In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import nltk
from nltk.corpus import stopwords
import string
import joblib
```

In [6]:
```
df = pd.read_csv('spam.csv', encoding='latin1')
```

In [7]:
```
df
```

Out[7]:

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| **0** | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| **1** | ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| **2** | spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| **3** | ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |
| **4** | ham | Nah I don't think he goes to usf, he lives aro... | NaN | NaN | NaN |
| **...** | ... | ... | ... | ... | ... |
| **5567** | spam | This is the 2nd time we have tried 2 contact u... | NaN | NaN | NaN |
| **5568** | ham | Will Ì_ b going to esplanade fr home? | NaN | NaN | NaN |
| **5569** | ham | Pity, * was in mood for that. So...any other s... | NaN | NaN | NaN |
| **5570** | ham | The guy did some bitching but I acted like i'd... | NaN | NaN | NaN |
| **5571** | ham | Rofl. Its true to its name | NaN | NaN | NaN |

5572 rows × 5 columns

In [16]:
```
# Encode labels
df['v1'] = df['v1'].map({'ham': 0, 'spam': 2})
```

In [10]:
```
# Text preprocessing
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\PoojaAarti\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

In [12]:
```
def preprocess(text):
    text = text.lower()
    text = "".join([char for char in text if char not in string.punctuation])
```

```
    tokens = text.split()
    tokens = [word for word in tokens if word not in stop_words]
    return " ".join(tokens)

df['v2'] = df['v2'].apply(preprocess)
```

In [13]:
```
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(df['v2'])
y = df['v1']
```

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [15]:
```
# Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)

print("Naive Bayes Accuracy:", accuracy_score(y_test, nb_predictions))
print("Naive Bayes Classification Report:\n", classification_report(y_test, nb_predict
```

```
Naive Bayes Accuracy: 0.9659192825112107
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       965
           1       1.00      0.75      0.85       150

    accuracy                           0.97      1115
   macro avg       0.98      0.87      0.92      1115
weighted avg       0.97      0.97      0.96      1115
```

In [17]:
```
# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)

print("Logistic Regression Accuracy:", accuracy_score(y_test, lr_predictions))
print("Logistic Regression Classification Report:\n", classification_report(y_test, lr
```

```
Logistic Regression Accuracy: 0.9426008968609866
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       965
           1       0.96      0.60      0.74       150

    accuracy                           0.94      1115
   macro avg       0.95      0.80      0.85      1115
weighted avg       0.94      0.94      0.94      1115
```

In [18]:
```
# Support Vector Machine
svm_model = SVC()
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)

print("SVM Accuracy:", accuracy_score(y_test, svm_predictions))
print("SVM Classification Report:\n", classification_report(y_test, svm_predictions))
```

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

           0       0.97      1.00      0.98       965
           1       0.98      0.77      0.87       150

    accuracy                           0.97      1115
   macro avg       0.97      0.89      0.92      1115
weighted avg       0.97      0.97      0.97      1115
```

In [19]:
```
joblib.dump(lr_model, 'best_sms_spam_model.pkl')
```

Out[19]:
```
['best_sms_spam_model.pkl']
```

In [20]:
```
loaded_model = joblib.load('best_sms_spam_model.pkl')
new_message = ["Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Te
new_message_tfidf = tfidf.transform(new_message)
prediction = loaded_model.predict(new_message_tfidf)
print("Prediction:", "Spam" if prediction[0] else "Legitimate")
```

```
Prediction: Spam
```

In [ ]: