Spam Detection in Email using Machine Learning

***Abstract*—In today's world, email is used in almost every industry, from business to education. Emails can be categorized into two categories: ham and spam. Junk emails, also known as spam messages, are emails that have been designed to harm recipients by wasting their time, computing resources, and stealing their valuable information. It is estimated that spam emails are increasing at a rapid rate. One of the most important and prominent spam prevention techniques is filtering email. Naive Bayes, Decision Trees, Neural Networks, and Random Forests are among the methods used for this purpose by researchers. In this project, I examine the Logistic Regression machine learning model for spam filtering in email by categorizing messages into appropriate groups. This study also compares the techniques based on accuracy, precision, recall, etc. The accuracy level for this project was around 97%. Towards the end, these insights and future research directions, and challenges are outlined.**

Keywords—Machine Learning, Logistic Regression, Spam Email Filtering, TfidfVectorizer, Random State, Deployment

I. INTRODUCTION

In the modern era of information technology, information sharing is easier than ever. Users can exchange information on a variety of platforms from anywhere across the globe. Email has become currently the easiest, cheapest, and most rapid method of transmitting information in the world among all information sharing mediums. Emails, on the other hand, are vulnerable to a variety of attacks, the most popular and destructive of which is spam due to their simplicity [1]. Aside from wasting recipients' time and resources, receiving emails that are not related to their interests may contain malicious content in the form of attachments or URLs which may compromise the host system's security [2]. The term spam refers to any irrelevant and unwanted messages or emails sent by an attacker to significant numbers of recipients by email or any other means of communication [2].

Therefore, the security of the email system requires a great deal of attention. In spam emails, viruses, rats, and Trojans may be contained. Users are often lured towards online services by this technique. It is possible for attackers to send spam emails with attachments that contain multiple-file extensions, link to malicious, spamming websites, and worse, result in data and financial fraud and identity theft [3, 4]. It is possible to create keywords-based rules that serve as filters for email messages with many email providers. Even so, this method is not very practical because it is difficult, and users do not want to customize their email messages, which leads to spammers attacking their accounts [4].

A. The importance of Spam Detection in Email using Machine Learning

IoT has become a part of our daily lives over the last few decades and is growing rapidly. The emergence of IoT is leading to increased spam email problems. In order to detect and filter spam and spammers, the researchers proposed a variety of spam detection methods. Currently, Spam email detection methods mainly fall into two categories: those based on behaviour patterns and those based on semantic patterns. Each type of approach has its drawbacks and limitations. Since the advent of the Internet and increased communication around the globe, spam emails have grown significantly [5]. Through the Internet, spammers can send spam from anywhere in the world by hiding their identities. Spam mail still dominates the internet despite all the antispam tools and techniques available. Those attacks most commonly involve malicious emails containing links to malicious websites that can cause harm to the victim's personal information. The memory or capacity of servers can also be occupied by spam emails, slowing down their response times. All organizations carefully evaluate the tools available to battle spam in their environment to accurately detect spam emails and avoid the increasing issue of spam in emails. Whitelists and blacklists, mail header analysis, keyword checking, and spam detection are some of the popular mechanisms for analyzing incoming emails [6].

B. Solution Proposed

According to researchers, 40% of social networks have accounts that are utilised for spam [7]. By sending hidden links in the text, spammers target specific segments, review pages, or fan pages to promote pornographic or other product sites from fraudulent accounts. The same kinds of noxious emails are sent to the same kinds of individuals or associations on a regular basis. A better detection of these types of emails can be achieved by investigating these highlights. In order to differentiate between spam and non-spam emails, artificial intelligence (AI) can be used [8].

Headers, subjects, and bodies of the messages can be used to extract feature information for this solution. These data can then be grouped into spam and ham based on their nature. Detecting spam today is commonplace by using learning- based classifiers. Using learning-based classification, spam emails are suspected of having a set of specific features to distinguish them from legitimate emails. In learning-based models, identifying spam has become more complex due to many factors. There are several factors contributing to spam subjectivity, including idea drift, language problems, overhead processing, and latency in texting. According to my proposed method, 97% of the accuracy rate of emails is classified as spam and ham based on their nature, which is an outstanding achievement since existing systems lack such precision.

II. RELATED WORK

Email spam is defined as unsolicited fake bulk emails sent from any account or automated system. Spam emails are becoming more widespread by today, and it has become a major issue over the last decade. Typically, Spambots (a computerized application that crawls email addresses through the Internet) are used to collect Email IDs to send spam emails. In the detection of spam emails, machine learning has been playing a vital role recently. A supervised approach with feature selection on email spam detection was presented by Kaur and Verma [9]. For spam detection systems, they introduced the knowledge discovery process.

This poll also addresses the selection of characteristics based on N-Gram. After detecting N - 1 terms in a sentence or text corpus, N- Gram is a predictive-based method that predicts the probability of the following word occurrence [09, 10]. They compare nonmachine learning (Signatures, Blacklist and Whitelist, and mail header checking) with machine learning (Nave Bayes, Support Vector Machine, multilayer perceptron Neural Network) techniques for detecting spam emails. Since they are using all these supervised machine learning algorithms and evaluate the results based on precision, recall, and accuracy false positives are generated at a high rate depending on the dataset.

DeBarr and Wechsler introduced another spam filtering system that uses Random Forest algorithms to categorize spam emails and active learning to refine the categorization [11]. In their approach, each email has divided into two sections. For that, they have used email messages from RFC 822 (Internet) [11]. For training the dataset, the researchers have used Support Vector Machine, Random Forest, Naive Bayes, and KNN [11]. However, since the research solely depends on term frequency and inverse document frequency of all features of each email which leads to stopping the accuracy of the model at 95.2%.

Takhmiri and Haroonabadi [12] use a fuzzy Decision Tree and the Naive Bayes algorithm to provide a new method for detecting spam. They extract spam behaviour patterns using the baked voting algorithm. They did this because, in the real world, apparent features do not exist. For spam and ham email classification, decision trees utilise fuzzy Mamdani rules according to the research. They next use the Nave Bayes classifier [12,13] in the dataset. Eventually, by separating votes into smaller portions, the baking approach is applied. This method provides them with an optimum weight to improve accuracy. The study utilised a dataset of 1000 emails, of which 350 (35%) were spam and 650 (65%) were ham emails [14] which kind a short dataset.

To identify the emails as junk mail or ham, Verma and Sofat utilized the supervised machine learning method ID3 to construct the decision trees of the study [15]. Further, the hidden Markov model was used to calculate the odds of many events occurring at the same time [16]. The proposed approach classifies all emails as spam or valid by calculating the total chance of each e-mail using previously classified email phrases. This study makes use of the Enron dataset, which contains 5172 emails together [15, 16]. 2086 of the 5172 emails were spam, while the other 2086 were legitimate. Using the feature set gathered from the Enron dataset, their algorithm can classify emails as spam or ham. Using the fitness function from the Scikit-Learn package in the suggested model, they got an 11% error. On the given dataset, their model had an accuracy rate of 89%.

III. DATASET

In this project, I am using a dataset obtained from Kaggle. I have named it “mail_data.csv”. In this dataset, there are 5500+ raw mail data in CSV format. I will discuss the basic properties of the dataset in the methodology section, and how I used it in my machine learning project with feature extraction, cleanups, removing redundant values, filling in missing values, etc.

IV. METHODOLOGY

Our primary objective with this product is to differentiate spam from ham emails that we receive daily. It determines which emails come to your inbox and which emails should go to the spam folder in real life more effective way. Here I will be using the Logistic Regression model to build this project. It is because the Logistic Regression model is the best model we can use when it comes to the binary classification problem. Further, we already discussed related works. My plan is to customize the code we built so that it can be used to its maximum potential.

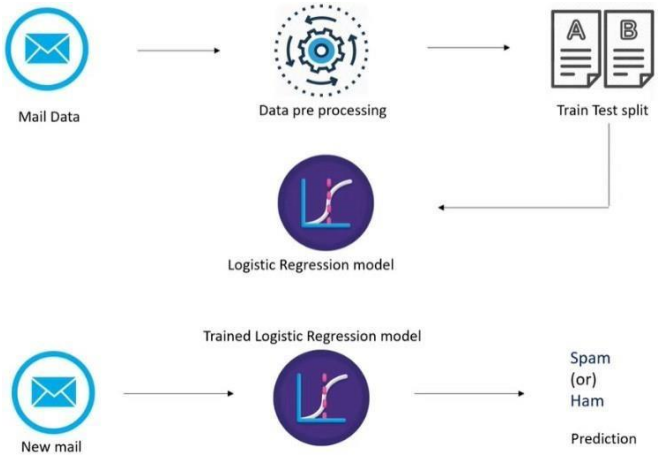


Figure 1: Logical Flow

We start with raw spam and ham email data, according to the above flow. After the data has been collected, we will train our machine learning model using the data. Nevertheless, they
aren’t directly applicable to our project. To accomplish this, our data needs to be preprocessed. Our text data will be converted into numbers during data preprocessing, since we know that machines can only understand numbers. Following that, I will split our data into training and test data, which will be used in training and evaluating our model. Once I have done that, I will feed the data into our Logistic Regression model. A trained model will eventually predict whether a mail is spam or ham by analyzing its contents.

A. Integrating Machine Learning into the project

We already have the Kaggle dataset, so let's explore how I'll use it in my Machine Learning project. I will start by importing dependencies (the libraries and functions) we will be using.

```
#Importing the Dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Figure 2: Importing dependencies

Model selection, features extraction, splitting the data and matrix import are all done through the sklearn learn library. As our algorithm always expects the input to be an integer or a

float, we must insert a feature extraction layer in the middle to convert the words to integers or floats.

There are a couple of methods of doing this: TfidfVectorizer, CountVectorizer, and Word Embedding. Counting words is good, but can we do better? The problem with simple word counts is that some words, such as “the” and “and”, appear repeatedly without adding any meaningful information. The word embedding technique attempts to convert a word into a vector-based format, and this vector describes where this word resides within a higher dimensional space. When two words have similar meanings, their cosine distances will be shorter, and they will be closer to one another. However, our purpose will not be achieved by doing so. When that happens, TfidfVectorizer comes into play. In addition to counting each word, the vectorizer will try to downscale words that appear across multiple documents or sentences.

```
#Data Collection & Pre-Processing
#Loading the data from csv file to a pandas Dataframe
raw_mail_data = pd.read_csv('mail_data.csv')
print(raw_mail_data)
```

```
     Category                                                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...
5        spam  FreeMsg Hey there darling it's been 3 week's n...
6         ham  Even my brother is not like to speak with me. ...
7         ham  As per your request 'Melle Melle (Oru Minnamin...
8        spam  WINNER!! As a valued network customer you have...
9        spam  Had your mobile 11 months or more? U R entitle...
10        ham  I'm gonna be home soon and i don't want to tal...
...       ...                                                ...
5560      ham                  Anything lor. Juz both of us lor.
5561      ham  Get me out of this dump heap. My mom decided t...
5562      ham  Ok lor... Sony ericsson salesman... I ask shuh...
5563      ham                                Ard 6 like dat lor.
5564      ham  Why don't you wait 'til at least wednesday to ...
5565      ham                                    Huh y lei...
5566     spam  REMINDER FROM O2: To get 2.50 pounds free call...
5567     spam  This is the 2nd time we have tried 2 contact u...
5568      ham              Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So...any other s...
5570      ham  The guy did some bitching but I acted like i'd...
5571      ham                         Rofl. Its true to its name

[5572 rows x 2 columns]
```

Figure 3: Data collection and Pre-Processing

The next step will be loading the dataset into a pandas data frame. The raw data is shown in the above figure. Since the dataset contains null values & missing values, this will pose a problem. This issue will be resolved by converting them into null strings in the next step.

```
#Replace the null values with a null string
mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)),'')
#Printing the first 5 rows of the dataframe
mail_data.head()
```

	Category	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...

Figure 4: Replace the null values with null string

Let's find out how many columns and rows we have in our dataset.

```
#Checking the number of rows and columns in the dataframe
mail_data.shape
```

```
(5572, 2)
```

Figure 5: Checking the number of rows and columns

As you can see, this is a fairly large dataset. We have a significant number of data here which is 5572 emails. We now see that the category column represents two labels. Therefore, in our next step, we need to encode those labels. In this case, I will label spam as 0 and ham as 1.

```
#Label Encoding
#Label spam mail as 0;  ham mail as 1;
mail_data.loc[mail_data['Category'] == 'spam', 'Category',] = 0
mail_data.loc[mail_data['Category'] == 'ham', 'Category',] = 1
#spam  -  0
#ham  -  1
```

Figure 6: Label Encoding

Now, I provide this message data and the labels separately to the machine learning model. It's like giving X-axis and Y- axis values. So, the features or message data will be the input, and the output or the target column will be the category. For this purpose, let's make two variables.

```
#Separating the data as texts and label
#X-input
#Y-Output/target
X = mail_data['Message']
Y = mail_data['Category']
```

```
print(X)
```

```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
5       FreeMsg Hey there darling it's been 3 week's n...
                              ...
5566    REMINDER FROM O2: To get 2.50 pounds free call...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will ü b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                         Rofl. Its true to its name
Name: Message, Length: 5572, dtype: object
```

```
print(Y)
```

```
0       1
1       1
2       0
3       1
4       1
5       0
       ..
5565    1
5566    0
5567    0
5568    1
5569    1
5570    1
5571    1
Name: Category, Length: 5572, dtype: object
```

Figure 7: Separating the data as text and label

The next part is the most important since we use one set of data to test our model and another set to evaluate it. In other words, part of the X will be our training data, and the other part will be our test data. The same applies to Y. In this instance, we will take advantage of the train split

function we imported above. A total of 80% of the 5572 emails will be used for the training data; the remaining 20% will be used for the test data. With the random state, I can be sure that our train_test_split will return the same split every time, which will give consistency to our model.

```
#Splitting the data into training data & test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)
print(X.shape)
print(X_train.shape)
print(X_test.shape)
```

```
(5572,)
(4457,)
(1115,)
```

Figure 8: Splitting the data into training data and test data

The next step will be feature extraction which we convert text values to feature vectors (which has meaningful numerical values) where Logistic Regression model can understand. max_df is used to remove terms that appear too frequently. A parameter stop_words = "english" will ignore words in English that add little meaning to a sentence. In the next step, fit_transform will convert it to feature vectors. Since we still have object data type in the data frame, it needs to convert to integer eventually.

```
#Feature Extraction
#Transform the text data to feature vectors that can be used as input to the Logistic regression

feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english', lowercase='True')

#Splited X has string values, those need to be fit & converted to integer

X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)

#Convert Y_train and Y_test values as integers [convert object type to int]

Y_train = Y_train.astype('int')
Y_test = Y_test.astype('int')
```

Figure 9: Feature extraction and transform text data into feature vectors

```
#not transformed data - original data
print(X_train)
```

```
4423              MMM ... Fuck .... Merry Christmas to me
4235    Now only i reached home. . . I am very tired n...
2577                  In sch but neva mind u eat 1st lor..
1361    Yo dude guess who just got arrested the other day
840     Last chance 2 claim ur £150 worth of discount ...
4977    You are gorgeous! keep those pix cumming :) th...
4068    You are being contacted by our Dating Service ...
4814            i can call in  &lt;#&gt;  min if thats ok
789     5 Free Top Polyphonic Tones call 087018728737,...
Name: Message, Length: 4457, dtype: object
```

```
#tansformed data - X train (same data) in numerical values
print(X_train_features)
```

```
  (1, 2746)     0.3398297002864083
  (1, 2957)     0.3398297002864083
  (1, 3325)     0.31610586766078863
  (1, 3185)     0.29694482957694585
  (1, 4080)     0.18880584110891163
  (2, 6601)     0.6056811524587518
  (2, 2404)     0.45287711070606745
  :     :
  (4456, 7150)  0.3677554681447669
  (4456, 7154)  0.24083218452280053
  (4456, 6028)  0.21034888000987115
  (4456, 5569)  0.4619395404299172
  (4456, 6311)  0.30133182431707617
  (4456, 647)   0.30133182431707617
  (4456, 141)   0.292943737785358
```

Figure 10: Displaying the transformed data

Now I am going to train my model. It will require importing the Logistic Regression model. Next, I will feed the model the training data (X-axis and Y-axis values).

```
#Training the Model
#Logistic Regression
model = LogisticRegression()
#Training the Logistic Regression model with the training data
model.fit(X_train_features, Y_train)
```

```
/home/shehan/.local/lib/python2.7/site-packages/sklearn/linear_model/logistic.py
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

Figure 11: Training the model

B. Evaluating the model

A training model must be evaluated before we proceed to build a predictive system. An array called prediction_on_training_data stores the values predicted by the trained model. We then compare the predicted values. Here I will utilize the accuracy_score function. We need to provide two parameters. In one, we have the "true" value, which is Y_train, and in the other, we have the "prediction_on_training data".

```
#Evaluating the trained model
#Prediction on training data

prediction_on_training_data = model.predict(X_train_features)
accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)
```

```
print('Accuracy on training data : ', accuracy_on_training_data)
```

```
('Accuracy on training data : ', 0.9670181736594121)
```

Figure 12: Evaluating the training data and prediction

Our model has been tested using training data, so let's try it with test data as well. Sometimes a model can overfit. Therefore, I am testing my model with test data as well as training data.

```
#Prediction on test data

prediction_on_test_data = model.predict(X_test_features)
accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)
```

```
print('Accuracy on test data : ', accuracy_on_test_data)
```

```
('Accuracy on test data : ', 0.9659192825112107)
```

Figure 13: Evaluating the test data and prediction

Now that I am confident in my model, let's build a predictive system. This can be achieved by submitting a random sample of emails to the model, which can then predict whether it is spam or not. Here are some examples based on some emails I selected from my dataset. The value of the label is predicted using the predict function.

```
#Building a Predictive System

input_mail = ["Valentines Day Special! Win over £1000 in our quiz...."]

#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('Ham mail')

else:
  print('Spam mail')
```

```
[0]
Spam mail
```

Figure 14: Building a Predictive system

C. Local Deployment on Ubuntu LTS 20.04.1 x64

app.py

```python
from flask import Flask, render_template, url_for, request
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

app=Flask(__name__)

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/predict', methods=['POST'])
def predict():

    #Loading the data from csv file to a pandas Dataframe
    raw_mail_data = pd.read_csv('mail_data.csv')

    #Replace the null values with a null string
    mail_data = raw_mail_data.where((pd.notnull(raw_mail_data)),'')

    #Label spam mail as 0;  ham mail as 1;
    mail_data.loc[mail_data['Category'] == 'spam', 'Category',] = 0
    mail_data.loc[mail_data['Category'] == 'ham', 'Category',] = 1

    #Separating the data as texts and label
    #X-input
    #Y-Output/target
    X = mail_data['Message']
    Y = mail_data['Category']

    #Splitting the data into training data & test data
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=3)

    #Transform the text data to feature vectors that can be used as input to the Logistic regression
    feature_extraction = TfidfVectorizer(min_df = 1, stop_words='english', lowercase='True')

    #Splited X has string values, those need to be fit & converted to integer
    X_train_features = feature_extraction.fit_transform(X_train)
    X_test_features = feature_extraction.transform(X_test)

    #Convert Y_train and Y_test values as integers [convert object type to int]
    Y_train = Y_train.astype('int')
    Y_test = Y_test.astype('int')

    #Training the Model
    model = LogisticRegression()

    #Training the Logistic Regression model with the training data
    model.fit(X_train_features, Y_train)

    #Prediction on training data
    prediction_on_training_data = model.predict(X_train_features)
    accuracy_on_training_data = accuracy_score(Y_train, prediction_on_training_data)

    #Prediction on test data
    prediction_on_test_data = model.predict(X_test_features)
    accuracy_on_test_data = accuracy_score(Y_test, prediction_on_test_data)

    if request.method=='POST':
        comment=request.form['comment']
        data=[comment]

        #Convert text to feature vectors
        input_data_features = feature_extraction.transform(data).toarray()

        #Making prediction
        my_prediction = model.predict(input_data_features)

    return render_template('result.html', prediction=my_prediction)

if __name__ == '__main__':
    app.run(debug=True)
```

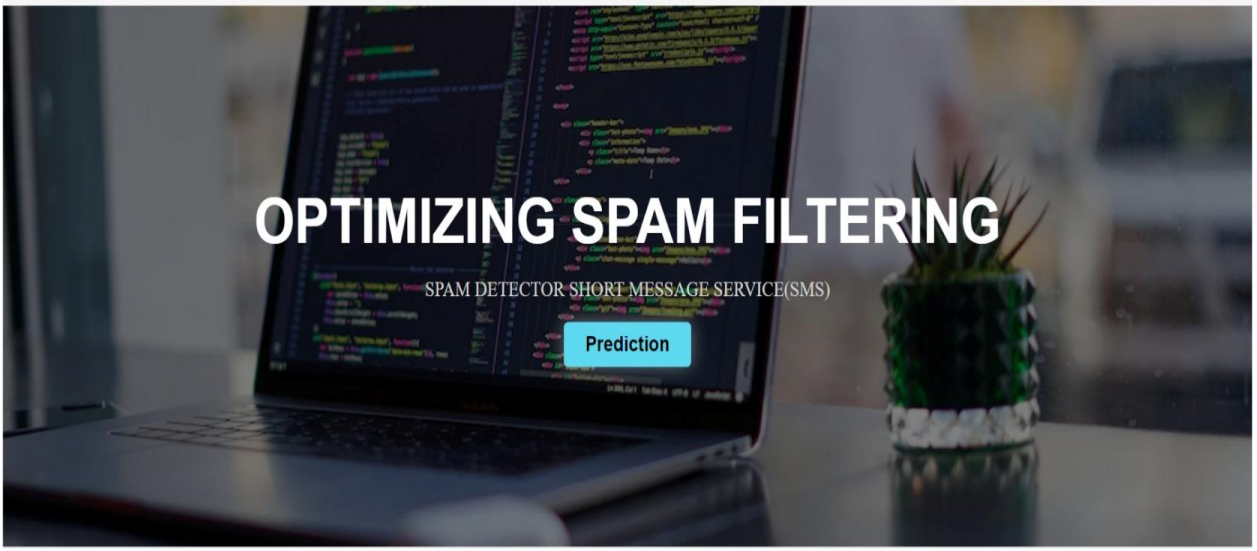
Figure 15: The Python script developed by me

To deploy my model as a web application on Ubuntu, I developed a Python script (app.py). Upon running the script, Flask (a Python-based web app framework) is imported to render the model into a web application.

...

SPAM DEDECTION

Home About Contact



ABOUT

We have a web developer and love to create websites. We very good developer and I am always looking for new projects. I am a very good developer and I am always looking for new projects.

MORE INFO

Home

About

Contact

SOCIAL LINKS

The system will produce the following results. Let's test the system by sending a spam email.

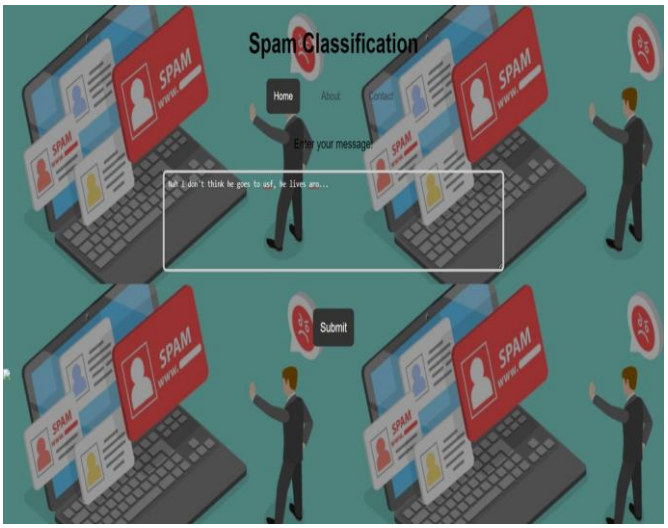


Figure 18: Checking the content of the mail to see if it's ham

The application works as expected. Let's see if it works when a ham mail is used.

Researchers have become increasingly interested in spam detection and filtering over the last two decades. Several studies have been conducted in this area because of its substantial impact on a variety of areas, such as consumer behavior or fake reviews. In the study, lessons learned from each machine learning category are compared with previous approaches. Additionally, spam filters find it challenging to evaluate features from multiple angles, including temporal, writing style, semantic and statistical ones. Models are trained primarily on balanced datasets, while self-learning models are not feasible. Deep fake is another challenge facing spam detection systems. According to the findings of this study, most proposed spam email detection techniques are based on supervised machine learning techniques. This project provides an in-depth analysis of these Logistic Regression algorithm and some future directions for searching and detecting spam email.

ACKNOWLEDGEMENT

I thank Dr Lakmal Rupasinghe, the lecturer in charge of the Machine Learning for Cyber Security - IE4092, Ms Chethana Liyanapathirana, the senior lecturer, and Ms Laneesha Ruggahakotuwa, the assistant lecturer and all associated lecturers and instructors of Sri Lanka Institute of Information Technology, for granting me an opportunity to conduct this Machine Learning Project Report with guidance. This work was supported in part by the Research Groups

CONCLUSIN

Faculty of Computing, Department of Computer Systems Engineering under Grant Machine Learning for Cyber Security - IE4092.

REFERENCES

[1] H. Faris, A. M. Al-Zoubi, A. A. Heidari et al., "An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks," *Information Fusion*, vol. 48, pp. 67–83, 2019

[2] S. O. Olatunji, "Extreme Learning machines and Support Vector Machines models for email spam detection," in *Proceedings of the 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, IEEE, Windsor, Canada, April 2017.

[3] A. Alghoul, S. Al Ajrami, G. Al Jarousha, G. Harb, and S. S. Abu-Naser, "Email classification using artificial neural network," *International Journal for Academic Development*, vol. 2, 2018.

[4] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, Article ID 102419, 2020.

[5] H. Bhuiyan, A. Ashiquzzaman, T. Islam Juthi, S. Biswas, and J. Ara, "A survey of existing e-mail spam filtering methods considering machine learning techniques," *Global Journal of Computer Science and Technology*, vol. 18, 2018.

[6] T. Vyas, P. Prajapati, and S. Gadhwal, "A survey and evaluation of supervised machine learning techniques for spam e-mail filtering," in *Proceedings of the 2015 IEEE international conference on electrical, computer and communication technologies (ICECCT)*, IEEE, Tamil Nadu, India, March 2015.

[7] A. K. Jain and B. B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 9, 2016.

[8] A. Subasi, S. Alzahrani, A. Aljuhani, and M. Aljedani, "Comparison of decision tree algorithms for spam E-mail filtering," in *Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, IEEE, Riyadh, Saudi Arabia, April 2018.

[9] H. Faris, I. Aljarah, and B. Al-Shboul, "A hybrid approach based on particle swarm optimization and random forests for e-mail spam filtering," in *Proceedings of the International Conference on Computational Collective Intelligence*, Springer, Halkidiki, Greece, September 2016.

[10] N. Sutta, Z. Liu, and X. Zhang, "A study of machine learning algorithms on email spam classification," in *Proceedings of the 35th International Conference, ISC High Performance 2020*, vol. 69, pp. 170–179, Frankfurt, Germany, 2020.

[11] Z. Guo, Y. Shen, A. K. Bashir et al., "Robust spammer detection using collaborative neural network in Internet of thing applications," *IEEE Internet of Things Journal*, vol. 8, 2020.

[12] Y. Dou, G. Ma, P. S. Yu, and S. Xie, "Robust spammer detection by nash reinforcement learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, Virtual Event CA, USA, July 2020.

[13] M. H. Arif, J. Li, M. Iqbal, and K. Liu, "Sentiment analysis and spam detection in short informal text using learning classifier systems," *Soft Computing*, vol. 22, no. 21, pp. 7281–7291, 2018.

[14] N. Kumar and S. Sonowal, "Email spam detection using machine learning algorithms," in *Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pp. 108–113, Coimbatore, India, 2020.

[15] A. J. Saleh, A. Karim, B. Shanmugam et al., "An intelligent spam detection model based on artificial immune system," *Information*, vol. 10, no. 6, p. 209, 2019.

[16] W. Peng, L. Huang, J. Jia, and E. Ingram, "Enhancing the naive bayes spam filter through intelligent text modification detection," in

Proceedings of the 2018 17th IEEE International Conference on Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference on Big Data Science And Engineering (TrustCom/BigDataSE), IEEE, New York, NY, USA, August 2018.