In [1]:
```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
```

In [2]:
```
data = pd.read_csv('spam (1).csv', encoding='latin-1')
```

In [3]:
```
data.head()
```

Out[3]:

	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

In [4]:
```
columns_to_drop = ['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4']
data = data.drop(columns_to_drop, axis=1, errors='ignore')
data.head()
```

Out[4]:

	v1	v2
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...

In [6]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to C:\Users\Siva
[nltk_data]     Lohith\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[6]:
```
True
```

In [7]:
```
data['v2'] = data['v2'].str.lower()


def preprocess_text(text):
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_words]
    stemmer = PorterStemmer()
    stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]
    return ' '.join(stemmed_tokens)

data['v2'] = data['v2'].apply(preprocess_text)

data.head()
```

Out[7]:

	v1	v2
0	ham	go jurong point crazi avail bugi n great world...
1	ham	ok lar joke wif u oni
2	spam	free entri 2 wkli comp win fa cup final tkt 21...
3	ham	u dun say earli hor u c alreadi say
4	ham	nah think goe usf live around though

In [12]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [13]:
```
from sklearn.model_selection import train_test_split

data = pd.read_csv('spam (1).csv', encoding='latin-1')


tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(data['v2'])


data['v1'] = data['v1'].map({'ham': 0, 'spam': 1})


X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix, data['v1'], test_size=0.2, random_state=42)


print("TF-IDF Matrix Shape:", tfidf_matrix.shape)
print("Training Data Shape:", X_train.shape)
print("Testing Data Shape:", X_test.shape)
```

```
TF-IDF Matrix Shape: (5572, 8672)
Training Data Shape: (4457, 8672)
Testing Data Shape: (1115, 8672)
```

In [14]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

rf_classifier = RandomForestClassifier(random_state=42)


rf_classifier.fit(X_train, y_train)

y_pred = rf_classifier.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print("Accuracy:", accuracy)
print("classification report:\n", classification_rep)
```

```
Accuracy: 0.9766816143497757
classification report:
               precision    recall  f1-score   support

           0       0.97      1.00      0.99       965
           1       1.00      0.83      0.91       150

    accuracy                           0.98      1115
   macro avg       0.99      0.91      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

In [15]:
```
input_text = """\apple Inc.Your iPhone 6 linked top***zm".edu) has been used a few minutes
ago. To localize it,login now to your apple account ."""

input_text = input_text.lower()


input_tfidf = tfidf_vectorizer.transform([input_text])


prediction = rf_classifier.predict(input_tfidf)


if prediction[0] == 1:
    print("This message is predicted to be SPAM by trained model.")
else:
    print("This message is predicted to be NOT SPAM by trained model.")
```

```
This message is predicted to be NOT SPAM by trained model.
```

In [16]:
```
input_text1 = "Hey, I'm mark. How are you?."

input_text1 = input_text1.lower()

input_tfidf = tfidf_vectorizer.transform([input_text1])

prediction = rf_classifier.predict(input_tfidf)

if prediction[0] == 1:
    print("This message is predicted to be SPAM by trained model.")
else:
    print("This message is predicted to be NOT SPAM by trained model.")
```

```
This message is predicted to be NOT SPAM by trained model.
```

In [ ]: