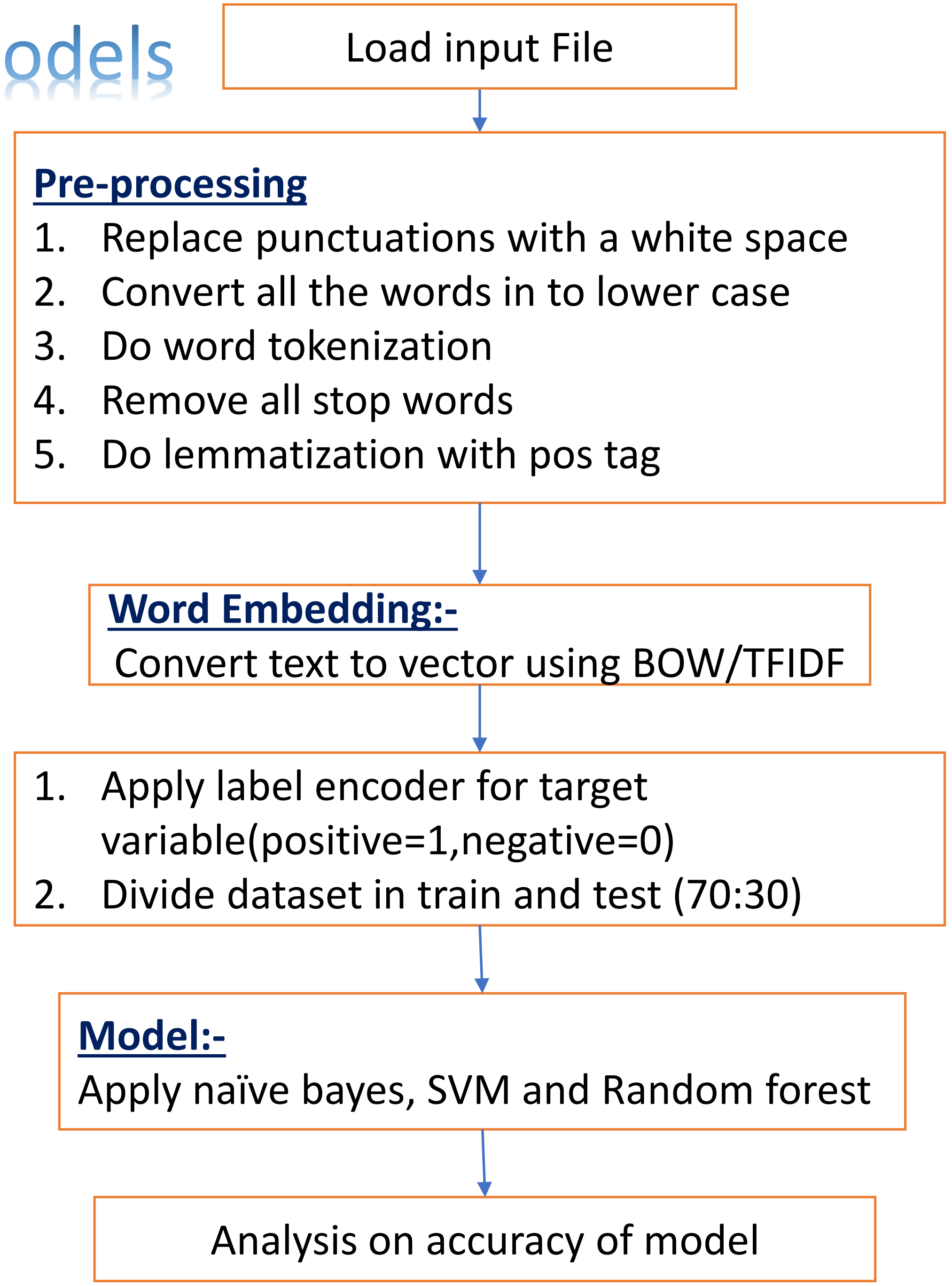
Case Study on Movie Review and Email Spam

BY:SHASHI KUMAR

INITIATIVE BY: PATHSHALA.AI

https://www.youtube.com/c/ShashiSAS

Steps for NLP models



Load input File

Pre-processing

1. Replace punctuations with a white space
2. Convert all the words in to lower case
3. Do word tokenization
4. Remove all stop words
5. Do lemmatization with pos tag

Word Embedding:-

Convert text to vector using BOW/TFIDF

1. Apply label encoder for target variable(positive=1,negative=0)
2. Divide dataset in train and test (70:30)

Model:-

Apply naïve bayes, SVM and Random forest

Analysis on accuracy of model

Movie Review Analysis using BOW

In [1]:

```
import pandas as pd
df = pd.read_csv('movie-Review.csv')
```

In [2]:

```
import numpy as np
np.random.seed(500)
```

In [3]:

```
#Remove number
import re # import all Regular expression functions
df['text']=[re.sub('\d','', i)for i in df['text']]
df.head(10)
```

Out[3]:

	class	text
0	Pos	films adapted from comic books have had plent...
1	Pos	every now and then a movie comes along from a...
2	Pos	you ve got mail works alot better than it des...
3	Pos	jaws is a rare film that grabs your atte...
4	Pos	moviemaking is a lot like being the general m...
5	Pos	on june a self taught idealistic ye...
6	Pos	apparently director tony kaye had a major b...
7	Pos	one of my colleagues was surprised when i tol...
8	Pos	after bloody clashes and independence won l...
9	Pos	the american action film has been slowly drow...

In [4]:

```
# Replace punctuations with a white space
import string
df['text']=[re.sub('[%s]' % re.escape(string.punctuation), ' ', i) for i in df['text']]
df.head(10)
```

Out[4]:

	class	text
0	Pos	films adapted from comic books have had plent...
1	Pos	every now and then a movie comes along from a...
2	Pos	you ve got mail works alot better than it des...
3	Pos	jaws is a rare film that grabs your atte...
4	Pos	moviemaking is a lot like being the general m...
5	Pos	on june a self taught idealistic ye...
6	Pos	apparently director tony kaye had a major b...
7	Pos	one of my colleagues was surprised when i tol...
8	Pos	after bloody clashes and independence won l...
9	Pos	the american action film has been slowly drow...

In [5]:

```
df['text']=[i.lower() for i in df['text']]
```

In [6]:

```
# import pandas as pd
import pandas as pd
#Word Tokenization
import nltk # import package for tokenization
#nltk.download('punkt') # download all spporting function /files for NLTK package
from nltk.tokenize import word_tokenize
df['text_wt'] = [word_tokenize(i) for i in df['text']]
df.head()
```

Out[6]:

	class	text	text_wt
0	Pos	films adapted from comic books have had plent...	[films, adapted, from, comic, books, have, had...
1	Pos	every now and then a movie comes along from a...	[every, now, and, then, a, movie, comes, along...
2	Pos	you ve got mail works alot better than it des...	[you, ve, got, mail, works, alot, better, than...
3	Pos	jaws is a rare film that grabs your atte...	[jaws, is, a, rare, film, that, grabs, your, a...
4	Pos	moviemaking is a lot like being the general m...	[moviemaking, is, a, lot, like, being, the, ge...

In [7]:

```
#To show the stop words
#nltk.download('stopwords') #download Stopwords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
#Remove All Stop Word
df['text_SW'] = [[i for i in j if not i in stop_words] for j in df['text_wt']]# remove the word which is aviable in stopword libr
df.head()
```

Out[7]:

	class	text	text_wt	text_SW
0	Pos	films adapted from comic books have had plent...	[films, adapted, from, comic, books, have, had...	[films, adapted, comic, books, plenty, success...
1	Pos	every now and then a movie comes along from a...	[every, now, and, then, a, movie, comes, along...	[every, movie, comes, along, suspect, studio, ...
2	Pos	you ve got mail works alot better than it des...	[you, ve, got, mail, works, alot, better, than...	[got, mail, works, alot, better, deserves, ord...
3	Pos	jaws is a rare film that grabs your atte...	[jaws, is, a, rare, film, that, grabs, your, a...	[jaws, rare, film, grabs, attention, shows, si...
4	Pos	moviemaking is a lot like being the general m...	[moviemaking, is, a, lot, like, being, the, ge...	[moviemaking, lot, like, general, manager, nfl...

In [8]:

```
#nltk.download('tagsets')
#nltk.help.upenn_tagset()# tagset documentation
#nltk.download('wordnet')
from collections import defaultdict #Default Dictionary is imported from collections
from nltk.corpus import wordnet as wn #the corpus reader wordnet is imported.
from nltk.tag import pos_tag
# WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc.
#By default it is set to Noun
tag_map = defaultdict(lambda : wn.NOUN) #Dictionary is created where pos_tag (first letter) are the key values
tag_map['J'] = wn.ADJ            #whose values are mapped with the value
tag_map['V'] = wn.VERB             #from wordnet dictionary. We have taken the only first letter as
tag_map['R'] = wn.ADV
# we will use it later in the loop.
#tag_map
```

In [9]:

```
#lemmatization
from nltk.stem import WordNetLemmatizer
 # Initializing WordNetLemmatizer()
lemmatizer = WordNetLemmatizer()

df['lemma']=[[lemmatizer.lemmatize(word,tag_map[tag[0]]) for word ,tag in pos_tag(i)] for i in df['text_SW']]
df.head()
```

Out[9]:

	class	text	text_wt	text_SW	lemma
0	Pos	films adapted from comic books have had plent...	[films, adapted, from, comic, books, have, had...	[films, adapted, comic, books, plenty, success...	[film, adapt, comic, book, plenty, success, wh...
1	Pos	every now and then a movie comes along from a...	[every, now, and, then, a, movie, comes, along...	[every, movie, comes, along, suspect, studio, ...	[every, movie, come, along, suspect, studio, e...
2	Pos	you ve got mail works alot better than it des...	[you, ve, got, mail, works, alot, better, than...	[got, mail, works, alot, better, deserves, ord...	[get, mail, work, alot, good, deserves, order,...
3	Pos	jaws is a rare film that grabs your atte...	[jaws, is, a, rare, film, that, grabs, your, a...	[jaws, rare, film, grabs, attention, shows, si...	[jaw, rare, film, grab, attention, show, singl...
4	Pos	moviemaking is a lot like being the general m...	[moviemaking, is, a, lot, like, being, the, ge...	[moviemaking, lot, like, general, manager, nfl...	[moviemaking, lot, like, general, manager, nfl...

