In [ ]:
```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

In [5]:
```
df = pd.read_csv("spam.csv", encoding='latin-1')
df.head()
```

Out[5]:

	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 [6]:
```
df.shape
```

Out[6]:
```
(5572, 5)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

In [8]:
```
df.drop_duplicates(inplace=True)
print(df.shape)
```

```
(5169, 5)
```

In [9]:
```
print(df.isnull().sum())
```

```
v1              0
v2              0
Unnamed: 2    5126
Unnamed: 3    5159
Unnamed: 4    5164
dtype: int64
```

In [10]:
```
df.drop(columns=['Unnamed: 2', 'Unnamed: 3','Unnamed: 4'], inplace=True)
```

In [11]:
```
print(df.isnull().sum())
```

```
v1    0
v2    0
dtype: int64
```

In [12]:
```
df.drop_duplicates(inplace=True)
print(df.shape)
```

```
(5169, 2)
```

In [16]:
```
nltk.download("stopwords")
```

```
[nltk_data] Error loading stopwords: <urlopen error [WinError 10060] A
[nltk_data]     connection attempt failed because the connected party
[nltk_data]     did not properly respond after a period of time, or
[nltk_data]     established connection failed because connected host
[nltk_data]     has failed to respond>
```

Out[16]:
```
False
```

In [15]:
```
def process(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    clean = [word for word in nopunc.split() if word.lower() not in stopwords.words
    return clean
# to show the tokenization
df['v2'].head().apply(process)
```

Out[15]:
```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: v2, dtype: object
```

In [20]:
```
from sklearn.feature_extraction.text import CountVectorizer
message = CountVectorizer(analyzer=process).fit_transform(df['v2'])
print(message)
```

```
  (0, 2027)     1
  (0, 7456)     1
  (0, 8809)     1
  (0, 5685)     1
  (0, 1096)     1
  (0, 5136)     1
  (0, 8231)     1
  (0, 6846)     1
  (0, 11043)    1
  (0, 7567)     1
  (0, 6131)     1
  (0, 5135)     1
  (0, 1461)     1
  (0, 6815)     1
  (0, 4574)     1
  (0, 10845)    1
  (1, 3012)     1
  (1, 7600)     1
  (1, 2407)     1
  (1, 10952)    1
  (1, 10582)    1
  (1, 8482)     1
  (2, 1909)     1
  (2, 6244)     2
  (2, 422)      1
  :     :
  (5165, 6791)  1
  (5165, 11239) 1
  (5165, 6604)  1
  (5165, 6267)  1
  (5166, 8147)  1
  (5166, 3169)  1
  (5166, 3655)  1
  (5166, 10087) 1
  (5167, 7698)  1
  (5167, 10886) 1
  (5167, 8314)  1
  (5167, 10669) 1
  (5167, 6612)  1
  (5167, 9804)  1
  (5167, 6196)  1
  (5167, 6710)  1
  (5167, 5169)  1
  (5167, 7297)  1
  (5167, 6892)  1
  (5167, 7190)  1
  (5167, 4430)  1
  (5167, 4973)  1
  (5168, 8243)  1
  (5168, 10532) 1
  (5168, 3370)  1
```

In [21]:
```
#split the data into 80% training and 20% testing
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(message, df['v1'], test_size=0.20,
# To see the shape of the data
print(message.shape)
```

```
(5169, 11304)
```

In [22]:
```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(xtrain, ytrain)
```

In [23]:
```
print(classifier.predict(xtrain))
print(ytrain.values)
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
```

In [24]:
```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(xtrain)
print(classification_report(ytrain, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytrain, pred))
print("Accuracy: \n", accuracy_score(ytrain, pred))
```

```
              precision    recall  f1-score   support

         ham       1.00      1.00      1.00      3631
        spam       0.98      0.98      0.98       504

    accuracy                           1.00      4135
   macro avg       0.99      0.99      0.99      4135
weighted avg       1.00      1.00      1.00      4135


Confusion Matrix:
 [[3623    8]
 [  11  493]]
Accuracy:
 0.9954050785973397
```

In [25]:
```
#print the predictions
print(classifier.predict(xtest))
#print the actual values
print(ytest.values)
```

```
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
['ham' 'ham' 'ham' ... 'ham' 'ham' 'ham']
```

In [26]:
```
# Evaluating the model on the testing data set
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
pred = classifier.predict(xtest)
print(classification_report(ytest, pred))
print()
print("Confusion Matrix: \n", confusion_matrix(ytest, pred))
print("Accuracy: \n", accuracy_score(ytest, pred))
```

```
                precision    recall  f1-score   support

           ham       0.99      0.96      0.97       885
          spam       0.80      0.93      0.86       149

      accuracy                           0.96      1034
     macro avg       0.89      0.94      0.92      1034
  weighted avg       0.96      0.96      0.96      1034


Confusion Matrix:
 [[850  35]
 [ 11 138]]
Accuracy:
 0.9555125725338491
```

In [ ]: