DOI : https://doi.org/10.56452/7-12-46

ISSN: 0974-5823 Vol. 7 No. 12 December, 2022

International Journal of Mechanical Engineering

Email Spam Detection using Machine Learning

Dr. Nilesh Jain

Associate Professor, Mandsaur University *nileshjainmca@gmail.com*

Dr. B. K. Sharma

[1]*Professor, Mandsaur University, Mandsaur, e-mail:* *bksharma7426@gmail.com*

ABSTRACT- *Spam emails are known as unrequested commercialized emails or deceptive emails sent to a specific person or a company [5]. Spams can be detected through natural language processing and machine learning methodologies. Machine learning methods are commonly used in spam filtering. These methods are used to render spam classifying emails to either ham (valid messages) or spam (unwanted messages) with the use of Machine Learning classifiers. The proposed work showcases differentiating features of the content of documents [4]. There has been a lot of work that has been performed in the area of spam filtering which is limited to some domains. Research on spam email detection either focuses on natural language processing methodologies [25] on single machine learning algorithms or one natural language processing technique [22] on multiple machine learning algorithms [2]. In this Project, a modeling pipeline is developed to review the machine learning methodologies.*

Keywords: *Email Spam Detection, Spam Detection, Machine Learning, Neural Networks, Naive Bayes, Support Vector Classifier, Logistic Regression, Spam, Social Media, Email.*

1. INTRODUCTION

Technology has become a vital part of life in today's time. With each passing day, the use of the internet increases exponentially, and with it, the use of email for the purpose of exchanging information and communicating has also increased, it has become second nature to most people. While e-mails are necessary for everyone, they also come with unnecessary, undesirable bulk mails, which are also called Spam Mails [29]. Anyone with access to the internet can receive spam on their devices. Most spam emails divert people's attention away from genuine and important emails and direct them towards detrimental situations. Spam emails are capable of filling up inboxes or storage

capacities, deteriorating the speed of the internet to a great extent. These emails have the capability of corrupting one's system by smuggling viruses into it, or steal useful information and scam gullible people. The identification of spam emails is a very tedious task and can get frustrating sometimes.

While spam detection can be done manually, filtering out a large number of spam emails can take very long and waste a lot of time. Hence, the need for spam detection softwares has become the need of the hour. To solve this problem, various spam detection techniques are used now. The most common technique for spam detection is the utilization of Naive Bayesian [5] method and feature sets that assess the presence of spam keywords. The main purpose is to demonstrate an alternative scheme, with the use of Neural Network (NN) [4] classification system that utilises a collection of emails sent by several users, is one of the objectives of this research. One other purpose is the development of spam detection with the help of Artificial Neural Networks, resulting in almost 98.8% accuracy.

2. LITERATURE SURVEY

Email :

Electronic mail (email) is a messaging system that electronically transmits messages across computer networks. Anyone is free to use email services through Gmail, Yahoo or people can even register with an Internet Service Provider (ISPs) and be provided with an email account. Only an internet connection is required, otherwise being a free service.

Spam :
Bulk mails that are unnecessary and undesirable can be classified as Spam Mails. These spam emails hold the power to corrupt one's system by filling up inboxes, degrading the speed of their internet connection.

Spam Detection :

Many spam detection techniques are being used now-a-days. The methods use filters which can prevent emails from causing any harm to the user. The contributions and their weakness have been identified.

There are several methods that are accessible to spam, for example location of sender, it's contents, checking IP address or space names. [26]. Spammers use refined variations to avoid spam identification. Few measures connected with spam identification are; Blacklist and white-list, Machine learning approaches, Naïve Bayes, Support Vector Machine, Neural Network Classification. [27]

A mobile system was proposed by Mahmoud et al. [28] with the motive of blocking and identifying spam SMS. In their work, they attempted to protect smartphones by filtering SMS spam that contains abbreviations and idioms. The system was based on the Artificial Immune System (AIS) and Naïve Bayesian (NB) algorithm. By the use of the Naive Bayes algorithm, the messages are classified based on their features. It used an SMS dataset with 1324 messages. Results from this system gave detection rate 82%, 6% positive rate and 91% accuracy.

Table 1 : Spam Categories

Categories	Descriptions
Health	The spam of fake medications
Promotional products	The spam of fake fashion items like clothes bags and watches
Adult content	The spam of adult content of pornography and prostitution
Finance & marketing	The spam of stock kiting, tax solutions, and loan packages
Phishing	The spam of phishing or fraud

An approach using random forest algorithm approach is proposed by Akinyelu and Adewumi [1] in order to identify the phishing or spam emails. It used 200 emails. The main motto of research was to reduce features and increase efficiency/accuracy. Accuracy of up to 99.7% with a minimal amount of 0.06% false positives is achieved by the proposed algorithm.
The research only covered the classification aspect without considering vital information which can affect the results, especially, in case of limited text in the email.

Yüksel et al. [3] aimed to resolve the problem of spam by inhibiting the spam emails from being spread within the

email systems. To achieve this, they propose a cloud base system, which involves the identification of spam emails using analytics and machine learning algorithms like support vector machines and decision trees. The results of the tests show that the SVM leads to a higher accuracy of up to 97.6% and a false-positive rate of 2.33%. The decision tree attains a lower accuracy of 82.6% and a false-positive rate of 17.3%. Results reveal that the increase in spam emails is affected by the no. of received emails. Lee et al. [28] proposed an optimal technique for spam detection.

2.1. EXISTING SYSTEMS

Due to the increase in the number of email users, the amount of spam emails have also risen in number in the past years. It has now become even more challenging to handle a wide range of emails for data mining and machine learning. Therefore, many researchers have executed comparative studies to see various classification algorithms performances and their results in classifying emails accurately with the help of a number of performance metrics. Hence, it is important to find an algorithm that gives the best possible outcome for any particular metric for correct classification of emails and spam or ham.
The present systems of spam detection are reliant on three major methods:-

A. Linguistic Based Methods
Unlike humans, who can grasp linguistic constructs along with their exposition, machines cannot and hence it is necessary to teach machines some languages to help them understand these constructs. This is the technique that is used in

places like search engines in order to ascertain the next terms for suggestions to the user while they are typing their search. Sentences are divided into two Unigrams (words taken are one by one) and two Bigrams (words that are taken two at a time). Since this technique requires that every expression be remembered, this method is not feasible and also time-intensive. [29]

B.Behavior-Based Methods

This technique is Metadata-based. This approach requires that users generate a set of rules, and the users must have a thorough understanding of these rules. Since the attributes of spam change over time so the rules also need to be reformed from time to time. As a result, it still requires a human to scrutinise the details and is majorly user-dependent. [29]

C.Graph-Based Methods

This technique uses a single graphical representation by incorporating numerous, heterogeneous particulars. Graph-based anomaly recognition algorithms are executed which detect abnormal forms in the data showing behaviours of spammers. This method is not dependable, so it is taxing to recognise faulty opinions. [29] Feature

Engineering mostly depends on the commercial appeal of terms and is absolutely content-oriented, and does not depend on statistics. All these attributes lead to a noteworthy decline of this structure.

3. PROPOSED METHOD

Many several techniques are present in the market to detect spam e-mails. If we want to classify broadly, there are 5 different techniques based on which algorithms decide whether any mail is spam or not.

1. Content-Based Filtering Technique

Algorithms analyze words, the occurrence of words, and the distribution of words and phrases inside the content of e-mails and segregate them into spam non-spam categories.

2. Case Base Spam Filtering Method

Algorithms trained on well-annotated spam/non-spam marked emails try to classify the incoming mails into two categories.

3. Heuristic or Rule-Based Spam Filtering Technique

Algorithms use pre-defined rules in the form of a regular expression to give a score to the messages present in the e-mails. Based on the scores generated, they segregate emails into spam non-spam categories.

4. The Previous Likeness Based Spam Filtering Technique

Algorithms extract the incoming mails' features and create a multi-dimensional space vector and draw points for every new instance. Based on the KNN algorithm, these new points get assigned to the closest class of spam and non-spam.

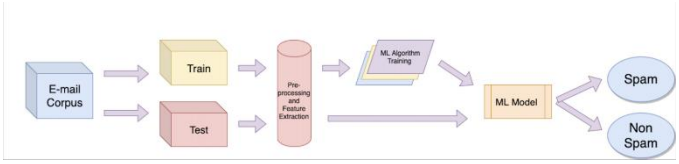
5. Adaptive Spam Filtering Technique

Algorithms classify the incoming mails in various groups and, based on the comparison scores of every group with the defined set of groups, spam and non-spam emails got segregated.

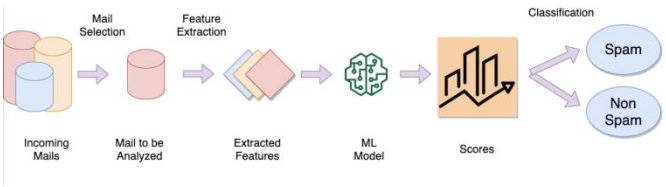
This article will give an idea for implementing content-based filtering using one of the most famous algorithms for spam detection, which is K-Nearest Neighbour (KNN).

k-NN based algorithms are widely used for clustering tasks. Let’s quickly know the entire architecture of this implementation first and then explore every step. Executing these 5 steps, one after the other, will help us implement our spam classifier smoothly.

Training Testing Phase



New Email Classification



Step 1: E-mail Data Collection

DOI : https://doi.org/10.56452/7-12-46

The dataset contained in a corpus plays a crucial role in assessing the performance of any spam filter. Many open-source datasets are freely available in the public domain.

Train/Test Split: Split the dataset into train and test datasets but make sure that both sets must balance numbers of ham and spam emails (ham is a fancy name for non-spam emails).

Below are a few of the famous repositories where you can easily get thousand kind of data set for free :

UC Irvine Machine Learning Repository

Kaggle datasets

AWS datasets

For this email spamming data set, it is distributed by Spam Assassin, you can click this link to go to the data set. There are a few categories of the data, you can read the readme.html to get more background information on the data.

In short, there is two types of data present in this repository, which is ham (non-spam) and spam data. Furthermore, in the ham data, there are easy and hard, which mean there is some non-spam data that has a very high similarity with spam data. This might pose a difficulty for our system to make a decision.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis is a very important process of data science. It helps the data scientist to understand the data at hand and relates it with the business context.

The open source tools that I will be using in visualizing and analyzing my data is Word Cloud.

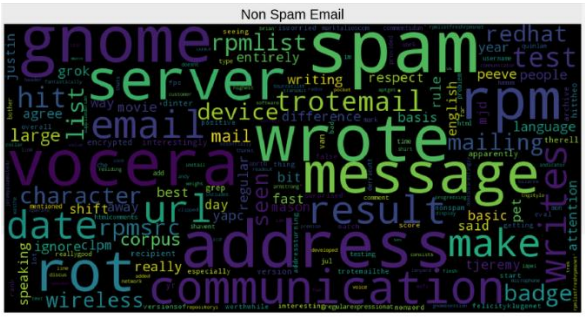
Word Cloud is a data visualization tool used for representing text data. The size of the texts in the image represent the frequency or importance of the words in the training data.

Visualization

Wordcloud

Wordcloud is a useful visualization tool for you to have a rough estimate of the words that has the highest frequency in the data that you have.

Visualization for spam email



Visualization for non spam email

From this visualization, you can notice something interesting about the spam email. A lot of them are having high number of “spammy” words such as: free, money, product etc. Having this awareness might help us to make better decision when it comes to designing the spam detection system.

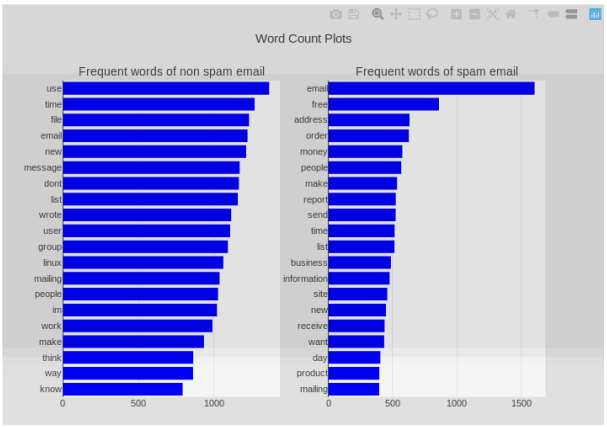
One important thing to note is that word cloud only displays the frequency of the words, not necessarily the importance of the words. Hence it is necessary to do some data cleaning such as removing stopwords, punctuation and so on from the data before visualizing it.

N-grams model visualization

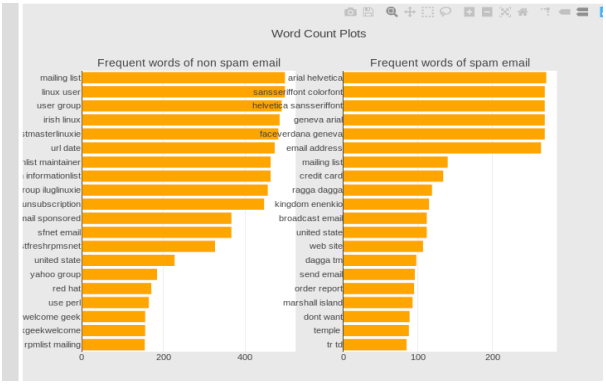
Another technique of visualization is by utilizing bar chart and display the frequency of the words that appear the most. N-gram means that how many words you are considering as a single unit when you are calculating the frequency of words.

Followings are the example of 1-gram, and 2-gram.

DOI : https://doi.org/10.56452/7-12-46



Bar chart visualization of 1-gram model



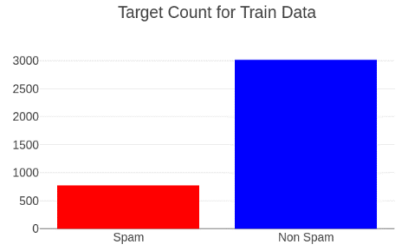
Bar chart visualization of 2-gram model

Train Test Split

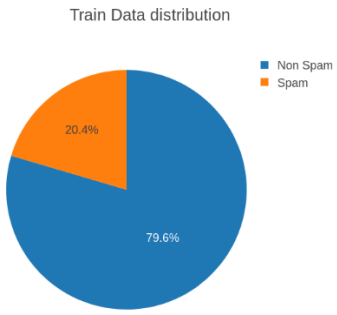
It is important to split your data set to training set and test set, so that you can evaluate the performance of your model using the test set before deploying it in a production environment.

One important thing to note when doing the train test split is to make sure the distribution of the data between the training set and testing set are similar.

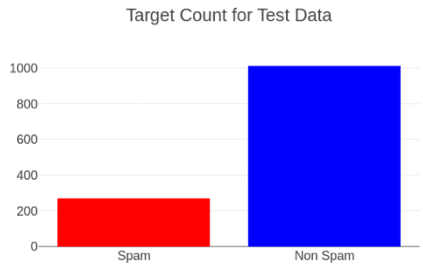
What it means in this context is that the percentage of spam email in the training set and test set should be similar.



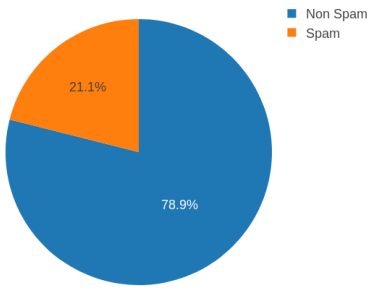
Target Count For Train Data



Train Data Distribution



Count For Test Data



Test Data Distribution

The distribution between train data and test data are quite similar which is around 20–21%, so we are good to go and start to process our data !

Data Preprocessing

Text Cleaning

Text Cleaning is a very important step in machine learning because your data may contains a lot of noise and unwanted character such as punctuation, white space, numbers, hyperlink and etc.

Some standard procedures generally used are:

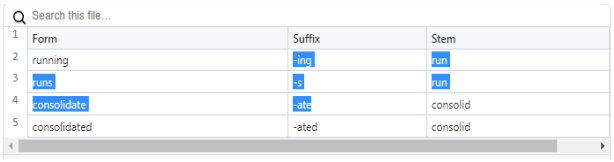
1. convert all letters to lower/upper case
2. removing numbers
3. removing punctuation
4. removing white spaces
5. removing hyperlink

removing stop words such as a, about, above, down,

doing and the list goes on…

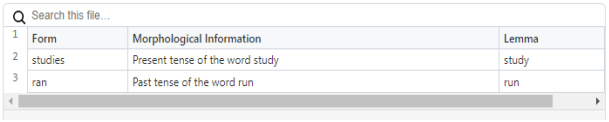
Word Stemming and Word lemmatization these are the two techniques are trying to reduce the words to its most basic form, but doing this with different approaches.

Word stemming — Stemming algorithms work by removing the end or the beginning of the words, using a list of common prefixes and suffixes that can be found in that language. Examples of Word Stemming for English words are as below:



Form	Suffix	Stem
running	-ing	run
runs	-s	run
consolidate	-ate	consolid
consolidated	-ated	consolid

Word Lemmatization — Lemmatization is utilizing the dictionary of a particular language and tried to convert the words back to its base form. It will try to take into account of the meaning of the verbs and convert it back to the most suitable base form.



Form	Morphological Information	Lemma
studies	Present tense of the word study	study
ran	Past tense of the word run	run

Implementing these two algorithms to deal with different edge cases.

Import the library and start designing some functions to help us understand the basic working of these two algorithms.

```
# Just import them and use it
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
dirty_text = "He studies in the house yesterday, unluckily, the fans breaks down"
def word_stemmer(words):
    stem_words = [stemmer.stem(o) for o in words]
    return " ".join(stem_words)
def word_lemmatizer(words):
  lemma_words = [lemmatizer.lemmatize(o) for o in words]
  return " ".join(lemma_words)
```

The output of word stemmer is very obvious, some of the endings of the words have been chopped off

```
clean_text = word_stemmer(dirty_text.split(" "))
clean_text
#Output
'He studi in the hous yesterday, unluckily, the fan break down'
```

The lemmatization has converted studies -> study, breaks -> break

```
clean_text = word_lemmatizer(dirty_text.split(" "))
clean_text

#Output

'I study in the house yesterday, unluckily, the fan break down'
```

Feature Extraction

Our algorithm always expect the input to be integers/floats, so we need to have some feature extraction layer in the middle to convert the words to integers/floats.

There are a couples ways of doing this as following

1. CountVectorizer
2. TfidfVectorizer
3. Word Embedding

CountVectorizer

First we need to input all the training data into CountVectorizer and the CountVectorizer will keep a dictionary of every word and its respective id and this id will relate to the word count of this word inside this whole training set.

For example, a sentence like ‘I like to eat apple and drink apple juice’

```
from sklearn.feature_extraction.text import CountVectorizer
# list of text documents
text = ["I like to eat apple and drink apple juice"]
# create the transform
vectorizer = CountVectorizer()
# tokenize and build vocab
vectorizer.fit(text)
# summarize
print(vectorizer.vocabulary_)
# encode document
vector = vectorizer.transform(text)
# summarize encoded vector
```

```
print(vector.shape)
print(type(vector))
print(vector.toarray())
# Output
# The number follow by the word are the index of the word
{'like': 5, 'to': 6, 'eat': 3, 'apple': 1, 'and': 0, 'drink': 2, 'juice': 4}
# The index relates to the position of the word count array below
# "I like to eat apple and drink apple juice" -> [1 2 1 1 1 1 1]
# apple which has the index 1 correspond to the word count of 2 in the array
```

TfidfVectorizer

Word counts are good but can we do better? One issue with simple word count is that some words like ‘the’, ‘and’ will appear many times and they don’t really add too much meaningful information.

Another popular alternative is TfidfVectorizer. Besides of taking the word count of every words, words that often appears across multiple documents or sentences, the vectorizer will try to downscale them.

For more info about CountVectorizer and TfidfVectorizer, please read from this great piece of article, which is also where I gain most of my understanding.

Word Embedding

Word embedding is trying to convert a word to a vectorized format and this vector represents the position of this word in a higher dimensional space.

For words that have similar meaning, the cosine distance of those two word vectors will be shorter and they will be closer to each other.

And in fact, these words are vectors, so you can even perform math operations on them ! The end results of these operation will be another vector that maps to a word. Unexpectedly, those operations produce some amazing result !

Example 1 : King- Man + Woman = Queen

Example 2: Madrid-Spain+France = Paris

Example 3: Walking-Swimming+Swam= Walked

Simply put, word embedding is a very powerful representation of the words and one of the well known techniques in generating this embedding is Word2Vec.

Algorithm Implementation

TfidfVectorizer + Naive Bayes Algorithm

The first approach to use the TfidfVectorizer as a feature extraction tools and Naive Bayes algorithm to do the prediction. Naive Bayes is a simple and a probabilistic traditional machine learning algorithm.

It is very popular even in the past in solving problems like spam detection. Using Naive Bayes library provided by sklearn library save us a lot of hassle in implementing this algorithm. This can be easily get in a few lines of codes

```
from sklearn.naive_bayes import GaussianNB
clf.fit(x_train_features.toarray(),y_train)
# Output of the score is the accuracy of the prediction
# Accuracy: 0.995
clf.score(x_train_features.toarray(),y_train)
# Accuracy: 0.932

clf.score(x_test_features.toarray(),y_test)
```

We achieve an accuracy of 93.2%. But accuracy is not solely the metrics to evaluate the performance of an algorithm. So other scoring metrics and that may help us to understand thoroughly how well this model is doing.

Scoring & Metrics

When it comes to evaluation of a data science model’s performance, sometimes accuracy may not be the best indicator.

Some problems that we are solving in real life might have a very imbalanced class and using accuracy might not give us enough confidence to understand the algorithm’s performance.

In the email spamming problem the spam data is approximately 20% of our data. If our algorithm predicts all the email as non-spam, it will achieve an accuracy of 80%.

And for some problem that has only 1% of positive data, predicting all the sample as negative will give them an accuracy of 99% but we all know this kind of model is useless in a real life scenario.

DOI : https://doi.org/10.56452/7-12-46

Precision & Recall

Precision & Recall is the common evaluation metrics that people use when they are evaluating class-imbalanced classification model.

Precision is evaluating, when a model predict something as positive, how accurate the model is. On the other hand, recall is evaluating how well a model in finding all the positive samples.

