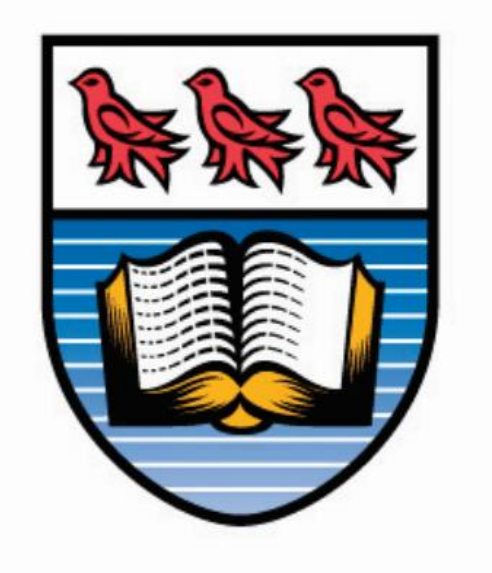
Spam Detection using N-gram Analysis and Machine Learning Techniques

by

SIMRAN KAUR



A Master's Project Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering.

Spam Detection using N-gram Analysis and Machine Learning Techniques

by

SIMRAN KAUR

Supervisory Committee

Dr. Issa Traore, Supervisor
Department of Electrical and Computer Engineering

Mihai Sima, Departmental Member
Department of Electrical and Computer Engineering

Abstract

There are many types of fraudulent activities happening today that open the loopholes of security, but email is a cheaper and widely known method for delivering false messages to potential victims. Spam is a form of email messages that is not only annoying for users but can provide a conduit for fraudulent or deceptive content delivery. In this project, a spam detector to identify an email as either spam or ham is built using n-gram analysis and supervised machine learning models. Three different algorithms are implemented and compared, namely naïve-Bayes, logistic regression and support vector machines (SVM). Experimental evaluation of the detector using a public dataset shows that the SVM and logistic regression attain the highest accuracy.

Table of Contents

List of Tables

List of Figures

List of Acronyms

NB Naïve-Bayes

LR Logistic Regression

SVM Support Vector Machine

BOW Bag of Words

TPR True Positive Rate

FPR False Positive Rate

ROC Receiver Operating Characteristic

Acknowledgments

First, I would like to thank my supervisor **Dr. Issa Traore**, who provided me valuable guidance, insightful advice and infinite knowledge throughout my program.

Besides, my supervisor, I would also like to thank Dr. Marcelo Brocardo, for initiating and helping me with the project, his continuous support and providing me proactive feedback on time.

Lastly, my deepest gratitude goes to my parents and my dear best friend Dr. Harleen Kaur, who constantly supported and helped me when I was in need. The final step to reach the milestone in my master's project would goes to her.

Chapter 1 – Introduction

1.1 Background

As the number of internet users is growing, there has been a dramatic increase in the use of electronic email by users as part of their daily life, and as more and people subscribe to different websites, products services, catalogs, newsletters, etc., users are more prone to receive annoying spam messages and fraudulent phishing messages.

Phishing is defined as a fraudulent attempt to obtain sensitive information such as usernames, passwords, bank account details by disguising oneself as a trustworthy entity in email communication [9]. Over the years, it has been discovered that people have gained an implicit knowledge, through which some users penetrate the walls of security and thus sending junk or spam messages which can direct them to provide sensitive information. It has been reported that the regular email messages sent around the world are about 50 billion [4]. Therefore, dealing and categorizing this huge number of emails is an important concern.

Before we discuss the main objective of this project, it is necessary to understand key aspects of spam messages. According to Information Technology [13], spam is defined as the unwanted email which generally contains content with an advertising or irrelevant content to multiple users, who never requested it. The main aim of spammers here is to attract the users to buy products and services of legitimate or prohibitive nature, with the end target of making money.

It is important to note that an email that contains a message with vendors website URL's is not associated with a spam message since it contains content that might interest some people, but others don't. This let us to conclude the common objectives of spamming email are:

- Promoting and selling products and services.
- Harvesting sensible information (from emails and passwords to bank accounts) via online lotteries, bank frauds and requests for help.
- Advertising concepts and ideologies.
- Sending viral spam such as infecting and turning the recipient computers into zombie PC's and frauds such as identity theft.

Therefore, sending too many spam messages usually overloads the mail-inbox which is the most disturbing factor faced by many users. Hence, to ensure that users only receive messages that are important to them (say an email from a friend) and to prevent malicious activities on the Internet many email service providers use different security mechanisms to filter incoming messages. Some server engines embed white and black distribution host lists and automatically accept email if it is a valid host, otherwise rejects it and sends it into the junk folder. Other server engines use security mechanisms built on machine learning algorithms that classify email messages as spam or ham.

As discussed in [13], the common spam detection techniques that server engines use to determine the likelihood for a message to be spam are:

1. Blacklist
   This is one of the popular spamming detection method that works by stopping unwanted emails from the current list of senders that system administrators create. The maintained list contains a record of email addresses and IP hosts that have been previously used to send spam or junk messages. However, this technique sometimes misidentifies legitimate senders as spammers. False positives can result if a spammer happens to be sending junk mail from an IP address that is also used by legitimate email users.

2. Whitelist
   This is the opposite of blacklist. Whitelist lets specify which senders to allow mail from, rather than specifying which senders to block mail from. However, this means a valid user who is not registered would automatically be blocked and will be marked as false positive.

3. Greylist
   In the greylist system, the receiving inbox will initially discard messages from unknown users and automatically sends failure alerts to the originating server. If the message happens to originate second time, greylist would then let it proceed to the receiver's mailbox and would add the name in the list of allowed senders. However, such filters cause delay in mail delivery, when you are expecting time sensitive messages.

4. Content-based Filters
   These filters evaluate the messages based on the phrases or words present in an email message. Though, such filters maintain a spam and ham words, maintaining the list is a tedious task.

5. Heuristic Filters
   As a rule of thumb, these filters process messages based on domain specific words. Instead of discarding messages that contain doubtful words, heuristic filters take multiple terms into concern found in a message. Skeptical words that are commonly found in spam messages such as "alert", "immediately" receive higher points, while terms commonly used in every other message receive lower points. The filter calculates the final score and evaluates it against a certain value. However, heuristic filters tend to falsely flag legitimate messages as malicious, if the message contains a certain pattern of words.

6. Bayesian Filters
   As the name suggests, these filters apply Bayes theorem to calculate the words probabilities that occur in the messages. These filters are considered the superior form of
   word based filters. To determine the probability of a class, these filters scan every word of a message and computes the probability using a list based probabilistic model.

As discussed, the trending spam detection filters offer security mechanisms that can be embedded into the mail server engines but provide severe drawbacks. The classifiers covered in this project cover some of the severe drawbacks that are discussed above. To accomplish the problem of spam filtering, we are using n-grams on the content of an email message, due to their simplicity and scalability. This happens through the use of larger *n* values; a model can store more context with a well understood space-time tradeoff, where each word is composed of contiguous sequence of *n* items from a given message of content [8].

1.2 Objective and Approach

In this project, our primary goal is to filter the messages in such a way that the users only read messages that are important to them based on the email body using n-grams feature extraction technique. The extracted features are analyzed using machine learning classification. We explore three different supervised machine learning algorithms, namely, naïve-Bayes, logistic regression and support vector machine (SVM).

In our proposed approach, the n-grams are generated from the training dataset and are preserved accordingly to predict the unlabeled data for the classifiers. In this project, a spam analyzer is built that initially learns from the Spam Corpus. This helps to increase the accuracy of the classifiers during the testing phase while maintaining the sensitivity of the algorithms.

Another aspect of the proposed approach consists of exploring all the n-grams generated, whose frequencies are typically very low. This methodology consists of using the number of "junk" words contained in the messages, as many spam messages have a lot of rare words (often common words being misspelled and flagged as rare). Also, to predict the class of an email the classifiers use the Bag of Words (BOW) approach.

1.3 Data Sources

The training data used in the project is the ADCG SPAM DATASET, connected with CSDMC2010 SPAM corpus. The dataset consists of a variety of email messages, which include a complete format of standard email MIME (Multipurpose Internet Mail Extensions) defined in RFC22 and some of them contain scripting tags such as HTML or XML.

The training dataset consists of 2,500 mails both in .eml file, including 1,721 ham messages labeled as 1 and 779 spam messages labeled as 0. The files are shuffled in a single folder and their associated labels are stored in different folders, to ensure that a specific .eml file maps to its correct label when training a classifier [1].

The testing dataset used is from a different source. This allows checking the accuracy of classifiers on data samples completely different from the training samples. The testing dataset consists of the SpamAssassin public email messages, fit for testing the trained classifiers [10]. The corpus is distributed into three parts – ham, spam, and hard spam/ham. For the testing, we used a subset of the emails consisting of typical ham and spam.

1.4 Report Outline

The remaining chapters in the report are structured as follows: Chapter 2 discusses the most popular algorithms used to classify an email as spam or ham. Background information on these classification models is provided in this chapter. Chapter 3 illustrates the data pre-processing technique used to extract an email body and for feature extraction. The implementation of the classifiers and the evaluation of their performances are presented in Chapter 4. Chapter 5 makes concluding remarks and discusses future work.

Chapter 2 – Supervised Models

In this project, we are using a supervised learning approach in which a model learns from the training data, and then make predictions on the new samples, based on generalizing trend. In simple words, a supervised model classifies new samples based on mapped input-output pairs in the training stage. In this, every input sample is mapped to an output object called input-output pairs in the training phase which is then used to predict new examples in the testing phase [12].

In summary, supervised learning algorithms analyze the training data and produce an inferred function. The inferred function would then be used for assigning new examples. This way, the classifier will determine the class labels for unseen instances [12].

There are many different supervised learning algorithms. In the current project, we explore three popular algorithms, namely, naïve Bayes, logistic regression and support vector machine (SVM). We provide an overview of each of these algorithms in the following.

2.1 Naïve-Bayes

In machine learning, naïve Bayes is a commonly used classifier, known for its simplicity. It is so-called "naïve" because it assumes that each feature is independent [3]. It is a probabilistic classifier, which uses Bayes theorem to calculate probabilities used to train the model.

In probability theory, the Bayesian theorem is used to calculate the posterior probability P(A|B) from the prior probability P(A), P(B) and P(B|A) the probability of B given A. The equation is shown below:

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

Equation 2.1: Bayes theorem

Where,

- P(A|B) is the conditional probability of A given B.
- P (A) is the prior probability of A.
- P(B|A) is the conditional probability of B given A.
- P(B) is the prior probability of B.

In other words, say we are given some data sample X and binary query variable Q1 and Q2. Using the information given, naïve-Bayes classifier calculates the probability P(X|Q1) and P(X| Q2) inferred from the data sample for all effect variables. The probabilities are calculated and are used in the testing stage to label a variable.

To better understand with an example, suppose we have a group of data samples, each sample contains a feature "review" (can have the possible values such as good, best, bad and word) for a movie and its label "feedback" as positive and negative, referring whether a response is negative or positive) as depicted by table 2.1.

Review	Feedback
Good	Positive
Bad	Negative
Good	Negative
Best	Positive
Best	Positive
Bad	Negative
Good	Positive
Bad	Positive
Bad	Negative
Best	Negative

Table 2.1: Sample training set

From the above data sample, we get the following probabilities as illustrated in table 2.2.

P(Feedback|Review)	Feedback
Good given positive	2/10
Good given negative	1/10
Bad given positive	1/10
Bad given negative	3/10
Best given positive	2/10
Best given negative	1/10

Table 2.2: Trained probabilistic model

To classify, whether a review "Best" is positive or negative feedback, we will calculate P(Positive|Best) and P(Negative|Best) and choose either of the maximum value. Now, according to naïve Bayes classifier:

$$P(Feedback|Review) = \frac{P(Review|Feedback)P(Feedback)}{P(Review)}$$

Equation 2.2: Probability function

Putting values in the above equation, as depicted in the equation below:

$$P(Positive|Best) = \left(\frac{2}{10}\right) * 0.5 = \ 0.1$$

Equation 2.3: Probability of Positive (feedback) given Best (review)

$$P(Negative|Best) = \left(\frac{1}{10}\right) * 0.5 = 0.05$$

Equation 2.4: Probability of Negative (feedback) given Best (review)

So, from the above calculations, it is evident that P(Positive|Best) > P(Negative|Best), and therefore labeled as "Positive". Since the probability of feedback (good, bad and best) is a fixed constant value, so its presence does not affect the overall probability. Therefore, we can ignore the denominator when calculating the spam and ham probabilities in our case.

From the above description, we can say that naïve Bayes classifier is easy to implement and very powerful, especially when used in large datasets. With its completely feature independent attribute, it is popular in various modern applications. This includes text classification, spam filtering, sentiment analysis, real-time predictions and recommendation systems [11].

In summary, Naïve Bayes is a straightforward classifier to categorize classes of data set. It uses the concept of Bayes theorem, for holding independence between features to predict values in the testing dataset. It requires less training data and outperforms many other complex classifiers.

2.2 Logistic Regression

Logistic regression is a supervised learning approach that is based on regression models [5]. Logistic regression classifier predicts the value of a binary (dependent) variable, given the target variable is categorical and builds the model using the sigmoid function, and the same goes for continuous variable. This can be better understood with an example taken from [6].

Assume, we want to know a person is a male or female, given a factor of height. We can talk about the likelihood of being male or a female. Since males are considered to have a longer height than females in the majority of the population. As an example, assume that given the height, odd of being male are 0.9. Therefore, the probability of being male would be calculated using equation 2.5.

$$Odds = \frac{P}{1 - P}$$

Equation 2.5: Odds function

Now, to calculate the odds of being female would be 0.1 (1 – 0.9). So, the probability of female to male would be 0.11 (1/9) from the sample distribution. After taking the natural log for each of the possibility calculated, we can also plot the logarithmic graph.

In logistic regression classifier, the dependent variable is a logit, which is the natural log of the odds, that is,

$$Log(Odds) = Logit(P) = \ln\left(\frac{P}{1-P}\right)$$

Equation 2.6: Odds Logit function

So, a logit is a log of odds and odds are a function of P. The classifier also involves a penalty factor C, with different values, that can make logistic regression to retain strength. This is typically used to regulate against overfitting, by not training the model to memorize, instead to generalize while learning from a trend.

In summary, logistic regression is more robust when compared with other classifiers, since the independent features don't have to be normally distributed. It uses the concept of logit function, to weigh the features for calculating their frequencies in a binary variable. It takes less time to train the high volume of data.

2.3 Support Vector Machine

Support vector machine (SVM) is a supervised learning approach that is used to categorize samples of data, typically used for classification and regression analysis. In simple words, an SVM classifier makes likelihoods on hyper-planes, given the decision boundaries. For a given labeled training data, an SVM classifier produces an optimal hyperplane that labels new samples.

To illustrate in detail, an SVM is a representation of the points drawn in space (say a graph), mapped in a way that the various labels of different classes are depicted by a clear gap as wide as possible. The gap defines the width of the region to categorize new samples, on either side of the region obtained. Therefore, an SVM classifier builds a model that organizes new instances to either category, thus, a non-probabilistic binary linear classifier [2].

To better understand SVM with an example, say we are given a set of training examples, each marked as belonging to one or the other between the two categories. Here, we are discussing an instance of SVM classification based on a dataset for diagnosing cancer given in the UCI machine learning repository. Our main goal here is to find a region (separating line, say) for the two classes, as given in figure 2.1.

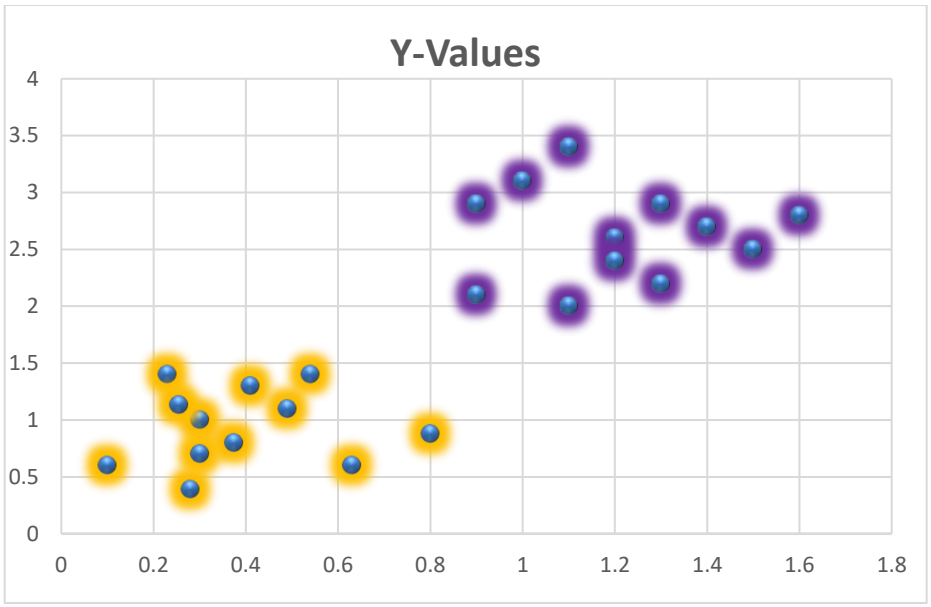


Figure 2.1: Purple and yellow class labels mapped in high dimensionality space

It is evident from figure 2.2 that the width of the region (separating line) should be somewhere between the purple and yellow dots mapped in space.

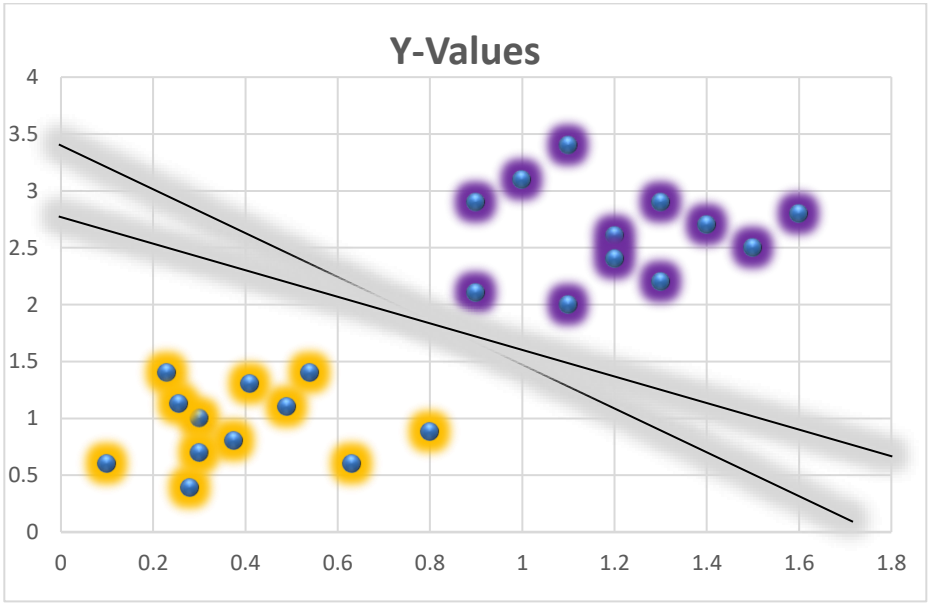


Figure 2.2: A sample partition of the region

The line separating the two classes should be between Y-axis 1.5 to 2, as shown in Figure 2.2. Therefore, given the separating line, if the new data point is mapped above the specific width of the region, it should be labeled to class purple, otherwise labeled as class yellow. SVM also provides a penalty factor C, which enables training a classifier in an efficient way.

In summary, SVM is more effective, in high dimensional spaces. It performs well when a suitable kernel function is used for the decision function. It is also memory efficient since it uses a subset of the training samples in the decision function called support vectors.

Chapter 3 – Data Analysis

An important aspect of data analysis is pre-processing. Pre-processing refers to the process of sanitizing the data, which means the conversion of raw data into a clean format and make it readable. This is done to ensure that the data gathered from various sources can be used in the analysis for making predictions. For example, various websites use analytical techniques to see what influences customers to buy products. Therefore, keeping only those words that will contribute to making assumptions increases the dimensionality of the problem.

For our project, pre-processing is done on all emails (.eml files) before any message is passed to any of the classifiers used. This includes all the training and testing emails. The processed data is then fed to an algorithm for the initial training, followed by the testing to classify the samples to either ham or spam. Our data analysis approach involves several phases; every phase is designed in a way that it takes input from an output received from its previous stage, to control the flow of data effectively. The basic flow diagram is illustrated in figure 3.1.

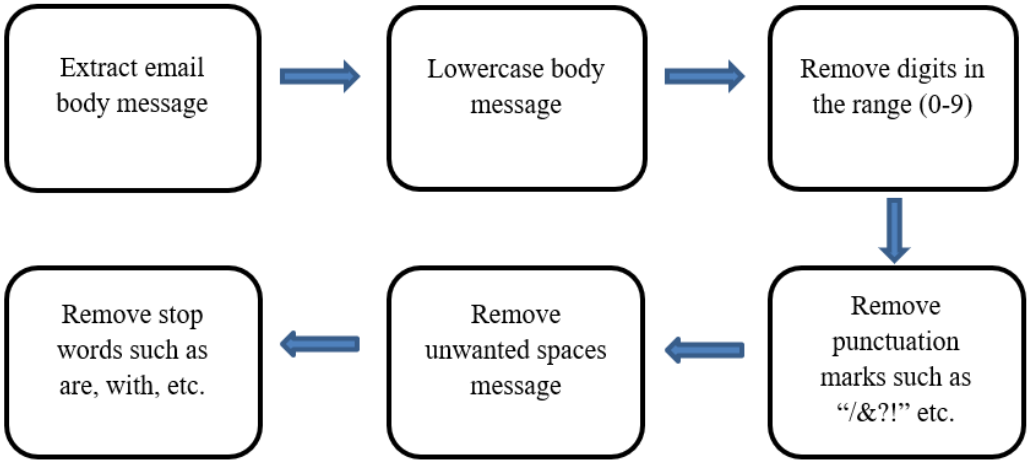


Figure 3.1: Email pre-processing

3.1 Email Extraction

The first step is the extraction of the email body. In this step, an email file is inputted to the program, where all HTML/XML tags, scripting, and the complete header information are removed. Header information such as sender/receiver address, subject, date, time and any other related data, included in an email header is eliminated and therefore returning a simple message for an algorithm to operate on. An example is shown in figure 3.2.

Martin A posted:
Tassos Papadopoulos, the Greek sculptor behind the plan, judged that the limestone of Mount Kerdylio, 70 miles east of Salonika and not far from the
Mount Athos monastic community, was ideal for the patriotic sculpture.
As well as Alexander's granite features, 240 ft high and 170 ft wide, a museum, a restored amphitheatre and car park for admiring crowds are planned
---
So is this mountain limestone or granite?
If it's limestone, it'll weather pretty fast.

Figure 3.2: Sample of email extraction

3.2 Email Lowercase

The second step is to lower case all email messages since we want our classifiers to treat two same words with different cases, the same. This means that keeping all features lowercase would have a higher impact on selecting features that contribute to marking text as either positive or negative. Therefore, this will help our classifiers to maintain similarity throughout the program, and we don't have to use any logistics to treat the same words with different cases. An example is shown in figure 3.3.

martin a posted:tassos papadopoulos, the greek sculptor behind the plan, judged that the limestone of mount kerdylio, 70 miles east of salonika and not far from the mount athos monastic community, was ideal for the patriotic sculpture. as well as alexander's granite features, 240 ft high and 170 ft wide, a museum, a restored amphitheatre and car park for admiring crowds areplanned----------so is this mountain limestone or granite?if it's limestone, it'll weather pretty fast.----------

Figure 3.3: Sample of email lowercase

3.3 Email Digits

The third step is to remove all the digits from the email messages, since numbers don't provide any relevant information for our analysis, except in visual form such as charts, graphs, etc. Also, going forward, our classifiers are designed to treat only text features and not any numbers, so any number in the range of (0 - 9) is removed. An example is shown in figure 3.4.

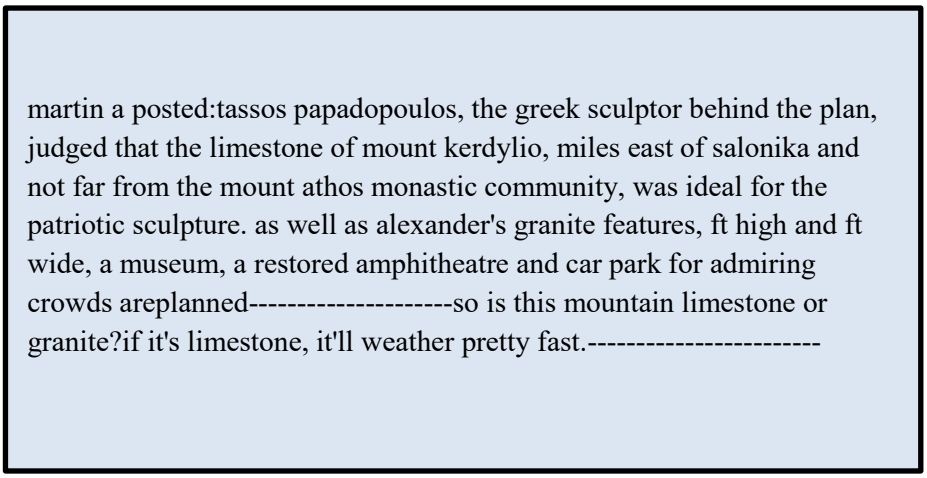


Figure 3.4: Sample of email digits removed

3.4 Email Punctuation

The fourth step is to remove all the punctuation marks because such marks are not helpful to operate on for an exploration of patterns, and messages including punctuation generally do not significantly convey any special meaning to classify whether an email is a spam or ham. Instead, such symbols terminate the programs from normal execution and break the flow of control. So, from the email messages, the set of symbols "[!" #$%&'()*+,-./:;<=>?@[\]^_`{|}~]:" are thoroughly removed. An example is shown in figure 3.5.

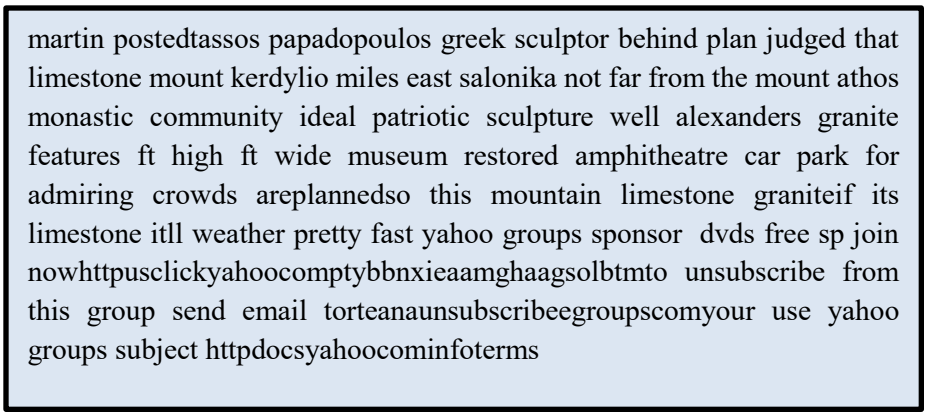


Figure 3.5: Sample of email punctuation removed

3.5 Email Spaces

The fifth step is to remove all the leading, trailing, middle and multiple duplicate spaces, to make the text more efficient and readable. This is done because invalid and extra spaces in the features generally increase the noise in text classification, and therefore reduces the precision and accuracy in the classifiers used. Also, n-grams generated for feature selection are generally composed of character phonemes constituting of spaces in between. Hence multiple spaces between the same words would let used classifiers to treat the same words differently.

3.6 Email Stop Words

The sixth step is to eliminate the words that are used in almost every email communication known as stop words. Words like the, are, all, are usually basic and non-informative words, and these words are not specific to either positive or negative domain and thus fail to provide any high-level meaning information to the classifiers used.

3.7 Email N-Grams

The seventh step is where the feature or attribute extraction is done to make n-grams for every email message called Bag of Words (BOW). These are then passed and processed by classifiers to initially train and then test the model by classifying an email sample as spam or ham. An instance of generated n-grams is shown in figure 3.6.

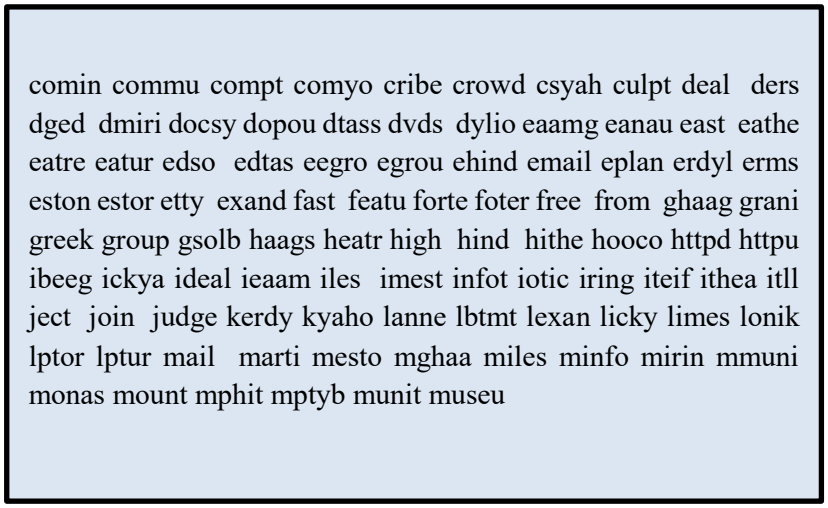


Figure 3.6: Email n-grams extraction

Chapter 4 – Evaluation and Results

The focus of this chapter is to test the email files with various classifiers and then discuss the conducted experiment built on top of the various supervised classifiers. The implementation is done in Python and the libraries used are pandas, NumPy, sci-kit-learn, together with the Jupyter Notebook and Sublime Text 3 used as an IDE.

4.1 Evaluation Metrics

To classify an email as spam and ham and for analyzing in-depth detail for each of the classifiers, the confusion matrix seems to be a perfect choice. This is because it is often used for binary classification and provides the required number of possible outcomes, which are best suited for predicting emails.

	Prediction Spam (0)	Prediction Ham (1)
Actual Spam (0)	TP	FN
Actual Ham (1)	FP	TN

Figure 4.1: Confusion matrix

As depicted in Figure 4.1, the matrix shows all possible outcomes, which are defined as follows:

- True Positive (TP) – Observation is spam and is predicted to be spam.
- False Positive (FP) – Observation is ham but is predicted to be spam.
- True Negative (TN) – Observation is ham and is predicted to be ham.
- False Negative (FN) – Observation is spam but is predicted to be ham.

For the above-stated illustrations, terms that are commonly used to evaluate the efficiency of the classifiers include accuracy rate, recall, and precision, which are defined as follows:

- Accuracy Rate – Defined as the ratio of a total number of correctly predicted samples (ham and spam), divided by the total number of samples taken. Equation 4.1 defines accuracy.

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

Equation 4.1: Accuracy

- Recall – Defined as the ratio of correctly predicted spam samples divided by the total number of actual spam samples. This means that higher the value, the lesser the wrongly classified spam samples and hence, the class is correctly assigned.

$$Recall = \frac{TP}{TP + FN}$$

Equation 4.2: Recall (TPR)

- Precision – Defined as the ratio of correctly assigned spam samples, divided to the total number of predicted spam samples (out of ham and spam samples). The higher value of precision indicates that the class assigned as spam is indeed spam.

$$Precision = \frac{TP}{TP + FP}$$

Equation 4.3: Precision

Given the confusion matrix and the above-stated terms, terms such as True Positive Rate (TPR) and False Positive Rate (FPR) are also used to evaluate the labels assigned to either ham or spam. The terms TPR and FPR are defined as follows:

- TPR – It is a synonym of recall and is also directly proportional to the number of correctly classified spam samples.
- FPR – Defined as the ratio of falsely predicted ham samples to the total number of ham samples.

$$FPR = \frac{FP}{FP + TN}$$

Equation 4.4: False Positive Rate

TPR and FPR are used to sketch an ROC (Receiver Operating Characteristic) curve, by pointing FPR on X-axis against TPR on Y-axis, keeping in mind that for each axis the maximum range is 1. The ROC curve is used for graphical plotting of different threshold values for a specific parameter (in this case, a penalty factor C in LR and SVM). The plot usually describes the proportion of correctly classified spam samples when compared with falsely classified ham samples to estimate a classifier as the best, good or an average one. An illustration of the ROC curve is shown in figure 4.2.

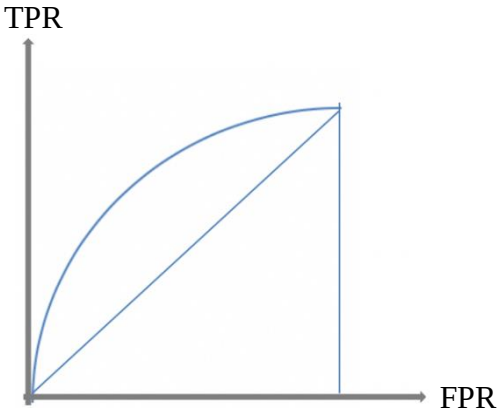


Figure 4.2: ROC curve

It is important to note from the figure 4.2 that when the ROC curve is more towards the left side of the diagonal line, the classifier is described to be the best one, and a curve closer evaluates a classifier as a good or an average one, depending on the rates calculated. The classifier is described to assume samples rather than generalizing from what was previously learned as a poor evaluator when the ROC curve is approximately towards the right side of the line that separates the two regions in the best and poor classification range.

4.2 Experimental Results

The conducted experiment results are presented and examined in this section. The dataset used is pre-processed in the first stage, followed by the training and then tested by each of the classifiers to analyze their accuracies in different contexts. Table 4.1 lists the metrics used to evaluate the classifiers.

Performance Metrics
TP
TN
Precision
Recall (TPR)
FPR
Accuracy
Time

Table 4.1: Evaluation metrics

It is important to note here that we are calculating running time in context of predicting new message samples. The following hardware setup is used to run and test the classifiers:

1. OS – 4-core Intel Core i7 (MacBook Pro)
2. RAM – 16 GB
3. Storage – 256 GB
4. Language used – Python
5. Libraries – Pandas, scikit-learn, NumPy
6. IDE – Jupyter Notebook and Sublime Text3
7. Websites – Leetcode, Geeksforgeeks

4.2.1 Data Pre-Processing

To convert the raw data into a clean format, a step by step procedure is followed as discussed in chapter 3. The output is a clean body message, that is understandable and can easily be read. An instance is shown in figure 4.3 of how the typical email message looks like and its corresponding message in figure 4.4 when the same email is processed.

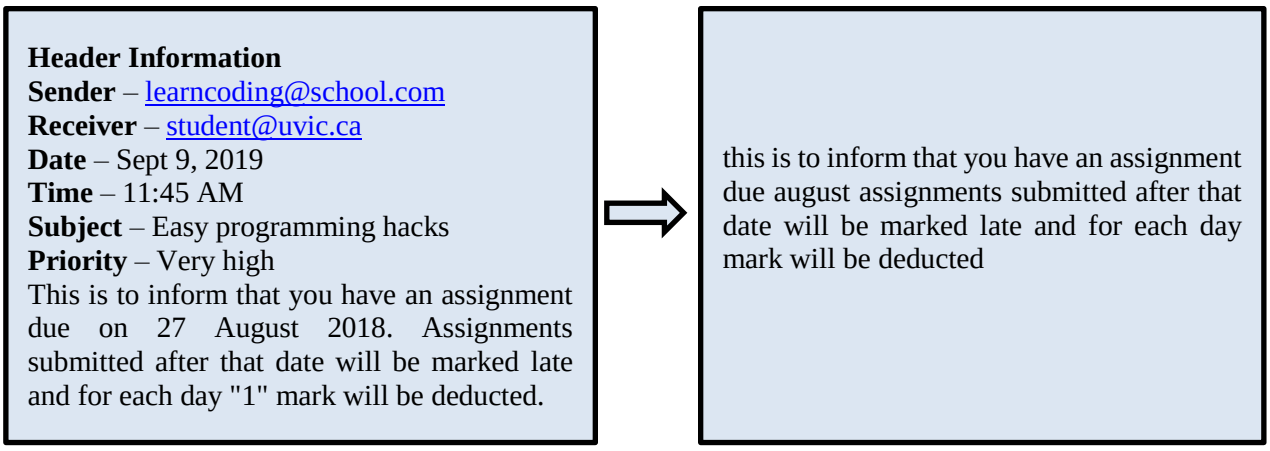


Figure 4.3: Typical email message

Figure 4.4: Email pre-processed

4.2.2 N-Grams Features and Labeling

After the body messages are obtained, the features are extracted using the character phonemes of (5,5) n-grams in a Bag of Words (BOW). We are using label 1 to associate a ham email and label 0 for spam email.

Then, for training the naïve-Bayes classifier, we constructed the spam and ham dictionary which contained the n-grams constructed from the training dataset contained in both spam and ham emails. Each dictionary contained approximately 200,000 n-grams associated with their frequencies in both ham and spam dataset. An illustration is shown in figure 4.5 and 4.6.

```
# coding=utf-8
NotSpam_Dict = {}

NotSpam_Dict = {'aaap':0.004617,'aaba':
0.004617,'aabb':0.004617,'aabc':0.013852,'aabd':
0.009235,'aabe':0.018469,'aabf':0.004617,'aac':
0.004617,'aaca':0.018469,'aacb':0.009235,'aacc':
0.009235,'aace':0.018469,'aacf':0.009235,'aaco':
0.018469,'aact':0.046173,'aacy':0.004617,'aada':
0.009235,'aadb':0.013852,'aade':0.009235,'aadf':
0.013852,'aaea':0.013852,'aaeb':0.009235,'aaec':
0.013852,'aaed':0.004617,'aaee':0.013852,'aaer':
0.004617,'aafa':0.009235,'aafb':0.013852,'aafd':
0.009235,'aafe':0.009235,'aaff':0.013852,'aaft':
0.004617,'aafy':0.004617,'aaga':0.004617,'aagi':
0.004617,'aags':0.15698900000000002,'aagx':
0.004617,'aahr':0.004617,'aahu':0.004617,'aaim':
0.004617,'aair':0.004617,'aait':0.004617,'aajt':
0.004617,'aaka':0.004617,'aaki':0.004617,'aakk':
0.004617,'aalg':0.004617,'aali':0.013852,'aall':
0.009235,'aals':0.009235,'aalt':0.004617,'aame':
0.004617,'aand':0.032320999999999996,'aang':
0.004617,'aani':0.004617,'aano':0.009235,'aant':
0.004617,'aanx':0.004617,'aany':0.004617,'aanz':
0.004617,'aaoc':0.004617,'aapo':0.004617,'aapp':
0.004617,'aapr':0.004617,'aarc':0.004617,'aarg':
0.004617,'aari':0.004617,'aarm':0.004617,'aaro':
0.041556,'aaru':0.004617,'aary':0.004617,'aas':
0.004617,'aasc':0.004617,'aasd':0.004617,'aast':
0.004617,'aatc':0.004617,'aate':0.004617,'aati':
0.004617,'aato':0.046173,'aatt':0.004617,'aauf':
```

Figure 4.5: Ham dictionary sample

```
# coding=utf-8
Spam_Dict = {}

Spam_Dict={'rgbu':0.005476,'dulh':0.005476,'bsite':
0.361394,'ellh':0.021903,'pwive':0.005476,'llcks':
0.005476,'qqqs':0.005476,'mbara':0.005476,'esnot':
0.049281,'anec':0.010951,'lyii':0.005476,'leaa':
0.005476,'afund':0.005476,'qocu':0.005476,'oplew':
0.049281,'dbust':0.005476,'proza':0.005476,'kdlu':
0.005476,'ksst':0.005476,'inyve':0.005476,'tecod':
0.005476,'uhea':0.005476,'ingkd':0.005476,'tsem':
0.005476,'ttpqt':0.005476,'vard':0.032854,'atinp':
0.005476,'wdra':0.005476,'tlyfo':0.010951,'rgspc':
0.005476,'ittge':0.005476,'hotra':0.005476,'ntmin':
0.005476,'pline':0.043805,'feeac':0.005476,'ncrr':
0.032854,'eirus':0.005476,'hrni':0.032854,'bzgjm':
0.005476,'else':0.191649,'fepow':0.005476,'jmqli':
0.005476,'elogd':0.005476,'vodo':0.005476,'mehrt':
0.005476,'urai':0.010951,'whyr':0.005476,'bloke':
0.005476,'lpho':0.093086,'byer':0.010951,'tgra':
0.021903,'rtple':0.016427,'icgcc':0.005476,'hrunf':
0.005476,'doesi':0.005476,'encr':0.027378,'gsrea':
0.005476,'qqzd':0.005476,'nohl':0.005476,'extd':
0.268308,'kalik':0.005476,'lidc':0.005476,'boutp':
0.005476,'aeeea':0.005476,'bimb':0.005476,'uelph':
0.005476,'vyde':0.005476,'keany':0.005476,'morri':
0.016427,'jwvj':0.005476,'agesz':0.005476,'pick':
0.14784299999999997,'eybu':0.005476,'safax':
0.005476,'lenj':0.005476,'ntbl':0.005476,'ssubs':
0.027378,'sjac':0.005476,'iagon':0.005476,'mque':
0.005476,'brids':0.005476,'sshco':0.005476,'gniro':
```

Figure 4.6: Spam dictionary sample

Then for logistic regression and support vector machine, we store all the training email messages into a single .txt file (in a way that each row will correspond to one message) and all the labels associated with it into a separate .txt file. The labels are stored in a way that a message stored at 1st row in a file, its corresponding label will also be at the same row in a labeling file. This is illustrated through an example given in figure 4.7 and 4.8. Now that we have both the messages and labels, the data can be fed into the classifiers.

1.ago alsa angl anyw apps aptg best bett box brai buil buti care chip clam clea disk doin dont dude eart easy
2.ess even ever file fire firs flys forw full futu geom get good hand happ hard hat icha icou imit inst iswa
3.itan kern know lapt linu list long look made mail matr matt mayb meth miss mixe modu mont musi mute
4.neve newe old olde one oss pack page past peop pers poss prob real reco red rpml saou soun star stin
5.tast tedi tell thin thos time took tric tryi turn unti use usin whic whil wow wrot xmms

Figure 4.7: Example of email format stored in .txt file

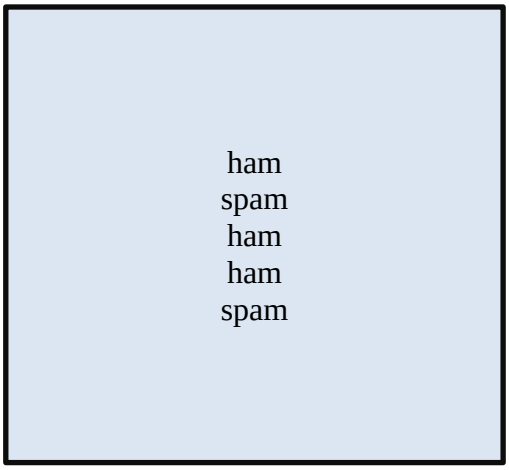


Figure 4.8: Example of email labels stored in .txt file

4.2.3 Naïve-Bayes Classifier

For the naïve-Bayes algorithm, we assume different values (say the probability of ham is greater than spam) to differentiate a ham message from a spam one. Since the features generated are discrete, so we do not have any tuning parameters. The results are shown in table 4.2.

Threshold	**Test < 1e-10**	**Test < 1e-20**	**Test < 1e-30**	**Test < 1e-40**
TP (1398)	1124	1135	1139	1143
TN (2552)	2030	1992	1936	1857
Precision	68.28%	66.96%	64.90%	62.18%
Recall (TPR)	80.40%	81.18%	81.47%	81.75%
FPR	20.45%	21.94%	24.13%	27.23%
Accuracy	79.8%	79.1%	77.8%	75.9%
Time	5s	7s	6s	5s

Table 4.2: Classification results for naïve-Bayes classifier by varying the threshold. Note that TP and FN refer to the total numbers of true positives and false negatives, respectively.

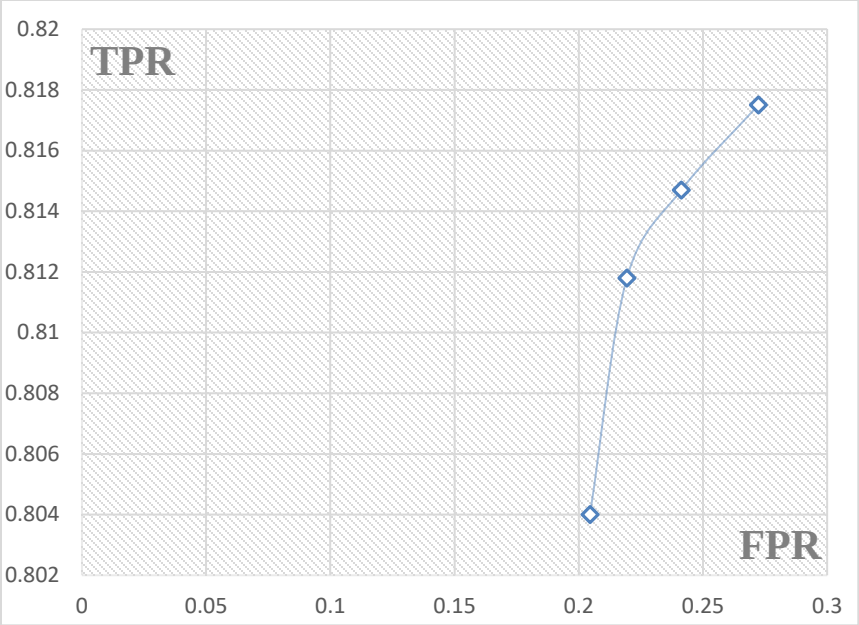


Figure 4.9: ROC curve for naïve-Bayes

It can be observed from the table 4.2 that NB gives encouraging results with an average accuracy of about 78.15%. Since the calculated probabilities are very small (decimal point to 50s and even more), we have used the thresholds in the range of [1e-10, 1e-20, 1e-30, 1e-40]. We see that the classifier predicts ham samples more correctly when the ham probability is about 1e-10, and the detection rate decreases when the probability is too small.

Also, it is seen from the ROC curve depicted in figure 4.9 that the false positive and true positive rate increases as the true positive rate increases since the number of ham samples are correctly recognized when the test probe is rounded to a maximum value.

4.2.4 Logistic Regression

For the LR classifier, we select a specific kernel value of "rbf" and the penalty factor C to be [0.1, 0.2, 1, 5, 10]. The results are shown in table 4.3.

Threshold	**C = 0.1**	**C = 0.2**	**C = 1**	**C = 5**	**C = 10**
TP (1398)	1341	1358	1362	1366	1377
TN (2552)	2412	2420	2443	2469	2493
Precision	90.54%	91.14%	92.59%	94.27%	95.89%
Recall (TPR)	95.92%	97.13%	97.42%	97.71%	98.49%
FPR	5.48%	5.17%	4.27%	3.25%	2.31%
Accuracy	95.01%	95.64%	96.32%	97.08%	97.97%
Time	8s	9s	8s	8s	10s

Table 4.3: Performance results for logistic regression by varying the penalty factor C.

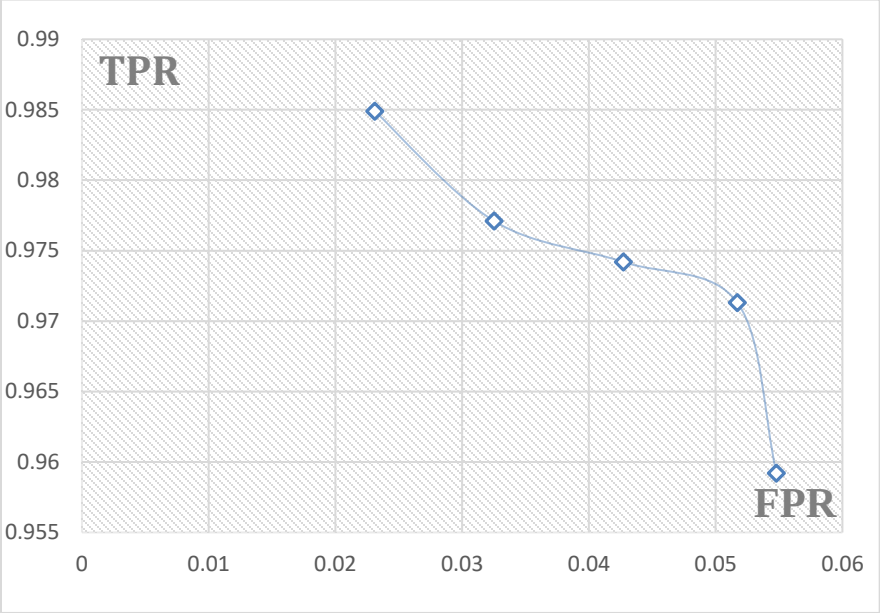


Figure 4.10: ROC curve for LR

It can be observed from the table 4.3 that LR guesses the number of spam and ham samples correctly when the penalty factor (C) is highest, and it then achieves the highest accuracy with top precision and recall values. We see that larger values of C provide more freedom to the model and conversely smaller values of C restrict the model more. It is also noted from the ROC curve depicted in figure 4.10 that LR provides the false positive rate with an average value of 4.09% and the rate is exponentially decreased as the detection rate (i.e. TPR) increases.

It is also noted that LR provides the moderate FPR value that consistently decreased as the true positive and true negative values increased with the increase in the regularization parameter C. This means that the classifier can offer a minimum value of 2.31 associated with the penalty factor C = 10, and after that point the rates calculated were same throughout.

4.2.5 SVM

For the SVM classifier also, we select a specific kernel value of “rbf” and the penalty factor C to be [0.1, 0.2, 1, 5, 10]. The results are shown in table 4.4.

Threshold	C = 0.1	C = 0.2	C = 1	C = 5	C = 10
TP (1398)	1325	1336	1331	1323	1328
TN (2552)	2482	2492	2540	2544	2542
Precision	94.98%	95.70%	99.10%	99.39%	99.25%
Recall (TPR)	94.77%	95.56%	95.20%	94.63%	94.99%
FPR	2.74%	2.35%	0.47%	0.31%	0.39%
Accuracy	96.37%	96.91%	98%	97.97%	97.89%
Time	14s	16s	15s	16s	15s

Table 4.4: Performance results for SVM by varying the penalty factor C.

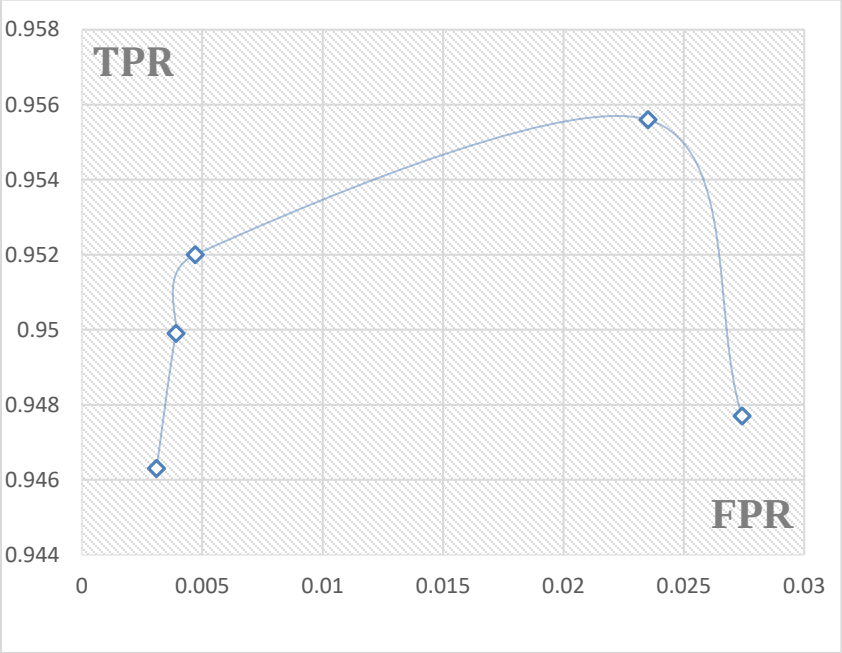


Figure 4.11: ROC curve for SVM

It can be observed from the table 4.4 that SVM produces the top results when the penalty factor (C) is chosen with the value of 1. The detection rate (i.e. TPR) increases with the increase in penalty factor and provides an average accuracy of about 97%, however at a certain point of time the detection rate decreases. This happens due to large value of penalty factor C, that makes SVM classifier to over fit the training data and thus leads to loss in generalization properties of the classifier. The recall value is approximately the same for all factors, but the precision value increased when the maximum number of samples were exactly determined. It is also noted from the ROC curve depicted in figure 4.11 that SVM provides the lowest false positive rate of about 1.25% that is continuously decreased when the penalty factor is highest.

It is also distinguished that the true positive and true negative rates calculated by using SVM classifier provided similar false positive rates when the value of C was chosen in the range of [1,10] and no such difference in the values of true positive and true negative.

4.3 Comparison and Analysis

To summarize the results obtained, table 4.5 illustrates a comparison between each of the classifiers used w.r.t the factors such as TP, TN, Precision, Recall (TPR), FPR, accuracy and the time taken. For each of the classifiers, the average results are presented.

Classifier	**Naïve-Bayes**	**LR**	**SVM**
TP	1135	1360	1328
TN	1953	2447	2520
Precision	65.58%	92.88%	97.68%
Recall (TPR)	81.19%	97.33%	95.03%
FPR	23.43%	4.09%	1.25%
Accuracy	78.15%	96.40%	97.42%
Time	6s	9s	15s

Table 4.5: Comparison of NB, LR and SVM

As evident from the table 4.5, the results strongly settle with the classifiers as explained in chapter 3. By analyzing the results, it is clear that:

- Naïve-Bayes classifier performs above average with an accuracy of approximately 78% which is a decent performance in this context. This means that the classifier predicts new samples correctly most of the time, considering all the n-grams and no tuning parameter required. However, NB has the highest FPR compared to other classifiers which make it less optimal.

- Logistic regression classifier offers an accuracy rate of about 96% comparatively higher than naïve-Bayes and takes more time when predicting new samples. On the other side, SVM also offered similar performance and produces satisfactory recall and precision values as is clear from the TPR value calculated.

- It is also noticed that out of the three classifiers, LR provides the best true positive rate (highest TP value means lower value of FN), since it guesses number of spam samples correctly than SVM, which provides the best precision value (highest TP value), means that the class is correctly recognized (small number of FP).

- It is also obvious that SVM offers the minimum FPR value when compared with LR. This is the result of providing a reasonable set of training data that guided SVM to provide

optimal results in the testing stage, whereas LR on the other side, attained the maximum detection rate with the increase in penalty factor C.

- In summary, for the given dataset used, SVM has demonstrated to be the best classifier and is efficient in terms of recognition capability. The performance rate calculated is approximately 97% and top recall and precision values. This happens because of the specific kernel value used. However, SVM might not be the optimal choice when the dataset is huge as it is demonstrated to consume the maximum time, whereas naïve-Bayes delivers the same results, but less robust than LR.

Chapter 5 – Conclusion

5.1 Summary

In this project, an attempt to review various machine learning classifiers for classification of messages as either spam or ham is discussed. The conducted study started with data analysis, followed by the training of supervised models used and then the testing for evaluating performance. The experiment used publicly available datasets and common evaluation metrics for estimating the performance.

During analysis, it is observed that naïve-Bayes is a good classifier for spam filtering problems by applying Bayes theorem with a strong assumption that the n-grams generated are individually contributing. However, the initial training time to construct both the spam and ham dictionaries (which contain n-grams with their associated probability from the total training dataset) took around 9 hours and hence proven to be time and memory consuming in the initial stage, but once the probabilistic model was trained, naïve-Bayes performed extremely fast when compared with other classifiers.

On the other hand, logistic regression and SVM offered similar performance and each of them has proven to be optimal for spam detection. Despite all the factors mentioned, it is noted that a classifier is best fitted when it provides the highest precision value (precision is inversely proportional to FP, therefore the lower value of FP provides higher precision value by predicting the number of ham samples correctly), even if it compensates accuracy.

5.2 Future Work

The supervised models discussed in this project have their pros and cons and, there is no universal classifier which is a suitable fit for every classification problem. To further improve the performance of the classifiers used, there exists the possibility of additional steps of data analysis. In the conducted experiment, we have only eliminated stop words that are generally used in every other email, therefore keeping those words that are specific to spam and ham domain can provide much more information to the classifiers during the training stage.

Another scope of this project is that we have studied classifiers that are in trend for almost every real-time analysis problem and so, there is still space for complex supervised models such as random forest, neural networks and decision trees, etc.

References

1. Huang Xiao, “Kaggle ADCG SS14 Challenge 01 – Spam Mails Detection: https://www.kaggle.com/c/adcg-ss14-challenge-02-spam-mails-detection/data”, 2013.

2. Afzal Ansari, “Classifying data using Support Vector Machines (SVMs) in Python: https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/”.

3. Ruiqi Hu, “Assessing IP Weight Metrics for Cloud Intrusion Detection using Machine Learning Techniques: https://dspace.library.uvic.ca/handle/1828/9088”, Master project report, University of Victoria, 2018.

4. Eman M.Bahgat, Sherine Rady, Walaa Gad, Ibrahim F. Moawad, “Efficient email classification approach based on semantic methods”, Ain Shams Engineering Journal, Cairo, Egypt – November 15, 2018.

5. Nikhil Kumar, “Understanding Logistic Regression: https://www.geeksforgeeks.org/understanding-logistic-regression/”.

6. Michael T. Brannick, Logistic Regression, “http://faculty.cas.usf.edu/mbrannick/regression/Logistic.html”.

7. Naïve Bayes classifier Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Naive_Bayes_classifier, 23 December, 2009.

8. N-grams from Wikipedia, the free encyclopedia “https://en.wikipedia.org/wiki/N-gram”, 14 September, 2004.

9. Phishing from Wikipedia, the free encyclopedia “https://en.wikipedia.org/wiki/Phishing”, 18 September, 2017.

10. Apache Spam Assassin, “Public Corpus: https://spamassassin.apache.org/old/publiccorpus/”, 2003-2018.

11. Sunil Ray, “6 easy steps to learn naive bayes algorithm (with codes in python and r): https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/”, 11 September, 2017.

12. Supervised learning from Wikipedia, the free encyclopedia “https://en.wikipedia.org/wiki/Supervised_learning”, January 8, 2002.

13. Brian Satterfield, “Ten Spam-Filtering Methods Explained – Learn how different spam-fighting techniques work” 30 November, 2006.