JUNAID GIRKAR
60004190057
TE COMPS A4

DMW
EXPERIMENT 2

DATASET 1: Diabetes

CODE:

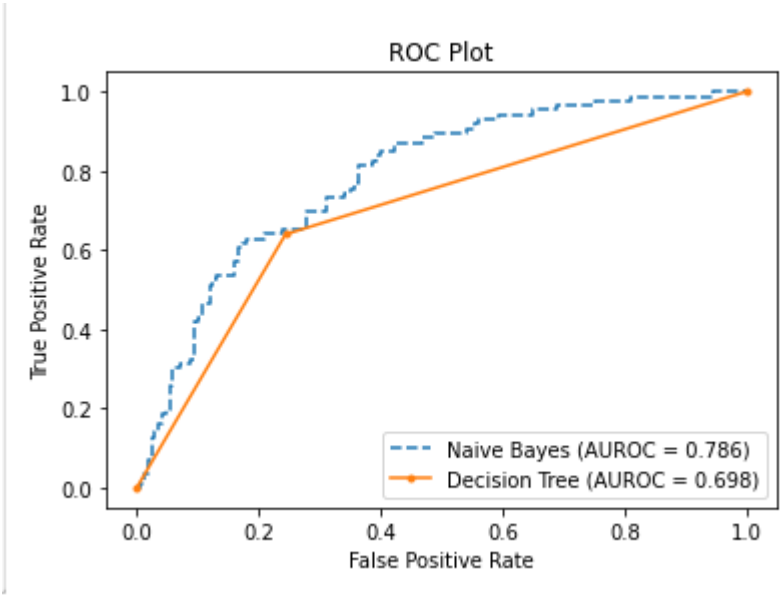
```
import pandas as pd
df = pd.read_csv('diabetes.csv')

df.head()

df.isnull().sum()

from sklearn.model_selection import train_test_split

X=df.drop(columns=['Outcome'])
y=df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print("NAIVE BAYERS CLASSIFICATION")

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()

nb.fit(X_train,y_train)

nb.score(X_test,y_test)

y_pred = nb.predict(X_test)

from sklearn.metrics import confusion_matrix,classification_report

print("Confusion Matrix")
confusion_matrix(y_test,y_pred)

print("Classification Report")
print(classification_report(y_test,y_pred))

X=df.drop(columns=['Outcome'])
```

```
y=df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

from sklearn import tree
dt = tree.DecisionTreeClassifier()

print("\nDECISION TREE CLASSIFICATION")
dt.fit(X_train,y_train)
print("Testing Score")
dt.score(X_test,y_test)

y_pred_dt = dt.predict(X_test)

print("Confusion Matrix")
confusion_matrix(y_test,y_pred_dt)

print("Classification Report")
print(classification_report(y_test,y_pred_dt))

nb_probs = nb.predict_proba(X_test)
dt_probs = dt.predict_proba(X_test)

dt_probs = dt_probs[:, 1]
nb_probs = nb_probs[:, 1]
nb_probs

from sklearn.metrics import roc_curve, roc_auc_score

nb_auc = roc_auc_score(y_test, nb_probs)
dt_auc = roc_auc_score(y_test, dt_probs)

print('Decision Tree AUROC = ' + str(dt_auc))
print('Naive Bayes AUROC = ' + str(nb_auc))

nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)

import matplotlib.pyplot as plt

plt.plot(nb_fpr, nb_tpr, linestyle='--', label='Naive Bayes (AUROC =
%0.3f)' % nb_auc)
plt.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree (AUROC =
%0.3f)' % dt_auc)
```

```
# Title
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
```

OUTPUT:

```
NAIVE BAYERS CLASSIFICATION
Confusion Matrix
Classification Report
              precision    recall  f1-score   support

           0       0.81      0.79      0.80       168
           1       0.61      0.63      0.62        86

    accuracy                           0.74       254
   macro avg       0.71      0.71      0.71       254
weighted avg       0.74      0.74      0.74       254


DECISION TREE CLASSIFICATION
Testing Score
Confusion Matrix
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.76      0.78       168
           1       0.57      0.64      0.60        86

    accuracy                           0.72       254
   macro avg       0.69      0.70      0.69       254
weighted avg       0.73      0.72      0.72       254

Decision Tree AUROC = 0.6977436323366556
Naive Bayes AUROC = 0.7857834994462902
```


ROC Plot
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes (AUROC = 0.786)
Decision Tree (AUROC = 0.698)

DATASET 2: Sonar

CODE:

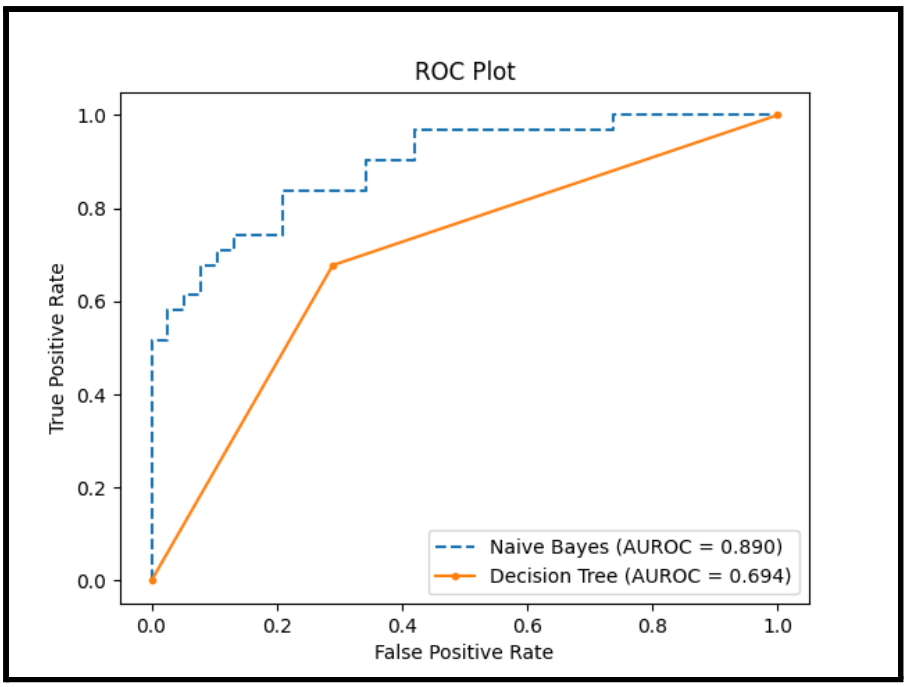
```
import pandas as pd
df = pd.read_csv('sonar.csv',header=None)
print("Showing First 5 rows of the database")
df.head()

print("Checking null fields in the dataset")
df.isnull().sum()

from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()

print("Data before using LavelEncoder")
df[60]

df[60]=le.fit_transform(df[60])
print("Data after using LavelEncoder")
df[60]


from sklearn.model_selection import train_test_split

X=df.drop(columns=[60])
y=df[60]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()

nb.fit(X_train,y_train)
print("Testing Score")
nb.score(X_test,y_test)

y_pred = nb.predict(X_test)

from sklearn.metrics import confusion_matrix,classification_report

print("Confusion Matrix for Naive Bayers")
confusion_matrix(y_test,y_pred)

print("Classification Report")
print(classification_report(y_test,y_pred))

X=df.drop(columns=[60])
```

```
y=df[60]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print("\n\n DECISION TREE CLASSIFIER")
from sklearn import tree
dt = tree.DecisionTreeClassifier()

dt.fit(X_train,y_train)
print("Testing Score")
dt.score(X_test,y_test)

y_pred_dt = dt.predict(X_test)
print("Classification Report")
print(classification_report(y_test,y_pred_dt))

print("Confusion Matrix for Decision Tree")
confusion_matrix(y_test,y_pred_dt)
nb_probs = nb.predict_proba(X_test)
dt_probs = dt.predict_proba(X_test)

dt_probs = dt_probs[:, 1]
nb_probs = nb_probs[:, 1]
nb_probs
from sklearn.metrics import roc_curve, roc_auc_score

nb_auc = roc_auc_score(y_test, nb_probs)
dt_auc = roc_auc_score(y_test, dt_probs)

print('Decision Tree AUROC = ' + str(dt_auc))
print('Naive Bayes AUROC = ' + str(nb_auc))

nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)

import matplotlib.pyplot as plt

plt.plot(nb_fpr, nb_tpr, linestyle='--', label='Naive Bayes (AUROC =
%0.3f)' % nb_auc)
plt.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree (AUROC =
%0.3f)' % dt_auc)


# Title
plt.title('ROC Plot')
# Axis labels
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
```

OUTPUT:

```
Showing First 5 rows of the database
Checking null fields in the dataset
Data before using LavelEncoder
Data after using LavelEncoder
Testing Score
Confusion Matrix for Naive Bayers
Classification Report
              precision    recall  f1-score   support

           0       0.86      0.66      0.75        38
           1       0.68      0.87      0.76        31

    accuracy                           0.75        69
   macro avg       0.77      0.76      0.75        69
weighted avg       0.78      0.75      0.75        69



 DECISION TREE CLASSIFIER
Testing Score
Classification Report
              precision    recall  f1-score   support

           0       0.76      0.82      0.78        38
           1       0.75      0.68      0.71        31

    accuracy                           0.75        69
   macro avg       0.75      0.75      0.75        69
weighted avg       0.75      0.75      0.75        69

Confusion Matrix for Decision Tree
Decision Tree AUROC = 0.74660441426146
Naive Bayes AUROC = 0.8904923599320883
```


ROC Plot
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Naive Bayes (AUROC = 0.890)
Decision Tree (AUROC = 0.694)

DATASET 3: Haberman

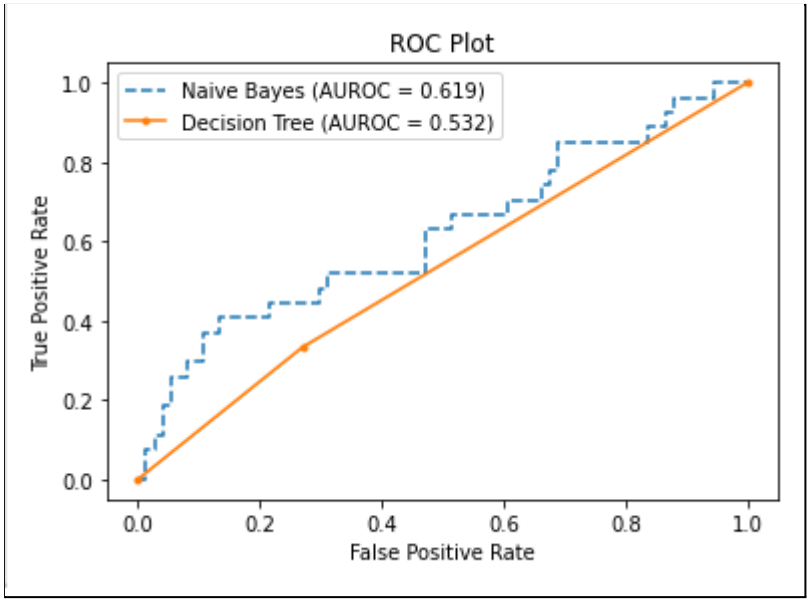
CODE:

```
import pandas as pd

df = pd.read_csv('haberman.csv',header=None)

df.head()
df.isnull().sum()

from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
df[3]

df[3]=le.fit_transform(df[3])
df[3]

from sklearn.model_selection import train_test_split

X=df.drop(columns=[3])
y=df[3]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print("NAIVE BAYERS CLASSIFICATION")

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()

nb.fit(X_train,y_train)
print("Testing Score")
nb.score(X_test,y_test)
y_pred = nb.predict(X_test)
from sklearn.metrics import confusion_matrix,classification_report

print("Naive Bayers Confusion Matrix")
confusion_matrix(y_test,y_pred)
print("Classification Report")
print(classification_report(y_test,y_pred))

X=df.drop(columns=[3])
y=df[3]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)
print("DECISION TREE CLASSIFIER")
from sklearn import tree
dt = tree.DecisionTreeClassifier()
```

```
dt.fit(X_train,y_train)

print("Testing Score")
dt.score(X_test,y_test)

y_pred_dt = dt.predict(X_test)

print(classification_report(y_test,y_pred_dt))
confusion_matrix(y_test,y_pred_dt)

nb_probs = nb.predict_proba(X_test)
dt_probs = dt.predict_proba(X_test)

dt_probs = dt_probs[:, 1]
nb_probs = nb_probs[:, 1]
nb_probs

from sklearn.metrics import roc_curve, roc_auc_score

nb_auc = roc_auc_score(y_test, nb_probs)
dt_auc = roc_auc_score(y_test, dt_probs)

print('Decision Tree AUROC = ' + str(dt_auc))
print('Naive Bayes AUROC = ' + str(nb_auc))

nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)

import matplotlib.pyplot as plt

plt.plot(nb_fpr, nb_tpr, linestyle='--', label='Naive Bayes (AUROC =
%0.3f)' % nb_auc)
plt.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree (AUROC =
%0.3f)' % dt_auc)


# Title
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
```

OUTPUT:

```
NAIVE BAYERS CLASSIFICATION
Testing Score
Naive Bayers Confusion Matrix
Classification Report
              precision    recall  f1-score   support

           0       0.78      0.91      0.84        74
           1       0.53      0.30      0.38        27

    accuracy                           0.74       101
   macro avg       0.66      0.60      0.61       101
weighted avg       0.71      0.74      0.72       101

DECISION TREE CLASSIFIER
Testing Score
              precision    recall  f1-score   support

           0       0.75      0.73      0.74        74
           1       0.31      0.33      0.32        27

    accuracy                           0.62       101
   macro avg       0.53      0.53      0.53       101
weighted avg       0.63      0.62      0.63       101

Decision Tree AUROC = 0.5315315315315315
Naive Bayes AUROC = 0.6191191191191191
```



DATASET 4: Ionosphere

CODE:

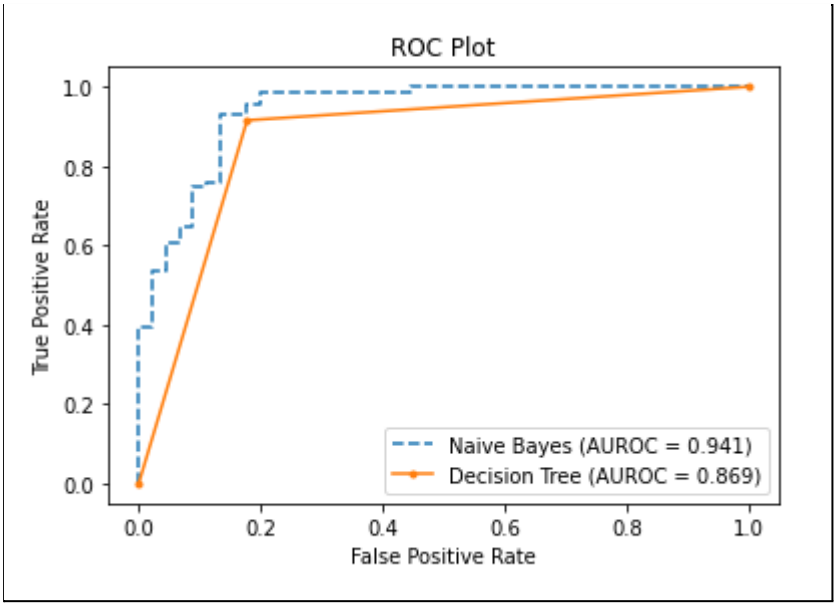
```
import pandas as pd

df = pd.read_csv('ionosphere_data.csv')

df.head()

df.isnull().sum()

from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
df['column_ai']

df['column_ai']=le.fit_transform(df['column_ai'])
df['column_ai']

from sklearn.model_selection import train_test_split

X=df.drop(columns=['column_ai'])
y=df['column_ai']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print("NAIVE BAYERS CLASSIFICATION\n")

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()

nb.fit(X_train,y_train)

print("Naive bayers Score:")
nb.score(X_test,y_test)

y_pred = nb.predict(X_test)

from sklearn.metrics import confusion_matrix,classification_report

print("Confusion Matrix")
confusion_matrix(y_test,y_pred)

print("Classification Report")
print(classification_report(y_test,y_pred))

X=df.drop(columns=['column_ai'])
y=df['column_ai']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

from sklearn import tree
dt = tree.DecisionTreeClassifier()

print("\n\nDECISION TREE CLASSIFIER")
dt.fit(X_train,y_train)

dt.score(X_test,y_test)

y_pred_dt = dt.predict(X_test)

print("Classification Report")
print(classification_report(y_test,y_pred_dt))

print("Confusion Matrix")
confusion_matrix(y_test,y_pred_dt)

nb_probs = nb.predict_proba(X_test)
dt_probs = dt.predict_proba(X_test)

dt_probs = dt_probs[:, 1]
nb_probs = nb_probs[:, 1]
nb_probs

from sklearn.metrics import roc_curve, roc_auc_score

nb_auc = roc_auc_score(y_test, nb_probs)
dt_auc = roc_auc_score(y_test, dt_probs)

print('Decision Tree AUROC = ' + str(dt_auc))
print('Naive Bayes AUROC = ' + str(nb_auc))

nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)

import matplotlib.pyplot as plt

plt.plot(nb_fpr, nb_tpr, linestyle='--', label='Naive Bayes (AUROC =
%0.3f)' % nb_auc)
plt.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree (AUROC =
%0.3f)' % dt_auc)


# Title
```

```
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
```

OUTPUT:

```
NAIVE BAYERS CLASSIFICATION

Naive bayers Score:
Confusion Matrix
Classification Report
              precision    recall  f1-score   support

           0       0.97      0.78      0.86        45
           1       0.88      0.99      0.93        71

    accuracy                           0.91       116
   macro avg       0.92      0.88      0.90       116
weighted avg       0.91      0.91      0.90       116



DECISION TREE CLASSIFIER
Classification Report
              precision    recall  f1-score   support

           0       0.86      0.82      0.84        45
           1       0.89      0.92      0.90        71

    accuracy                           0.88       116
   macro avg       0.88      0.87      0.87       116
weighted avg       0.88      0.88      0.88       116

Confusion Matrix
Decision Tree AUROC = 0.8688575899843505
Naive Bayes AUROC = 0.9411580594679188
```



DATASET 5: BankNote Authentication

CODE:

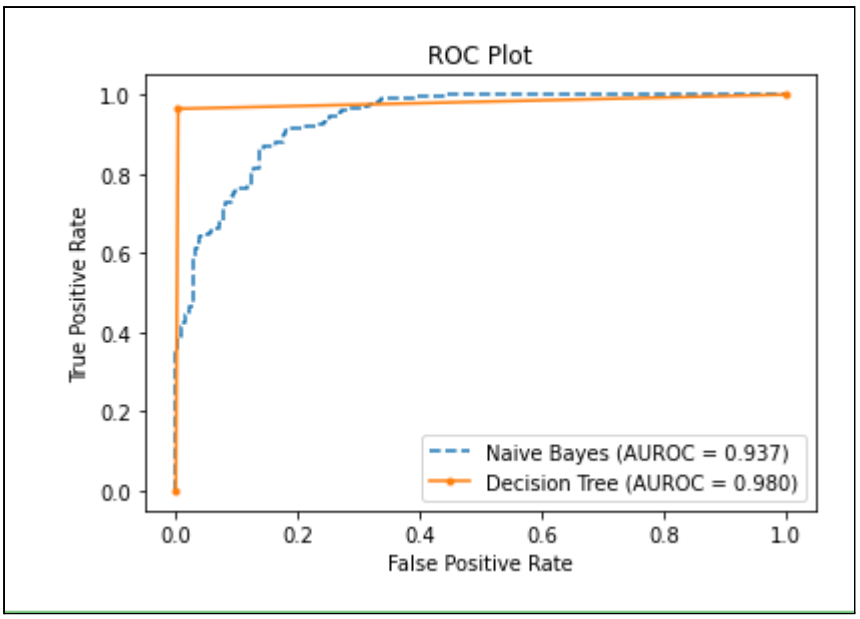
```
import pandas as pd

df = pd.read_csv('BankNoteAuthentication.csv')

df.head()

df.isnull().sum()

from sklearn.model_selection import train_test_split

X=df.drop(columns=['class'])
y=df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print("NAIVE BAYERS CLASSFICATION")

from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()

nb.fit(X_train,y_train)

print("TESTING SCORE")
nb.score(X_test,y_test)

y_pred = nb.predict(X_test)

from sklearn.metrics import confusion_matrix,classification_report

print("CONFUSION MATRIX")
confusion_matrix(y_test,y_pred)

print("CLASSIFICATION REPORT")
print(classification_report(y_test,y_pred))

X=df.drop(columns=['class'])
y=df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=42)

print("\nDECISION TREE CLASSIFIER")
from sklearn import tree
dt = tree.DecisionTreeClassifier()
```

```
dt.fit(X_train,y_train)

print("Testing Score")
dt.score(X_test,y_test)

y_pred_dt = dt.predict(X_test)

print("Confusion Matrix")
confusion_matrix(y_test,y_pred_dt)

print("Classification Report")
print(classification_report(y_test,y_pred_dt))

nb_probs = nb.predict_proba(X_test)
dt_probs = dt.predict_proba(X_test)

dt_probs = dt_probs[:, 1]
nb_probs = nb_probs[:, 1]
nb_probs

from sklearn.metrics import roc_curve, roc_auc_score

nb_auc = roc_auc_score(y_test, nb_probs)
dt_auc = roc_auc_score(y_test, dt_probs)

print('Decision Tree AUROC = ' + str(dt_auc))
print('Naive Bayes AUROC = ' + str(nb_auc))

nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_probs)

import matplotlib.pyplot as plt

plt.plot(nb_fpr, nb_tpr, linestyle='--', label='Naive Bayes (AUROC =
%0.3f)' % nb_auc)
plt.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree (AUROC =
%0.3f)' % dt_auc)


# Title
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
```

```
# Show plot
plt.show()
```

OUTPUT:

```
NAIVE BAYERS CLASSFICATION
TESTING SCORE
CONFUSION MATRIX
CLASSIFICATION REPORT
              precision    recall  f1-score   support

           0       0.83      0.91      0.86       257
           1       0.86      0.75      0.80       196

    accuracy                           0.84       453
   macro avg       0.84      0.83      0.83       453
weighted avg       0.84      0.84      0.84       453


DECISION TREE CLASSIFIER
Testing Score
Confusion Matrix
Classification Report
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       257
           1       0.99      0.96      0.98       196

    accuracy                           0.98       453
   macro avg       0.98      0.98      0.98       453
weighted avg       0.98      0.98      0.98       453

Decision Tree AUROC = 0.9801973318510284
Naive Bayes AUROC = 0.9371476216945922
```



Comparison:

CODE:

```
import numpy as np
import matplotlib.pyplot as plt

barWidth = 0.25
fig = plt.subplots(figsize =(12, 8))

naive_bayes = [79, 89, 93, 94, 62]
decision_tree = [69,74, 98, 83, 50]

br1 = np.arange(len(naive_bayes))
br2 = [x + barWidth for x in br1]

plt.bar(br1, naive_bayes, color ='b', width = barWidth,
        edgecolor ='grey', label ='Naive Bayes')
plt.bar(br2, decision_tree, color ='y', width = barWidth,
        edgecolor ='grey', label ='Decision Tree')
plt.xlabel('Datasets', fontweight ='bold', fontsize = 15)
plt.ylabel('Percentage', fontweight ='bold', fontsize = 15)
plt.xticks([r+ barWidth for r in range(len(naive_bayes))],
        ['diabeties.csv', 'mines_rock.csv',
'BankNoteAuthentication.csv', 'ionosphere_data.csv',
'haberman.csv'],rotation=30)

plt.legend()
```

```
plt.show()
```

OUTPUT: