

# Build a Naive Bayes and KNN classifier

```
In [1]: 1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from sklearn.naive_bayes import MultinomialNB
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.metrics import accuracy_score, confusion_matrix
7
8 # Load your dataset
9 data = pd.read_csv("C:/Users/jagad/Downloads/spam_dataset.csv")
10
11 # Split the data into features (X) and the target variable (y)
12 X = data.drop('spam', axis=1) # Features
13 y = data['spam'] # Target variable
14
15 # Split the data into training and testing sets
16 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
17
18 # Naive Bayes Classifier
19 nb_classifier = MultinomialNB()
20 nb_classifier.fit(X_train, y_train)
21 nb_pred = nb_classifier.predict(X_test)
22 nb_accuracy = accuracy_score(y_test, nb_pred)
23
24 # K-Nearest Neighbors (KNN) Classifier
25 knn_classifier = KNeighborsClassifier(n_neighbors=1) # You can adjust n_neighbors
26 knn_classifier.fit(X_train, y_train)
27 knn_pred = knn_classifier.predict(X_test)
28 knn_accuracy = accuracy_score(y_test, knn_pred)
29
30 # Print the accuracy of both classifiers
31 print("Naive Bayes Accuracy: {:.2f}%".format(nb_accuracy * 100))
32 print("K-Nearest Neighbors Accuracy: {:.2f}%".format(knn_accuracy * 100))
33
34 # Create confusion matrices for both classifiers
35 nb_confusion = confusion_matrix(y_test, nb_pred)
36 knn_confusion = confusion_matrix(y_test, knn_pred)
37
38 # Print the confusion matrices
39 print("Naive Bayes Confusion Matrix:")
40 print(nb_confusion)
41
42 print("\nK-Nearest Neighbors Confusion Matrix:")
43 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]: 1 from sklearn.metrics import classification_report # Import classification_report
2 from sklearn.neighbors import KNeighborsClassifier
3 from sklearn.metrics import accuracy_score, confusion_matrix
4
5 # Split the data into training and testing sets
6 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
7
8 # Naive Bayes Classifier
9 nb_classifier = MultinomialNB()
10 nb_classifier.fit(X_train, y_train)
11 nb_pred = nb_classifier.predict(X_test)
12 nb_accuracy = accuracy_score(y_test, nb_pred)
13
14 # K-Nearest Neighbors (KNN) Classifier
15 knn_classifier = KNeighborsClassifier(n_neighbors=1) # You can adjust the number of neighbors
16 knn_classifier.fit(X_train, y_train)
17 knn_pred = knn_classifier.predict(X_test)
18 knn_accuracy = accuracy_score(y_test, knn_pred)
19
20 # Print the accuracy of both classifiers
21 print("Naive Bayes Accuracy: {:.2f}%".format(nb_accuracy * 100))
22 print("K-Nearest Neighbors Accuracy: {:.2f}%".format(knn_accuracy * 100))
23
24 # Create confusion matrices for both classifiers
25 nb_confusion = confusion_matrix(y_test, nb_pred)
26 knn_confusion = confusion_matrix(y_test, knn_pred)
27
28 # Initialize and train the Naive Bayes classifier
29 naive_bayes = MultinomialNB()
30 naive_bayes.fit(X_train, y_train)
31
32 # Predictions and evaluation for Naive Bayes
33 naive_bayes_predictions = naive_bayes.predict(X_test)
34 naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions)
35 print(f'Naive Bayes Accuracy: {naive_bayes_accuracy}')
36 print(classification_report(y_test, naive_bayes_predictions))
37
38 # Initialize and train the Naive Bayes classifier
39 naive_bayes = MultinomialNB()
40 naive_bayes.fit(X_train, y_train)
41
42 # Predictions and evaluation for Naive Bayes
43 naive_bayes_predictions = naive_bayes.predict(X_test)
44 naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions)
45 print(f'Naive Bayes Accuracy: {naive_bayes_accuracy}')
46
47 # Use classification_report for precision, recall, F1-score, and support
48 print(classification_report(y_test, naive_bayes_predictions))
49
```

