1.Importing Libraries

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

2.Loading Data

In [2]:
```
data = pd.read_csv(r'C:\Users\vamsi\Desktop\M.Tech\ML\Natural Language Processing
```

In [3]:
```
data.head()
```

Out[3]:

	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 [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

3.Data Preprocessing

In [5]:
```
#Handling Null Values
data = data.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
```

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

Out[6]:

	v1	v2
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 [7]:
```
data.rename(columns = {"v1" : "Label", "v2" : "Message"},inplace=True)
```

In [8]:
```
data.head()
```

Out[8]:

	Label	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...

Handling Categorical data

In [9]:
```
data['Label'].unique()
```

Out[9]:
```
array(['ham', 'spam'], dtype=object)
```

In [10]:
```
data['Label'].value_counts()
```

Out[10]:
```
ham     4825
spam     747
Name: Label, dtype: int64
```

In [11]:
```
data = pd.get_dummies(data,columns=['Label'])
data.head()
```

Out[11]:

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

In [12]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Message     5572 non-null   object
 1   Label_ham   5572 non-null   uint8
 2   Label_spam  5572 non-null   uint8
dtypes: object(1), uint8(2)
memory usage: 54.5+ KB
```

In [13]:
```
data['Count'] = 0
for i in np.arange(0,len(data.Message)):
    data.loc[i,'Count']=len(data.loc[i,'Message'])
data.head()
```

Out[13]:

	Message	Label_ham	Label_spam	Count
0	Go until jurong point, crazy.. Available only ...	1	0	111
1	Ok lar... Joking wif u oni...	1	0	29
2	Free entry in 2 a wkly comp to win FA Cup fina...	0	1	155
3	U dun say so early hor... U c already then say...	1	0	49
4	Nah I don't think he goes to usf, he lives aro...	1	0	61

In [14]:
```
data.describe()
#No Statistics about message
```

Out[14]:

	Label_ham	Label_spam	Count
count	5572.000000	5572.000000	5572.000000
mean	0.865937	0.134063	80.118808
std	0.340751	0.340751	59.690841
min	0.000000	0.000000	2.000000
25%	1.000000	0.000000	36.000000
50%	1.000000	0.000000	61.000000
75%	1.000000	0.000000	121.000000
max	1.000000	1.000000	910.000000

Processing Message

In [15]:
```
data['Message'][0]
```

Out[15]:
```
'Go until jurong point, crazy.. Available only in bugis n great world la e buff
et... Cine there got amore wat...'
```

In [16]:
```
data['Message'][1]
```

Out[16]:
```
'Ok lar... Joking wif u oni...'
```

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

Out[17]:
```
(5572, 4)
```

Preparing WordVector Corpus

In [18]:
```
corpus = []
```

Using Porter stemmer

In [19]:
```
from nltk.stem.porter import PorterStemmer
import re
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer

ps = PorterStemmer()
```

In [20]:
```
stopwords.words('english')
```

Out[20]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his'
```

