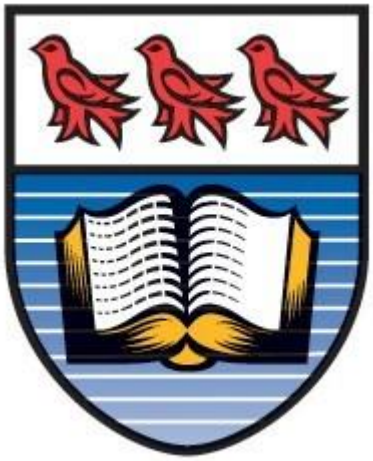
User-centered Spam Detection using Linear and Non-Linear Machine Learning Models

by

Manpreet Singh



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.

Manpreet Singh, 2019 University of Victoria

All rights reserved. This project may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

Supervisory Committee

User-centered Spam Detection using Linear and Non-Linear Machine Learning Models

by

Manpreet Singh

Supervisory Committee

Dr. Issa Traore, Supervisor

Department of Electrical and Computer Engineering

ABSTRACT

The Enron dataset is one of the very few datasets in the world of spam ham detection that has helped the data science community understand the relationship of ham and spam mails for specific users and build powerful models around it. The Enron dataset being textual in nature poses unique challenges in the manner in which information is extracted from the text and supplied to the models. The purpose of the MEng project is to replicate the results obtained by Metsis et al. [1] on spam detection using different strains of Naïve Bayes (NB) classification models, and identify areas for improvement. While Metsis et al. focused solely on linear models, we have explored the performance of non-linear models as well. We have compared the existing NB models with the nonlinear models and simulated the mails that a typical mailbox receives in real time with incremental training. We have also created new datasets from the raw data of the Enron mails, and used these datasets to test the different models. They show interesting results that prove that the proposed approach works for personalized mails more accurately than being generalist in nature.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to thank **Dr. Issa Traore** for his continuous support and mentoring throughout the project for his valuable feedback and suggestions.

Also, I would like to thank my friends and family who encouraged me during this project.

CHAPTER 1 - INTRODUCTION

1.1 CONTEXT AND OBJECTIVES

The Enron dataset is a collection of emails belonging to the employees of a company called "**Enron Corporation**" that went down due to corruption in its financial dealings. A subset of the original Enron dataset was compiled by Metsis et al. [1] and was used to study the spam detection capability of different Naïve Bayes classification techniques. The compiled dataset consists of legitimate emails from six employees from the Enron corpus and spams collected from four different sources. We will refer to this dataset as the Enron spam-ham dataset.

The Enron dataset is a very interesting spam-ham dataset that can be used to explore many insightful ways into which spam can be distinguished from hams. Two ways provided by Metsis et al. [1] are, by splitting the data into train-test set, and by creating a simulation study where each batch of next hundred emails is the test set and the train set slowly increases with time. The first approach is a simple one which helps us understand which model performs best whereas the second approach is a very novel way of simulating the real-world scenario where emails come at regular intervals when the number of hams may be more or vice-versa.

The work conducted in this project involves reproducing the same scenario as reported by Metsis et al. [1] as well as exploring ways to extend the proposed approaches and results. From the perspective of exploring the goodness of the proposed new models, we have used the same metrics as used by Metsis et al. [1].

We compare the linear and non-linear models on Enron spam-ham dataset to find out which model gives better performance and which of the two categories gives the best results in terms of classification accuracy. The best among all is further tested on new extended datasets, which we have created from the raw emails given along the Enron dataset. These new datasets are completely different from the ones used to train the algorithms.

In their paper [1], Metsis et al. suggested several future works that can be performed on the Enron spam-ham dataset. One prominent suggestion is to replace the spam messages by fresher ones and the claim that the same methodology can be used to filter the spam from the ham messages on a personalized setup, which would be worthwhile exploring.

After going through the study and suggestions provided by the authors, the following are the main ideas explored in this project.

1. As the paper only explores the various forms of linear NB algorithms, the project performs a similar study with non-linear algorithms, such as random forest, decision trees, SVM and neural networks. I applied the same performance metrics and presented various evaluation methods, such as the area

under curve (AUC) and receiver operating characteristic (ROC) analysis, recall across various attributes and spam-ham ratio effect on various algorithms.

2. Another interesting area to explore is the cross training and testing mechanism, which is not covered in the paper of Metsis et al. as it is unrelated to its primary objective. The experiment performed in the paper does the training and testing on the same dataset. This is achieved by sequentially adding blocks of 100 messages into the training corpus and testing the trained model on the next block of 100 messages. I have built a new system that learns across all the six user-specific datasets involved in the Enron spam-ham corpus simultaneously and then normalizes the results according to each person and then performs an ensemble study on it.

3. Another contribution of the project consists of exploring a novel usage of rare tokens that Metsis et al. have discarded in their work [1]; specifically, the tokens/words whose occurrence is very low were discarded. My approach consists of using the number of "garbage" words contained in the messages, as many spam messages have a lot of rare words (often common words being misspelled and flagged as rare).

4. It was worth adding new spam messages to the existing Enron spam-ham data, which are more current and reflective of the current social interactions. This helped increase the sensitivity of the algorithms. So, spam messages from the raw spam emails were added assuming the spam messages are not personalized to a user even today.

1.2 REPORT OUTLINE

The remaining chapters in the report are structured as follows. Chapter 2 presents the background of work, which contains key results from the work of Metsis et al. and gives an overview of the linear and non-linear models studied in the project. Chapter 3 presents the initial datasets compiled by Mestis et al., and the extended datasets generated in the project for follow-up testing.
Chapter 4 focuses on the feature building and process flow of token engineering. The important approaches followed like bag of words (BOW) and TF-IDF are explained to help understand the study. Chapter 5 presents the evaluation results based on both the initial and extended datasets. Finally, Chapter 6 makes some concluding remarks and outlines directions for future work.

CHAPTER 2 – BACKGROUND

2.1 KEY RESULTS FROM THE WORK OF METSIS ET AL.

We summarize in this section the broad points underlying the work of Vangelis Metsis presented in [1].

Our main objective was to understand the research paper in detail and draw parallels to the proposed work. The work presented by Metsis et al. in [1] mainly focuses on building a personalized email classification system in a simulated environment where messages are being received in a real-time scenario. So, the system is technically getting retrained as time progresses and helps evaluate the new incoming messages.

The authors explored five different flavors of NB algorithms, including multivariate Bernoulli NB, Multinomial NB with term frequency attributes and its slightly different form -- multinomial NB with Boolean attributes, multivariate Gauss NB, and flexible Bayes (FB). The paper looks deeply into exploring the potential of the NB algorithms in differentiating spam and ham messages.

The authors proposed a novel way of creating a simulated environment where spam and ham messages are received in a timely sequential fashion. In short, the approach places a random number of spam messages (maintaining the time order within the spam messages) with replacement from the disparate sources between ordered ham messages of each of the legitimate users. This helps in creating a real-world scenario where each mailbox gets messages from the spam world at random instances and has a regular ham message distribution.

The results obtained by the authors have been reported based on three contexts:

- **Size of attribute set:** The paper reports for 500, 1000 and 3000 attributes with the best result being obtained for 3000 features set. But the paper also argues that the use of more attributes does not necessarily result in an increase in accuracy performance.
- **Comparison of NB versions:** There is no clear winner over all the six datasets. In some cases, the multinomial NB with Boolean attributes performed well in 4 out of 6 datasets, and in other cases, the FB outperformed the other classifiers.
- **Learning Curves:** The learning curves illustrating the spam-ham recall ratios show a very true picture of the nature of the Enron datasets. As the authors know the ground truth at the individual level as to which Enron dataset out of the user samples is the toughest and the easiest, the results seem to support the fact that Enron 4 (i.e. subset including ham messages belonging to the 4^{th} user) is said to be the easiest dataset and it turns out the spam-ham recall jumps to near-perfect performance in few accumulations of the training data of 4000 messages. Whereas, the Enron 1 dataset which is the most difficult dataset has a lot of fluctuations as seen in the spam-ham recall curve.

2.2 CLASSIFICATION MODELS

We have explored in the project both linear and non-linear models. In this chapter, we give an overview of the different classification models considered.

2.2.1 LINEAR MODELS

Naive Bayes classifiers

Different strains of Naïve Bayes (NB) have been explored in the project, including Gaussian NB, Multinomial NB, and Bernoulli NB. These classifiers are among the group of simple "probabilistic approach classifiers" based on implementing Bayes theorem with feature independent assumptions. Bayes theorem can find the probability of event A, given that event B has occurred, based on the assumption that the features are independent.

- Multinomial Naive Bayes is specially used for text classification scenarios. The classifier uses the frequency of words in the text as features.