Naive Bayes Accuracy: 40.00%

K-Nearest Neighbors Accuracy: 40.00%

Naive Bayes Accuracy: 0.4

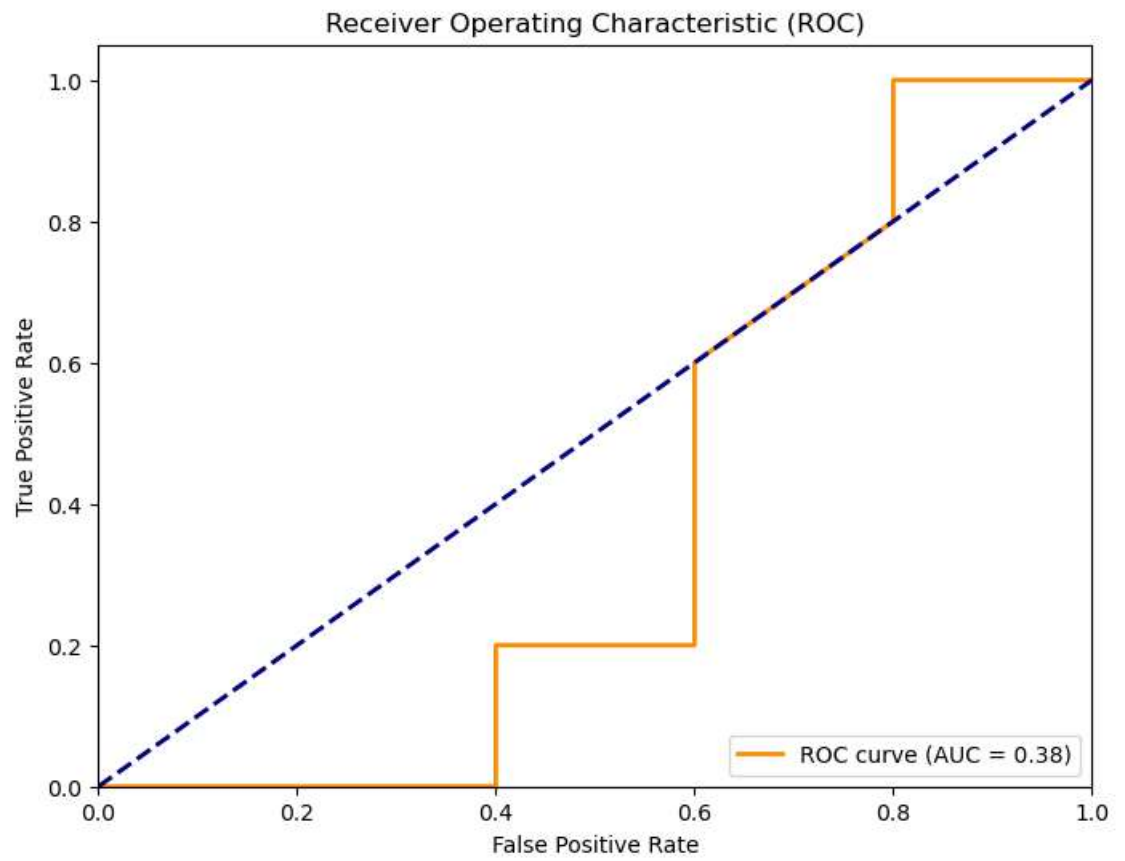
	precision	recall	f1-score	support
0	0.44	0.80	0.57	5
1	0.00	0.00	0.00	5
accuracy			0.40	10
macro avg	0.22	0.40	0.29	10
weighted avg	0.22	0.40	0.29	10

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

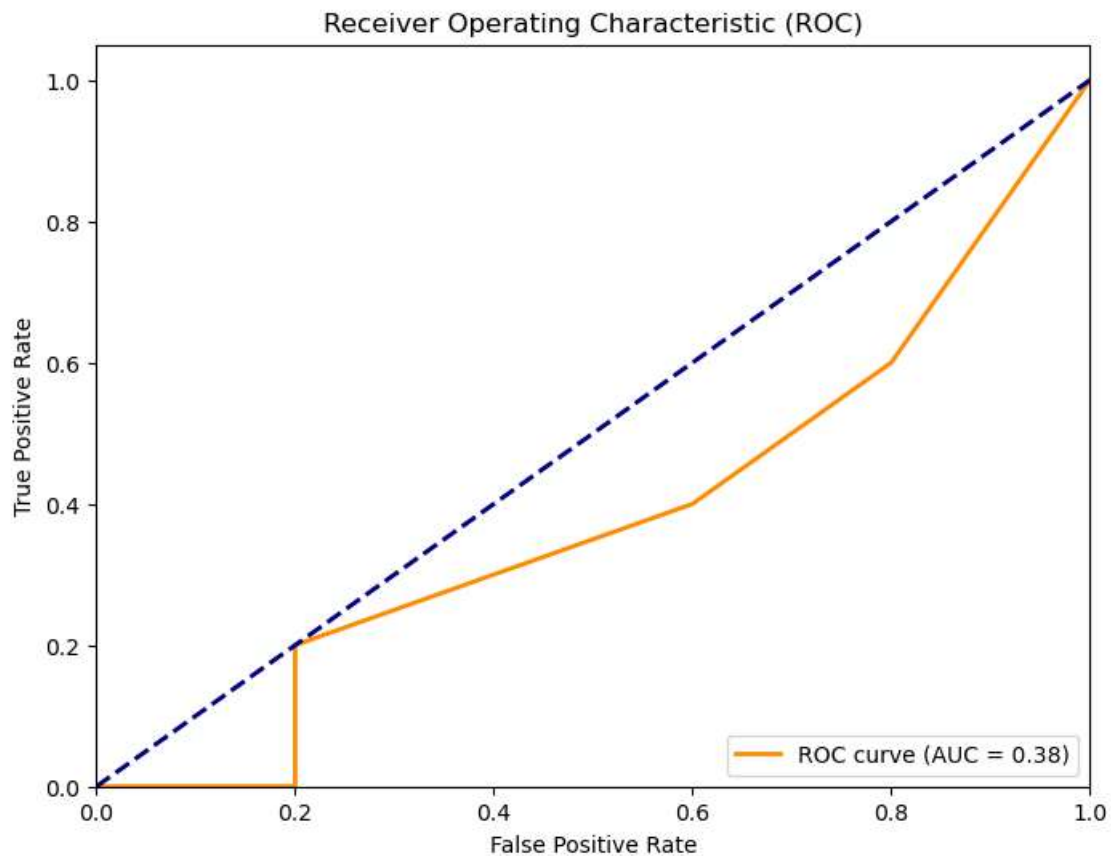
## ROC curve and AUC score (NB & Decision Tree)

```
In [3]: ▶ 1 from sklearn.metrics import roc_curve, roc_auc_score, auc
2 import matplotlib.pyplot as plt
3
4
5 # Split the data into features (X) and the target variable (y)
6 X = data.iloc[:, :-1] # All columns except the last 'spam' column
7 y = data['spam'] # Target variable
8
9 # Split the data into training and testing sets
10 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
11
12 # Naive Bayes Classifier
13 nb_classifier = MultinomialNB()
14 nb_classifier.fit(X_train, y_train)
15
16 # Predict probabilities of being spam
17 y_prob = nb_classifier.predict_proba(X_test)[: , 1]
18
19 # Calculate ROC curve
20 fpr, tpr, thresholds = roc_curve(y_test, y_prob)
21
22 # Calculate AUC score
23 roc_auc = auc(fpr, tpr)
24
25 # Plot ROC curve
26 plt.figure(figsize=(8, 6))
27 plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
28 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
29 plt.xlim([0.0, 1.0])
30 plt.ylim([0.0, 1.05])
31 plt.xlabel('False Positive Rate')
32 plt.ylabel('True Positive Rate')
33 plt.title('Receiver Operating Characteristic (ROC)')
34 plt.legend(loc='lower right')
35 plt.show()
36
37 # Print AUC score
38 print(f'AUC Score: {roc_auc:.2f}')
39
```



AUC Score: 0.38

```
In [4]: ▶ 1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.model_selection import train_test_split
3 from sklearn.metrics import roc_curve, roc_auc_score, auc
4 import matplotlib.pyplot as plt
5
6 # Split the data into features (X) and the target variable (y)
7 X = data.iloc[:, :-1] # All columns except the last 'spam' column
8 y = data['spam'] # Target variable
9
10 # Split the data into training and testing sets
11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
12
13 # Decision Tree Classifier
14 dt_classifier = DecisionTreeClassifier(random_state=42)
15 dt_classifier.fit(X_train, y_train)
16
17 # Predict probabilities of being spam
18 y_prob = dt_classifier.predict_proba(X_test)[:, 1]
19
20 # Calculate ROC curve
21 fpr, tpr, thresholds = roc_curve(y_test, y_prob)
22
23 # Calculate AUC score
24 roc_auc = auc(fpr, tpr)
25
26 # Plot ROC curve
27 plt.figure(figsize=(8, 6))
28 plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
29 plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
30 plt.xlim([0.0, 1.0])
31 plt.ylim([0.0, 1.05])
32 plt.xlabel('False Positive Rate')
33 plt.ylabel('True Positive Rate')
34 plt.title('Receiver Operating Characteristic (ROC)')
35 plt.legend(loc='lower right')
36 plt.show()
37
38 # Print AUC score
39 print(f'AUC Score: {roc_auc:.2f}')
40
```



AUC Score: 0.38

In [ ]: ▶

1