Build a Naive Bayes and KNN classifier

In [1]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Load your dataset
data = pd.read_csv("C:/Users/jagad/Downloads/spam_dataset.csv")

# Split the data into features (X) and the target variable (y)
X = data.drop('spam', axis=1)  # Features
y = data['spam']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# Naive Bayes Classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)
nb_pred = nb_classifier.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_pred)

# K-Nearest Neighbors (KNN) Classifier
knn_classifier = KNeighborsClassifier(n_neighbors=1)  # You can adjus
knn_classifier.fit(X_train, y_train)
knn_pred = knn_classifier.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_pred)

# Print the accuracy of both classifiers
print("Naive Bayes Accuracy: {:.2f}%".format(nb_accuracy * 100))
print("K-Nearest Neighbors Accuracy: {:.2f}%".format(knn_accuracy * 1

# Create confusion matrices for both classifiers
nb_confusion = confusion_matrix(y_test, nb_pred)
knn_confusion = confusion_matrix(y_test, knn_pred)

# Print the confusion matrices
print("Naive Bayes Confusion Matrix:")
print(nb_confusion)

print("\nK-Nearest Neighbors Confusion Matrix:")
print(knn_confusion)
```

```
Naive Bayes Accuracy: 40.00%
K-Nearest Neighbors Accuracy: 40.00%
Naive Bayes Confusion Matrix:
[[4 1]
 [5 0]]

K-Nearest Neighbors Confusion Matrix:
[[3 2]
 [4 1]]
```

In [2]:

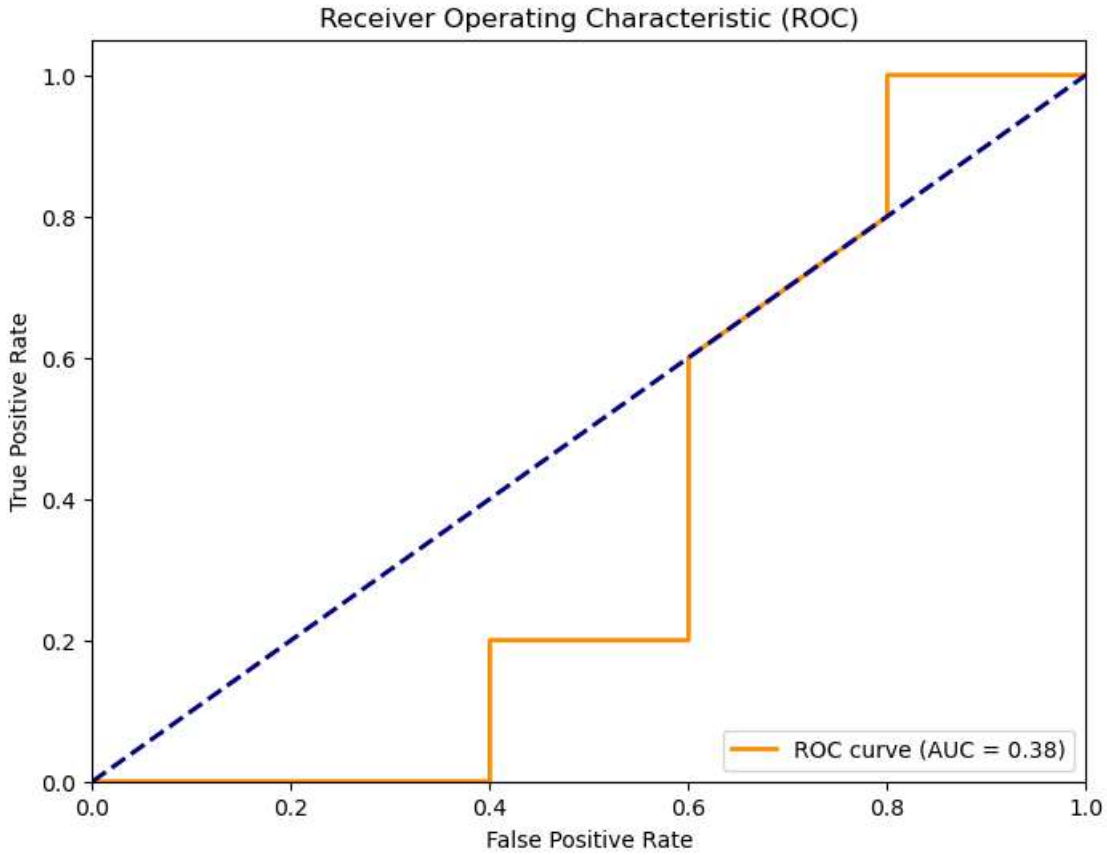
```
from sklearn.metrics import classification_report  # Import classific
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# Naive Bayes Classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)
nb_pred = nb_classifier.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_pred)

# K-Nearest Neighbors (KNN) Classifier
knn_classifier = KNeighborsClassifier(n_neighbors=1)  # You can adjus
knn_classifier.fit(X_train, y_train)
knn_pred = knn_classifier.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_pred)

# Print the accuracy of both classifiers
print("Naive Bayes Accuracy: {:.2f}%".format(nb_accuracy * 100))
print("K-Nearest Neighbors Accuracy: {:.2f}%".format(knn_accuracy * 1

# Create confusion matrices for both classifiers
nb_confusion = confusion_matrix(y_test, nb_pred)
knn_confusion = confusion_matrix(y_test, knn_pred)

# Initialize and train the Naive Bayes classifier
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train,y_train)

# Predictions and evaluation for Naive Bayes
naive_bayes_predictions = naive_bayes.predict(X_test)
naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions
print(f'Naive Bayes Accuracy: {naive_bayes_accuracy}')
print(classification_report(y_test, naive_bayes_predictions))

# Initialize and train the Naive Bayes classifier
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)

# Predictions and evaluation for Naive Bayes
naive_bayes_predictions = naive_bayes.predict(X_test)
naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions
print(f'Naive Bayes Accuracy: {naive_bayes_accuracy}')

# Use classification_report for precision, recall, F1-score, and suppo
print(classification_report(y_test, naive_bayes_predictions))

```

```
Naive Bayes Accuracy: 40.00%
K-Nearest Neighbors Accuracy: 40.00%
Naive Bayes Accuracy: 0.4
              precision    recall  f1-score   support

           0       0.44      0.80      0.57         5
           1       0.00      0.00      0.00         5

    accuracy                           0.40        10
   macro avg       0.22      0.40      0.29        10
weighted avg       0.22      0.40      0.29        10

Naive Bayes Accuracy: 0.4
              precision    recall  f1-score   support

           0       0.44      0.80      0.57         5
           1       0.00      0.00      0.00         5

    accuracy                           0.40        10
   macro avg       0.22      0.40      0.29        10
weighted avg       0.22      0.40      0.29        10
```

ROC curve and AUC score (NB & Decision Tree)

In [3]:

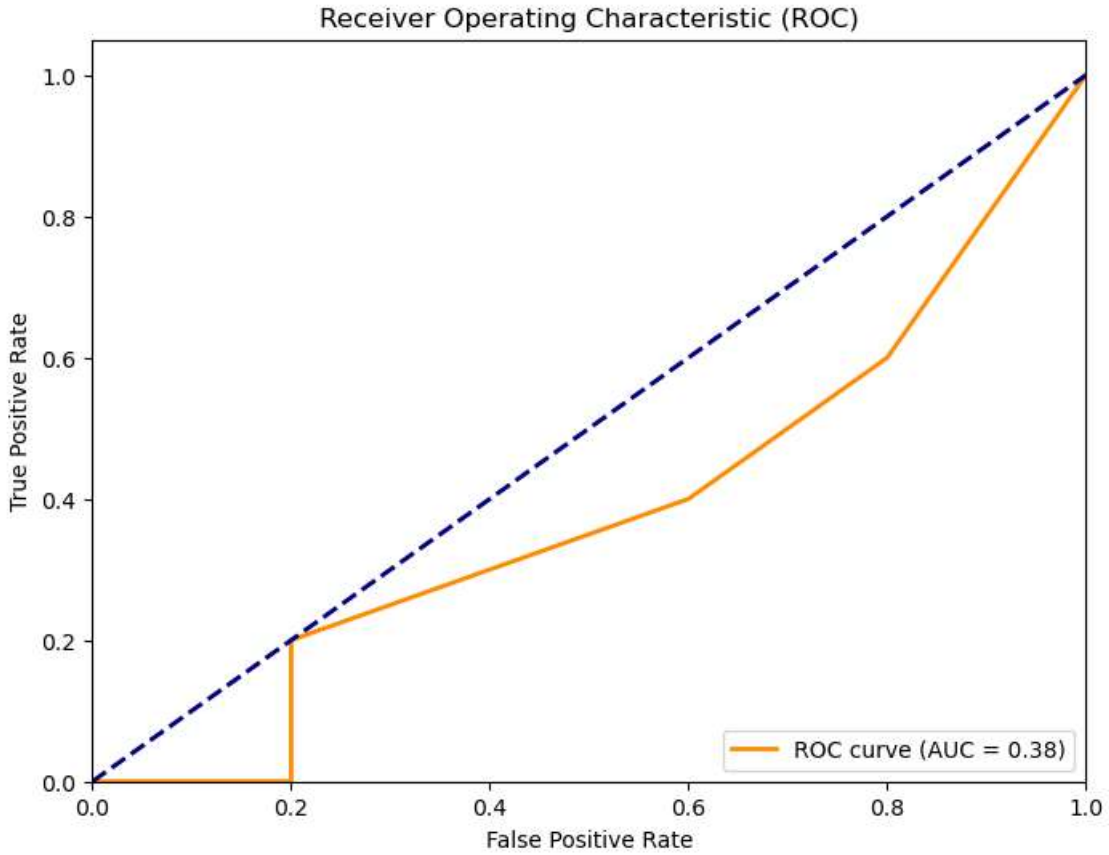
```
from sklearn.metrics import roc_curve, roc_auc_score, auc
import matplotlib.pyplot as plt


# Split the data into features (X) and the target variable (y)
X = data.iloc[:, :-1]  # All columns except the last 'spam' column
y = data['spam']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# Naive Bayes Classifier
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train, y_train)

# Predict probabilities of being spam
y_prob = nb_classifier.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)

# Calculate AUC score
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
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()

# Print AUC score
print(f'AUC Score: {roc_auc:.2f}')

```




```
AUC Score: 0.38
```

In [4]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score, auc
import matplotlib.pyplot as plt

# Split the data into features (X) and the target variable (y)
X = data.iloc[:, :-1]  # All columns except the last 'spam' column
y = data['spam']  # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

# Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)

# Predict probabilities of being spam
y_prob = dt_classifier.predict_proba(X_test)[:, 1]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_prob)

# Calculate AUC score
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
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()

# Print AUC score
print(f'AUC Score: {roc_auc:.2f}')

```




```
AUC Score: 0.38
```

In [ ]:

```
1
```