In [10]:

```
df['lemma2']= df['lemma'].apply(lambda x: ' '.join(x))
```

In [11]:

```
df['lemma2'].head()
```

Out[11]:

```
0    film adapt comic book plenty success whether s...
1    every movie come along suspect studio every in...
2    get mail work alot good deserves order make fi...
3    jaw rare film grab attention show single image...
4    moviemaking lot like general manager nfl team ...
Name: lemma2, dtype: object
```

In [12]:

```
# Bag of Words
from sklearn.feature_extraction.text import CountVectorizer
#instantiate CountVectorizer()# CountVectorizer to count the number of words (term frequency)
cv = CountVectorizer(max_features=5000)
#this steps generates word counts for the words in your docs and return term-document matrix.
BOW = cv.fit_transform(df['lemma2']).toarray()
```

In [13]:

```
pd.DataFrame(BOW, columns=cv.get_feature_names()).head()
```

Out[13]:

	aaron	abandon	ability	able	aboard	abound	abraham	absence	absent	absolute	...	youth	zane	zany	zellweger	zero	zeta	zombie	zone
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 5000 columns

In [14]:

```
from sklearn.preprocessing import LabelEncoder
Encoder = LabelEncoder()
df['class2'] = Encoder.fit_transform(df['class'])
print(df['class'])
print(df['class2'])
```

```
0      Pos
1      Pos
2      Pos
3      Pos
4      Pos
5      Pos
6      Pos
7      Pos
8      Pos
9      Pos
```

```
9       Pos
10      Pos
11      Pos
12      Pos
13      Pos
14      Pos
15      Pos
16      Pos
17      Pos
18      Pos
19      Pos
20      Pos
21      Pos
22      Pos
23      Pos
24      Pos
25      Pos
26      Pos
27      Pos
28      Pos
29      Pos
       ...
1970    Neg
1971    Neg
1972    Neg
1973    Neg
1974    Neg
1975    Neg
1976    Neg
1977    Neg
1978    Neg
1979    Neg
1980    Neg
1981    Neg
1982    Neg
1983    Neg
1984    Neg
1985    Neg
1986    Neg
1987    Neg
1988    Neg
1989    Neg
1990    Neg
1991    Neg
1992    Neg
1993    Neg
1994    Neg
1995    Neg
1996    Neg
1997    Neg
1998    Neg
1999    Neg
Name: class, Length: 2000, dtype: object
0       1
1       1
2       1
3       1
4       1
5       1
6       1
7       1
8       1
9       1
10      1
11      1
12      1
13      1
14      1
15      1
16      1
17      1
18      1
19      1
20      1
21      1
22      1
23      1
24      1
25      1
26      1
27      1
28      1
29      1
```

```
        ..
1970    0
1971    0
1972    0
1973    0
1974    0
1975    0
1976    0
1977    0
1978    0
1979    0
1980    0
1981    0
1982    0
1983    0
1984    0
1985    0
1986    0
1987    0
1988    0
1989    0
1990    0
1991    0
1992    0
1993    0
1994    0
1995    0
1996    0
1997    0
1998    0
1999    0
Name: class2, Length: 2000, dtype: int32
```

In [15]:

```
from sklearn.model_selection import train_test_split
Train_X, Test_X, Train_Y, Test_Y = train_test_split(BOW,df['class2'],test_size=0.2)
```

In [16]:

```
pd.DataFrame(Test_X, columns=cv.get_feature_names()).head()
```

Out[16]:

	aaron	abandon	ability	able	aboard	abound	abraham	absence	absent	absolute	...	youth	zane	zany	zellweger	zero	zeta	zombie	zone
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 5000 columns

In [17]:

```
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",round(accuracy_score(predictions_NB, Test_Y)*100,2),"%")
```

```
Naive Bayes Accuracy Score ->  83.25 %
```

In [18]:

```
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X,Train_Y)
# predict the labels on validation dataset
```

```
predictions_SVM = SVM.predict(Test_X)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",round(accuracy_score(predictions_SVM, Test_Y)*100,2),"%")
```

```
SVM Accuracy Score ->  83.25 %
```

In [19]:

```
# Fitting Random Forest Classification
# to the Training set
from sklearn.ensemble import RandomForestClassifier

# n_estimators can be said as number of
# trees, experiment with n_estimators
# to get better results
model = RandomForestClassifier(n_estimators = 501, criterion = 'entropy')
model.fit(Train_X, Train_Y)
```

Out[19]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=501,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [20]:

```
# Predicting the Test set results
y_pred = model.predict(Test_X)
```

In [21]:

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Test_Y, y_pred)
# Use accuracy_score function to get the accuracy
print("Random forest Accuracy Score -> ",round(accuracy_score(y_pred, Test_Y)*100,2),"%")
```

```
Random forest Accuracy Score ->  84.75 %
```

In [ ]:

```

```

Movie Review Analysis using TF-IDF

In [1]:

```
import pandas as pd
df = pd.read_csv('movie-Review.csv')
```

In [2]:

```
import numpy as np
np.random.seed(500)
```

In [3]:

```
#Remove number
import re # import all Regular expression functions
df['text']=[re.sub('\d','', i)for i in df['text']]
df.head(10)
```

Out[3]:

	class	text
0	Pos	films adapted from comic books have had plent...
1	Pos	every now and then a movie comes along from a...
2	Pos	you ve got mail works alot better than it des...
3	Pos	jaws is a rare film that grabs your atte...
4	Pos	moviemaking is a lot like being the general m...
5	Pos	on june a self taught idealistic ye...
6	Pos	apparently director tony kaye had a major b...
7	Pos	one of my colleagues was surprised when i tol...
8	Pos	after bloody clashes and independence won l...
9	Pos	the american action film has been slowly drow...

In [4]:

```
# Replace punctuations with a white space
import string
df['text']=[re.sub('[%s]' % re.escape(string.punctuation), ' ', i) for i in df['text']]
df.head(10)
```

Out[4]:

	class	text
0	Pos	films adapted from comic books have had plent...
1	Pos	every now and then a movie comes along from a...
2	Pos	you ve got mail works alot better than it des...
3	Pos	jaws is a rare film that grabs your atte...
4	Pos	moviemaking is a lot like being the general m...
5	Pos	on june a self taught idealistic ye...
6	Pos	apparently director tony kaye had a major b...
7	Pos	one of my colleagues was surprised when i tol...
8	Pos	after bloody clashes and independence won l...
9	Pos	the american action film has been slowly drow...

In [5]:

