ADAPTIVE PREDICTION OF SPAM EMAILS

A Project Report

Submitted in the partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Department of Computer Science Engineering

By

Buddi Sita Rama Krishna 150030138
Ponugoti Vamshi 150030738

Under the Supervision of

Lakshmana Phaneendra Maguluri
(Asst. Professor)

KONERU LAKSHMAIAH EDUCATION FOUNDATION

Green Fields, Vaddeswaram- 522502, Guntur (Dist.),

Andhra Pradesh, India.

April-2019

KONERU LAKSHMAIAH EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Declaration

The Project Report entitled "**ADAPTIVE PREDICTION OF SPAM EMAILS"** is a record of bonafide work of Buddi Sita Rama Krishna (150030138), Ponugoti Vamshi (150030738), submitted in partial fulfilment for the award of Bachelor of Technology in Computer Science Engineering during the academic year 2018-19.

We also declare that this report is of our own effort and it has not been submitted to any other university for the award of any degree.

Buddi Sita Rama Krishna (150030138)

Ponugoti Vamshi (150030738)

KONERU LAKSHMAIAH EDUCATION FOUNDATION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Certificate

This is to certify that the Project Report entitled "**ADAPTIVE PREDICTION OF SPAM EMAILS**" is being submitted by Buddi Sita Rama Krishna (150030138), Ponugoti Vamshi (150030738) in partial fulfilment for the award of Bachelor of Technology in Computer Science Engineering during the academic year 2018-19.

Signature of the Supervisor	**Signature of the HOD**
Lakshmana Phaneendra Maguluri	Dr. V. HARI KIRAN
(Asst.Professor)	(Professor)

Signature of the EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

Apart from the efforts by us, the success of any work depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this thesis.

We would like to show my greatest appreciation to my internal guide **Mr. M L PHANEENDRA,** Dept. of Computer Science and Engineering. We feel motivated and encouraged every time we attend his meetings. Without her encouragement and guidance this thesis would not have materialized and implemented.

We deeply indebted to my Head of the Department **Dr. V. HARI KIRAN** Dept. of Computer Science and Engineering who modelled us both technically and normally for achieving greater success in life.

Finally, we owe a lot to the teaching and non-teaching staff of the Dept., of Computer Science and Engineering for their direct or indirect support in the entire course of our thesis work.

Buddi Sita Rama Krishna (150030138)

Ponugoti Vamshi (150030738)

Abstract

We get hundreds of messages each day, it is difficult for us to say which of them are relevant. We face this problem of spam messages every day. Spam severely risks an internet company's reputation. These days, most of the brands and companies use spam as a source of publicity for their products and services. It can be observed in most popular mailing services that spam filters are being biased for the profit from the ads, i.e. they are allowing some exception for some companies that pay for promotion. This is not an ethical practice, but it is very profitable for their business for sure. Our aim is to build a spam detector using machine learning in python with the packages NLTK, Matplotlib, Word cloud, Math, pandas, NumPy. it can state a stated message is spam or not. It can be implemented by using Bayes' Theorem, a simple yet powerful theorem.

Machine learning empowers investigation of huge amounts of information. While it by and large conveys quicker, more precise outcomes with the end goal to distinguish productive chances or perilous dangers, it might likewise require extra time and assets to prepare it appropriately. Consolidating machine learning with AI and subjective advances can make it much more powerful in handling huge volumes of data.

It is important for ever in this technology era to learn how to build small machine learning systems for performing our day to day tasks it improves efficiency and accuracy.

So, our aim is to build and train a system ourselves for the detection of spam with the help of PYTHON (ANACONDA) packages.

Table of Contents Pages

1. Introduction

Machine learning is a subdivision of Artificial intelligence (AI) that gives frameworks the capacity to certainly take in and enhance as a matter of fact without being expressly modified. Machine learning revolves around the change of PC programs that can get to data and use it learn for themselves.

The way toward learning starts with perceptions or in-formation, for example, models, coordinate understanding, or guidance, with the end goal to search for examples in information and settle on better choices later de-pendent on the precedents that we give. The essential point is to permit the PCs to learn consequently without human mediation or help and alter activities as needs are.

Some machine learning techniques

Machine learning algorithms are characterized as un-supervised and supervised.

Supervised machine learning algorithms can apply what has been realized in the past to new information utilizing marked models to foresee future occasions. Beginning from the examination of a known preparing dataset, the learning calculation delivers a deduced capacity to make forecasts about the yield esteems. The framework can offer concentrations to any new commitment after suitable preparing. The learning calculation can likewise contrast its yield and the right, proposed yield and discover blunders with the end goal to adjust the model in like manner.

Interestingly, unsupervised machine learning calculations are utilized when the data used to prepare is neither characterized nor named. Unsupervised learning contemplates how frameworks can derive a capacity to depict a concealed structure from unlabelled information. The framework doesn't make sense of the correct yield, yet it investigates the information and can attract deductions from datasets to depict concealed structures from unlabelled information.

The semi-supervised machine is a class of machine learning assignments and systems that additionally make utilization of unlabelled data for training, normally a lit-tle measure of marked data with a lot of unlabelled data. Semi-supervised learning sandwiched between super-vised learning and unsupervised learning. Many ma-chine-learning specialists have discovered that unlabelled data, when utilized related to a little measure of labelled data, can deliver impressive change in learning precision over unsupervised learning, however without the time and costs required for supervised learning.

Reinforcement machine learning calculations is a learning technique that associates with its condition by delivering activities and finds mistakes or rewards. Ex-peri mentation seeks and postponed compensate are the most significant attributes of support learning. This technique enables machines and programming operators to consequently decide the perfect conduct inside a setting with the end goal to boost its execution. Basic re-ward input is required for the specialist to realize which activity is ideal; this is known as the Reinforcement signal.

Machine learning enables investigation of enormous amounts of data. While it, for the most part, conveys quicker, more exact outcomes with the end goal to distinguish beneficial chances or perilous dangers, it might likewise require extra time and assets to prepare it appropriately. Joining machine learning with AI and subjective advancements can make it much more viable in processing large volumes of information.

1.1 Bayes' Theorem

It identifies the probability of an event based on the prior knowledge of the conditions that might be related to the event. For a hypothesis, it obtains the posterior probability P(H|D) as a product of the prob-ability of the data P(D|H), multiplied by the probability of the P(H), divided by the probability of the data P(D).

$$\boxed{P(H|D) = \frac{P(D|H).P(H)}{P(D)}}$$

1.2 Naive Bayes Algorithm

The complexity of the above Bayesian classifier needs to be reduced, for it to be practical. The naive Bayes algorithm does that by making an assumption of conditional independence over the training dataset. This drastically reduces the complexity of above-mentioned problem to just 2n.

The assumption of conditional independence states that, given random variables X, Y and Z, we say X is conditionally independent of Y given Z, if and only if the probability distribution governing X is independent of the value of Y given Z.

In other words, X and Y are conditionally independent given Z if and only if, given knowledge that Z occurs, knowledge of whether X occurs provides no information on the likelihood of Y occurring, and knowledge of whether Y occurs provides no information on the likelihood of X occurring.

This assumption makes the Bayes algorithm, naive.

Given, n different attribute values, the likelihood now can be written as

$$P(X_1 \dots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

Here, X represents the attributes, and Y is the response variable. Now, P(X|Y) becomes equal to the products of, probability distribution of each attribute X given Y.

1.2.1 Maximizing a Posteriori

What we are interested in, is finding the posterior probability or P(Y|X). Now, for multiple values of Y, we will need to calculate this expression for each of them.

Given a new instance X New, we need to calculate the probability that Y will take on any given value, given the observed attribute values of X new and given the distributions P(Y) and P(X|Y) estimated from the training data.

So, how will we predict the class of the response variable, based on the different values we attain for P(Y|X). We simply take the most probable or maximum of these values. Therefore, this procedure is also known as maximizing posteriori.

1.2.2 Maximizing Likelihood

If we assume that the response variable is uniformly distributed, that is it is equally likely to get any response, then we can further simplify the algorithm. With this assumption the priori or P(Y) becomes a constant value, which is 1/categories of the response.

As, the priori and evidence are now independent of the response variable, these can be removed from the equation. Therefore, the maximizing the posteriori is reduced to maximizing the likelihood problem.

1.2.3 Feature Distribution

As seen above, we need to estimate the distribution of the response variable from training set or assume uniform distribution. Similarly, to estimate the parameters for a feature's distribution, one must assume a distribution or generate nonparametric models for the features from the training set. Such assumptions are known as event models. The variations in these assumptions generates different algorithms for different purposes. For continuous distributions, the Gaussian naive Bayes is the algorithm of choice. For discrete features, multinomial and Bernoulli distributions as popular. Detailed discussion of these variations is out of the scope of this article.

Naive Bayes classifiers work well in complex situations, despite the simplified assumptions and naivety. The advantage of these classifiers is that they require small number of training data for estimating the parameters necessary for classification. This is the algorithm of choice for text categorization. This is the basic idea behind naive Bayes classifiers, that you need to start experimenting with the algorithm.

2. Literature review

Walmir, M. Caminhas.

They present a thorough survey of late improvements in the use of machine learning calculations to Spam channelling, concentrating on both literary and picture-based methodologies. Rather than considering Spam shifting as a standard classifica-tion issue, we feature the significance of thinking about particular qualities of the issue, particularly idea float, in planning new channels. Two especially essential view-points not generally perceived in the writing are talked about: the difficulties in refreshing a classifier dependent on the sack of-words representation and a noteworthy distinction between two early gullible Bayes models. Generally, we reason that while critical progressions have been made in the most recent years, a few aspects stay to be investigated, particularly under more sensible assessment settings

Taghi M. Khoshgoftaar, Joseph D. Prusa, Aaron N. Richter and Hamzah Al

Online reviews are often the primary factor in a customer's decision to purchase a product or service and are a valuable source of information that can be used to govern public view on some of their products or services. Because of their influence, manufacturers, retailers are highly concerned about customer feedback and reviews. Dependence on online re-views gives rise to the potential apprehension that criminals may create false evaluations on reviews to artificially promote services and products. This practice is known as Opinion or Review Spam, where spammers manipulate reviews (i.e., making untruthful, fake or deceptive reviews) for profit. Since most online reviews are not truthful and trustworthy, it is important to develop techniques for detecting review spam. By extracting evocative features from the text using NLP, it is possible spam detection using various machine learning methods. Furthermore, reviewer information, to one side from the text itself, can be used to help in this process.

Kyumin Lee, James Caverlee, Steve Webb.

Web-based social systems enable new community-based opportunities for members to engage, share, and interact. This community value and associated services like advertising and search are vulnerable by spammers, mal-ware disseminators and content polluters. To preserve communal value and ensure long-standing success, they proposed a honeypot-based approach for finding social spammers on online social systems.

Alex Hai Wang

As online social networking sites become more and more popular, they have also attracted the attentions of the spammers. In this paper they took Twitter, a popular micro-blogging service as an example of spam bots' detection in social networking sites online. To differentiate the spam bots from normal ones we use machine learning approach. To enable the spam bot's detection, three graph-based features, such as the number of friends and the number of followers, are extracted to explore the unique follower and friend associations among users on Twitter. From user's most recent tweets content-based features are also extracted. A real data set is collected from Twitter's publicly available information using two different methods. Evaluation experiments show that the detection system is efficient and accurate to identify spam bots in Twitter.

3. Theoretical Analysis

Let us consider we take a string or message m = (word1, word2…… word n), where (word1, word2…… word n) is a set of unique words that are present in the message.

We need to find

$$\boxed{\frac{\boldsymbol{P(word1|spam)P(word2|spam)\ldots P(word\ n|\ spam).P(spam)}}{\boldsymbol{P(word1\ ).P(word2\ )\ldots P(word\ n)}}}$$

Let us consider that occurrence of a word independent of all other words, we can simplify the above equation into.

$$\boxed{P(spam|word1 \cap word2\ldots \cap word\ n) = \frac{P(word1 \cap word2 \cap \ldots\ldots word\ n|\ spam).P(spam)}{P(word1 \cap word2 \cap \ldots word\ n|\ spam)}}$$

To classify we must determine which is greater

$$\boxed{P(spam|word1 \cap word2\ldots \cap word\ n) \text{ versus } P(\sim spam|\ word1 \cap word2\ldots \cap word\ n)}$$

$$P(spam|w1 \cap w2\cap\ldots\cap wn)\ versus\ P(\sim spam|w1 \cap w2\cap\ldots\cap wn)$$

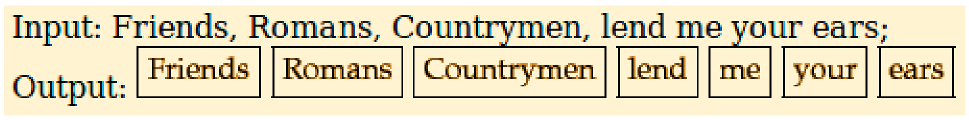
3.1. Training of the model

To train our model, we use two techniques first one is Bag-of-words and second one is TF-IDF.

3.1.1. Pre-processing stage:

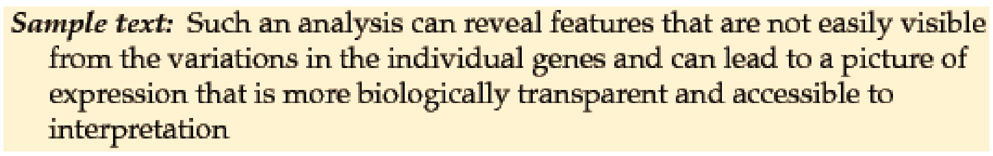
Before the start of training, we must first pre-process the input messages. We need to convert all the input characters to lowercase. We do this need to treat ‘free’ and ‘FREE’ as the same, if not the model treats them as two different words which lead in the waste of more pro-cessing power.

Then we split up the text into small pieces and also re-moving the punctuations. The task of removing punctuations and splitting messages is called **Tokenization** For example.

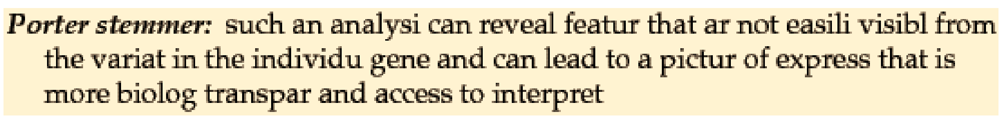


Input: Friends, Romans, Countrymen, lend me your ears;
Output: Friends | Romans | Countrymen | lend | me | your | ears

There are some similar words like ‘eat’, ‘eats’, ‘eating’ direct the same activity, we can replace all such words by one word ‘eat’. This process is called stemming. We are going to use the most popular stemming algorithm The Porter-Stemmer.



Sample text: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation



Porter stemmer: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

Next, we remove the **Stop-words**.

In any text, stop-words occurs extremely frequently. Some of them are ‘or’, ‘as’, ‘is’, ‘an’, ‘a’, ‘the’, ‘to’ and so on. These words are not very important for identification of the content of the text. Thus, we remove these words from the text, and also, we can further optimize the performance by removing words with opposite meaning such as ‘sweet’ and ‘not sweet’. Suppose a text contains ‘not good’, we consider ‘not pretty’ as one token because it is better that way rather than ‘not’ and ‘pretty’ as two tokens. But sometimes accuracy may be improved when we split them into multiple tokens than splitting into the only word.

3.1.2. Bag of Words:

In this model, we find the number of occurrences of each word in the dataset or frequency of the terms.

For a word

$$\boxed{\boldsymbol{P(word) = \frac{Total\ occurrences\ of\ word}{Total\ number\ of\ words}}}$$

Then

$$\boxed{\boldsymbol{P(word|spam) = \frac{Total\ occurrences\ of\ the\ word\ in\ spam\ message}{Total\ number\ of\ words\ in\ the\ spam\ message}}}$$

3.1.3. Term Frequency-Inverse Document Frequency:

In short TF-IDF. On top of Term Frequency, we also compute Inverse Document frequency.

$$\boldsymbol{IDF(word) = log\frac{Total\ number\ of\ messages}{Total\ number\ of\ messages\ contain\ the\ word}}$$

For instance, consider two messages from the dataset. ‘hello world’ and ‘hello foo bar’. Term Frequency(‘hello’) is 2. Inverse Document Frequency (‘hello’) is log (2/2).

If the frequency of a word is more then it gives very less information.

Herein our model each word will have a score, which is

The product of Term Frequency and Inverse Document Frequency i.e., TF (word)*IDF (word).

The probability of each word can be calculated by.

$$P(w) = \frac{TF(w) * IDF(w)}{\sum_{\forall\ words\ x \in train\ dataset} TF(x) * IDF(x)}$$

$$P(w|spam) = \frac{TF(w|spam) * IDF(w)}{\sum_{\forall\ words\ x \in train\ dataset} TF(x|spam) * IDF(x)}$$

3.1.4.Additive Smoothing Technique:

So, suppose we come across a word in test dataset which is not a part of train dataset? In such cases P(word) will be 0, then the P (spam | word) will become infinite (Ac-cording to the formula we would have to divide it by P(word) which is 0). To confrontation such issues we introduce additive-smoothing technique. In this procedure we add a number say alpha to the numerator and add no. of classes over-which the probability is on the de-nominator alpha times.

$$P(w|spam) = \frac{TF(w|spam) \; + \; \alpha}{\sum_{\forall \; words \; x \; \in \; spam \; in \; train \; dataset} TF(x) \; + \; \alpha \sum_{\forall \; words \; x \; \in \; spam \; in \; train \; dataset} 1}$$

If we are using Term Frequency-Inverse Document Frequency, then we can also use soothing as.

$$P(w|spam) = \frac{TF(w|spam) * IDF(w) \; + \; \alpha}{\sum_{\forall \; words \; x \; \in \; train \; dataset} TF(x) * IDF(x) \; + \; \alpha \sum_{\forall \; words \; x \; \in \; spam \; in \; train \; dataset} 1}$$

This is done as such that minimal likelihood of any word currently ought to be a finite number. Addition in the denominator is to make the resultant total of the sum of probabilities of words in the spam messages to one.

When alpha is equal, it is called **Laplace-smoothing**.

3.1.5. Classification of Message

To classify the given message, initially, we must pre-process it. For every word w in the processed messaged we will obtain the product of P (word |spam). If there is any case where w does not exist in the train dataset, we take TF (word) as 0 and find P (word |spam) using above formula, then multiply this product with P(spam) Then the resultant product is the P (spam | message). Similarly, we will obtain P (not-spam |message). In among those whose probability is the greater-corresponding tag (spam or ham) is assigned to the input message. Consider in this case we are not dividing by P (word) as given in the formula because both the numbers will be divided by that and it would not affect the comparison between the two.

4. Experimental Investigations

4.1. Importing Packages

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from math import log, sqrt
import pandas as pd
import numpy as np
import re
%matplotlib inline
```

With the help of NLTK (Natural Language Tool Kit) for the text processing, matplotlib and Word Cloud for visualization and pandas for loading and managing large data, NumPy basically is used to implement Bayes' theorem for generating probabilities for train and test split in ran-dom.

4.2. Importing data

```
mails = pd.read_csv('spam.csv', encoding = 'latin-1')
mails.head()
```

For training our algorithm we use a data set from Kaggle. It has many fields, some of these columns of the dataset are not required, so after we import the data we remove some columns which are not required. We modify the names of some columns so that it is convenient for us in later steps.

```
 mails.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis = 1, inplace = True)
mails.head()

mails.rename(columns = {'v1': 'labels', 'v2': 'message'}, inplace = True)
mails.head()

mails['labels'].value_counts()

mails['label'] = mails['labels'].map({'ham': 0, 'spam': 1})
mails.head()

mails.drop(['labels'], axis = 1, inplace = True)
mails.head()
```

4.3. Splitting Train-Test Data

To verify our model, we need to split the data into test dataset and train dataset. We will use the train dataset for training the model and then the model will be tested on the test dataset. We shall use 75% of the dataset as train dataset and the rest as test set. This selection is totally random.

```
totalMails = 4825 + 747
trainIndex, testIndex = list(), list()
for i in range(mails.shape[0]):
    if np.random.uniform(0, 1) < 0.75:
        trainIndex += [i]
    else:
        testIndex += [i]
trainData = mails.loc[trainIndex]
testData = mails.loc[testIndex]

trainData.reset_index(inplace = True)
trainData.drop(['index'], axis = 1, inplace = True)
trainData.head()

testData.reset_index(inplace = True)
testData.drop(['index'], axis = 1, inplace = True)
testData.head()
trainData['label'].value_counts()
testData['label'].value_counts()
```

4.4. Visualization of data

We need to find out which are the most repeated words in the spam messages, to do this we use WordCloud library.

```
spam_words = ' '.join(list(mails[mails['label'] == 1]['message']))


spam_wc = WordCloud(width = 512,height = 512).generate(spam_words)
plt.figure(figsize = (10, 8), facecolor = 'k')


plt.imshow(spam_wc)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

for words people mostly use in their messages daily the size of the word in these images respective frequency.

```
ham_words = ' '.join(list(mails[mails['label'] == 0]['message']))
ham_wc = WordCloud(width = 512,height = 512).generate(ham_words)
plt.figure(figsize = (10, 8), facecolor = 'k')
plt.imshow(ham_wc)
plt.axis('off')
plt.tight_layout(pad = 0)
plt.show()
```

4.5. Training of the model

```
trainData['label'].value_counts()

testData.head()

testData['label'].value_counts()
```

we use two models one is Bag-of-words and other is TF-IDF.

4.6. Preprocessing stage

```
def process_message(message, lower_case = True, stem = True, stop_words = True, gram = 2):
    if lower_case:
        message = message.lower()
    words = word_tokenize(message)
    words = [w for w in words if len(w) > 2]
    if gram > 1:
        w = []
        for i in range(len(words) - gram + 1):
            w += [' '.join(words[i:i + gram])]
        return w
    if stop_words:
        sw = stopwords.words('english')
        words = [word for word in words if word not in sw]
    if stem:
        stemmer = PorterStemmer()
        words = [stemmer.stem(word) for word in words]
    return words
```

```
class SpamClassifier(object):
    def __init__(self, trainData, method = 'tf-idf'):
        self.mails, self.labels = trainData['message'], trainData['label']
        self.method = method

    def train(self):
        self.calc_TF_and_IDF()
        if self.method == 'tf-idf':
            self.calc_TF_IDF()
        else:
            self.calc_prob()

    def calc_prob(self):
        self.prob_spam = dict()
        self.prob_ham = dict()
        for word in self.tf_spam:
            self.prob_spam[word] = (self.tf_spam[word] + 1) / (self.spam_words + \
                                                len(list(self.tf_spam.keys())))
        for word in self.tf_ham:
            self.prob_ham[word] = (self.tf_ham[word] + 1) / (self.ham_words + \
                                                len(list(self.tf_ham.keys())))
        self.prob_spam_mail, self.prob_ham_mail = self.spam_mails / 
self.total_mails, self.ham_mails / self.total_mails
```

```
 def calc_TF_and_IDF(self):
        noOfMessages = self.mails.shape[0]
        self.spam_mails,        self.ham_mails        =        self.labels.value_counts()[1],
self.labels.value_counts()[0]
        self.total_mails = self.spam_mails + self.ham_mails
        self.spam_words = 0
        self.ham_words = 0
        self.tf_spam = dict()
        self.tf_ham = dict()
        self.idf_spam = dict()
        self.idf_ham = dict()
        for i in range(noOfMessages):
            message_processed = process_message(self.mails[i])
            count = list() #To keep track of whether the word has ocured in the message
or not.

#For IDF

            for word in message_processed:
                if self.labels[i]:
                    self.tf_spam[word] = self.tf_spam.get(word, 0) + 1
                    self.spam_words += 1
                else:
                    self.tf_ham[word] = self.tf_ham.get(word, 0) + 1
                    self.ham_words += 1
                if word not in count:
```

```
                count += [word]
            for word in count:
                if self.labels[i]:
                    self.idf_spam[word] = self.idf_spam.get(word, 0) + 1
                else:
                    self.idf_ham[word] = self.idf_ham.get(word, 0) + 1

    def calc_TF_IDF(self):
        self.prob_spam = dict()
        self.prob_ham = dict()
        self.sum_tf_idf_spam = 0
        self.sum_tf_idf_ham = 0
        for word in self.tf_spam:
            self.prob_spam[word] = (self.tf_spam[word]) * log((self.spam_mails + self.ham_mails) \
                                                          / (self.idf_spam[word] + self.idf_ham.get(word, 0)))
            self.sum_tf_idf_spam += self.prob_spam[word]
        for word in self.tf_spam:
            self.prob_spam[word] = (self.prob_spam[word] + 1) / (self.sum_tf_idf_spam + len(list(self.prob_spam.keys())))

        for word in self.tf_ham:
            self.prob_ham[word] = (self.tf_ham[word]) * log((self.spam_mails + self.ham_mails) \
                                                          / (self.idf_spam.get(word, 0) + self.idf_ham[word]))
            self.sum_tf_idf_ham += self.prob_ham[word]
        for word in self.tf_ham:
            self.prob_ham[word] = (self.prob_ham[word] + 1) / (self.sum_tf_idf_ham
+ len(list(self.prob_ham.keys())))



        self.prob_spam_mail, self.prob_ham_mail = self.spam_mails /
self.total_mails, self.ham_mails / self.total_mails


    def classify (self, processed_message):
        pSpam, pHam = 0, 0
        for word in processed_message:
            if word in self.prob_spam:
                pSpam += log(self.prob_spam[word])
            else:
                if self.method == 'tf-idf':
                    pSpam -= log(self.sum_tf_idf_spam + len(list(self.prob_spam.keys())))
                else:
                    pSpam -= log(self.spam_words + len(list(self.prob_spam.keys())))
            if word in self.prob_ham:
                pHam += log(self.prob_ham[word])
            else:
                if self.method == 'tf-idf':
                    pHam -= log(self.sum_tf_idf_ham + len(list(self.prob_ham.keys())))
                else:
                    pHam -= log(self.ham_words + len(list(self.prob_ham.keys())))
            pSpam += log(self.prob_spam_mail)
            pHam += log(self.prob_ham_mail)
        return pSpam >= pHam
    def predict(self, testData):
        result = dict()
        for (i, message) in enumerate(testData):
            processed_message = process_message(message)
            result[i] = int(self.classify(processed_message))
        return result
```

```
def metrics(labels, predictions):
    true_pos, true_neg, false_pos, false_neg = 0, 0, 0, 0
    for i in range(len(labels)):
        true_pos += int(labels[i] == 1 and predictions[i] == 1)
        true_neg += int(labels[i] == 0 and predictions[i] == 0)
        false_pos += int(labels[i] == 0 and predictions[i] == 1)
        false_neg += int(labels[i] == 1 and predictions[i] == 0)
    precision = true_pos / (true_pos + false_pos)
    recall = true_pos / (true_pos + false_neg)
    Fscore = 2 * precision * recall / (precision + recall)
    accuracy = (true_pos + true_neg) / (true_pos + true_neg + false_pos +
false_neg)

    print("Precision: ", precision)
    print("Recall: ", recall)
    print("F-score: ", Fscore)
    print("Accuracy: ", accuracy)
```

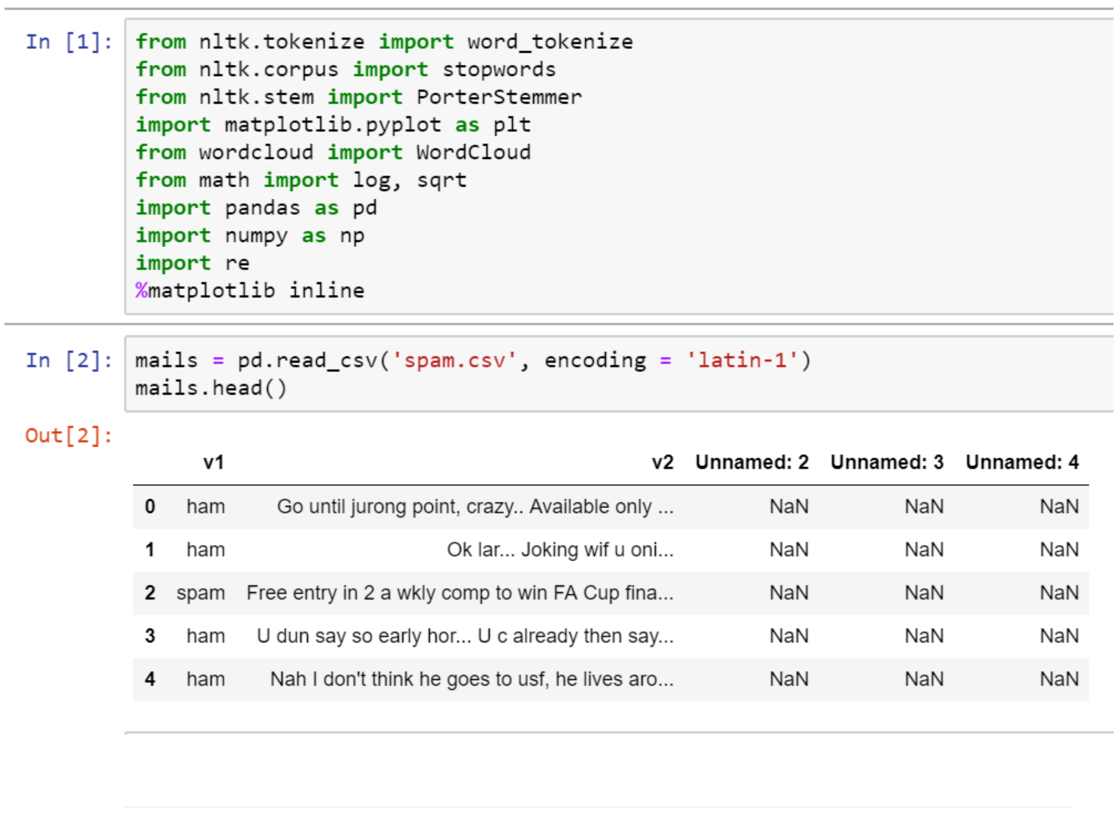
4.7 Term Frequency-Inverse Document Frequency

```
sc_tf_idf = SpamClassifier(trainData, 'tf-idf')
sc_tf_idf.train()
preds_tf_idf = sc_tf_idf.predict(testData['message'])
metrics(testData['label'], preds_tf_idf)
```

4.8 Bag of Words

```
sc_bow = SpamClassifier(trainData, 'bow')
sc_bow.train()
preds_bow = sc_bow.predict(testData['message'])
metrics(testData['label'], preds_bow)
```

5. Experimental Results

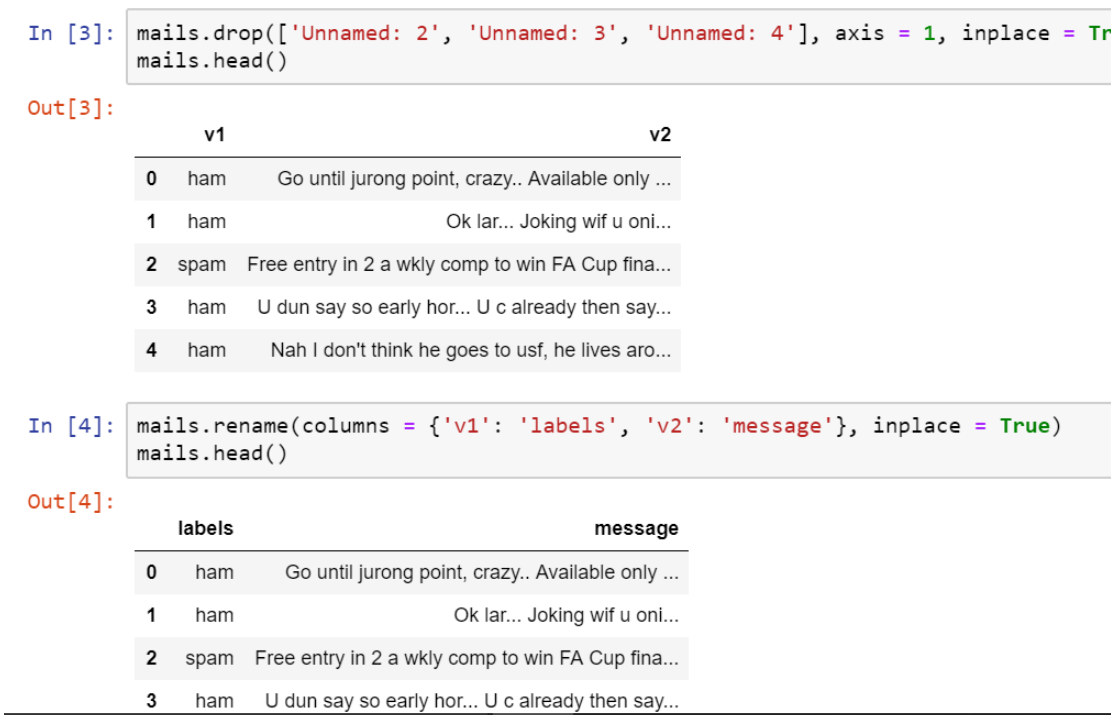


In [1]:
```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from math import log, sqrt
import pandas as pd
import numpy as np
import re
%matplotlib inline
```

In [2]:
```
mails = pd.read_csv('spam.csv', encoding = 'latin-1')
mails.head()
```

Out[2]:

	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 [3]:
```
mails.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis = 1, inplace = Tr
mails.head()
```

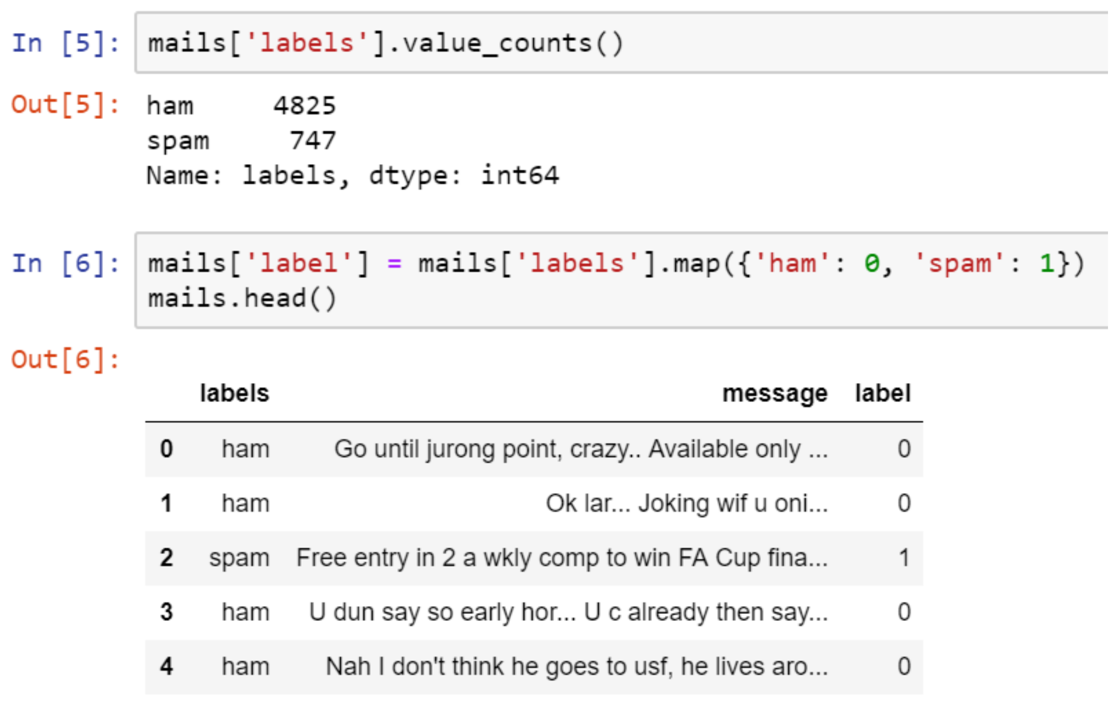
Out[3]:

	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 [4]:
```
mails.rename(columns = {'v1': 'labels', 'v2': 'message'}, inplace = True)
mails.head()
```

Out[4]:

	labels	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...



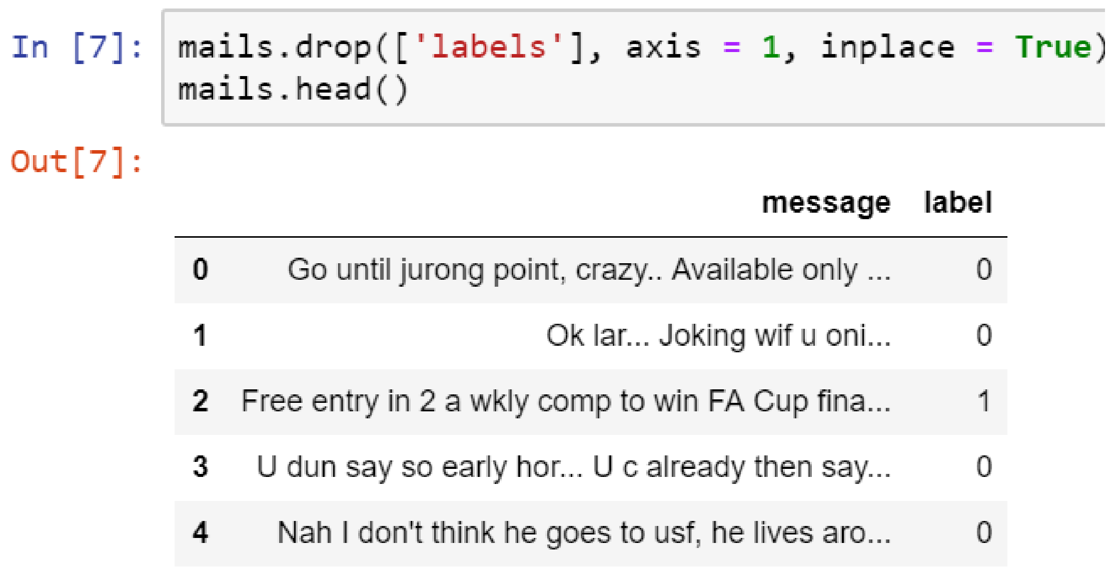
In [5]:
```
mails['labels'].value_counts()
```

Out[5]:
```
ham      4825
spam      747
Name: labels, dtype: int64
```

In [6]:
```
mails['label'] = mails['labels'].map({'ham': 0, 'spam': 1})
mails.head()
```

Out[6]:

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



In [7]:
```
mails.drop(['labels'], axis = 1, inplace = True)
mails.head()
```

Out[7]:

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

In [8]:
```python
totalMails = 4825 + 747
trainIndex, testIndex = list(), list()
for i in range(mails.shape[0]):
    if np.random.uniform(0, 1) < 0.75:
        trainIndex += [i]
    else:
        testIndex += [i]
trainData = mails.loc[trainIndex]
testData = mails.loc[testIndex]
```

In [9]:
```python
trainData.reset_index(inplace = True)
trainData.drop(['index'], axis = 1, inplace = True)
trainData.head()
```

Out[9]:

	message	label
0	Ok lar... Joking wif u oni...	0
1	Free entry in 2 a wkly comp to win FA Cup fina...	1
2	Nah I don't think he goes to usf, he lives aro...	0
3	Even my brother is not like to speak with me. ...	0
4	As per your request 'Melle Melle (Oru Minnamin...	0

In [10]:
```python
testData.reset_index(inplace = True)
testData.drop(['index'], axis = 1, inplace = True)
testData.head()
```

Out[10]:

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	U dun say so early hor... U c already then say...	0
2	FreeMsg Hey there darling it's been 3 week's n...	1
3	Eh u remember how 2 spell his name... Yes i di...	0
4	Ahhh. Work. I vaguely remember that! What does...	0

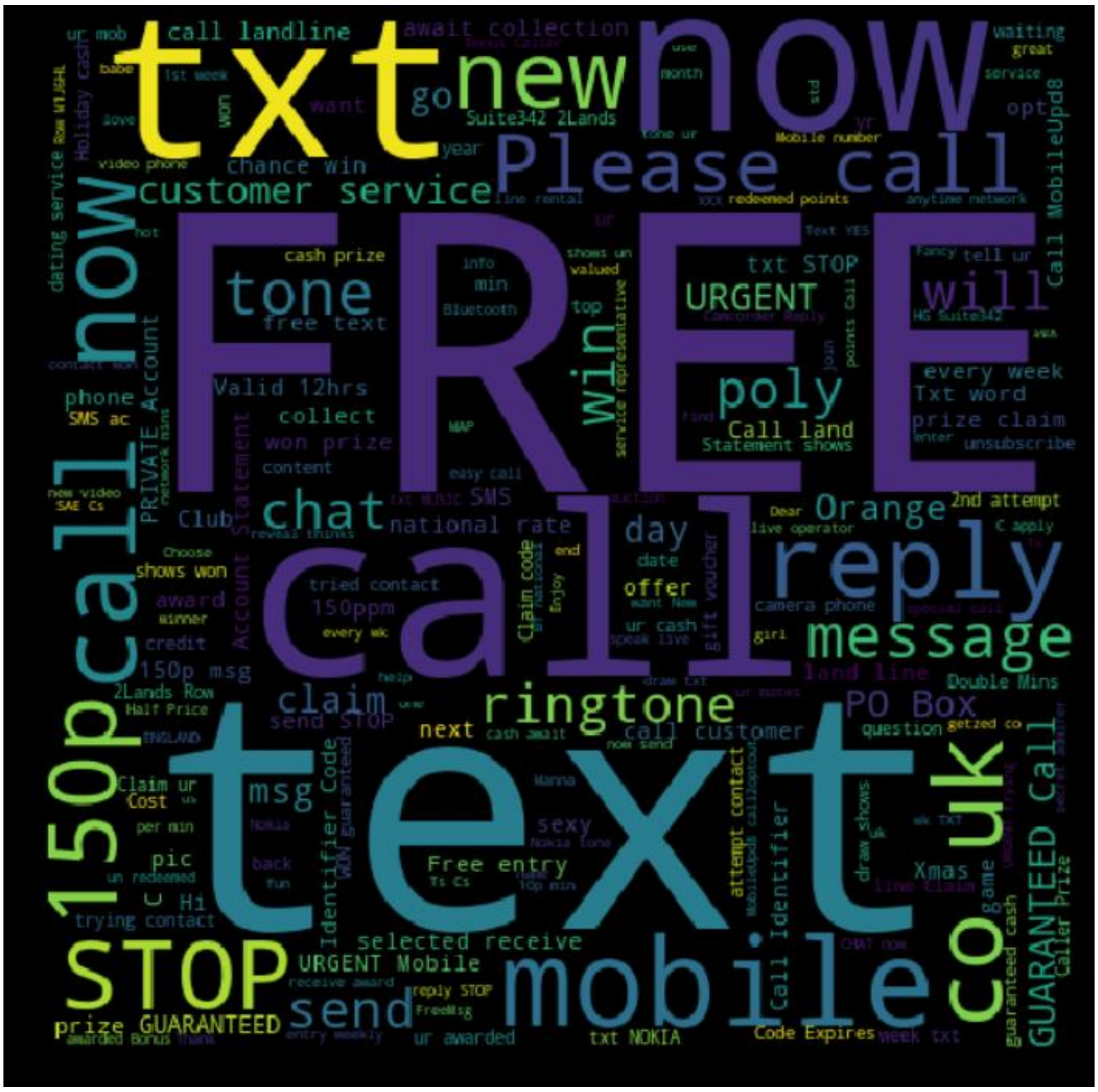
In [11]:
```python
trainData['label'].value_counts()
```

Out[11]:
```
0    3580
1     546
Name: label, dtype: int64
```

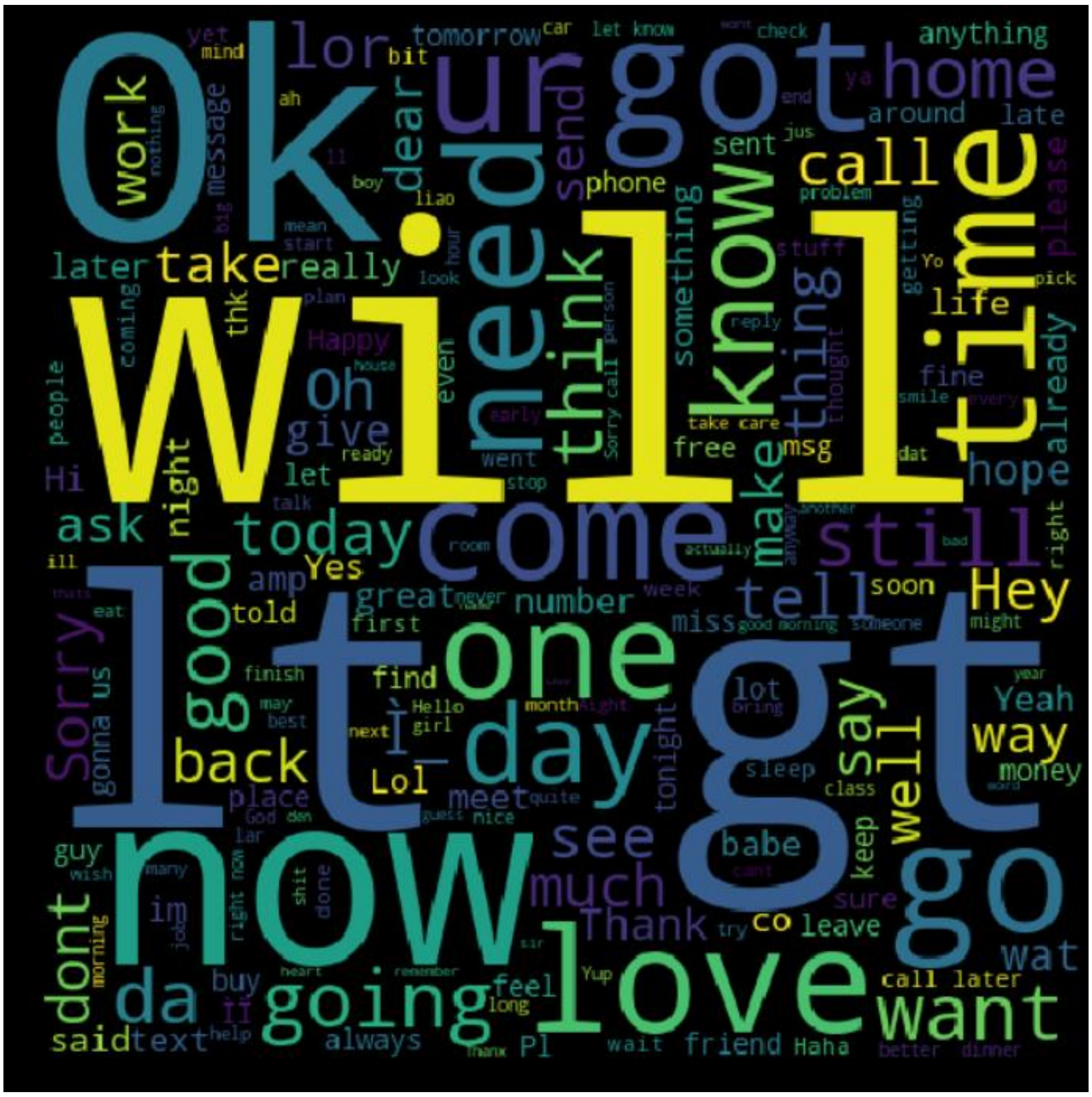
In [12]:
```python
testData['label'].value_counts()
```

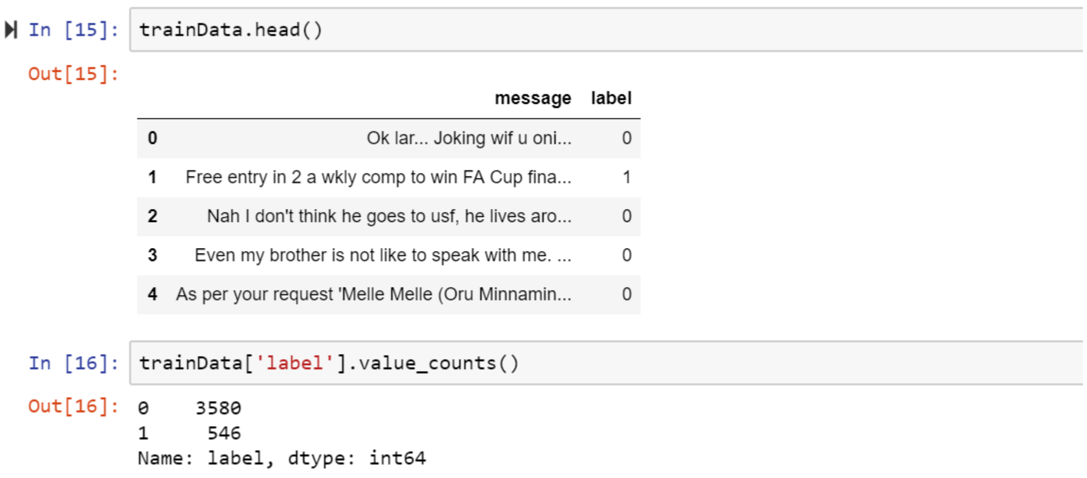
Out[12]:
```
0    1245
1     201
Name: label, dtype: int64
```

```
In [13]: spam_words = ' '.join(list(mails[mails['label'] == 1]['message']))
         spam_wc = WordCloud(width = 512,height = 512).generate(spam_words)
         plt.figure(figsize = (10, 8), facecolor = 'k')
         plt.imshow(spam_wc)
         plt.axis('off')
         plt.tight_layout(pad = 0)
         plt.show()
```



```
In [14]:  ham_words = ' '.join(list(mails[mails['label'] == 0]['message']))
           ham_wc = WordCloud(width = 512,height = 512).generate(ham_words)
           plt.figure(figsize = (10, 8), facecolor = 'k')
           plt.imshow(ham_wc)
           plt.axis('off')
           plt.tight_layout(pad = 0)
           plt.show()
```

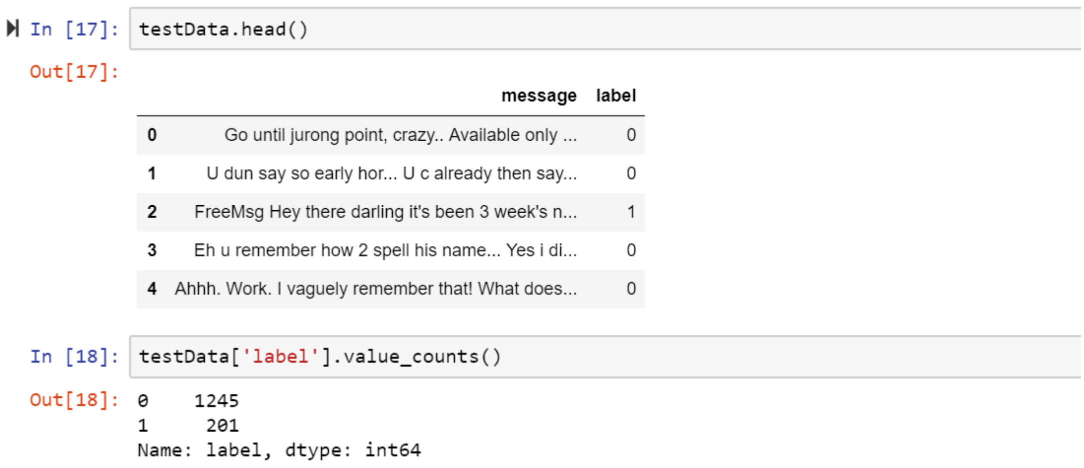




```
In [15]: trainData.head()
Out[15]:
```

	message	label
0	Ok lar... Joking wif u oni...	0
1	Free entry in 2 a wkly comp to win FA Cup fina...	1
2	Nah I don't think he goes to usf, he lives aro...	0
3	Even my brother is not like to speak with me. ...	0
4	As per your request 'Melle Melle (Oru Minnamin...	0

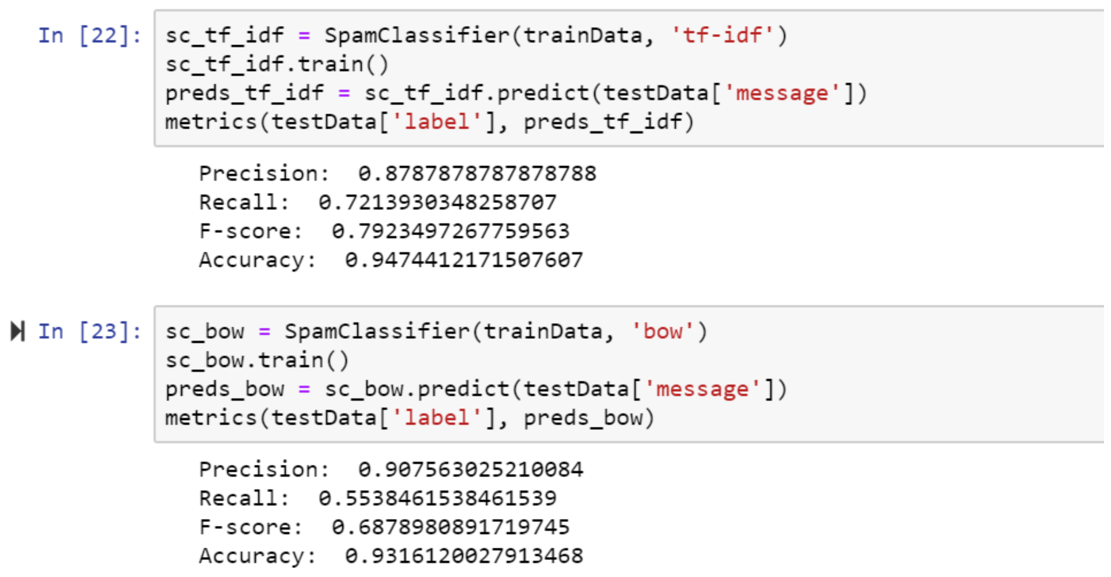
```
In [16]: trainData['label'].value_counts()
Out[16]: 0    3580
         1     546
         Name: label, dtype: int64
```



```
In [17]: testData.head()
Out[17]:
```

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	U dun say so early hor... U c already then say...	0
2	FreeMsg Hey there darling it's been 3 week's n...	1
3	Eh u remember how 2 spell his name... Yes i di...	0
4	Ahhh. Work. I vaguely remember that! What does...	0

```
In [18]: testData['label'].value_counts()
Out[18]: 0    1245
         1     201
         Name: label, dtype: int64
```



```
In [22]: sc_tf_idf = SpamClassifier(trainData, 'tf-idf')
         sc_tf_idf.train()
         preds_tf_idf = sc_tf_idf.predict(testData['message'])
         metrics(testData['label'], preds_tf_idf)

           Precision:  0.8787878787878788
           Recall:  0.7213930348258707
           F-score:  0.7923497267759563
           Accuracy:  0.9474412171507607

In [23]: sc_bow = SpamClassifier(trainData, 'bow')
         sc_bow.train()
         preds_bow = sc_bow.predict(testData['message'])
         metrics(testData['label'], preds_bow)

           Precision:  0.907563025210084
           Recall:  0.5538461538461539
           F-score:  0.6878980891719745
           Accuracy:  0.9316120027913468
```

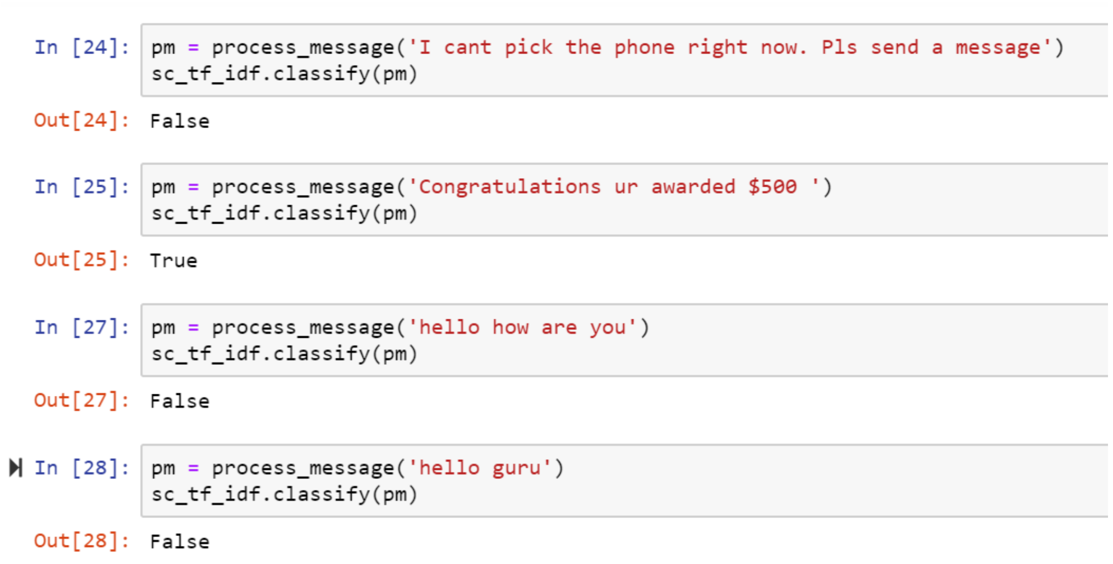
In information retrieval, pattern recognition, binary classification, precision (additionally called positive prescient esteem) is the division of significant occasions among the recovered examples

while Recall(sensitivity) is the division of applicable cases that have been recovered over the aggregate sum of significant cases. Both precision and review are in this way dependent on an understanding and proportion of significance.

F-score is a measure of a test's accuracy In statistical analysis of binary-classification.

Result

When we pass a string to the function "process_message" then it will process the string or the message and then it predicts weather it is spam or ham I returns either true or false and that string is stored in string and is printed to show the result.



```
In [24]: pm = process_message('I cant pick the phone right now. Pls send a message')
         sc_tf_idf.classify(pm)

Out[24]: False

In [25]: pm = process_message('Congratulations ur awarded $500 ')
         sc_tf_idf.classify(pm)

Out[25]: True

In [27]: pm = process_message('hello how are you')
         sc_tf_idf.classify(pm)

Out[27]: False

In [28]: pm = process_message('hello guru')
         sc_tf_idf.classify(pm)

Out[28]: False
```

When the function returns False it means that it is not a spam and similarly for True it is spam

The function may not predict properly for very small length sentences if the sentence is large it works good or in other words the larger the sentence higher the accuracy.

Conclusion

This thesis has addressed the problem of Spam messages. In this work a methodology has been proposed to detect the most common and frequent list of words which are leading a mail to be spam filtered. With the help of these methodologies we are successful in determining the spam behavioural word and e-mails.

Email is an effective, brisk and minimal effort correspondence approach. Email Spam is non-asked for information sent to the E-letter drops. Spam could be an enormous downside each for clients and for ISPs. As per examination these days client gets a considerable measure of spam messages then non-spam messages. To stay away from spam/unessential sends we'd like successful spam separating systems. Spam sends zone unit utilized for spreading infection or malevolent code, for extortion in managing an account, for phishing, and for publicizing. Spam messages are disturbance and gigantic issue to most clients since they mess their letter drops and waste their opportunity to erase all the garbage sends before perusing the real ones. They additionally cost client cash with dial up associations; squander arrange data transfer capacity and circle space. Bayesian classifier is a standout amongst the most essential and generally utilized classifier and furthermore it's the easiest order strategy because of its controlling capacities of tokens and related probabilities as indicated by the users‟ grouping choice and exact execution. In this undertaking, we executed the framework to break down every single mail. And furthermore, give security-based location framework to encode the messages utilizing Digest based framework. Upgrade the oddity indicator, to foresee messages with spring up window with email following framework. Later on work we have an arrangement to execute other calculation to our grouping strategy to accomplish better execution.

References and Resources

[1] [Alurkar, Aakash Atul, et al. "A proposed data science approach for email spam classification using machine learning techniques." *Internet of Things Business Models, Users, and Networks, 2017*. IEEE, 2017.

[2] Amin, Rohan, Julie Ryan, and Johan van Dorp. "Detecting targeted malicious email." *IEEE Security & Privacy* 10.3.

[3] Renuka, D. Karthika, et al. "Spam classification based on supervised learning using machine learning techniques." *Process Automation, Control and Computing (PACC), 2011 International Conference on*. IEEE.

[4] Harisinghaney, Anirudh, et al. "Text and image-based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm." *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*. IEEE.

[5] Choo, Jaegul, and Haesun Park. "Customizing computational methods for visual analytics with big data." *IEEE Computer Graphics and Applications* 33.4.

[6] Liu, Yulin, et al. "Effect of demographic structure on resource utilisation using term frequency–inverse document frequency algorithm–evidence from China." *The Journal of Engineering*2018.16 (2018).

[7] Duan, Lixin, Dong Xu, and Ivor Wai-Hung Tsang. "Domain adaptation from multiple sources: A domain-dependent regularization approach." *IEEE Transactions on Neural Networks and Learning Systems* 23.3.

[8] Mujtaba, Ghulam, et al. "Email classification research trends: Review and open issues." *IEEE Access* 5 (2017).

[9] Trivedi, Shrawan Kumar. "A study of machine learning classifiers for spam detection." *Computational and Business Intelligence (ISCBI), 2016 4th International Symposium on*. IEEE, 2016.

[10] You, Wanqing, et al. "Web Service-Enabled Spam Filtering with Naïve Bayes Classification." *2015 IEEE First International Conference on Big Data Computing Service and Applications (BigDataService)*. IEEE, 2015.

[11] Rathod, Sunil B., and Tareek M. Pattewar. "Content based spam detection in email using Bayesian classifier." *International Conference on*. IEEE, 2015.

[12] Sahın, Esra, Murat Aydos, and Fatih Orhan. "Spam/ham e-mail classification using machine learning methods based on bag of words technique." *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2018.

[13] Guzella, Thiago S., and Walmir M. Caminhas. "A review of machine learning approaches to spam filtering." *Expert Systems with Applications* 36.7.

[14] Crawford, Michael, et al. "Survey of review spam detection using machine learning techniques." *Journal of Big Data* 2.1 (2015).

[15] Lee, Kyumin, James Caverlee, and Steve Webb. "Uncovering social spammers: social honeypots+ machine learning." *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM.

[16] Wang, Alex Hai. "Detecting spam bots in online social networking sites: a machine learning approach." *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, Berlin, Heidelberg.