

# **Amrit Science Campus**

(Tribhuvan University)



A Project Report

On

## **“Email Spam Classification”**

Under the supervision of

Balkrishna Subedi

### **Submitted By:**

Aayush Paudel (10074/073)

Bishal Chapagain (10086/073)

Giriraj Khanal (10091/073)

Satish Kandel (10122/073)

### **Submitted To:**

Department of Computer Science and Information Technology

Amrit Science Campus

Lainchaur, Kathmandu, Nepal

April 2021

# EMAIL SPAM CLASSIFICATION



## Submitted By:

Aayush Paudel (10074/073)

Bishal Chapagain (10086/073)

Giriraj Khanal (10091/073)

Satish Kandel (10122/073)

A project submitted in partial fulfillment of the requirements for the  
Degree of Bachelor of Science (B.Sc.) in  
Computer Science and Information Technology awarded by  
IOST, Tribhuvan University

AMRIT SCIENCE CAMPUS

Lainchaur, Kathamandu

Nepal

April 2021

## ACKNOWLEDGEMENTS

Firstly, we would like to express our sincere gratitude to our supervisor, **Mr. Balkrishna Subedi** Sir for the continuous support throughout this project. His guidance and motivation helped us throughout the study and research process.

A very special thanks goes to our Head of Department **Mr. Hikmat Rokaya** Sir for giving us this opportunity to undertake this project and believing in us.

Last but not the least, we would like to express our sincere thanks to all our friends who helped us either directly or indirectly during this project. The quality time spent with our teacher and friends will remain forever in our heart.

Group Members:

Aayush Paudel (10074/073)

Bishal Chapagain (10086/073)

Giriraj Khanal (10091/073)

Satish Kandel (10122/073)

## **ABSTRACT**

Spam email is one of the biggest problems in today's world of the Internet. Spam emails not only affect the organizations financially but is also a major problem to individual email users. The email labeled as spam is nothing but advertisement of any company/product or any kind of virus being received by the email client mailbox without prior expectation of the user. Another aspect of the problem is that due to the large frequency of incoming emails, it is very difficult to separate important emails from spams. To support ease of access, emails are needed to be categorized based on the type of information they contain which will let the person know the content before even opening the email. To solve this problem different spam email filtering techniques and algorithms are used to protect our mailbox from spam emails.

In this project, we are using Naive Bayes Classifier for spam email classification enhanced with Laplace Smoothing for numerical stability. The Naive Bayesian Classifier is a very simple and efficient method for spam classification. Here we are using a dataset from Apache Spam Assain's site for classification of spam and non-spam email. The frequency of words arranged in a full-matrix table is used as the feature for the model. The result is to increase the accuracy of the system.

# TABLE OF CONTENTS

## Contents

ACKNOWLEDGEMENTS.....	i
ABSTRACT.....	ii
List of Figures .....	v
List of Tables .....	vi
LIST OF ABBREVIATIONS .....	vii
CHAPTER 1 .....	1
INTRODUCTION.....	1
1.1 Background.....	1
1.2 Problem Statement.....	1
1.3 Objective .....	2
1.4 Scope of the Project .....	2
1.5 Applications .....	2
CHAPTER 2 .....	3
LITERATURE REVIEW .....	3
2.1 Document Preprocessing .....	3
2.1.1 Removal of HTML tags .....	3
2.1.2 Tokenization.....	4
2.1.3 Removal of Stop Words .....	4
2.1.4 Stemming words .....	5
2.1.5 Sparse and Full Matrix Representation.....	6
CHAPTER 3 .....	8
REQUIREMENT ANALYSIS AND FEASIBILITY STUDY .....	8
3.1 Requirement Collection .....	8
3.2 Data Flow Diagram.....	8
3.3 Feasibility Study .....	10
3.3.1 Technical Feasibility .....	10

3.3.2 Operational Feasibility .....	10
3.3.3 Economical Feasibility .....	10
3.4 Gantt Chart.....	10
CHAPTER 4 .....	12
SYSTEM DESIGN .....	12
4.1 Flowchart .....	12
4.2 Spam Filter Algorithm .....	12
4.3 Pre-processing.....	13
4.4 Feature Selection.....	13
4.5 Naïve Bayes Classifier.....	13
CHAPTER 5 .....	16
IMPLEMENTATION .....	16
5.1 Feature Extraction.....	16
5.2 Testing .....	17
5.3 Dataset .....	18
CHAPTER 6 .....	21
EXPERIMENTAL RESULT .....	21

## List of Figures

Figure	Title	Page
2.1.5.1	Sparse matrix representation.....	6
2.1.5.1	Full matrix representation .....	7
3.2.1	Level-0 DFD .....	8
3.2.2	Level-1 DFD .....	9
3.2.3	Level-2 DFD .....	9
3.4.1	Gantt chart showing activities and time duration of project .....	11
4.1.1	Flow Chart Showing the Proposed System.....	12
6.1	Home Page Showing User-Interface to write email messages .....	21
6.2	Prediction Page Displaying the Result that the email is not spam...	22

## List of Tables

Table	Title	Page
6.2	Confusion Matrix .....	22



## **LIST OF ABBREVIATIONS**

<b>NLTK</b>	Natural Language ToolKit
<b>PIL</b>	Pillow
<b>HTML</b>	Hyper Text Markup Language
<b>CSS</b>	Cascading Style Sheet
<b>HTTP</b>	Hyper Text Transfer Protocol
<b>URL</b>	Uniform Resource Locator
<b>IT</b>	Information Technology

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Background**

Due to wide use of the internet for communication, it became a popular medium for advertising and marketing. Although sending emails through the network is quick and cost effective, it gave rise to another problem in today's internet world i.e., sending bulk or unsolicited emails to numerous users [1]. Email spam comes under electronic spam which sends bulk of unnecessary or junk mail of duplicate emails to recipients without any request by hiding the identity of the sender.

Email spam follows three properties i.e., Anonymity, mass mailing and unsolicited emails.

Anonymity is the property of hiding the uniqueness and whereabouts of the email sender. Mass mailing is defined as the sending of bulk identical emails to a large number of groups and unsolicited emails are the emails transferring to the recipients who do not request [2].

### **1.2 Problem Statement**

It is estimated that 70 percent of all emails sent globally is spam, and the volume of spam continues to grow because spam remains a lucrative business[2]. It is important to stop as much spam as we can, to protect the network from many possible risks: viruses, phishing attacks, compromised web links and other malicious content. Spam filters also protect our server from being overloaded with non-essential emails, and the worse problem of being infected with spam software that may turn them into spam servers themselves. We have chosen this project of making a spam classifier which will act as the key first line of defense.

### **1.3 Objective**

The goal of this project is to construct an email spam filter using machine learning technique - Naive Bayesian Classifier.

This helps to classify whether the email is spam or ham (not-spam).

### **1.4 Scope of the Project**

This project considers the email body and uses a bag of words approach to classify the email. It looks at each word in isolation and keeps track of frequency of each word, which is used as the feature for the Naive Bayes algorithm during the training.

Although the classifier does a good job classifying the email, the context is lost while constructing the features. Also, since the header of the email is stripped out, it does not consider other possibilities for the email to be spam like: routing information, including IP address.

### **1.5 Applications**

- It is used to make real-time predictions.
- Email services like Gmail use this algorithm to figure out whether an email is spam or not. This algorithm is excellent for spam filtering.
- Collaborative filtering and the Naive Bayes algorithm work together to build recommendation systems.
- This algorithm is popular for multiclass predictions.

## **CHAPTER 2**

### **LITERATURE REVIEW**

There has been much research and discussions on the topic of email spam classification. Many research papers have been published and numerous algorithms have been developed to resolve the issue of spam email adulterating the inbox of users. Algorithms like K-means clustering, Logistic Regression, Decision Trees etc. have been used to classify the received emails.

In the paper[3], authors have highlighted several features contained in the email header which will be used to identify and classify spam messages efficiently. Those features are selected based on their performance in detecting spam messages. Although header information provides a lot of useful insights in detecting the spam emails, they do not take into consideration the other part which is the email body.

Spam filtering when implemented in the recipient mailbox can significantly improve the user performance in terms of time and effort as it eliminates the need to manually mark the email as being spam. Using a simple probabilistic approach such as Naive Bayes rule, we can automatically infer the class of email as being spam and ham with high accuracy. Our project is based on this very approach and aims to classify spam emails from ham emails.

## **2.1 Document Preprocessing**

### **2.1.1 Removal of HTML tags**

Since the email body contains a lot of HTML tags which does not provide additional information on the context of the email, we use the BeautifulSoup package in python to remove all the HTML tags and get a clean email body without any HTML tags.

### **2.1.2 Tokenization**

Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or sub-words. Hence, tokenization can be broadly classified into 3 types: word, character, and subword (n-gram characters) tokenization.

For example, consider the sentence: “Computer Science Domain”.

The most common way of forming tokens is based on space. Assuming space as a delimiter, the tokenization of the sentence results in 3 tokens: Computer-Science-Domain. As each token is a word, it becomes an example of Word tokenization.

As tokens are the building blocks of Natural Language, the most common way of processing the raw text happens at the token level. Tokenization is the foremost step while modeling text data. Tokenization is performed on the corpus to obtain tokens. The following tokens are then used to prepare a vocabulary. Vocabulary refers to the set of unique tokens in the corpus. Vocabulary can be constructed by considering each unique token in the corpus or by considering the top K Frequently Occurring Words.

The major drawback of using word tokenization is when dealing with Out Of Vocabulary (OOV) words.

Here we are using word tokenization.

### **2.1.3 Removal of Stop Words**

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query [4].

We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

To determine if the email is spam or not, stop word does not contribute much. Thus, it is desirable to remove them from the email body while preprocessing.

We have used the NLTK package to remove stop words present in the English Language.

Furthermore, we have also removed punctuations from the email body, as it is of least importance to us.

#### **2.1.4 Stemming words**

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language [5].

Stem (root) is the part of the word to which you add inflectional (changing/deriving) affixes such as (-ed, -ize, -s, -de,mis). So, stemming a word or sentence may result in words that are not actual words. Stems are created by removing the suffixes or prefixes used with a word.

Here we have used the Python's nltk package to stem the word. Specifically, we have used PorterStemmer class which uses suffix stripping to produce stems. PorterStemmer is known for its simplicity and speed.

Example:

Connections, Connecting, Connection, Connected are all mapped to the stem word Connect.

Application of Stemming:

- Stemming is used in information retrieval systems like search engines.
- It is used to determine domain vocabularies in domain analysis.\
- Stemming is desirable as it may reduce redundancy as most of the time the word stem and their inflected/derived words mean the same.

### 2.1.5 Sparse and Full Matrix Representation

Initially for collection of stemmed words in each document we count the occurrences of each unique word and create a sparse matrix which is a Pandas Dataframe having the column Document\_ID, word\_ID, Label and Occurrences.

```
In [112]: sparse_train_df[:5]
```

```
Out[112]:
```

	LABEL	DOC_ID	OCCURENCES	WORD_ID
0	0	4844	1	394
1	0	4844	1	492
2	0	4844	1	2353
3	0	4844	1	496
4	0	4844	1	37

*Figure 2.1.5.1: Sparse matrix representation*

After this, we create a Full Matrix, which is also a Pandas Dataframe consisting of Document\_ID, Labels, and most frequent 2500 word\_id's as the columns. Each cell value below the word\_id consists of the occurrences for the corresponding words in the given Document\_Id.

Out[17]:

	DOC_ID	LABEL	0	1	2	3	4	5	6	7	...	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

*Figure 2.1.5.2: Full matrix representation*

We do this for both Training (70% of the total emails) and Test Data (30% of the remaining emails)



## CHAPTER 3

# REQUIREMENT ANALYSIS AND FEASIBILITY STUDY

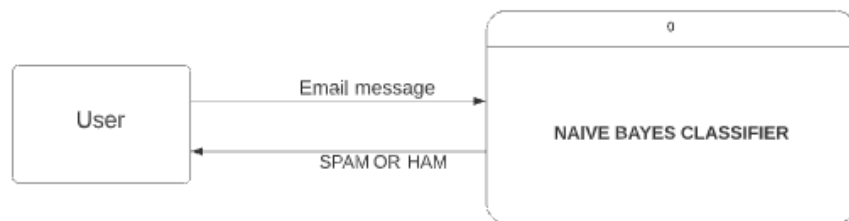
### 3.1 Requirement Collection

As a part of data collection, we relied primarily on a secondary data source: a website called spamassassin.apache.org. This is an open-source anti-spam platform.

The corpus obtained from this site is used for both training and testing phase of model formation.

### 3.2 Data Flow Diagram

DFD is one of the popular system analysis and design tools. It is a graphical representation of flow of data in a system. Process, Data Flow, Data Store and External entity are major components used in DFD to represent a system.



*Figure 3.2.1: Level-0 DFD*

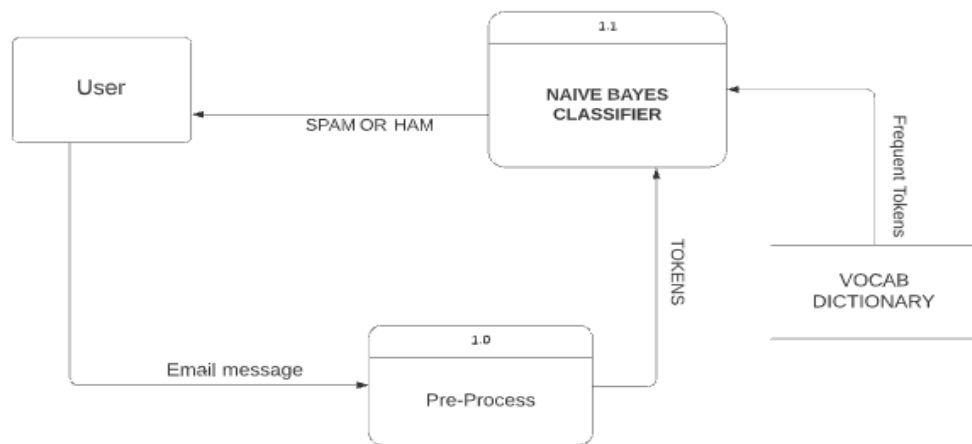


Figure 3.2.2: Level-1 DFD

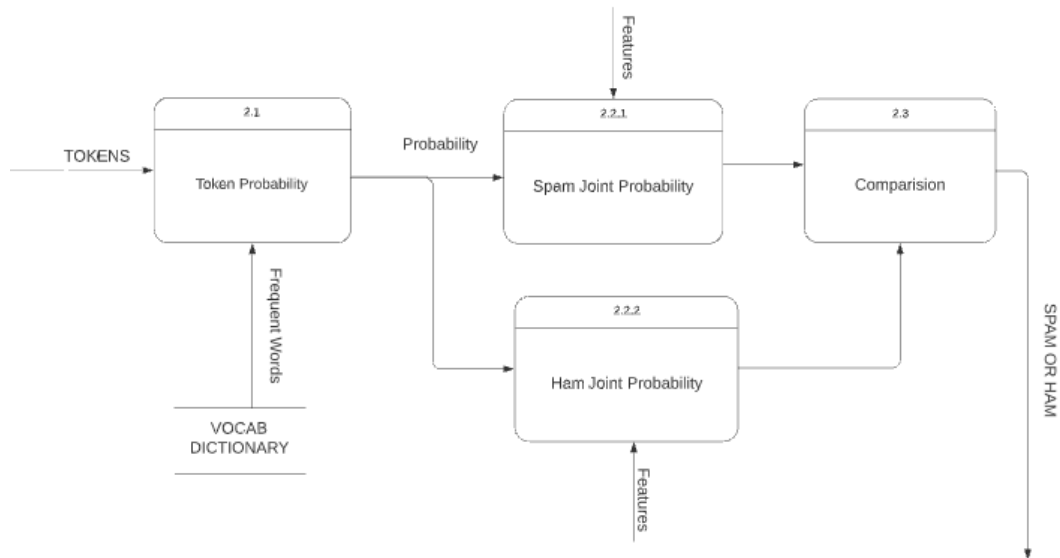


Figure 3.2.3: Level-2 DFD

### **3.3 Feasibility Study**

#### **3.3.1 Technical Feasibility**

The system we are trying to develop is technologically feasible. We are using Python as the programming language. We will be using Pycharm CE as the IDE, mainly for testing and deployment. For the majority of the task, we will be using Jupyter Notebook, until a final machine learning model is developed.

#### **3.3.2 Operational Feasibility**

The user-friendly interface provided by WTforms, HTML and CSS, make it easy even for the non-technical user to test the email using any web browser. Also, the response is quick.

This makes our system operationally feasible.

#### **3.3.3 Economical Feasibility**

The system we are trying to build is also feasible economically. All the tools required and data collected are available publicly and are open-sourced. Since this is a small project, we will be deploying our model using the Flask package locally in our machine. Flask is a lightweight web application microframework

### **3.4 Gantt Chart**

We had followed the following schedule during our project planning and implementation:

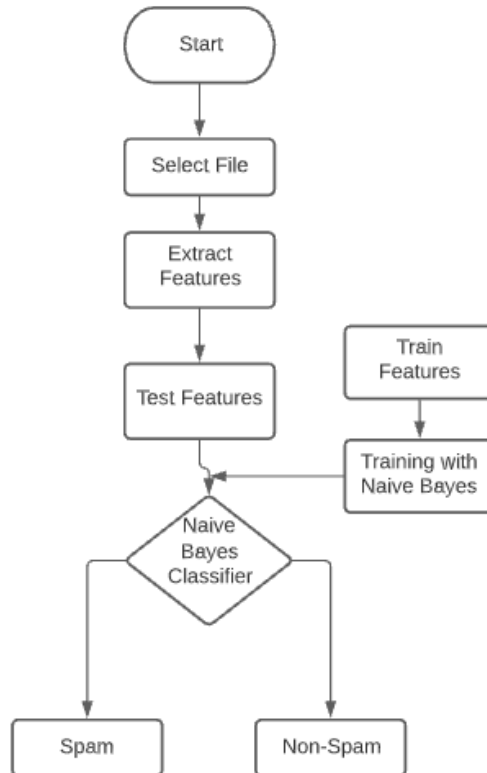
ACTIVITIES	WEEKS												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Study and Analysis													
Data Collection													
Data Exploration and Visulaization													
Implementation													
Testing													
Deployment													
Documentation													
Review													

*Figure 3.4.1: Gantt chart showing activities and time duration of project*

## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 Flowchart



*Figure 4.1.1: Flow Chart Showing the Proposed System*

#### 4.2 Spam Filter Algorithm

1. Load the Corpus and Generate Features
2. Split the Features into Train and Validation set
3. Use Training set to get Statistics for the Classifier
4. Use Validation Set for evaluating the overall performance

5. Put all the pieces together to build a standalone Classifier and test it on a fresh email data.

### **4.3 Pre-processing**

Data in real-world is never clean, complete or in the desirable format. Thus, the first step is to preprocess this data which involves extracting the email body from the entire email, removing blank emails and emails containing bad data.

Then for each clean email body, we convert it to lowercase, tokenize the sentence to obtain words, remove the stop words, stem the words and remove the punctuations to get the data in the desirable format for further analysis.

### **4.4 Feature Selection**

After preprocessing, we convert it to the full matrix representation, which is used as the feature for the Naive Bayes classifier to train on. This is basically a pandas dataframe consisting of word count for the most frequent words in the training data.

### **4.5 Naïve Bayes Classifier**

A Naive Bayes classifier is a probabilistic machine learning model that is used for classification tasks. The gist of the classifier is based on the Bayes theorem.

Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes uses the Bag of Word approach. That is, every word in the email is treated independently (each word is looked at in isolation.) This method does not give importance to the sequence; hence the context is lost. Thus, the name - Naive.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Using this algorithm and assuming that the words are independent, we calculate the joint probability of the words in an email or document.

Here is the Formula to calculate conditional probability for each word in the email to determine if it is more likely to be Spam or Ham.

$$P(Spam | X) = \frac{P(X | Spam) P(Spam)}{P(X)}$$

$$P(Ham | X) = \frac{P(X | Ham) (1 - P(Spam))}{P(X)}$$

We then make the decision based on the conditional probability and joint probability.

If the condition:

$$P(Spam|word) > P(Ham|word)$$

is true, we predict that the email is spam. Otherwise, it is Ham.

For example:

If the email has the body: “Hello all the Strangers!”

We first Split the body into tokens, remove the stop-words, remove punctuations and stem the words to its root word.

After Preprocessing:

[ Hello, Strange]

We then make the feature matrix out of these words by counting the occurrences of these words in spam and ham email context.

Finally, we calculate the joint probability of the email being spam as:

$P(\text{Spam}|\text{Hello}) * P(\text{Spam}|\text{Strange})$

And joint probability of the email being ham as:

$P(\text{Ham}|\text{Hello}) * P(\text{Ham}|\text{Strange})$

If the former is greater, this email is likely to be spam rather than ham.



# CHAPTER 5

## IMPLEMENTATION

### 5.1 Feature Extraction

For each email we want to classify as either spam or ham, it has to go through a series of modules after which we get the features that we can feed into the algorithm to make the prediction.

Full matrix is the final representation for the feature for our data. Below is the python code which takes into sparse matrix as the parameter along with the number of tokens in the given email body and index for document id, word id and frequency.

```
def make_full_matrix(sparse_matrix, nr_words, doc_idx=0, word_idx=1, freq_idx=3):
    """
    Create a full_matrix from the sparse matrix and return pandas dataframe.
    sparse_matrix --- numpy array
    nr_words --- size of vocabulary
    doc_idx --- position of document id in the sparse matrix. default is 0
    word_idx -- position of word id in the sparse matrix. default is 1
    cat_idx --- position of category id in the sparse matrix, default is 2
    freq_idx -- position of frequency id in the sparse matrix, default is 3
    """
    column_name = ['DOC_ID'] + list(range(nr_words))
    doc_id_names = [0]
    full_matrix = pd.DataFrame(index=doc_id_names, columns=column_name)
    full_matrix = full_matrix.fillna(value=0)
    for i in range(sparse_matrix.shape[0]): # looping row by row
        # doc_nr = sparse_matrix[i][doc_idx]
```

```

word_id = sparse_matrix[i][word_idx]
occurrence = sparse_matrix[i][freq_idx]
full_matrix.at[0, 'DOC_ID'] = 0
# full_matrix.at[0, 'CATEGORY'] = label
full_matrix.at[0, word_id] = occurrence
full_matrix.set_index('DOC_ID', inplace=True)
return full_matrix

```

Using this feature matrix, we can easily calculate the conditional probability for each token being spam and ham, which will form the basis for joint probability calculation.

## 5.2 Testing

Supervised Machine Learning algorithms work on three major phases. The first is called the Training phase where we teach the algorithm about the data using the features and the labels present in the training dataset. Once the algorithm learns about the data from the training phase, we use the rest of the data to evaluate the model performance. This is called Validation phase and the data being used is called the Validation dataset. The Validation dataset is different from the training dataset which is something the model has not seen during the training phase. This helps to prevent the problem of our model being overfit to the data so that we can make a sense of how our model performs on the previously unseen data. The labels present in the validation dataset will be used to calculate evaluation metrics of the model.

We should make sure the validation dataset should be representative of real-world data on which the model will be performing its actual evaluation on.

The final phase is called testing where we give this trained model the real-world data and let the model evaluate on its own. We train the model so as to ensure it gives

sensible results during the testing phase and helps to achieve the objective of actually classifying spam email from the ham email.

### 5.3 Dataset

A data set is simply a collection of data. In other words, a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question.

In the context of spam classification, we call these dataset corpus. Corpus are large and structured sets of text. To put it in simple words, they are a set of all documents.

There are two categories of dataset being used in this project. They are spam data and non-spam(ham) data.

Spam Dataset is a collection of all the spam emails which are labeled as 1 to indicate they are spam. They are undesirable to the recipient. Thus, we want to separate them from the legit email.

Spam Email Example:

'ATTENTION: This is a MUST for ALL Computer Users!!!\n\n\n\*NEW-Special  
Package Deal!\*\n\n\nNorton SystemWorks 2002 Software Suite -Professional  
Edition-\n\n\nIncludes Six - Yes 6! - Feature-Packed Utilities\n\n\nALL For 1 Special  
LOW Price!\n\n\nThis Software Will:\n\n- Protect your computer from unwanted  
and hazardous viruses\n\n- Help secure your private & valuable information\n\n- Allow  
you to transfer files and send e-mails safely\n\n- Backup your ALL your data quick and  
easily\n\n- Improve your PC\'s performance w/superior integral diagnostics!\n\n\n6  
Feature-Packed Utilities...1 Great Price!\n\n\nA \$300+ Combined Retail

Value!\n\n\nYOURS for Only \$29.99! <Includes FREE Shipping!>\n\n\nDon't  
 fall prey to destructive viruses or hackers!\n\nProtect your computer and your valuable  
 information!\n\n\n\nSo don't delay...get your copy  
 TODAY!\n\n\n\nhttp://euro.specialdiscounts4u.com/\n\n+++++\n\n  
 +++++\n\n  
 This email has been screened and filtered by our in house ""OPT-OUT"" system in  
 \n\ncompliance with state laws. If you wish to "OPT-OUT" from this mailing as well  
 \n\nas the lists of thousands of other email providers please  
 visit \n\n\nhttp://dvd.specialdiscounts4u.com/optoutd.html\n\n+++++\n\n  
 +++++\n\n  
 \n\n\n\n'

Ham Dataset is a collection of all the ham emails which are labeled as 0 to indicate they are non-spam. They are important to the recipient. Thus, we want to separate them from the spam emails and put it in the users inbox.

Ham Email Example:

