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

In [2]:
```
df=pd.read_csv("D:\swand\spam - Email Dataset.csv")
df.head()
```

Out[2]:

	Category	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...

In [3]:
```
df.Category.value_counts()
```

Out[3]:
```
ham     2111
spam     341
Name: Category, dtype: int64
```

In [4]:
```
df['spam'] = df['Category'].apply(lambda x: 1 if x =='spam' else 0)
```

In [5]:
```
df.shape
```

Out[5]:
```
(2452, 3)
```

In [6]:
```
df.head()
```

Out[6]:

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

Train test split

In [7]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.Message, df.spam, test_
```

In [8]:
```
X_train.shape
```

Out[8]:
```
(1961,)
```

In [9]:
```
X_test.shape
```

Out[9]:
```
(491,)
```

In [10]:
```
type(X_train)
```

Out[10]:
```
pandas.core.series.Series
```

In [11]:
```
X_train[:4]
```

Out[11]:
```
906     Hey what's up charles sorry about the late reply.
2226    Alrite jod hows the revision goin? Keris bin d...
1583      Yep, at derek's house now, see you Sunday &lt;3
2165    Nothing really, just making sure everybody's u...
Name: Message, dtype: object
```

In [12]:
```
type(y_train)
```

Out[12]:
```
pandas.core.series.Series
```

In [13]:
```
y_train[:4]
```

Out[13]:
```
906     0
2226    0
1583    0
2165    0
Name: spam, dtype: int64
```

In [14]:
```
type(X_train.values)
```

Out[14]:
```
numpy.ndarray
```

Create bag of words representation using CountVectorizer

In [15]:
```
from sklearn.feature_extraction.text import CountVectorizer

v = CountVectorizer()

X_train_cv = v.fit_transform(X_train.values)
X_train_cv
```

Out[15]:
```
<1961x5006 sparse matrix of type '<class 'numpy.int64'>'
        with 26515 stored elements in Compressed Sparse Row format>
```

In [16]:
```
X_train_cv.toarray()[:2][0]
```

Out[16]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [17]:
```
X_train_cv.shape
```

Out[17]:
```
(1961, 5006)
```

In [19]:
```
X_train_np = X_train_cv.toarray()
X_train_np[0]
```

Out[19]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [20]:
```
np.where(X_train_np[0]!=0)
```

Out[20]:
```
(array([ 444, 1077, 2193, 2582, 3661, 4074, 4367, 4613, 4814], dtype=int64),)
```

Train the naive bayes model

In [23]:
```
from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()
model.fit(X_train_cv, y_train)
```

Out[23]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [24]:
```
X_test_cv = v.transform(X_test)
```

Evaluate Performance

In [25]:
```
from sklearn.metrics import classification_report

y_pred = model.predict(X_test_cv)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       416
           1       1.00      0.87      0.93        75

    accuracy                           0.98       491
   macro avg       0.99      0.93      0.96       491
weighted avg       0.98      0.98      0.98       491
```

In [26]:
```
emails = [
    'Hey mohan, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss thi
]

emails_count = v.transform(emails)
model.predict(emails_count)
```

Out[26]:
```
array([0, 1], dtype=int64)
```

Train the model using sklearn pipeline and reduce number of lines of code

In [27]:
```
from sklearn.pipeline import Pipeline

clf = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('nb', MultinomialNB())
])
```

In [28]:
```
clf.fit(X_train, y_train)
```

Out[28]:
```
Pipeline(memory=None,
         steps=[('vectorizer',
                 CountVectorizer(analyzer='word', binary=False,
                                 decode_error='strict',
                                 dtype=<class 'numpy.int64'>, encoding='utf-
8',
                                 input='content', lowercase=True, max_df=1.0,
                                 max_features=None, min_df=1,
                                 ngram_range=(1, 1), preprocessor=None,
                                 stop_words=None, strip_accents=None,
                                 token_pattern='(?u)\\b\\w\\w+\\b',
                                 tokenizer=None, vocabulary=None)),
                ('nb',
                 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e))],
         verbose=False)
```

In [29]:
```
y_pred = clf.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       416
           1       1.00      0.87      0.93        75

    accuracy                           0.98       491
   macro avg       0.99      0.93      0.96       491
weighted avg       0.98      0.98      0.98       491
```

In [ ]: