In [1]:

```
import numpy as np
```

In [2]:

```
import pandas as pd
```

1) LOADING DATA SET

In [3]:

```
dt = pd.read_csv('SPAM-210331-134237.csv',encoding="ISO-8859-1")
dt.head(10)
```

Out[3]:

	type	text
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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

In [4]:

```
dt['spam'] = dt['type'].map({'spam' : 1, 'ham' : 0 }).astype(int)
dt.head(5)
```

Out[4]:

	type	text	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

In [35]:

```
dt.isnull().sum()
```

Out[35]:

```
type     0
text     0
spam     0
dtype: int64
```

In [5]:

```
print("COLUMNS IN THE GIVEN DATA:")
for col in dt.columns:
    print(col)
```

```
COLUMNS IN THE GIVEN DATA:
type
text
spam
```

In [6]:

```
t = len(dt['type'])
print("NO OF ROWS IN REVIEW COLUMN:",t)
t = len(dt['text'])
print("NO OF ROW IN LINKED COLUMN:" ,t)
```

```
NO OF ROWS IN REVIEW COLUMN: 116
NO OF ROW IN LINKED COLUMN: 116
```

TOKENIZATION

In [7]:

```
dt['text'][1]
```

Out[7]:

```
'Ok lar... Joking wif u oni...'
```

In [8]:

```
def tokenizer(text):
    return text.split()
```

In [9]:

```
dt['text'] = dt['text'].apply(tokenizer)
```

In [10]:

```
dt['text'][1]
```

Out[10]:

```
['Ok', 'lar...', 'Joking', 'wif', 'u', 'oni...']
```

STEMMING

In [11]:

```
dt['text'][1]
```

Out[11]:

```
['Ok', 'lar...', 'Joking', 'wif', 'u', 'oni...']
```

In [12]:

```
from nltk.stem.snowball import SnowballStemmer
porter = SnowballStemmer("english" ,ignore_stopwords = False )
```

In [13]:

```
def stem_it(text):
    return [porter.stem(word) for word in text]
```

In [14]:

```
dt['text'] = dt['text'].apply(stem_it)
```

In [15]:

```
dt['text'][1]
```

Out[15]:

```
['ok', 'lar...', 'joke', 'wif', 'u', 'oni...']
```

lemmitization

In [16]:

```
dt['text'][109]
```

Out[16]:

```
['i',
 'know!',
 'grumpi',
 'old',
 'people.',
 'my',
 'mom',
 'was',
 'like',
 'you',
 'better',
 'not',
 'be',
 'lying.',
 'then',
 'again',
 'i',
 'am',
 'alway',
 'the',
 'one',
 'to',
 'play',
 'jokes...']
```

In [17]:

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

In [18]:

```
def lemmit_it(text):
    return [lemmatizer.lemmatize(word, pos ="a")for word in text]
```

In [19]:

```
dt['text'] = dt['text'].apply(lemmit_it)
```

In [20]:

```
dt['text'][109]
```

Out[20]:

```
['i',
 'know!',
 'grumpi',
 'old',
 'people.',
 'my',
 'mom',
 'was',
 'like',
 'you',
 'good',
 'not',
 'be',
 'lying.',
 'then',
 'again',
 'i',
 'am',
 'alway',
 'the',
 'one',
 'to',
 'play',
 'jokes...']
```

STOP WORD REMOVAL

In [21]:

```
dt['text'][110]
```

Out[21]:

```
['dont', 'worry.', 'i', 'guess', 'he', 'busy.']
```

In [22]:

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
```

In [23]:

```
def stop_it(text):
    review = [word for word in text if not word in stop_words ]
    return review
```

In [24]:

```
dt['text'] = dt['text'].apply(stop_it)
```

In [25]:

```
dt['text'][110]
```

Out[25]:

```
['dont', 'worry.', 'guess', 'busy.']
```

In [26]:

```
dt.head(10)
```

Out[26]:

	type	text	spam
0	ham	[go, jurong, point,, crazy.., avail, onli, bug...	0
1	ham	[ok, lar..., joke, wif, u, oni...]	0
2	spam	[free, entri, 2, wkli, comp, win, fa, cup, fin...	1
3	ham	[u, dun, say, earli, hor..., u, c, alreadi, sa...	0
4	ham	[nah, think, goe, usf,, live, around, though]	0
5	spam	[freemsg, hey, darl, 3, week, word, back!, i'd...	1
6	ham	[even, brother, like, speak, me., treat, like,...	0
7	ham	[per, request, mell, mell, (oru, minnaminungin	0

	type	text	spam
7	ham	[per, request, mell, mell, (oru, minnaminungin...	0
8	spam	[winner!!, valu, network, custom, select, rece...	1
9	spam	[mobil, 11, month, more?, u, r, entitl, updat,...	1

In [27]:

```
dt['text'] = dt['text'].apply(' '.join)
```

In [28]:

```
dt.head(10)
```

Out[28]:

	type	text	spam
0	ham	go jurong point, crazy.. avail onli bugi n gre...	0
1	ham	ok lar... joke wif u oni...	0
2	spam	free entri 2 wkli comp win fa cup final tkts 2...	1
3	ham	u dun say earli hor... u c alreadi say...	0
4	ham	nah think goe usf, live around though	0
5	spam	freemsg hey darl 3 week word back! i'd like fu...	1
6	ham	even brother like speak me. treat like aid pat...	0
7	ham	per request mell mell (oru minnaminungint nuru...	0
8	spam	winner!! valu network custom select receivea â...	1
9	spam	mobil 11 month more? u r entitl updat late col...	1

Transform text data into TDF/TF-IDF Vectors

In [29]:

```
#Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a
#  corresponding vector of real numbers which used to find word predictions, word similar
ities/semantics
```

In [30]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
y = dt.spam.values
x = tfidf.fit_transform(dt['text'])
```

In [31]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=1,test_size=0.2,shuffl
e = False)

#If you don't specify the random_state in your code, then every time you run(execute) you
r code a
  #new random value is generated and the train and test datasets would have different val
ues each time.

#This parameter decides the size of the data that has to be split as the test dataset.

# does not shuffe the data
```

Classfication using logistic regression

In [32]:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
from sklearn.metrics import accuracy_score
acc_log = accuracy_score(y_pred, y_test)*100
print("accuracy:",acc_log)
```

```
accuracy: 87.5
```

Classification using LinearSVC Accuracy

In [33]:

```
from sklearn.svm import LinearSVC
linear_svc = LinearSVC(random_state = 0)
linear_svc.fit(x_train,y_train)
y_pred = linear_svc.predict(x_test)
acc_linear_svc = accuracy_score(y_pred, y_test)*100
print("accuracy:",acc_linear_svc)
```

```
accuracy: 87.5
```

In [ ]: