Machine Learning for Detection of Spam

Lakshmana Phaneendra Maguluri, Buddi Sita Rama Krishna, Kalimili Tharun Sai, Ponugoti Vamshi

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 advertising. 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 specified message is spam or not. It can be implemented by using Bayes' Theorem, a simple yet powerful theorem.

1. Introduction

Machine learning is a subdivision of artificial intelligence that provides frameworks the ability to naturally take in and improve as a matter of fact without being expressly modified. Machine learning will revolve around changing PC programs, which can learn for themselves by using data. The way to learning begins with perceptions or information, such as models, coordination of understanding or guidance, with the ultimate goal of finding examples in information and making better choices depending on the precedents we give. The key point is to enable PCs to learn without human involvement or help and to alter activities as needed.

Some machine learning techniques [2]

Algorithms for machine learning are characterized as unsupervised and supervised.

Supervised machine learning algorithms can apply what was previously realized to new information using marked models to anticipate future events. Starting from the examination of a known preparation data set, the learning calculation results in a deduced capacity to predict the yield estimations. After proper preparation, the framework can offer concentrations for any new commitment. The learning calculation can also contrast its yield with the right yield and find blunders with the end goal of adjusting the model in the same way.

Interestingly, Calculations of unsupervised machine learning are used when the data used to prepare is neither characterized nor named. unsupervised learning [7] contemplates how frameworks can derive from unlabeled information the ability to represent a hidden structure. The framework does not make sense of the correct output, but it examines the information and can attract conclusions from data sets to describe hidden structures from unlabeled data.

The semi-supervised [1] machine is a class of machine learning assignments and systems that additionally make utilization of unlabeled data for training, normally a little measure of marked data with a lot of unlabeled data. Semi-supervised learning sandwiched between supervised learning and unsupervised learning. Many machine-learning specialists [10] have discovered that unlabeled 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 [14] are learning techniques that relate to their condition by carrying out activities and finding errors or rewards. Experimental searches and postponed compensation are the most important attributes of support learning. This technique enables machines and programming operators to decide the perfect performance in a setting with the ultimate goal of enhancing its execution. Basic feedback is required for the specialist to realize which activity is ideal; this is called the Reinforcement signal.

Machine learning allows large amounts of data to be investigated. While, for the most part, it conveys faster, more accurate results with the ultimate goal of distinguishing beneficial chances or insecure dangers, it may also require extra time and resources to prepare it appropriately. It can be made much more viable in the processing of large volumes of information by joining machine learning with AI and subjective advancements.

Bayes' Law [11] finds the possibility of an event that is based on the prior information of the conditions that could be related to an event. For a hypothesis, that posterior probability P(A|B) is obtained as a product of the possibility of data P(B|A) multiplied by the possibility of P(A), divided by the possibility of data P(B).

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Eq_n 1.1 Bayes Law

2.0 Literature review:

Walmir, M. Caminhas [13]. They present a thorough survey of late improvements in the use of machine learning calculations to Spam channeling, concentrating on both literary and picture-based methodologies. Rather than considering Spam shifting as a standard classification issue, we feature the significance of thinking about particular qualities of the issue [8], particularly idea float, in planning new channels. Two especially essential viewpoints not generally perceived in the writing are talked about: the difficulties in refreshing a classifier depending on the representation of the sack of words and a notable distinction between two early gullible models of Bayes. Generally speaking, while critical progress has been made in recent years, a number of aspects need to be investigated, especially in the context of more sensible evaluation settings.
Taghi M. Khoshgoftaar, Joseph D. Prusa, Aaron N. Richter and Hamzah Al The primary factor in online reviews of the customer's decision to buy a product or service is valuable information sources that can be used to overlook the public's view of some of their products or services. Manufacturing companies are highly concerned about customer feedback and reviews due to their impact. Dependency on online reviews leads to the potential apprehension that criminals can artificially make false reviews to promote their services and products. This kind of practice is called Opinion or Review Spam, where spammers manipulate reviews for their benefit. As most online reviews are not true and accurate, spam filtering techniques need to be developed. If evocative functionality is obtained from text using NLP, many machine learning methods can be used to detect spam.
Kyumin Lee, James Caverlee, Steve Webb [15] Web-based social systems allow the members to engage, share and interact with a new community- based opportunities. Spammers, malware distributors and content polluters are vulnerable to this community value and related services, such as advertising and search. In order to preserve community value and ensure long- term success, they proposed a honeypot approach to find social spammers in social systems online.
Alex Hai Wang [16]. As social networking sites are becoming increasingly popular online, They also attracted spammers ' attention. Twitter, a popular blogging service, was used in this paper as an example of spam detection on online social networking sites. We use the learning method of the machine to distinguish spam bots from normal ones. In order to detect the spam bot, three graphic features, such as the number of friends and the number of followers, are extracted to explore the unique associations between Twitter users and friends. The content-based features of the latest user tweets are also extracted. Two different methods are used to collect real data. The publicly available Twitter information. Evaluation experiments show that the System for detection is precise and efficient to detect spam bots on twitter.

3.0 Spam detection using machine learning

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

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

We can simplify the above equation into that occurrence of a word independent of all other words.

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

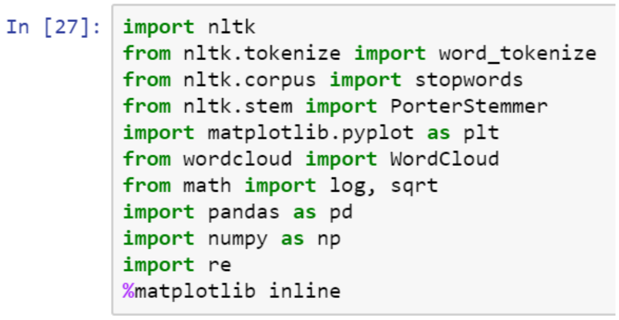
Eq$_n$ 3.1 Word Occurrence

To classify we must determine which is greater

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

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

3.1. Importing Packages

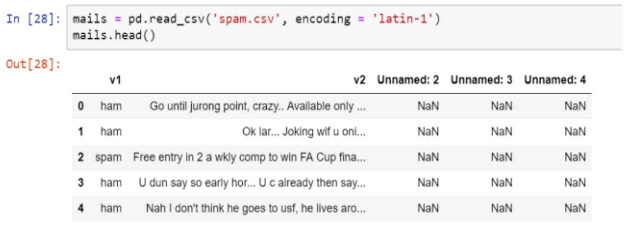


```
In [27]: import nltk
         from nltk.tokenize import word_tokenize
         from nltk.corpus import stopwords
         from nltk.stem import PorterStemmer
         import matplotlib.pyplot as plt
         from wordcloud import WordCloud
         from math import log, sqrt
         import pandas as pd
         import numpy as np
         import re
         %matplotlib inline
```

Fig 3.1.1 Packages

With the help of NLTK (Natural Language Tool Kit) [9] 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 random.

3.2. Importing data

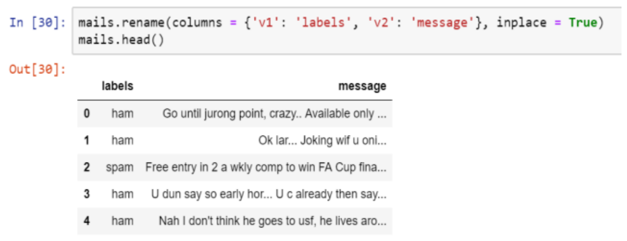


```
In [28]: mails = pd.read_csv('spam.csv', encoding = 'latin-1')
         mails.head()

Out[28]:
```

	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

Fig 3.2.1 Data Set

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.



```
In [30]: mails.rename(columns = {'v1': 'labels', 'v2': 'message'}, inplace = True)
         mails.head()
Out[30]:
```

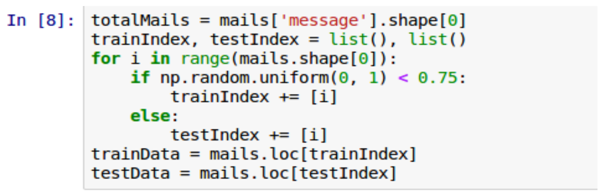
	labels	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Fig 3.2.2 Classification Data Set

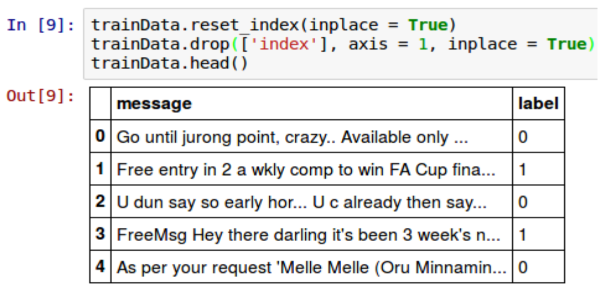
The final output data frame that we obtain after performing the above-required operations is.

3.3. Splitting Train and Test Data

We need to split the data into test dataset and train dataset to verify our model. The model is trained using train dataset , and tested on the test dataset. 75% of the data set is used as train dataset and the rest as a test dataset. This choice is completely random.



```
In [8]: totalMails = mails['message'].shape[0]
        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]
```



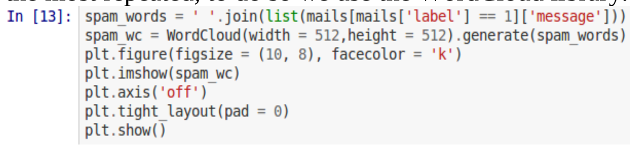
```
In [9]: trainData.reset_index(inplace = True)
        trainData.drop(['index'], axis = 1, inplace = True)
        trainData.head()
Out[9]:
```

	message	label
0	Go until jurong point, crazy.. Available only ...	0
1	Free entry in 2 a wkly comp to win FA Cup fina...	1
2	U dun say so early hor... U c already then say...	0
3	FreeMsg Hey there darling it's been 3 week's n...	1
4	As per your request 'Melle Melle (Oru Minnamin...	0

Fig 3.3.1 Train Dataset

3.4. Visualization of data [5]

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



```
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()
```

Fig 3.4.1 Generating Word Cloud

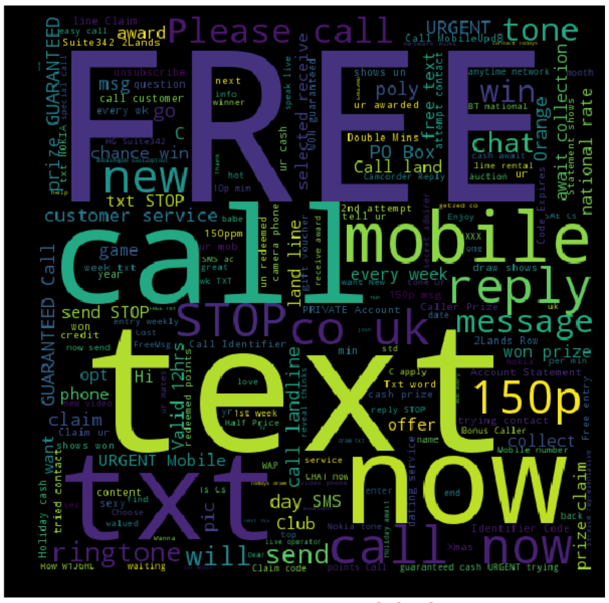


Fig 3.4.2 Spam WordCloud

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

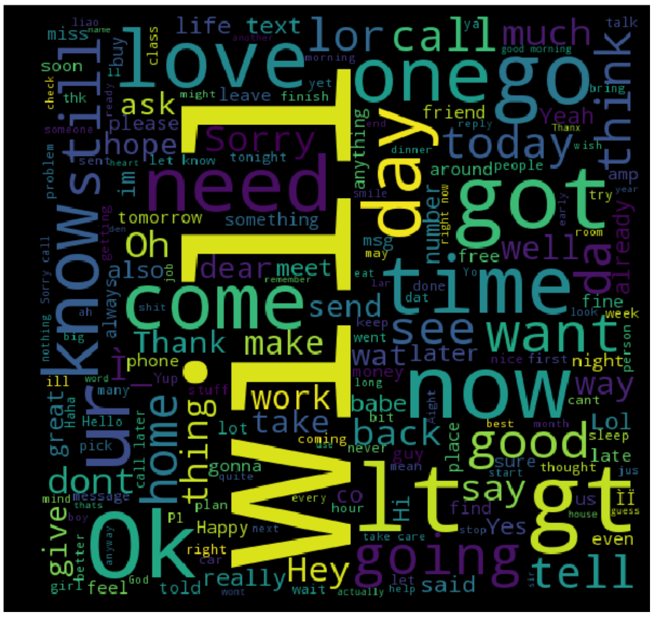


Fig 3.4.3 Ham WordCloud

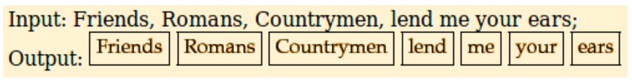
3.5. Training of the model

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

Preprocessing stage:

Before the start of training, we must first preprocess 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 processing power.

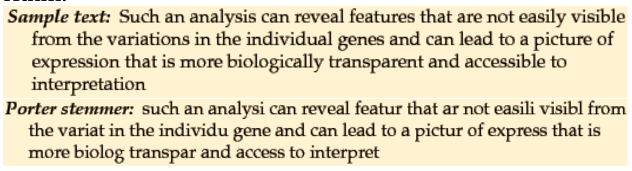
Then we split up the text into small pieces and also removing the punctuations. The task of removing punctuations and splitting messages is called **Tokenization** For example.



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

Output: Friends | Romans | Countrymen | lend | me | your | ears

There are similar words such as ' eat," eat," eat," eat,' directly the same activity, We can substitute one word for all these words. This process is called stemming. We will use the most popular Porter-Stemmer stemming algorithm.



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.

Bag of Words: [12]
In this model, we find how many times each word is occurs in the dataset or frequency of terms in the dataset. For a word

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

Eqn 3.5.1 frequency of a word

Then

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

Eqn 3.5.2spam word frequency

Term Frequency-Inverse Document Frequency: [6]
In short TF-IDF. On top of Term Frequency, we also compute Inverse Document frequency.

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

Eqn 3.5.3 Words Frequency

Consider two messages from the dataset, for example. "Hello world" and "Hello foo bar." Term Hello Frequency is 2. Log is the inverted document frequency (' hello'). If a word is more frequently, it gives very little information.

Here in our model each word will have a score, Term Frequency and Inverse Document Frequency i.e., TF (word)*IDF (word).
Is probability of each word can be calculated by.

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

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

Eqn 3.5.4 Document Frequency

Additive Smoothing Technique: [4]
So, suppose we find a word in the test data set that doesn't form part of the train data set? In such cases, P(word) is 0, then the P(spam | word) becomes infinite(according to the formula we should divide it by 0). To tackle such problems, We introduce smoothing additive technology. In this procedure, We consider an alpha number to the numerator and increase a number of classes over which the probability is alpha times the denominator.

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

Eqn 3.5.5 Alpha Addition

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

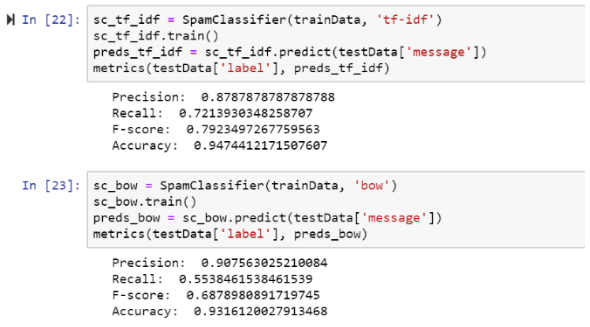
$$P(w|spam) = \frac{TF(w|spam) * IDF(w) + \alpha}{\sum_{\forall\ words\ x\ \in\ train\ dataset} TF(x) * IDF(x) + \alpha \sum_{\forall\ words\ x\ \in\ spam\ in\ train\ dataset} 1}$$

Eqn 3.5.6 Term Frequency

This is done so that at present the lowest probability of any word has to be the finite number. The denominator adds to one the resulting number of word probabilities in spam messages.When alpha is equal, it is called Laplace-smoothing.

3.6. Classification of Message

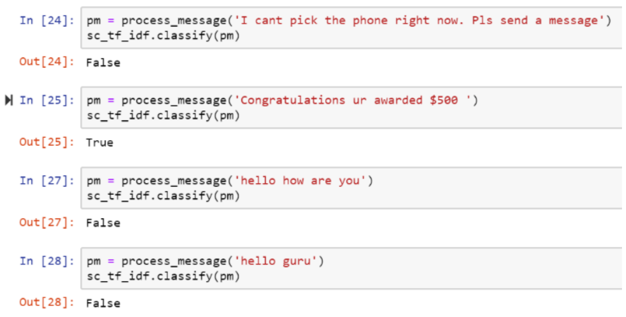
In order to classify [3] the given message, we must first process it. For each word w in the processed message we get P (word |spam). If w does not exist in the train dataset, we take TF(word) as 0 and find P(word spam) using the above formula, then multiply P(spam) with this product. Then P(spam | message) is the resulting product. We will also receive P (not-spam | message). The input message is assigned to those whose probability is the larger corresponding tag (spam or ham). In this case, consider that we are not divided by P(word) according to the formula, because it divides both numbers, and the comparison between the two is not affected.



```
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
```

Fig 3.6.1 Accuracy

3.7 Final 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
```

Fig 3.6.2 Email Classification

When it shows False it means that it is not a spam and similarly for True it is spam

This thesis has addressed the problem of Spam E-mails. 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 behavioral word and e-mails.

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 (2012).

[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, 2011.

[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, 2014.

[5] Choo, Jaegul, and Haesun Park. "Customizing computational methods for visual analytics with big data." *IEEE Computer Graphics and Applications* 33.4 (2013).

[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 (2012).

[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 Interntional 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 (2009).

[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, 2010.

[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, 2010.