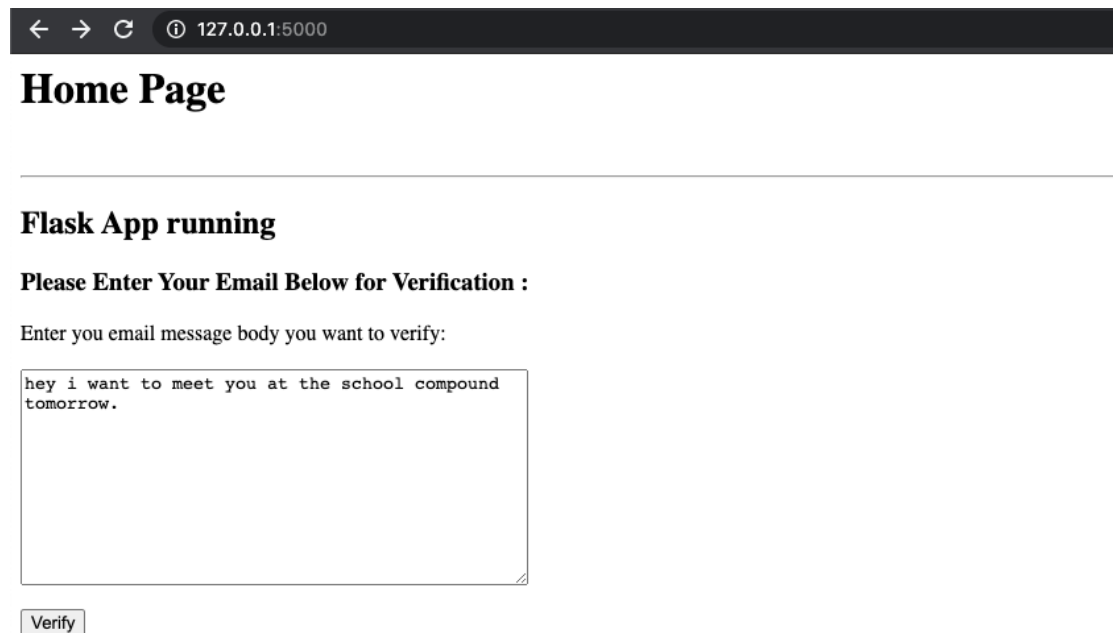
'Once upon a time, Manfred wrote :\n\n\n> I would like to install RPM itself. I have  
 tried to get the information\n\n> by visiting www.rpm.org <http://www.rpm.org> and  
 the related links they\n\n> give but they all seems to assume that RPM already is  
 installed.\n\n> I have a firewall based on linux-2.2.20 (Smoothwall) for private  
 use.\n\n> I would like to install the RPM package/program but there is no\n\n>  
 information how to do this from scratch.\n\n> Found this site and hopefully some have  
 the knowledge.\n\n> Best regards Manfred Grobosch\n\n\n\nWell, you can simply use  
 an rpm tarball (or extract one from a source rpm\n\non a machine that has rpm scripts  
 install "rpm2cpio <file.src.rpm> | cpio\n\n-dimv" and "./configure && make && make  
 install" as usual. You need db3 or\n\nndb4 development files at least, and once  
 everything installed you'll need\n\nto initialize your rpm database.\n\n\n\nIf you need  
 more help, I suggest you join the rpm-list@redhat.com by\n\nsubscribing at

<https://listman.redhat.com/>  
 Matthias Saou World Trade  
 Center Edificio Norte 4 Planta  
 System and Network  
 Engineer 08039 Barcelona, Spain  
 Electronic Group  
 Interactive Phone : +34 936 00 23  
 23  
 RPM-List  
 mailing list <RPM-  
 List@freshrpms.net>  
<http://lists.freshrpms.net/mailman/listinfo/rpm-list>

## CHAPTER 6

### EXPERIMENTAL RESULT

Naive Bayes Email Spam Classification is a Binary Classification problem. The result for the experiment is in the form of 1 and 0. 1 indicates that the given email is Spam and 0 indicates it is Ham



← → ↻ ⓘ 127.0.0.1:5000

## Home Page

---

### Flask App running

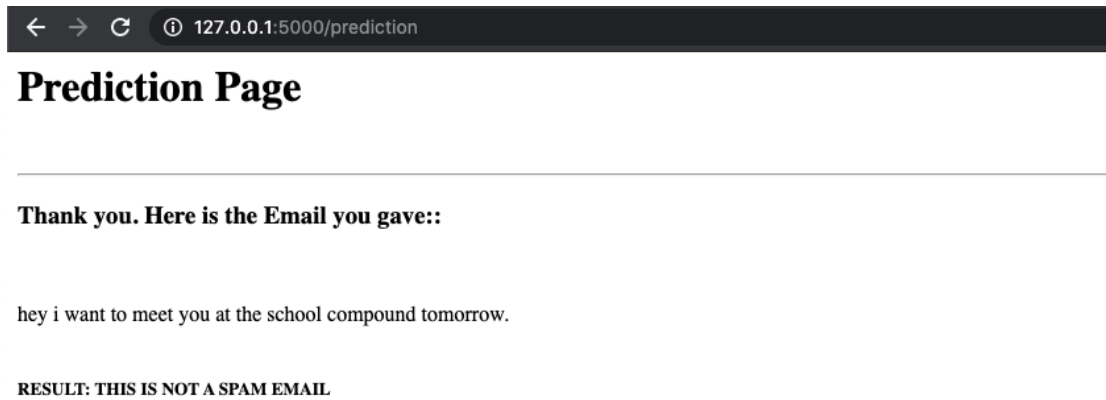
**Please Enter Your Email Below for Verification :**

Enter you email message body you want to verify:

```
hey i want to meet you at the school compound  
tomorrow.
```

Verify

*Figure 6.1: Home Page Showing User-Interface to write email messages*



*Figure 6.2: Prediction Page Displaying the Result that the email is not spam*

## 6.1 Result and Analysis

The following table summarizes the evaluation metrics used for the Naive Bayes algorithm in this project.