```
df['text']=[i.lower() for i in df['text']]
```

In [6]:

```
# import pandas as pd
import pandas as pd
#Word Tokenization
import nltk # import package for tokenization
#nltk.download('punkt') # download all spporting function /files for NLTK package
from nltk.tokenize import word_tokenize
df['text_wt'] = [word_tokenize(i) for i in df['text']]
df.head()
```

Out[6]:

	class	text	text_wt
0	Pos	films adapted from comic books have had plent...	[films, adapted, from, comic, books, have, had...
1	Pos	every now and then a movie comes along from a...	[every, now, and, then, a, movie, comes, along...
2	Pos	you ve got mail works alot better than it des...	[you, ve, got, mail, works, alot, better, than...
3	Pos	jaws is a rare film that grabs your atte...	[jaws, is, a, rare, film, that, grabs, your, a...
4	Pos	moviemaking is a lot like being the general m...	[moviemaking, is, a, lot, like, being, the, ge...

In [7]:

```
#To show the stop words
#nltk.download('stopwords') #download Stopwords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
#Remove All Stop Word
df['text_SW'] = [[i for i in j if not i in stop_words] for j in df['text_wt']]# remove the word which is aviable in stopword libr
df.head()
```

Out[7]:

	class	text	text_wt	text_SW
0	Pos	films adapted from comic books have had plent...	[films, adapted, from, comic, books, have, had...	[films, adapted, comic, books, plenty, success...
1	Pos	every now and then a movie comes along from a...	[every, now, and, then, a, movie, comes, along...	[every, movie, comes, along, suspect, studio, ...
2	Pos	you ve got mail works alot better than it des...	[you, ve, got, mail, works, alot, better, than...	[got, mail, works, alot, better, deserves, ord...
3	Pos	jaws is a rare film that grabs your atte...	[jaws, is, a, rare, film, that, grabs, your, a...	[jaws, rare, film, grabs, attention, shows, si...
4	Pos	moviemaking is a lot like being the general m...	[moviemaking, is, a, lot, like, being, the, ge...	[moviemaking, lot, like, general, manager, nfl...

In [8]:

```
#nltk.download('tagsets')
#nltk.help.upenn_tagset()# tagset documentation
#nltk.download('wordnet')
from collections import defaultdict #Default Dictionary is imported from collections
from nltk.corpus import wordnet as wn #the corpus reader wordnet is imported.
from nltk.tag import pos_tag
# WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc.
#By default it is set to Noun
tag_map = defaultdict(lambda : wn.NOUN) #Dictionary is created where pos_tag (first letter) are the key values
tag_map['J'] = wn.ADJ             #whose values are mapped with the value
tag_map['V'] = wn.VERB              #from wordnet dictionary. We have taken the only first letter as
tag_map['R'] = wn.ADV
# we will use it later in the loop.
#tag_map
```

In [9]:

```
#lemmatization
from nltk.stem import WordNetLemmatizer
 # Initializing WordNetLemmatizer()
lemmatizer = WordNetLemmatizer()

df['lemma']=[[lemmatizer.lemmatize(word,tag_map[tag[0]]) for word ,tag in pos_tag(i)] for i in df['text_SW']]
df.head()
```

Out[9]:

	class	text	text_wt	text_SW	lemma
0	Pos	films adapted from comic books have had plent...	[films, adapted, from, comic, books, have, had...	[films, adapted, comic, books, plenty, success...	[film, adapt, comic, book, plenty, success, wh...
1	Pos	every now and then a movie comes along from a...	[every, now, and, then, a, movie, comes, along...	[every, movie, comes, along, suspect, studio, ...	[every, movie, come, along, suspect, studio, e...
2	Pos	you ve got mail works alot better than it des...	[you, ve, got, mail, works, alot, better, than...	[got, mail, works, alot, better, deserves, ord...	[get, mail, work, alot, good, deserves, order,...
3	Pos	jaws is a rare film that grabs your atte...	[jaws, is, a, rare, film, that, grabs, your, a...	[jaws, rare, film, grabs, attention, shows, si...	[jaw, rare, film, grab, attention, show, singl...
4	Pos	moviemaking is a lot like being the general m...	[moviemaking, is, a, lot, like, being, the, ge...	[moviemaking, lot, like, general, manager, nfl...	[moviemaking, lot, like, general, manager, nfl...

In [10]:

```
df['lemma2']= df['lemma'].apply(lambda x: ' '.join(x))
```

In [11]:

```
df['lemma2'].head()
```

Out[11]:

```
0    film adapt comic book plenty success whether s...
1    every movie come along suspect studio every in...
2    get mail work alot good deserves order make fi...
3    jaw rare film grab attention show single image...
4    moviemaking lot like general manager nfl team ...
Name: lemma2, dtype: object
```

In [12]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer(max_features=5000)
Tfidf= tf.fit_transform(df['lemma2']).toarray()
```

In [13]:

```
pd.DataFrame(Tfidf, columns=tf.get_feature_names()).head()
```

Out[13]:

	aaron	abandon	ability	able	aboard	abound	abraham	absence	absent	absolute	...	youth	zane	zany	zellweger	zero	zeta	zombie	zone
0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.036021	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 5000 columns

