EMAIL SPAM DETECTION WITH MACHINE LEARNING

We have all been the recipient of spam mails before. Spam mail,or junk mail, is a type of email that is sent to a massive number of users at one time, frequently containing cryptic messages, scams ,or most dangerously, phishing content. In this project , we use Python to build an email spam detector. Then use machine learning to train the spam detector to recognize and classify emails into spam and non-spam.

**Modules needed:

pandas: Pandas is an opensource library that allows you to perform data manipulation in Python. Pandas provide an easy way to create, manipulate and wrangle the data.

numpy: Numpy is the fundamental package for scientific computing with Python. numpy can be used as an efficient multi-dimensional container of generic data.

matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of formats.

seaborn: Seaborn is a Python data-visualization library that is based on matplotlib. Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.

scipy: Scipy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

In [4]:

```python
#importing Libraries
import pandas as pd
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
import re
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\meghana\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\meghana\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\meghana\AppData\Roaming\nltk_data...
```

In [6]:

```python
#reading of csv file
```

```python
df = pd.read_csv('email_spam_detection.csv',encoding = 'ISO-8859-1')
```

In [7]:

```python
#Displaying dataset
print("Email spam detection dataset is: \n",df)
```

```
Email spam detection dataset is:
         v1                                                 v2 Unnamed: 2  \
0      ham  Go until jurong point, crazy.. Available only ...        NaN
1      ham                      Ok lar... Joking wif u oni...        NaN
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN
3      ham  U dun say so early hor... U c already then say...        NaN
4      ham  Nah I don't think he goes to usf, he lives aro...        NaN
...    ...                                                ...        ...
5567  spam  This is the 2nd time we have tried 2 contact u...        NaN
5568   ham               Will Ì_ b going to esplanade fr home?        NaN
5569   ham  Pity, * was in mood for that. So...any other s...        NaN
5570   ham  The guy did some bitching but I acted like i'd...        NaN
5571   ham                         Rofl. Its true to its name        NaN

     Unnamed: 3 Unnamed: 4
0           NaN        NaN
1           NaN        NaN
2           NaN        NaN
3           NaN        NaN
4           NaN        NaN
...         ...        ...
5567        NaN        NaN
5568        NaN        NaN
5569        NaN        NaN
5570        NaN        NaN
5571        NaN        NaN

[5572 rows x 5 columns]
```

In [10]:

```python
#Top and Botton 5 rows of car price prediction dataset
print("Top 5 rows of the dataset are: \n",df.head())
print("\n\nBottom 5 rows of the dataset are: \n",df.tail())
```

```
Top 5 rows of the dataset are:
      v1                                                 v2 Unnamed: 2  \
0   ham  Go until jurong point, crazy.. Available only ...        NaN
1   ham                      Ok lar... Joking wif u oni...        NaN
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN
3   ham  U dun say so early hor... U c already then say...        NaN
4   ham  Nah I don't think he goes to usf, he lives aro...        NaN

  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN


Bottom 5 rows of the dataset are:
         v1                                                 v2 Unnamed: 2  \
5567  spam  This is the 2nd time we have tried 2 contact u...        NaN
5568   ham               Will Ì_ b going to esplanade fr home?        NaN
5569   ham  Pity, * was in mood for that. So...any other s...        NaN
```

```
5570   ham  The guy did some bitching but I acted like i'd...        NaN
5571   ham                        Rofl. Its true to its name         NaN

     Unnamed: 3 Unnamed: 4
5567        NaN        NaN
5568        NaN        NaN
5569        NaN        NaN
5570        NaN        NaN
5571        NaN        NaN
```

The Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalisation process that is usually done when setting up Information Retrieval systems.

In [13]:

```
ps=PorterStemmer()
lemmatize=WordNetLemmatizer()
corpus=[]
for i in range(0,len(df)):
  review=re.sub('[^a-zA-Z]', ' ', df['v2'][i])
  review = review.lower()
  review = review.split()

  review = [ps.stem(word) for word in review if not word in stopwords.words('
  review = ' '.join(review)
  corpus.append(review)
```

CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text.

In [15]:

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2500)
X = cv.fit_transform(corpus).toarray()
y=pd.get_dummies(df['v1'])
y=y.iloc[:,1].values
```

Train-Test Split

With train_test_split (), you need to provide the sequences that you want to split as well as any optional arguments. It returns a list of NumPy arrays, other sequences, or SciPy sparse matrices if appropriate: arrays is the sequence of lists, NumPy arrays, pandas DataFrames, or similar array-like objects that hold the data you want to split.

In [16]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, r
```

The Naive Bayes classifier separates data into different classes according to the Bayes' Theorem, along with the assumption that all the predictors are independent of one another. It assumes that a particular feature in a class is not related to the presence of other features.

Naive Bayes classifier for multinomial models. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

In [17]:
```
from sklearn.naive_bayes import MultinomialNB
spam_detect_model = MultinomialNB().fit(X_train, y_train)
```

In [18]:
```
y_pred=spam_detect_model.predict(X_test)
```

**Model precision:

The model precision score measures the proportion of positively predicted labels that are actually correct. Precision is also known as the positive predictive value. Precision is used in conjunction with the recall to trade-off false positives and false negatives. Precision is affected by the class distribution. **Precision Score = TP / (FP + TP)

**Accuracy Score:

Model recall score represents the model's ability to correctly predict the positives out of actual positives. This is unlike precision which measures how many predictions made by models are actually positive out of all positive predictions made. **Recall Score = TP / (FN + TP) **Confusion Matrix:

A confusion matrix, also referred to as an error matrix, is a process that helps to assess and predict the validity of a classification model. Using confusion matrices allows you to see different errors which you could make when you make predictions.

**Classification Report

A classification report is a performance evaluation metric in machine learning. It is used to show the precision, recall, F1 Score, and support of your trained classification model.