In [21]:
```
# re.sub(pattern,replacement_string,string)

for i in range(0,5572):
    #Regular expressions
    msg = data['Message'][i]
    #email_addresses
    msg = re.sub('\\b[\\w\\-.]+?@\\w+?\\.\\w{2,4}\\b', 'emailaddr', msg)
    #url's
    msg = re.sub('(http[s]?\\S+)|(\\w+\\.[A-Za-z]{2,4}\\S*)', 'httpaddr', msg)
    #money
    msg = re.sub('£|\\$', 'moneysymb', msg)
    #phone_number
    msg = re.sub('\\b(\\+\\d{1,2}\\s)?\\d?[\\-(.]?\\d{3}\\)?[\\s.-]?\\d{3}[\\s.-]
    #numbers
    msg = re.sub('\\d+(\\.\\d+)?', 'numbr', msg)
    #removing_punctuations
    msg = re.sub('[^\\w\\d\\s]', ' ', msg)

    #lower case
    msg = msg.lower()

    #Splitting to tokens
    msg = msg.split()

    #Stop-word Removal & Stemming
    msg = [ps.stem(word) for word in msg if not word in set(stopwords.words('engl

    #join
    msg = ' '.join(msg)

    print(msg,"\n")
    #adding to corpus list
    corpus.append(msg)
```

```
go jurong point crazi avail bugi n great world la e buffet cine got amor wat

ok lar joke wif u oni

free entri numbr wkli comp win fa cup final tkt numbrst may numbr text fa num
br receiv entri question std txt rate c appli numbrovernumbr

u dun say earli hor u c alreadi say

nah think goe usf live around though

freemsg hey darl numbr week word back like fun still tb ok xxx std chg send å
moneysymbnumbr rcv

even brother like speak treat like aid patent

per request mell mell oru minnaminungint nurungu vettam set callertun caller
press numbr copi friend callertun
```

In [22]:
```
len(corpus)
```

Out[22]:
```
5572
```

Preparing Vectors for each message

In [23]:
```
cv = CountVectorizer()
data_input = cv.fit_transform(corpus).toarray()
```

In [24]:
```
data_input[0]
```

Out[24]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [25]:
```
len(data_input[0])
```

Out[25]:
```
6386
```

Applying Classification

- Input : Vector for each message
- Output : Lables i.e. spam/ham

In [26]:
```
data_input
```

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

In [27]:
```
data_output = data['Label_ham']
```

In [28]:
```
data_output
```

Out[28]:
```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Label_ham, Length: 5572, dtype: uint8
```

Splitting data for training and testing

In [29]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(data_input,data_output,test_size
```

Training the Model

In [30]:
```
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report
```

In [31]:
```
model_nvb = GaussianNB()
model_nvb.fit(x_train,y_train)

model_rfc = RandomForestClassifier(n_estimators=100)
model_rfc.fit(x_train,y_train)

model_dtc = DecisionTreeClassifier()
model_dtc.fit(x_train,y_train)
```

Out[31]:
```
DecisionTreeClassifier()
```

Predictions

In [32]:
```
pred_nvb = model_nvb.predict(x_test)
pred_rfc = model_rfc.predict(x_test)
pred_dtc = model_dtc.predict(x_test)
```

In [33]:
```
#Naive Bayes Classifier Result
print("\t\tNaive Bayes Accuracy : ",accuracy_score(y_test,pred_nvb)*100," %\n")
print("Classification Report : \n",classification_report(y_test,pred_nvb))
#Random Forest Classifier Result
print("\t\tRandom Forest Accuracy : ",accuracy_score(y_test,pred_rfc)*100," %\n")
print("Classification Report : \n",classification_report(y_test,pred_rfc))
#Decision Tree Classifier Result
print("\t\Random Forest Accuracy : ",accuracy_score(y_test,pred_dtc)*100," %\n")
print("Classification Report : \n",classification_report(y_test,pred_dtc))
```

```
                Naive Bayes Accuracy :  86.3677130044843  %

Classification Report :
               precision    recall  f1-score   support

           0       0.53      0.85      0.65       166
           1       0.97      0.87      0.92       949

    accuracy                           0.86      1115
   macro avg       0.75      0.86      0.78      1115
weighted avg       0.90      0.86      0.88      1115

                Random Forest Accuracy :  97.75784753363229  %

Classification Report :
               precision    recall  f1-score   support

           0       1.00      0.85      0.92       166
           1       0.97      1.00      0.99       949

    accuracy                           0.98      1115
   macro avg       0.99      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115

        \Random Forest Accuracy :  96.95067264573991  %

Classification Report :
               precision    recall  f1-score   support

           0       0.92      0.87      0.90       166
           1       0.98      0.99      0.98       949

    accuracy                           0.97      1115
   macro avg       0.95      0.93      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```