NLP

In [2]:
```
import pandas as pd
import numpy as np
import nltk # natual language toolkit
```

In [3]:
```
messages = pd.read_csv(r"F:\Imaticus\Data set\spam1.csv", encoding = 'cp1252')
```

In [4]:
```
messages.shape
```

Out[4]:
```
(6776, 5)
```

In [5]:
```
messages.head(2)
```

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

Data Cleaning

In [6]:
```
messages.isnull().sum()
```

Out[6]:
```
v1              0
v2              0
Unnamed: 2    6720
Unnamed: 3    6760
Unnamed: 4    6768
dtype: int64
```

In [7]:
```
messages = messages.loc[:, ['v1','v2']]
# get rid of columns which have huge nulls
```

In [8]:
```
messages.head(2)
```

Out[8]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...

In [9]:
```
messages.rename(columns={'v1' : 'label' , 'v2' : 'message'}, inplace= True) # mess
```

In [10]:
```
messages.head(2)
```

Out[10]:

	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...

In [11]:
```
messages.label.value_counts()
```

Out[11]:
```
ham     5854
spam     922
Name: label, dtype: int64
```

converting Cateorical variable to Numerical

In [12]:
```
messages.label.replace({'spam' : 1 , 'ham' : 0}, inplace=True)

# message.lebel = y variable
```

In [13]:
```
messages.label.value_counts()
```

Out[13]:
```
0    5854
1     922
Name: label, dtype: int64
```

In [14]:
```
messages.message = messages.message.str.lower() # to convert string into lower cas
```

In [15]:
```
messages.message
```

Out[15]:
```
0       go until jurong point, crazy.. available only ...
1                           ok lar... joking wif u oni...
2       free entry in 2 a wkly comp to win fa cup fina...
3       u dun say so early hor... u c already then say...
4       nah i don't think he goes to usf, he lives aro...
                              ...
6771    this is the 2nd time we have tried 2 contact u...
6772                will ì_ b going to esplanade fr home?
6773    pity, * was in mood for that. so...any other s...
6774    the guy did some bitching but i acted like i'd...
6775                           rofl. its true to its name
Name: message, Length: 6776, dtype: object
```

In [16]:
```
from nltk.corpus import stopwords
```

In [17]:
```
#stopwords.words('english')
```

In [18]:
```
import nltk
```

In [19]:
```
len(stopwords.words('english'))

abcd = stopwords.words('english')
```

In [20]:
```
import string
```

In [21]:
```
string.punctuation
```

Out[21]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [22]:
```
# i wish to remove stop words and punctuation bcz they do not add any meaning to do

# till this line we have only seen stopwords & punctuation but we have not remove t
```

In [23]:
```
# lets create user defined function
```

In [24]:
```
def text_process(mess):            ### creating a function
    """                                                    ## a docstring
    1. remove the punctuation
    2. remove the stopwords
    3. return the list of clean textwords

    """
    nopunc = [char for char in mess if char not in string.punctuation]
    nopunc = "".join(nopunc)

    return [ word for word in nopunc.split() if word not in abcd]
```

In [25]:
```
messages.message.apply(text_process) # this will take time to run
```

Out[25]:
```
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...
                              ...
6771    [2nd, time, tried, 2, contact, u, u, å£750, po...
6772                    [ì, b, going, esplanade, fr, home]
6773                      [pity, mood, soany, suggestions]
6774    [guy, bitching, acted, like, id, interested, b...
6775                                  [rofl, true, name]
Name: message, Length: 6776, dtype: object
```

In [26]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [28]:
```
import time

start = time.time() # Get the current time

bow_transformer =    CountVectorizer(analyzer = text_process   ).fit(messages["mess

end = time.time() # get the current time
print(end - start) #time taken
```

```
1.9618027210235596
```

In [29]:
```
bow_transformer.vocabulary_
# count of each unique word
```

```
 'comp': 2290,
 'win': 9010,
 'fa': 3257,
 'cup': 2514,
 'final': 3381,
 'tkts': 8304,
 '21st': 434,
 'may': 5283,
 '2005': 421,
 'text': 8143,
 '87121': 836,
 'receive': 6769,
 'questionstd': 6660,
 'txt': 8511,
 'ratetcs': 6713,
 'apply': 1226,
 '08452810075over18s': 71,
 'dun': 2970,
 'say': 7123,
 'early': 2991,
```

In [30]:
```
len(bow_transformer.vocabulary_)

# 9422 unique words
# TDM it will have 9422 columns
```

Out[30]: 9422

In [31]:
```
tdm = bow_transformer.transform(messages["message"])
```

In [32]:
```
tdm.shape # x variable
```

Out[32]: (6776, 9422)

In [ ]:
```
#6776 rows,  9422 columns
```

In [33]:
```
type(tdm) # SPARSE matrix
```

Out[33]: scipy.sparse.csr.csr_matrix

In [34]:
```
# this tdm will act as my x(independent) messages.label column is my Y(dependent)
```

In [35]:
```
from sklearn.model_selection import train_test_split
```

In [36]:
```
x_train, x_test, y_train, y_test = train_test_split(tdm, messages.label, test_size
```

In [37]:
```
print(x_train.shape)
print(y_train.shape)
print("----")
print(x_test.shape)
print(y_test.shape)

# just to check record shapes
```

```
(5420, 9422)
(5420,)
----
(1356, 9422)
(1356,)
```

In [38]:
```
#TDM build till above code
```

now build the model using NaiveBayes

In [40]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [41]:
```
nb_model = MultinomialNB()
```

In [42]:
```
nb_model.fit(x_train, y_train)
```

Out[42]:
```
MultinomialNB()
```

In [43]:
```
pred = nb_model.predict(x_test)
```

In [44]:
```
from sklearn.metrics import confusion_matrix
```

In [45]:
```
tab =  confusion_matrix(y_test , pred)
```

Confusion Matrix

In [46]:
```
tab
```

Out[46]:
```
array([[1155,   16],
       [   8,  177]], dtype=int64)
```

Accuracy

In [47]:
```
tab.diagonal().sum() * 100 / tab.sum()
```

Out[47]:
```
98.23008849557522
```

Using RandomForest

In [50]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [51]:
```
rfc_model = RandomForestClassifier()
```

In [52]:
```
rfc_model.fit(x_train, y_train)
```

Out[52]:
```
RandomForestClassifier()
```

In [53]:
```
pred_rfc = rfc_model.predict(x_test)
```

In [54]:
```
tab_rfc = confusion_matrix(y_test , pred_rfc)
tab_rfc
```

Out[54]:
```
array([[1171,    0],
       [  33,  152]], dtype=int64)
```

In [55]:
```
tab_rfc.diagonal().sum() * 100 / tab_rfc.sum()
```

Out[55]:
```
97.56637168141593
```

using Logestic regression

In [57]:
```
from sklearn.linear_model import LogisticRegression
```

In [58]:
```
logreg = LogisticRegression()
```

In [59]:
```
logreg.fit(x_train, y_train)
```

Out[59]:
```
LogisticRegression()
```

In [60]:
```
pred1 = logreg.predict(x_test)
```

In [61]:
```
tab1 = confusion_matrix(y_test , pred1)
tab1
```

Out[61]:
```
array([[1170,    1],
       [  18,  167]], dtype=int64)
```

In [62]:
```
tab1.diagonal().sum() * 100 / tab1.sum()
```

Out[62]:
```
98.59882005899705
```

Using Decisoin tree

In [64]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [65]:
```
dt = DecisionTreeClassifier(max_depth=4)
```

In [66]:
```
dt.fit(x_train, y_train)
```

Out[66]:
```
DecisionTreeClassifier(max_depth=4)
```

In [67]:
```
pred_dt = dt.predict(x_test)
```

In [68]:
```
tab2 = confusion_matrix(y_test , pred_dt)
tab2
```

Out[68]:
```
array([[1134,   37],
       [  61,  124]], dtype=int64)
```

In [69]:
```
tab2.diagonal().sum() * 100/tab.sum()
```

Out[69]:
```
92.77286135693215
```

plot a word cloud

In [73]:
```
from wordcloud import WordCloud #required import
```

In [74]:
```
import matplotlib.pyplot as plt
```

In [75]:
```
from wordcloud import WordCloud

# messages["message"]
cloud  = WordCloud(stopwords = stopwords.words("english"), max_words= 10).generate
plt.figure(figsize=(10 , 10))
plt.imshow(cloud)
```

Out[75]:
```
<matplotlib.image.AxesImage at 0x1e1bab8fdf0>
```