1. **Train the Naive Bayes model using the training dataset. The model calculates the probabilities of features given each class.**

Evaluate the performance of the Naive Bayes model using the test dataset. The performance metrics include accuracy, precision, recall, and F1-score.

The performance of the model depends on various factors, such as the size of the training dataset, the diversity of the data, and the complexity of the problem. In general, the Naive Bayes model is suitable for classification problems with high-dimensional data and large datasets.

program for model:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score

from sklearn.linear_model import LogisticRegression

data = pd.read_csv('spam.csv')

X = data['EmailText']

y = data['Label']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

vectorizer = CountVectorizer()

X_train_counts = vectorizer.fit_transform(X_train)

X_test_counts = vectorizer.transform(X_test)

model = MultinomialNB()

model.fit(X_train_counts, y_train)

y_pred = model.predict(X_test_counts)

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

```
classifier = LogisticRegression(solver='liblinear')

classifier.fit(X_train_counts, y_train)

y_pred = classifier.predict(X_test_counts)

print("Accuracy: ", accuracy_score(y_test, y_pred))

print("Precision: ", precision_score(y_test, y_pred))

print("Recall: ", recall_score(y_test, y_pred))

print("F1-score: ", f1_score(y_test, y_pred))
```

output for the program :

mentions accuracy first

```
Accuracy:  0.9703703703703703

Precision:  0.9583333333333333

Recall:  0.975

F1-score:  0.9666666666666667
```