

Phishing Mail Detection

A PROJECT REPORT

Submitted by

Vaibhav Shrivastava (21BCY10227)

Yash Garg (21BCY10032)

Lakshay Gaur (21BCY10212)

Kavya Singhla (21BCY10204)

*in partial fulfillment for the award of the degree
of*

BACHELOR OF TECHNOLOGY

in

Computer Science Engineering

Spec. Cybersecurity and Digital Forensics



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

**KOTRIKALAN, SEHORE
MADHYA PRADESH - 466114**

February 2023

**VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE
MADHYA PRADESH – 466114**

BONAFIDE CERTIFICATE

Certified that this project report titled “**Phishing Mail Detection**” is the bonafide work of “**Vaibhav Shrivastava(21BCY10227), Yash Garg(21BCY10032), Lakshay Gaur(21BCY10212), Kavya Singhla(21BCY10204)**” who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR

Dr D. Sarvanan,
Cyber Security and Digital Forensics
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE

Dr. Shahana Qureshi
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition II Examination is held on _____

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude o Dr. S Poonkuntran Dean, School of Computer Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Ms. Dr. Shahana Qureshi ,for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Aeronautical Science, who extended directly or indirectly all support.

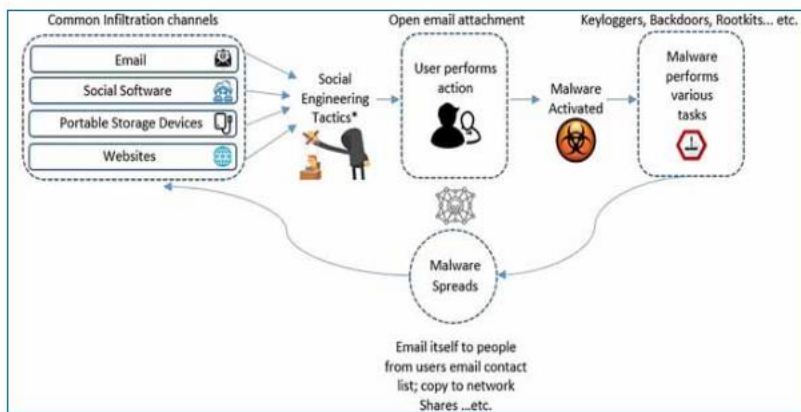
Last, but not the least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

TABLE OF CONTENTS

Sr. No.	TOPIC	Page No.
1.	Abstract	5
2.	Literature Survey	6
3.	Project Procedure	7
4.	Work done	8
5.	Observation	9
6.	Conclusion	10
7.	Recommendation for further work	11
8.	References	11

ABSTRACT

Phishing is a cybercrime, which involves luring the user into providing sensitive and confidential information to the attacker. The information could include credit card details, username and passwords, bank details, etc. These phishing attacks occur through malicious emails, text messages and telephone calls. After obtaining the information, the attacker could commit crimes such as financial losses and identity thefts. The target could be an individual, an organization or a cluster in an organization. In this type of cyber-attack, the attacker sends malicious links or attachments through phishing e-mails that can perform various functions, including capturing the login credentials or account information of the victim. These e-mails harm victims because of money loss and identity theft. In this study, a software called “Anti Phishing mail detector” was designed, giving information about the detection problem of phishing and how to detect phishing emails. With this software, phishing and spam mails are detected by examining mail contents. Classification of spam words added to the database by Bayesian and Naïve algorithm is provided.



LITERATURE SURVEY

The systematic literature review is a research process that follows a set of rules. The review methodology includes constructing research questions, identifying the list of electronic databases to be explored, data collection, data analysis, discussion on findings, and a comparison study of final selected research articles once all exclusion criteria have been applied. This systematic literature review aims to find the best approach, data set, and algorithm researchers employ for phishing mail detection. The methodology that is used for the filtering method is machine learning techniques that divide by three phase. The methodology is used for the process of e-mail spam filtering based on Naïve Bayes algorithm.

The Naïve Bayes algorithm is a simple probabilistic classifier that calculates a set of probabilities by counting the frequency and combination of values in a given dataset [4]. In this research, Naïve Bayes classifier use bag of words features to identify spam e-mail and a text is representing as the bag of its word. The bag of words is always used in methods of document classification, where the

frequency of occurrence of each word is used as a feature for training classifier. This bag of words features are included in the chosen datasets. Naïve Bayes technique used Bayes theorem to determine that probabilities spam e-mail. Some words have particular probabilities of occurring in spam e-mail or non-spam e-mail. Example, suppose that we know exactly, that the word Free could never occur in a non-spam e-mail. Then, when we saw a message containing this word, we could tell for sure that were spam email. Bayesian spam filters have learned a very high spam probability for the words such as Free and Viagra, but a very low spam probability for words seen in non-spam e-mail, such as the names of friend and family member. So, to calculate the probability that e-mail is spam or non-spam Naïve Bayes technique used Bayes theorem as shown in formula below. Where:

(i) $P(\text{spam}|\text{word})$ is probability that an e-mail has particular word given the e-mail is spam. (ii) $P(\text{spam})$ is probability that any given message is spam. (iii) $P(\text{word}|\text{spam})$ is probability that the particular word appears in spam message. (iv) $P(\text{non-spam})$ is the probability that any particular word is not spam. (v) $P(\text{word}|\text{non-spam})$ is the probability that the particular word appears in non-spam message. To achieve the objective, the research and procedure is conducted in three phases. The phases involved are as follows: (i) Phase 1: Pre-processing (ii) Phase 2: Feature Selection (iii) Phase 3: Naïve Bayes Classifier The following sections will explain the activities that involve in each phases in order to develop this project. Figure 2 shows the process for e-mail spam filtering based on Naïve Bayes algorithm.

3.2. Pre-processing

Today, most of the data in the real world are incomplete containing aggregate, noisy and missing values [9]. Pre-processing of e-mails in next step of training filter, some words like conjunction words, articles are removed from email body because those words are not useful in classification. As mentioned earlier, we are using WEKA tool to facilitate the experiments. For both experiments, the datasets are presented in Attribute-Relation File Format (ARFF) file (Refer to Figure 3 for sample of data).

4 1234567890 International Research and Innovation Summit (IRIS2017) IOP Publishing IOP Conf. Series: Materials Science and Engineering 226 (2017) 012091 doi:10.1088/1757-899X/226/1/012091

Figure 2. Process of E-mail spam filtering based on Naïve Bayes Algorithm

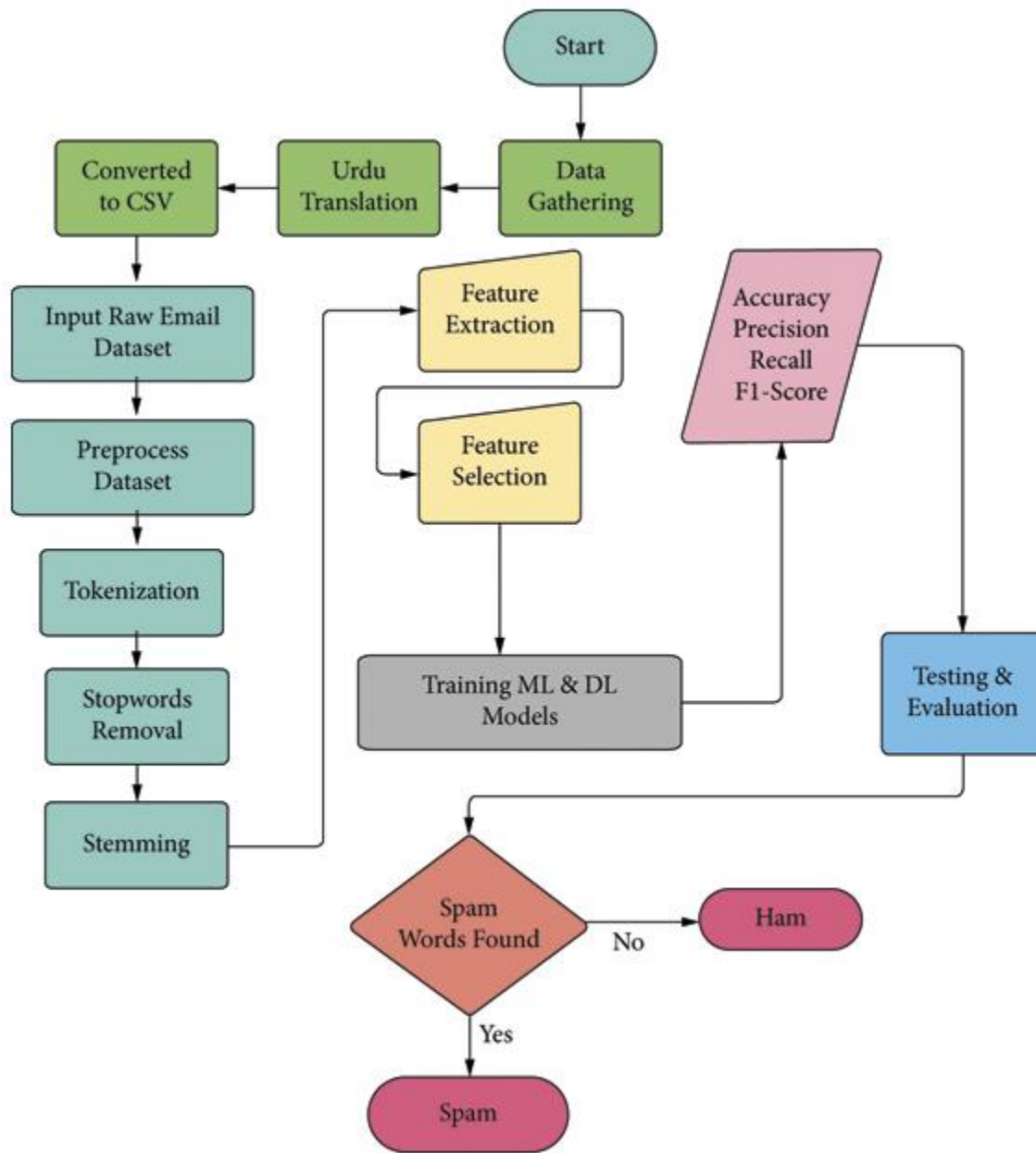
Figure 3. Sample of spam data fragment in ARFF format

A full list of the attributes in this data set appears in the "Attributes" frame as shown in Figure 4. Random selection of attribute are performed for the further process. Attributes capital run length average, capital run length longest and capital run length total are removed from the list by checking the box to their left and hitting the Remove button.

3.3. Feature Selection

After the pre-processing step, we apply the feature selection algorithm, the algorithm which deploy here is Best First Feature Selection algorithm.

PROJECT PROCEDURE



WORK DONE

We'll use [machine learning to train our spam detector](#) to recognize and classify emails into spam and non-spam. Let's get started!

Prerequisites

First, we'll import the necessary dependencies. Pandas is a library used mostly used by data scientists for data cleaning and analysis.

[Scikit-learn](#), also called Sklearn, is a robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction via a consistent interface.

We'll use a train-test split method to train our email spam detector to recognize and categorize spam emails. The train-test split is a technique for evaluating the performance of a machine learning algorithm. We can use it for either classification or regression of any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two separate datasets. The first dataset is used to fit the model and is referred to as the training dataset. For the second dataset, the test dataset, we provide the input element to the model. Finally, we make predictions, comparing them against the actual output.

- Train dataset: used to fit the machine learning model
- Test dataset: used to evaluate the fit of the machine learning model

In practice, we'd fit the model on available data with known inputs and outputs. Then, we'd make predictions based on new examples for which we don't have the expected output or target values. We'll take the data from our sample .csv file, which contains examples pre-classified into spam and non-spam, using the labels spam and ham, respectively.

To split the data into our two datasets, we'll use scikit-learn's `train_test_split()` method.

Let's say we have 100 records in the loaded dataset. If we specify the test dataset is 30 percent, we'll split 70 records for training and use the remaining 30 records for testing.

In `cv = CountVectorizer()`, `CountVectorizer()` randomly assigns a number to each word in a process called tokenizing. Then, it counts the number of occurrences of words and saves it to `cv`. At this point, we've only assigned a method to `cv`.

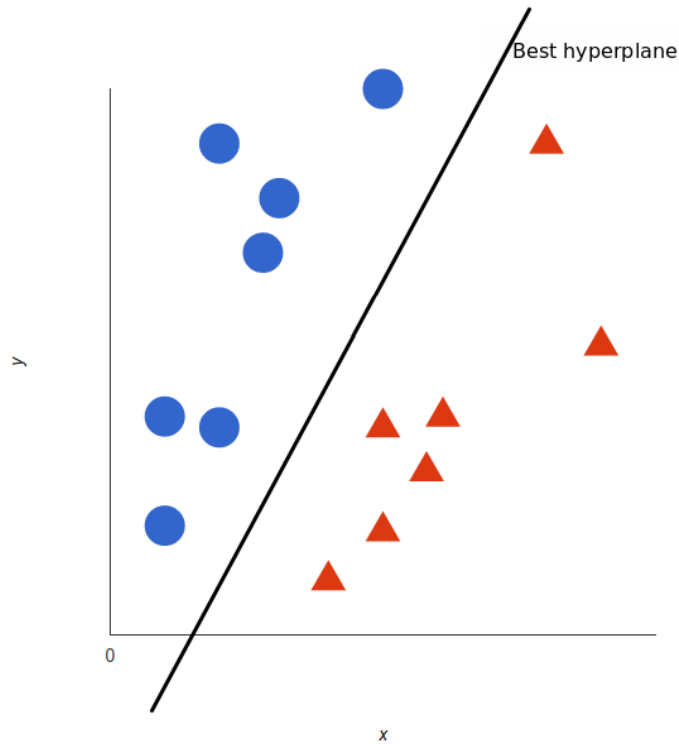
`features = cv.fit_transform(z_train)` randomly assigns a number to each word. It counts the number of occurrences of each word, then saves it to `cv`. In the image below, 0 represents the index of the email. The number sequences in the middle column represent a word recognized by our function, and the numbers on the right indicate the number of times that word was counted:

For example, in the image above, the word corresponding to 1841 is used twice in email number 0.

Now, our machine learning model will be able to predict spam emails based on the number of occurrences of certain words that are common in spam emails.

Building the model

SVM, the support vector machine algorithm, is a linear model for classification and regression. The idea of SVM is simple, the algorithm creates a line, or a hyperplane, which separates the data into classes. SVM can solve both linear and non-linear problems:



Let's create an SVM model with the code below:

`model = svm.SVC()` assigns `svm.SVC()` to the model. In the `model.fit(features,y_train)` function, `model.fit` trains the model with `features` and `y_train`. Then, it checks the prediction against the `y_train` label and adjusts its parameters until it reaches the highest possible accuracy.


Testing our email spam detector

Now, to ensure accuracy, let's test our application. Run the code below:

The `features_test = cv.transform(z_test)` function makes predictions from `z_test` that will go through count vectorization. It saves the results to the `features_test` file.

In the `print(model.score(features_test,y_test))` function, `model.score()` scores the prediction of `features_test` against the actual labels in `y_test`.

OBSERVATION



```

input_mail = ["Dear User, All Hotmail customers have been upgraded to Outlook.com. Your Hotmail Account services has expired.Due to our new system upgrade to Outlook. I

# convert text to feature vectors
input_data_features = feature_extraction.transform(input_mail)

# making prediction

prediction = model.predict(input_data_features)
print(prediction)

if (prediction[0]==1):
    print('Safe')
else:
    print('Pshing mail')

```

[1]
Safe

CONCLUSION

Given a set of words, we used feature selection to obtain words which allow us to distinguish between spam and ham emails. We also compared the accuracy of various classifiers in predicting the class attribute. With the right tools and techniques, it is possible to build highly effective spam mail detection systems using machine learning. By leveraging the power of these techniques, we can help protect individuals and organizations from the growing threat of spam and other email-based attacks. Implementing spam filtering is extremely important for any organization. Not only does spam filtering help keep garbage out of email inboxes, it helps with the quality of life of business emails because they run smoothly and are only used for their desired purpose.

RECOMMENDATION FOR FUTURE WORK

Further research work needs to be conducted to tackle the fact that email spam filtering is a concept drift problem. As such, while the spam filter researchers are trying to increase the prognostic accuracy of the filter, the spammers are also evolving and trying to surpass the efficiency of the spam filters. It becomes very important to develop more efficient techniques that will adequately handle the trend or progression in spam features that makes them to evade many spam filters undetected. The most successful technique applied in filtering spam is the content-based spam filtering approach, which classifies emails as either spam or ham depending on the data that made up the content of the message. Examples of this technique include Bayesian Filtering, SVM, kNN classifier, Neural Network, AdaBoost classifier, and others. Systems based on machine learning approach facilitate learning and adjustment to recent dangers posed to the security of spam filters. They also have the capacity to counter curative channels that spammers are using. We hereby suggest that the future of email spam filters lies in deep learning for content-based classification and deep adversarial learning techniques.

REFERENCES

- Porter, M (n.d). The Porter Stemming Algorithm. Retrieved March 2011, from <http://tartarus.org/~martin/PorterStemmer>

- Retrieved March 2011, from <http://spamassassin.apache.org/publiccorpus/>