MLP Matrix:
 [[67 13]
 [19 81]]
```

```
#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()
```

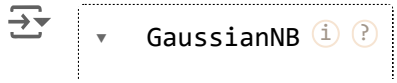



```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.84      0.81        80
           1       0.86      0.81      0.84       100

    accuracy                           0.82       180
   macro avg       0.82      0.82      0.82       180
weighted avg       0.82      0.82      0.82       180
```

Model 9 Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

nb_classifier=GaussianNB()
nb_classifier
```




```
nb_classifier.fit(X_train,y_train)
```