```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix, classification_report
```

```
df = pd.read_csv('/content/SMSSpamCollection', sep='\t', header=None, names=['label', 'message'])
```

```
print("Dataset Shape:",df.shape)
print(df.head())
```

```
Dataset Shape: (5572, 2)
  label                                            message
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

```
df['label_num'] = df['label'].map({'ham':0, 'spam':1})
```

```
X = df['message']
Y = df['label_num']
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=y, random_state=42)
```

```
vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

```
nb = MultinomialNB()
nb.fit(X_train_tfidf, Y_train)
Y_pred_nb = nb.predict(X_test_tfidf)
```

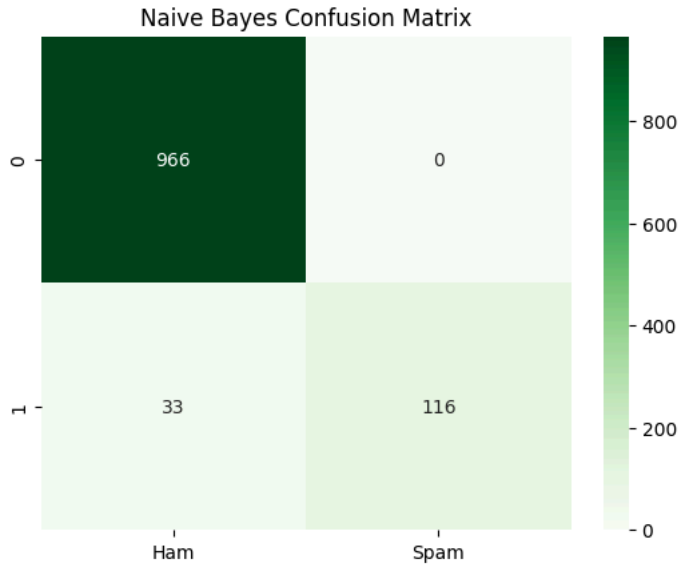
```
print("n-- Naive Bayes ---")
print("Accuracy:", accuracy_score(Y_test, Y_pred_nb))
print("Classification Report:\n", classification_report(Y_test, Y_pred_nb))
```

```
n-- Naive Bayes ---
Accuracy: 0.9704035874439462
Classification Report:
               precision    recall  f1-score   support

           0       0.97      1.00      0.98       966
           1       1.00      0.78      0.88       149

    accuracy                           0.97      1115
   macro avg       0.98      0.89      0.93      1115
weighted avg       0.97      0.97      0.97      1115
```

```
sns.heatmap(confusion_matrix(Y_test, Y_pred_nb), annot=True, fmt='d', cmap="Greens",xticklabels=['Ham','Spam'])
plt.title("Naive Bayes Confusion Matrix")
plt.show()
```




```
lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(X_train_tfidf, y_train)
y_pred_lr = lr.predict(X_test_tfidf)

print("\n--- Logistic Regression ---")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print("Classification Report:\n", classification_report(y_test, y_pred_lr))

# Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_pred_lr), annot=True, fmt='d', cmap="Greens",
            xticklabels=['Ham','Spam'], yticklabels=['Ham','Spam'])
plt.title("Logistic Regression Confusion Matrix")
plt.show()
```

```
--- Logistic Regression ---
Accuracy: 0.967713004484305
Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       966
           1       1.00      0.76      0.86       149

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.97      1115
```


```
cv_scores = cross_val_score(nb, vectorizer.transform(X), Y, cv=5, scoring="accuracy")
print("\nCross-validation Scores (Naive Bayes):", cv_scores)
print("Mean Accuracy:", cv_scores.mean())
```

```
Cross-validation Scores (Naive Bayes): [0.97847534 0.96681614 0.96319569 0.97127469 0.97217235]
Mean Accuracy: 0.970386841745095
```

```
params = {'alpha':[0.1,0.5,1.0]}
grid = GridSearchCV(MultinomialNB(), param_grid=params, cv=5, scoring="accuracy")
grid.fit(X_train_tfidf, Y_train)
print("\nBest Params (Naive Bayes):", grid.best_params_)
print("Best CV Score:", grid.best_score_)
```

```
Best Params (Naive Bayes): {'alpha': 0.5}
Best CV Score: 0.9831730357888804
```