Actual vs Predicted	T	F
T	TP	FN
F	FP	TN

*Table 6.1: Confusion Matrix*

Accuracy is the measure of actual true and false values captured by the model.

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Recall is the measure of actual true values captured by the model.

$$\text{Recall} = \frac{TP}{(TP+FN)}$$

Precision is the measure of relevant true values predicted by the model.

Precision=  $TP/(TP+FP)$

F1-Score is the harmonic mean of recall and precision.

F1-score =  $2*Recall*Precision/(Recall+Precision)$

Evaluation on Validation set:

True Positive: 547

True Negative: 1126

False Positive: 9

False Negative: 41

Accuracy: 97.10%

Precision Score: 98.38%

Recall Score: 93.03%

F1-Score: 95.63%



## **CHAPTER 7**

### **CONCLUSION AND FUTURE WORK**

Taking into account the importance of electronic messages, in this paper we have proposed an approach for email classification which takes into account the content present in the email body as tokens and uses Naive Bayes to classify it as either being spam or ham. The main advantage of this approach is that it is fast, robust and gives pretty good classification accuracy. The algorithm was implemented using the Python programming language in PyCharm IDE. The implementation results were studied and the system was tested in the new instances of email. The results show that the approach used is reasonable and takes away the manual load of users to classify spam email from legit ones.

Some of the future research work can be done on, (1) utilize Email headers information to further strengthen the system. (2) Making the algorithm to self-learn based on misclassified data (3) Making the algorithm to classify email written in different languages.

## REFERENCES

- [1] Cheruvu, A. (2012) Email Spam Detector: A Tool to Monitor and Detect Spam Attack. [online] Available at: <http://sci.tamucc.edu/~cams/projects/410.pdf> [Accessed Sept. 2020]
- [2] Tope, M. (2019) Email Spam Detection using Naive Bayes Classifier. [online] Available at: <https://www.ijedr.org/papers/IJEDR1906001.pdf> [Accessed Sept. 2020]
- [3] Razak,S., Mohamad, A. (2013) Identification of Spam Email Based on Information from Email Header 13th International Conference on Intelligent Systems Design and Applications (ISDA), 2013.
- [4] What is Tokenization in NLP? [Blog] Available at: <https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/> [Accessed Sept. 2020]
- [5] Removing stop words with NLTK in python. [Blog] Available at: <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/> [Accessed Sept. 2020]
- [6] Stemming with python nltk package. [Blog] Available at: <https://www.datacamp.com/community/tutorials/stemming-lemmatization-python> [Accessed Sept. 2020]
- [7] Naive Bayes Classifier. [Blog] Available at: <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c> [Accessed Oct. 2020]

## APPENDICES

This is the python file that does the task of data preprocessing and part of feature extraction

```
from os import walk
from os.path import join

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

import nltk
from nltk.stem import PorterStemmer, SnowballStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

from bs4 import BeautifulSoup

def clean_message(msg, stemmer=PorterStemmer(),
stop_words=set(stopwords.words('english'))):
    filtered_words = []

    soup = BeautifulSoup(msg, 'html.parser')
    msg_no_html = soup.get_text()

    words = word_tokenize(msg_no_html.lower())

    for word in words:
        # remove stopwords and punctuation
        if word not in stop_words and word.isalpha():
            filtered_words.append(stemmer.stem(word))

    return filtered_words

def make_dataframe(word_list):
    word_columns_df = pd.DataFrame.from_records(word_list)
    return word_columns_df

def make_sparse_matrix(df, indexed_words, labels=0):
```

```

nr_rows = df.shape[0]
nr_cols = df.shape[1]
word_set = set(indexed_words)
dict_list = []

for i in range(nr_rows):
    for j in range(nr_cols):
        word = df.iat[i, j]
        if word in word_set:
            doc_id = df.index[i]
            word_id = indexed_words.get_loc(word)
            category = labels

            item = {'LABEL': category, 'DOC_ID': doc_id, 'OCCURENCES': 1,
'WORD_ID': word_id}
            dict_list.append(item)

sparse_df = pd.DataFrame(dict_list)
return sparse_df.groupby(['DOC_ID', 'WORD_ID', 'LABEL']).sum().reset_index()

```