In [14]:

```
from sklearn.preprocessing import LabelEncoder
Encoder = LabelEncoder()
df['class2'] = Encoder.fit_transform(df['class'])
print(df['class'])
print(df['class2'])
```

```
0      Pos
1      Pos
2      Pos
3      Pos
4      Pos
5      Pos
6      Pos
7      Pos
8      Pos
9      Pos
10     Pos
11     Pos
```

```
12     Pos
13     Pos
14     Pos
15     Pos
16     Pos
17     Pos
18     Pos
19     Pos
20     Pos
21     Pos
22     Pos
23     Pos
24     Pos
25     Pos
26     Pos
27     Pos
28     Pos
29     Pos
      ...
1970    Neg
1971    Neg
1972    Neg
1973    Neg
1974    Neg
1975    Neg
1976    Neg
1977    Neg
1978    Neg
1979    Neg
1980    Neg
1981    Neg
1982    Neg
1983    Neg
1984    Neg
1985    Neg
1986    Neg
1987    Neg
1988    Neg
1989    Neg
1990    Neg
1991    Neg
1992    Neg
1993    Neg
1994    Neg
1995    Neg
1996    Neg
1997    Neg
1998    Neg
1999    Neg
Name: class, Length: 2000, dtype: object
0       1
1       1
2       1
3       1
4       1
5       1
6       1
7       1
8       1
9       1
10      1
11      1
12      1
13      1
14      1
15      1
16      1
17      1
18      1
19      1
20      1
21      1
22      1
23      1
24      1
25      1
26      1
27      1
28      1
29      1
       ..
1970    0
1971    0
```

```
1972    0
1973    0
1974    0
1975    0
1976    0
1977    0
1978    0
1979    0
1980    0
1981    0
1982    0
1983    0
1984    0
1985    0
1986    0
1987    0
1988    0
1989    0
1990    0
1991    0
1992    0
1993    0
1994    0
1995    0
1996    0
1997    0
1998    0
1999    0
Name: class2, Length: 2000, dtype: int32
```

In [21]:

```python
from sklearn.model_selection import train_test_split
Train_X, Test_X, Train_Y, Test_Y = train_test_split(Tfidf,df['class2'],test_size=0.2)
```

In [22]:

```python
pd.DataFrame(Test_X, columns=tf.get_feature_names()).head()
```

Out[22]:

	aaron	abandon	ability	able	aboard	abound	abraham	absence	absent	absolute	...	youth	zane	zany	zellweger	zero	zeta	zombie	zone
0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.03945	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 5000 columns

In [23]:

```python
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
# fit the training dataset on the NB classifier
Naive = naive_bayes.MultinomialNB()
Naive.fit(Train_X,Train_Y)
# predict the labels on validation dataset
predictions_NB = Naive.predict(Test_X)
# Use accuracy_score function to get the accuracy
print("Naive Bayes Accuracy Score -> ",round(accuracy_score(predictions_NB, Test_Y)*100,2),"%")
```

```
Naive Bayes Accuracy Score ->  80.25 %
```

In [24]:

```python
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",round(accuracy_score(predictions_SVM, Test_Y)*100,2),"%")
```

```
SVM Accuracy Score ->  81.75 %
```

In [25]:

```python
# Fitting Random Forest Classification
# to the Training set
from sklearn.ensemble import RandomForestClassifier

# n_estimators can be said as number of
# trees, experiment with n_estimators
# to get better results
model = RandomForestClassifier(n_estimators = 501, criterion = 'entropy')
model.fit(Train_X, Train_Y)
```

Out[25]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=501,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [26]:

```python
# Predicting the Test set results
y_pred = model.predict(Test_X)
```

In [27]:

```python
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Test_Y, y_pred)
# Use accuracy_score function to get the accuracy
print("Random forest Accuracy Score -> ",round(accuracy_score(y_pred, Test_Y)*100,2),"%")
```

```
Random forest Accuracy Score ->  81.0 %
```

In [ ]:

```python

```

SMS Spam Analysis using BOW

Data Source:https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

In [1]:

```
import numpy as np
np.random.seed(500)
import pandas as pd
df=pd.read_csv('smsspamcollection/SMSSpamCollection',sep='\t',names=["class","text"])
```

In [2]:

```
df.head()
```

Out[2]:

	class	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...

In [3]:

```
class_count=df.groupby('class').count()
print(class_count)
import matplotlib.pyplot as plt
plt.bar(class_count.index.values, class_count['text'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```

```
       text
class
ham    4825
spam    747
```

```
<Figure size 640x480 with 1 Axes>
```

In [4]:

```
#Remove number
import re # import all Regular expression functions
df['text_RN']=[re.sub('\d','', i)for i in df['text']]
df.head(10)
```

Out[4]:

	class	text	text_RN
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...
3	ham	U dun say so early hor... U c already then say...	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...	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...	FreeMsg Hey there darling it's been week's no...
6	ham	Even my brother is not like to speak with me. ...	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...	Had your mobile months or more? U R entitled ...

class text text_RN

In [5]:

```
# Replace punctuations with a white space
import string
df['text_RP']=[re.sub('[%s]' % re.escape(string.punctuation), ' ', i) for i in df['text_RN']]
df.head(10)
```

Out[5]:

	class	text	text_RN	text_RP
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	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...	Nah I don't think he goes to usf, he lives aro...	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...	FreeMsg Hey there darling it's been week's no...	FreeMsg Hey there darling it s been week s no...
6	ham	Even my brother is not like to speak with me. ...	Even my brother is not like to speak with me. ...	Even my brother is not like to speak with me ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...	As per your request 'Melle Melle (Oru Minnamin...	As per your request Melle Melle Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...	WINNER!! As a valued network customer you have...	WINNER As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...	Had your mobile months or more? U R entitled ...	Had your mobile months or more U R entitled ...

In [6]:

```
df['text_lw']=[i.lower() for i in df['text_RN']]
df.head()
```

Out[6]:

	class	text	text_RN	text_RP	text_lw
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	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...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...

In [7]:

```
# import pandas as pd
import pandas as pd
#Word Tokenization
import nltk # import package for tokenization
#nltk.download('punkt') # download all spporting function /files for NLTK package
from nltk.tokenize import word_tokenize
df['text_wt'] = [word_tokenize(i) for i in df['text_lw']]
df.head()
```

Out[7]:

	class	text	text_RN	text_RP	text_lw	text_wt
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...	[go, until, jurong, point, ,, crazy.., availab...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...	[ok, lar, ..., joking, wif, u, oni, ...]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...	[free, entry, in, a, wkly, comp, to, win, fa, ...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	u dun say so early hor... u c already then say...	[u, dun, say, so, early, hor, ..., u, c, alrea...
4	ham	Nah I don't think he goes to usf, he lives aro...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...	[nah, i, do, n't, think, he, goes, to, usf, ,,...

In [8]:

```
#To show the stop words
#nltk.download('stopwords') #download Stopwords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
#Remove All Stop Word
df['text_SW'] = [[i for i in j if not i in stop_words] for j in df['text_wt']]# remove the word which is aviable in stopword libr
df.head()
```

Out[8]:

	class	text	text_RN	text_RP	text_lw	text_wt	text_SW
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...	[go, until, jurong, point, ,, crazy.., availab...	[go, jurong, point, ,, crazy.., available, bug...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...	[ok, lar, ..., joking, wif, u, oni, ...]	[ok, lar, ..., joking, wif, u, oni, ...]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...	[free, entry, in, a, wkly, comp, to, win, fa, ...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	u dun say so early hor... u c already then say...	[u, dun, say, so, early, hor, ..., u, c, alrea...	[u, dun, say, early, hor, ..., u, c, already, ...
4	ham	Nah I don't think he goes to usf, he lives aro...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...	[nah, i, do, n't, think, he, goes, to, usf, ,,...	[nah, n't, think, goes, usf, ,, lives, around,...

In [9]:

```
#nltk.download('tagsets')
#nltk.help.upenn_tagset()# tagset documentation
#nltk.download('wordnet')
from collections import defaultdict #Default Dictionary is imported from collections
from nltk.corpus import wordnet as wn #the corpus reader wordnet is imported.
from nltk.tag import pos_tag
# WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc.
#By default it is set to Noun
tag_map = defaultdict(lambda : wn.NOUN) #Dictionary is created where pos_tag (first letter) are the key values
tag_map['J'] = wn.ADJ                   #whose values are mapped with the value
tag_map['V'] = wn.VERB                    #from wordnet dictionary. We have taken the only first letter as
tag_map['R'] = wn.ADV
# we will use it later in the loop.
#tag_map
```

In [10]:

```
#lemmatization
from nltk.stem import WordNetLemmatizer
 # Initializing WordNetLemmatizer()
lemmatizer = WordNetLemmatizer()

df['lemma']=[[lemmatizer.lemmatize(word,tag_map[tag[0]]) for word ,tag in pos_tag(i)] for i in df['text_SW']]
df.head()
```

Out[10]:

	class	text	text_RN	text_RP	text_lw	text_wt	text_SW	lemma
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...	[go, until, jurong, point, ,, crazy.., availab...	[go, jurong, point, ,, crazy.., available, bug...	[go, jurong, point, ,, crazy.., available, bug...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...	[ok, lar, ..., joking, wif, u, oni, ...]	[ok, lar, ..., joking, wif, u, oni, ...]	[ok, lar, ..., joke, wif, u, oni, ...]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...	[free, entry, in, a, wkly, comp, to, win, fa, ...	[free, entry, wkly, comp, win, fa, cup, final,...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	u dun say so early hor... u c already then say...	[u, dun, say, so, early, hor, ..., u, c, alrea...	[u, dun, say, early, hor, ..., u, c, already, ...	[u, dun, say, early, hor, ..., u, c, already, ...
4	ham	Nah I don't think he goes to usf, he lives aro...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...	[nah, i, do, n't, think, he, goes, to, usf, ,,...	[nah, n't, think, goes, usf, ,, lives, around,...	[nah, n't, think, go, usf, ,, live, around, th...

In [11]:

```
df['lemma2']= df['lemma'].apply(lambda x: ' '.join(x))
```

In [14]:

```
# Bag of Words
from sklearn.feature_extraction.text import CountVectorizer
#instantiate CountVectorizer()# CountVectorizer to count the number of words (term frequency)
cv = CountVectorizer(max_features=5000)
#this steps generates word counts for the words in your docs and return term-document matrix.
BOW = cv.fit_transform(df['lemma2']).toarray()
```

In [15]:

```
pd.DataFrame(BOW, columns=cv.get_feature_names()).head()
```

Out[15]:

	____	aa	aah	aaniye	aaooooright	aathi	ab	abbey	abdomen	abeg	...	zed	zero	zf	zhong	zindgi	zoe	zogtorius	zoom	zouk	zyada
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 5000 columns

In [16]:

```
from sklearn.model_selection import train_test_split
Train_X, Test_X, Train_Y1, Test_Y1 = train_test_split(BOW,df['class'],test_size=0.3)
```

In [17]:

```
from sklearn.preprocessing import LabelEncoder
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y1)
Test_Y = Encoder.fit_transform(Test_Y1)
print(Train_Y1[1:5])
print(Train_Y[1:5])
print(Test_Y1[1:5])
print(Test_Y[1:5])
```