The mathematical equation for precision & recall are as respective

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN}$$

TP: True Positive

FP : False Positive

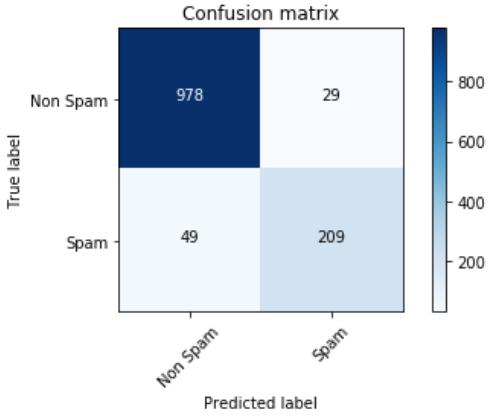
TN: True Negative

FN: False Negative

Confusion Matrix

Confusion Matrix is a very good way to understand results like true positive, false positive, true negative and so on.

Sklearn documentation has provided a sample code of how to plot nice looking confusion matrix to visualize your result.



Confusion Matrix of the result

Precision: 87.82%

Recall: 81.01%

The recall of this model is rather low, it might not be doing a good enough job in discovering the spam email.

Summary

I have showed you all the necessary steps needed in designing a spam detection algorithm. Just a brief recap:

Explore and understand your data

Visualize the data at hand to gain a better intuition — Wordcloud, N-gram Bar Chart

Text Cleaning — Word Stemmer and Word Lemmatization

Feature Extraction — Count Vectorizer, Tfidf Vectorizer, Word Embedding

Algorithm — Naive Bayes

Scoring & Metrics — Accuracy, Precision, Recall

Here concludes the first part of demonstration in designing spam detection algorithm.

4. CONCLUSION

As shown in Figure 4, all the models based on the feature set 2 most-frequent-word-count have higher accuracy and F1 score than those based on the feature set 1 stop words
+ n-gram + tf-IDF.

If the use case is to introduce a beta version of an email spam detector like no-spam in the inbox. In this case, the model: Neural Network with tanh activation function and the feature set 1 stop words + n-gram + tf-IDF serves this purpose.

According to the graphs in Figure 4, if the use case is to introduce an email spam detector to reduce bad user experience in searching for important emails from junk mailboxes and filtering spam from the inbox. In this case, Neural Network with a feature set 2 - ‘most frequent word count’ gives a better user experience in general.

The future work includes testing the model with various standard datasets. This research proposes that the outcome that is obtained should be compared with additional spam datasets from various sources. Also, more classification and

feature algorithms should be analyzed with email spam datasets.

5. REFERENCES

[1] AKINYELU, A. A., & ADEWUMI, A. O. (2014). "Classification of phishing email using random forest machine learning technique". Journal of Applied Mathematics.

[2] Vinodhini. M, Prithvi. D, Balaji. S "Spam Detection Framework using ML Algorithm" in IJRTE ISSN: 2277-3878, Vol.8 Issue.6, March 2020.

[3] YUsKSEL, A. S., CANKAYA, S. F., & UsNCUs, It. S. (2017). "Design of a Machine Learning Based Predictive Analytics System for Spam Problem." Acta Physica Polonica, A.,132(3).[26] GOODMAN, J. (2004, July). "IP Addresses in

[4] Email Clients." In CEAS.

[5] Deepika Mallampati, Nagaratna P. Hegde "A Machine Learning Based Email Spam Classification Framework Model" in IJITEE, ISSN: 2278-3075, Vol.9 Issue.4, February 2020.

[6] Javatpoint, "Machine Learning Tutorial" 2017 https://www.javatpoint.com/machi ne- learning

[7] SpamAssassin, "Spam and Ham Dataset", Kaggle, 2018. https://www.kaggle.com/veleon/ham-and-spam-dataset

[8] Apache, "open-source Apache SpamAssassin Dataset", 2019 https://spamassassin.apache.org/old/publiccorpus/

[9] SpamAssassin, "Spam Classification Kernel", 2018 https://www.kaggle.com/veleon/spam-classification

[10] SpamAssassin, "REVISION HISTORY OF THIS CORPUS", 2016 https://spamassassin.apache.org/old/publiccorpus/read me.html

[11] Jason Brownlee, "Naive Bayes for Machine Learning" The Machine Learning Mastery, April 11, 2015. https://machinelearningmastery.com/naive-bayes-for-machine- learning/

[12] Wikipedia, "History of email spam," Internet Free Encyclopedia, 2001. https://en.wikipedia.org/wiki/History_of_ email_spam

[13] Rohith Gandhi, "Support Vector Machine" The Machine Learning Mastery, June 7, 2018. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[14] Jason Brownlee, "Logistic Regression for Machine Learning" The Machine Learning Mastery, April 1, 2016. https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[15] Jason Brownlee, "How to Encode Text Data for Machine Learning with scikit- learn" The Machine Learning Mastery, September 29, 2017. https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

[16] I. Androutsopoulos, J. Koutsias, K. Chandrinos and C.
D. Spyropoulos, "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal email messages," Computation and Language, pp. 160-167, 2000.

[17] G. V. Cormack, "Email Spam Filtering: A Systematic Review," Foundations and Trends® in Information Retrieval, vol. 1, no. 4, pp. 335-455, 2006.

[18] M. Siponen and C. Stucke, "Effective Anti-Spam Strategies in Companies: An International Study," Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), 2006.

[19] Guzella, T. S. and Caminhas, W. M."A review of machine learning approaches to Spam filtering." Expert Syst. Appl., 2009.

[20] Jianying Zhou, Wee-Yung Chin, Rodrigo Roman, and Javier Lopez, (2007) "An Effective MultiLayered Defense Framework against Spam", Information Security Technical Report 01/2007.

[21] Xiao Mang Li, Ung Mo Kim, (2012) "A hierarchical framework for content-based image spam filtering", 8th International Conference on Information Science and Digital Content Technology (ICIDT), Jeju, June, pp. 149-155.

[22] Linda Huang, Julia Jia, Emma Ingram, Wuxu Peng, "Enhancing the Naive Bayes Spam Filter through Intelligent Text Modification Detection", 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.

[23] W.A. Awad, S.M. Elseuofi, Machine learning methods for spam E-mail classification, Int. J. Comput. Sci. Inf. Technol. 3 (1) (2011) 173–184.

[24] K.R. Dhanaraj, V. Palaniswami, Firefly and Bayes classifier for email spam classification in a distributed environment, Aust. J. Basic Appl. Sci. 8 (17) (2014) 118–130.

[25] M. Zavvar, M. Rezaei, S. Garavand, Email spam detection using combination of particle swarm optimization and artificial neural network and support vector machine Int. J Mod Educ. Comput.Sci. (2016) 68-74.

[26] Deepika Mallampati, "An Efficient Spam Filtering using Supervised Machine Learning Techniques" in IJSRCSE, Vol.6, Issue.2, pp.33-37, April (2018).

[27] [Deepika Mallampati, K.Chandra Shekar and K.Ravikanth "Supervised Machine Learning Classifier for Email Spam Filtering", © Springer Nature Singapore Pte Ltd. 2019 and Engineering, https://doi.org/10.1007/978-981-13-7082-341.

[28] GUPTA, H., JAMAL, M. S., MADISETTY, S., & DESARKAR, M. S. (2018, January). "A framework for real-time spam detection in Twitter." In Communication Systems & Networks (COMSNETS), 2018 10th International Conference on (pp. 380-383).

[29] MAHMOUD, T. M., & MAHFOUZ, A. M. (2012). "SMS spam filtering technique based on artificial immune system." International Journal of Computer Science Issues (IJCSI), 9(2), 589.

[30] AN ANTI-SPAM DETECTION MODEL FOR EMAILS OF MULTI-NATURAL LANGUAGE Mazin Abed Mohammed a,*, Salama A. Mostafa b,*, Omar Ibrahim Obaid