In [5]:

```
## import statements ##
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline
```

```
UsageError: Line magic function `%` not found.
```

Type *Markdown* and LaTeX: α^2

In [6]:

```
train_data = []
```

In [22]:

```
data_files =['C:\\Users\\singh\\OneDrive\\Desktop\\Youtube-Comments-Spam-Detection-maste
for file in data_files:
    data = pd.read_csv(file)
    train_data.append(data)
train_data = pd.concat(train_data)
```

In [23]:

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1956 entries, 0 to 369
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   COMMENT_ID  1956 non-null   object
 1   AUTHOR      1956 non-null   object
 2   DATE        1711 non-null   object
 3   CONTENT     1956 non-null   object
 4   CLASS       1956 non-null   int64 
dtypes: int64(1), object(4)
memory usage: 91.7+ KB
```

In [24]:

```
train_data.head()
```

Out[24]:

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
0	LZQPQhLyRh80UYxNuaDWhIGQYNQ96IuCg-AYWqNPjpU	Julius NM	2013-11-07T06:20:48	Huh, anyway check out this you[tube] channel: ...	1
1	LZQPQhLyRh_C2cTtd9MvFRJedxydaVW-2sNg5Diuo4A	adam riyati	2013-11-07T12:37:15	Hey guys check out my new channel and our firs...	1
2	LZQPQhLyRh9MSZYnf8djyk0gEF9BHDPYrrK-qCczIY8	Evgeny Murashkin	2013-11-08T17:34:21	just for test I have to say murdev.com	1
3	z13jhp0bxqncu512g22wvzkasxmvvzjaz04	ElNino Melendez	2013-11-09T08:28:43	me shaking my sexy ass on my channel enjoy ^_^	1
4	z13fwbwp1oujthgqj04chlngpvzmtt3r3dw	GsMega	2013-11-10T16:05:38	watch?v=vtaRGgvGtWQ Check this out .	1

In [25]:

```
train_data['CLASS'].value_counts()
```

Out[25]:

```
1    1005
0     951
Name: CLASS, dtype: int64
```

In [ ]:

```
#Data Cleaning
#We don't need all the features. Hence drop all the columns except CONTENT, CLASS
```

In [26]:

```
def drop_fectures(features,data):
    data.drop(features,axis=1,inplace=True)
```

In [27]:

```
drop_fectures(['COMMENT_ID','AUTHOR','DATE'],train_data)
```

In [28]:

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1956 entries, 0 to 369
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   CONTENT  1956 non-null   object
 1   CLASS    1956 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 45.8+ KB
```

In [29]:

```
import re
re.findall("[A-Za-z]+","Hey Ravi57,How are you doing?")
```

Out[29]:

```
['Hey', 'Ravi', 'How', 'are', 'you', 'doing']
```

In [ ]:

```
#Processing the comments in such a way that they contains only alphabet
```

In [30]:

```
def process_content(content):
    return " ".join(re.findall("[A-Za-z]+",content.lower()))
```

In [31]:

```
train_data['processed_content'] = train_data['CONTENT'].apply(process_content)
```

In [32]:

```
train_data.head()
```

Out[32]:

	CONTENT	CLASS	processed_content
0	Huh, anyway check out this you[tube] channel: ...	1	huh anyway check out this you tube channel kob...
1	Hey guys check out my new channel and our firs...	1	hey guys check out my new channel and our firs...
2	just for test I have to say murdev.com	1	just for test i have to say murdev com
3	me shaking my sexy ass on my channel enjoy ^_^	1	me shaking my sexy ass on my channel enjoy
4	watch?v=vtaRGgvGtWQ Check this out .	1	watch v vtarggvgtwq check this out

In [33]:

```
drop_fectures(['CONTENT'],train_data)
```

In [ ]:

```
##Splitting the whole data into train and test sets
```

In [34]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(train_data['processed_content'],trai
```

In [35]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [36]:

```
count_vect = CountVectorizer(stop_words='english')
x_train_counts = count_vect.fit_transform(x_train)
```

In [37]:

```
x_train_counts.shape
```

Out[37]:

```
(1564, 3345)
```

In [ ]:

```
##  From occurrences to frequencies
##Occurrence count is a good start but there is an issue: longer documents will have higl
##To avoid these potential discrepancies it suffices to divide the number of occurrences
##Another refinement on top of tf is to downscale weights for words that occur in many d
##This downscaling is called tf-idf for “Term Frequency times Inverse Document Frequency
#Both tf and tf-idf can be computed as follows:
```

In [38]:

```
from sklearn.feature_extraction.text import TfidfTransformer
tranformer = TfidfTransformer()
x_train_tfidf = tranformer.fit_transform(x_train_counts)
x_train_tfidf.shape
```

Out[38]:

```
(1564, 3345)
```

In [39]:

```
x_test_counts = count_vect.transform(x_test)
x_test_counts
```

Out[39]:

```
<392x3345 sparse matrix of type '<class 'numpy.int64'>'
        with 2384 stored elements in Compressed Sparse Row format>
```

In [40]:

```
x_test_tfidf = tranformer.transform(x_test_counts)
x_test_tfidf
```

Out[40]:

```
<392x3345 sparse matrix of type '<class 'numpy.float64'>'
        with 2384 stored elements in Compressed Sparse Row format>
```

In [41]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train_tfidf,y_train)
```

Out[41]:

```
LogisticRegression()
```

In [42]:

```
predictions = model.predict(x_test_tfidf)
```

In [43]:

```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

In [44]:

```
confusion_matrix(y_test,predictions)
```

Out[44]:

```
array([[176,   5],
       [ 24, 187]], dtype=int64)
```

In [ ]:

```
#The above confusion matrix states that we classified (176 + 187) = 363 out of 392 comme
```

In [45]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92       181
           1       0.97      0.89      0.93       211

    accuracy                           0.93       392
   macro avg       0.93      0.93      0.93       392
weighted avg       0.93      0.93      0.93       392
```

In [46]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train_tfidf,y_train)
```

Out[46]:

```
RandomForestClassifier()
```

In [56]:

```
predictions = model.predict(x_test_tfidf)
```

In [57]:

```
confusion_matrix(y_test,predictions)
```

Out[57]:

```
array([[170,  11],
       [ 47, 164]], dtype=int64)
```

In [58]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.78      0.94      0.85       181
           1       0.94      0.78      0.85       211

    accuracy                           0.85       392
   macro avg       0.86      0.86      0.85       392
weighted avg       0.87      0.85      0.85       392
```

In [59]:

```
from sklearn.model_selection import GridSearchCV
parameters = {
                    'max_depth' : [1,3,4],
                    'n_estimators': [10,30,50],
                    'max_features': ['sqrt', 'auto', 'log2'],
                    'min_samples_split': [10,20,30],
                    'min_samples_leaf': [1, 3, 10],
                    'bootstrap': [True, False],
                    }
model = GridSearchCV(RandomForestClassifier(),parameters)
model.fit(x_train_tfidf,y_train)
```

Out[59]:

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'bootstrap': [True, False], 'max_depth': [1, 3,
4],
                         'max_features': ['sqrt', 'auto', 'log2'],
                         'min_samples_leaf': [1, 3, 10],
                         'min_samples_split': [10, 20, 30],
                         'n_estimators': [10, 30, 50]})
```

In [60]:

```
model.best_params_
```

Out[60]:

```
{'bootstrap': False,
 'max_depth': 3,
 'max_features': 'sqrt',
 'min_samples_leaf': 3,
 'min_samples_split': 30,
 'n_estimators': 50}
```

In [64]:

```
predictions  = model.predict(x_test_tfidf)
```

In [65]:

```
confusion_matrix(y_test,predictions)
```

Out[65]:

```
array([[173,   8],
       [ 47, 164]], dtype=int64)
```

In [66]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.79      0.96      0.86       181
           1       0.95      0.78      0.86       211

    accuracy                           0.86       392
   macro avg       0.87      0.87      0.86       392
weighted avg       0.88      0.86      0.86       392
```

In [ ]:

```

```

In [ ]:

```

```