```
1481     ham
3894     ham
4050     ham
3010    spam
Name: class, dtype: object
[0 0 0 1]
3758    spam
3089     ham
1298     ham
3574    spam
Name: class, dtype: object
[1 0 0 1]
```

In [18]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(Train_X, Train_Y)
predicted= clf.predict(Test_X)
print("MultinomialNB Accuracy:",round(accuracy_score(predicted,Test_Y)*100,2),"%")
```

```
MultinomialNB Accuracy: 97.91 %
```

In [19]:

```
from sklearn import model_selection, svm
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",round(accuracy_score(predictions_SVM, Test_Y)*100,2),"%")
```

```
SVM Accuracy Score ->  97.85 %
```

In [20]:

```
# Fitting Random Forest Classification
# to the Training set
from sklearn.ensemble import RandomForestClassifier

# n_estimators can be said as number of
# trees, experiment with n_estimators
# to get better results
model = RandomForestClassifier(n_estimators = 501, criterion = 'entropy')
model.fit(Train_X, Train_Y)
```

Out[20]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=501,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [21]:

```
# Predicting the Test set results
y_pred = model.predict(Test_X)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Test_Y, y_pred)
print("Confusion Matrix ->",cm)
# Use accuracy_score function to get the accuracy
print("Random forest Accuracy Score -> ",round(accuracy_score(y_pred, Test_Y)*100,2),"%")
```

```
Confusion Matrix -> [[1436    0]
 [  48  188]]
Random forest Accuracy Score ->  97.13 %
```

In [ ]:

```

```

SMS Spam Analysis using TF-IDF

Data Source:https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

In [1]:

```
import numpy as np
np.random.seed(500)
import pandas as pd
df=pd.read_csv('smsspamcollection/SMSSpamCollection',sep='\t',names=["class","text"])
```

In [2]:

```
df.head()
```

Out[2]:

	class	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...

In [3]:

```
class_count=df.groupby('class').count()
print(class_count)
import matplotlib.pyplot as plt
plt.bar(class_count.index.values, class_count['text'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```

```
       text
class
ham    4825
spam    747
```

```
<Figure size 640x480 with 1 Axes>
```

In [4]:

```
#Remove number
import re # import all Regular expression functions
df['text_RN']=[re.sub('\d','', i)for i in df['text']]
df.head(10)
```

Out[4]:

	class	text	text_RN
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...
3	ham	U dun say so early hor... U c already then say...	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...	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...	FreeMsg Hey there darling it's been week's no...
6	ham	Even my brother is not like to speak with me. ...	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...	Had your mobile months or more? U R entitled ...

class	text	text_RN

In [5]:

```
# Replace punctuations with a white space
import string
df['text_RP']=[re.sub('[%s]' % re.escape(string.punctuation), ' ', i) for i in df['text_RN']]
df.head(10)
```

Out[5]:

	class	text	text_RN	text_RP
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	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...	Nah I don't think he goes to usf, he lives aro...	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...	FreeMsg Hey there darling it's been week's no...	FreeMsg Hey there darling it s been week s no...
6	ham	Even my brother is not like to speak with me. ...	Even my brother is not like to speak with me. ...	Even my brother is not like to speak with me ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...	As per your request 'Melle Melle (Oru Minnamin...	As per your request Melle Melle Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...	WINNER!! As a valued network customer you have...	WINNER As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...	Had your mobile months or more? U R entitled ...	Had your mobile months or more U R entitled ...

In [6]:

```
df['text_lw']=[i.lower() for i in df['text_RN']]
df.head()
```

Out[6]:

	class	text	text_RN	text_RP	text_lw
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	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...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...

In [7]:

```
# import pandas as pd
import pandas as pd
#Word Tokenization
import nltk # import package for tokenization
#nltk.download('punkt') # download all spporting function /files for NLTK package
from nltk.tokenize import word_tokenize
df['text_wt'] = [word_tokenize(i) for i in df['text_lw']]
df.head()
```

Out[7]:

	class	text	text_RN	text_RP	text_lw	text_wt
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...	[go, until, jurong, point, ,, crazy.., availab...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...	[ok, lar, ..., joking, wif, u, oni, ...]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...	[free, entry, in, a, wkly, comp, to, win, fa, ...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	u dun say so early hor... u c already then say...	[u, dun, say, so, early, hor, ..., u, c, alrea...
4	ham	Nah I don't think he goes to usf, he lives aro...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...	[nah, i, do, n't, think, he, goes, to, usf, ,,...

In [8]:

```
#To show the stop words
#nltk.download('stopwords') #download Stopwords
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
#Remove All Stop Word
df['text_SW'] = [[i for i in j if not i in stop_words] for j in df['text_wt']]# remove the word which is aviable in stopword libr
df.head()
```

Out[8]:

	class	text	text_RN	text_RP	text_lw	text_wt	text_SW
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...	[go, until, jurong, point, ,, crazy.., availab...	[go, jurong, point, ,, crazy.., available, bug...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...	[ok, lar, ..., joking, wif, u, oni, ...]	[ok, lar, ..., joking, wif, u, oni, ...]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...	[free, entry, in, a, wkly, comp, to, win, fa, ...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	u dun say so early hor... u c already then say...	[u, dun, say, so, early, hor, ..., u, c, alrea...	[u, dun, say, early, hor, ..., u, c, already, ...
4	ham	Nah I don't think he goes to usf, he lives aro...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...	[nah, i, do, n't, think, he, goes, to, usf, ,,...	[nah, n't, think, goes, usf, ,, lives, around,...

In [9]:

```
#nltk.download('tagsets')
#nltk.help.upenn_tagset()# tagset documentation
#nltk.download('wordnet')
from collections import defaultdict #Default Dictionary is imported from collections
from nltk.corpus import wordnet as wn #the corpus reader wordnet is imported.
from nltk.tag import pos_tag
# WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc.
#By default it is set to Noun
tag_map = defaultdict(lambda : wn.NOUN) #Dictionary is created where pos_tag (first letter) are the key values
tag_map['J'] = wn.ADJ                #whose values are mapped with the value
tag_map['V'] = wn.VERB                 #from wordnet dictionary. We have taken the only first letter as
tag_map['R'] = wn.ADV
# we will use it later in the loop.
#tag_map
```

In [10]:

```
#lemmatization
from nltk.stem import WordNetLemmatizer
 # Initializing WordNetLemmatizer()
lemmatizer = WordNetLemmatizer()

df['lemma']=[[lemmatizer.lemmatize(word,tag_map[tag[0]]) for word ,tag in pos_tag(i)] for i in df['text_SW']]
df.head()
```

Out[10]:

	class	text	text_RN	text_RP	text_lw	text_wt	text_SW	lemma
0	ham	Go until jurong point, crazy.. Available only ...	Go until jurong point, crazy.. Available only ...	Go until jurong point crazy Available only ...	go until jurong point, crazy.. available only ...	[go, until, jurong, point, ,, crazy.., availab...	[go, jurong, point, ,, crazy.., available, bug...	[go, jurong, point, ,, crazy.., available, bug...
1	ham	Ok lar... Joking wif u oni...	Ok lar... Joking wif u oni...	Ok lar Joking wif u oni	ok lar... joking wif u oni...	[ok, lar, ..., joking, wif, u, oni, ...]	[ok, lar, ..., joking, wif, u, oni, ...]	[ok, lar, ..., joke, wif, u, oni, ...]
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry in a wkly comp to win FA Cup final...	Free entry in a wkly comp to win FA Cup final...	free entry in a wkly comp to win fa cup final...	[free, entry, in, a, wkly, comp, to, win, fa, ...	[free, entry, wkly, comp, win, fa, cup, final,...	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	U dun say so early hor... U c already then say...	U dun say so early hor... U c already then say...	U dun say so early hor U c already then say	u dun say so early hor... u c already then say...	[u, dun, say, so, early, hor, ..., u, c, alrea...	[u, dun, say, early, hor, ..., u, c, already, ...	[u, dun, say, early, hor, ..., u, c, already, ...
4	ham	Nah I don't think he goes to usf, he lives aro...	Nah I don't think he goes to usf, he lives aro...	Nah I don t think he goes to usf he lives aro...	nah i don't think he goes to usf, he lives aro...	[nah, i, do, n't, think, he, goes, to, usf, ,,...	[nah, n't, think, goes, usf, ,, lives, around,...	[nah, n't, think, go, usf, ,, live, around, th...

In [11]:

```
df['lemma2']= df['lemma'].apply(lambda x: ' '.join(x))
```

In [12]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer(max_features=5000)
Tfidf= tf.fit_transform(df['lemma2']).toarray()
```

In [14]:

```
pd.DataFrame(Tfidf, columns=tf.get_feature_names()).head()
```

Out[14]:

	____	aa	aah	aaniye	aaooooright	aathi	ab	abbey	abdomen	abeg	...	zed	zero	zf	zhong	zindgi	zoe	zogtorius	zoom	zouk	zyada
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 5000 columns

In [15]:

```
from sklearn.model_selection import train_test_split
Train_X, Test_X, Train_Y1, Test_Y1 = train_test_split(Tfidf,df['class'],test_size=0.3)
```

In [16]:

```
from sklearn.preprocessing import LabelEncoder
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y1)
Test_Y = Encoder.fit_transform(Test_Y1)
print(Train_Y1[1:5])
print(Train_Y[1:5])
print(Test_Y1[1:5])
print(Test_Y[1:5])
```

```
1481     ham
3894     ham
4050     ham
3010    spam
Name: class, dtype: object
[0 0 0 1]
3758    spam
3089     ham
1298     ham
3574    spam
Name: class, dtype: object
[1 0 0 1]
```

In [17]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(Train_X, Train_Y)
predicted= clf.predict(Test_X)
print("MultinomialNB Accuracy:",round(accuracy_score(predicted,Test_Y)*100,2),"%")
```

```
MultinomialNB Accuracy: 96.17 %
```

In [18]:

```
from sklearn import model_selection, svm
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X,Train_Y)
# predict the labels on validation dataset
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",round(accuracy_score(predictions_SVM, Test_Y)*100,2),"%")
```

```
SVM Accuracy Score ->  97.79 %
```

In [19]:

```
# Fitting Random Forest Classification
# to the Training set
from sklearn.ensemble import RandomForestClassifier

# n_estimators can be said as number of
# trees, experiment with n_estimators
# to get better results
model = RandomForestClassifier(n_estimators = 501, criterion = 'entropy')
model.fit(Train_X, Train_Y)
```

Out[19]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=501,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [20]:

```
# Predicting the Test set results
y_pred = model.predict(Test_X)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Test_Y, y_pred)
print("Confusion Matrix ->",cm)
# Use accuracy_score function to get the accuracy
print("Random forest Accuracy Score -> ",round(accuracy_score(y_pred, Test_Y)*100,2),"%")
```

```
Confusion Matrix -> [[1436    0]
 [  44  192]]
Random forest Accuracy Score ->  97.37 %
```

In [ ]:

```

```