AIM: Implementation of Spam Mail Filter.

Tool: Jupyter Notebook

Roll No. 412039

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
```

In [2]:

```
#Data Preprocessing
#Load Data Sets to pandas Data Frame
raw_mail_data = pd.read_csv('D:/SpamHam/spam.csv', encoding='ISO-8859-1')
raw_mail_data.head()
```

Out[2]:

	Category	Message	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 [3]:

```
#replace null values with null string
mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)),'')
mail_data.shape
```

Out[3]:

```
(5572, 5)
```

In [4]:

```
mail_data.head() #sample Data
```

Out[4]:

	Category	Message	Unnamed: 2	Unnamed: 3	Unnamed: 4
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 [6]:

```
#label spam mail as 0 ; Non-spam mail as (ham) mail as 1
mail_data.loc[mail_data['Category'] == 'spam','Category',] = 0
mail_data.loc[mail_data['Category'] == 'ham','Category',] = 1
```

In [7]:

```
#seperate data as text and label
# X --> text;
# Y --> label
X = mail_data['Message']
Y = mail_data['Category']
print(X)
print("-------------------------------------------------------")
print(Y)
```

```
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...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                            Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
-------------------------------------------------------
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

Train Test Split

In [8]:

```
#split the data as train data and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size = 0.8, test_size =
```

Feature Extraction

In [9]:

```
#transform text data to feature vectors that can be used as input to SVM model using Tfi
# Convert text to lower case
feature_extraction = TfidfVectorizer(min_df = 1, stop_words = 'english', lowercase = 'Tr
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

#convert Y_train and Y_test values as integer
Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

Training the model --> Support Vector Machine

In [10]:

```
#training the support vector machine model with training data
model = LinearSVC()
model.fit(X_train_features, Y_train)
```

Out[10]:

```
LinearSVC()
```

Evaluation of model

In [13]:

```
#prediction on training data
prediction_on_train_data = model.predict(X_train_features)
accuracy_on_train_data = accuracy_score(Y_train, prediction_on_train_data)
print("Accuracy on training data: ",accuracy_on_train_data)
```

```
Accuracy on training data:  0.9995512676688355
```

In [14]:

```
#prediction on test data
prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
print("Accuracy on test data: ",accuracy_on_test_data)
```

```
Accuracy on test data:  0.9856502242152466
```

Prediction on new mail

In [18]:

```
input_mail = ["I've been searching for the right words to thank you for this breather. I
#convert text to feature vectors
input_mail_features = feature_extraction.transform(input_mail)

#making predictions
prediction = model.predict(input_mail_features)
print(prediction)
if (prediction[0] == 1):
    print("It is Ham Mail!!")
else:
    print("It is a Spam!!")
```

```
[1]
It is Ham Mail!!
```

Conclusion: Hence, We have implemented the Spam Mail Prediction using the library called TfidfVectorizer which uses an in-memory vocabulary (a python dict) to map the most frequent words to feature indices and hence compute a word occurrence frequency (sparse) matrix to Filter out the spam mails.