- Bernoulli Naïve Bayes is somewhat similar to Multinomial Naïve Bayes but the predictors here are the Boolean variables, which is not the case in multinomial Naïve Bayes. The parameters here will just take yes/ no values for a word that has occurred or not.

- Gaussian Naïve Bayes helps with continuous predictors. The conditional probability formula includes the Gaussian distribution.

Logistic Regression:

Logistic Regression is a popular and simple algorithm used to solve any classification task. The underlying model uses a technique like linear regression and uses the logistic function for classification.

Support Vector Machine Linear Classifier:

Support Vector Machine (SVM) is a machine learning technique that can be used for both classification and regression tasks.

2.2.2 NON-LINEAR MODELS

The information stored in textual formats are more complicated than images or other content formats, and the main reason behind this is the huge variation in which similar words can be used to mean absolutely different ideas. Therefore, a model that relies on the presence or absence of specific words, such as Naive Bayes models which try to classify emails based on the probability of the occurrence of the words, tend to overlook the textual relationship between words.

Non-linear models can ingest much more complicated information and make decisions with higher accuracy compared to linear models, such as Naive Bayes. The nonlinear models we investigated are presented below.

Neural Network:

A neural network is a multilayer network, which consists of units called neurons, which process input vector into the output. Each neuron, which is taken as input, applies a non-linear or linear function to the layer and passes the output of one layer to the input of the next layer. The weighted sum of the inputs is applied to the signals, passing from one layer to another, which are tuned in the training to get adapted to the neural network.

Support Vector Machine Non-Linear Classifier:

In addition to linear classification, SVM can also perform nonlinear classification. This is obtained with the help of kernel trick in which you implicitly map their inputs to high dimensional feature spaces.

Decision Tree:

A decision tree classifier creates a flowchart-like top to bottom branches where each branch represents a decision rule and each leaf node represents certain output. The topmost node is called the root node. The training time of decision tree is faster than NN algorithm. It is capable of handling high dimensional data with great accuracy.

Random Forest:

Random forest classifier is a meta estimator which fits many decision tree classifiers on various subsets of the dataset. It uses the averaging of the predictions by the separate decision trees to improve the accuracy of prediction and control the overfitting.

Extreme Gradient Boosting:

XGBoost stands for eXtreme Gradient Boosting. XGBoost is a prototype designed to achieve execution speed and model performance of gradient boosted decision trees. As compared to the implementations of other gradient boosting, XGBoost is fast. The implementation of XGBoost is based on the principle of gradient boosting decision tree algorithm. It is mainly used in tasks involving supervised learning, such as regression, classification, and ranking. The feature that makes XGBoost unique is that it helps in reducing overfitting.

Light Gradient Boosting:

It is a model that uses a tree learning algorithm called Light GBM. It is a fast, distributed, high-performance gradient boosting model. It is mainly used for ranking, classification, and other machine learning tasks. It divides the tree leaf wise giving it the best fit. On the other hand, in case of other boosting algorithms, the tree is divided either depth wise or level wise. The complexity is increased significantly leading to overfitting which can be overcome by identifying the depth of splitting.

Cat Gradient Boosting:

CatBoost is a combination of two words mainly “Category” and “Boosting”. It is an open-source machine learning algorithm from Yandex. Its easy integration with deep learning algorithms helps to solve a variety of challenges arising in today business applications. Its distinguishing features are:

1. There is no requirement to train the data extensively.
2. To solve business problems, CatBoost yields powerful support for more descriptive data formats. The CatBoost library performs effectively with numerous data format, like, audio, text, and image.

CHAPTER 3 – DATASETS

In the original Enron email corpus, the personal files of about 150 Enron employees were made available to the public. These personal files contained a substantial amount of personal emails, which were used in creating email classification benchmark and public benchmark corpus for the TREC 2005 Spam Track [4]. While the latter benchmark was being constructed, various spam filters were exercised to eliminate spam from the Enron message collection. That collection was extended by adding spam messages being collected in 2005, which led to a benchmark with approximately 43,000 ham and 50,000 spam messages. It was possible to divide the resulting corpus into personal mailboxes via 'To' field, but the 2005 Spam Track experiments did not perform any such task. Therefore, the experiment was similar to the situation in which training of single filter was done on the basis of messages received by different users, rather than using personalized filters.

The Enron spam-ham dataset created by Metsis et al. caters to a very specific purpose of simulating the real-world scenario of spam ham detection for a particular person. Spam for one person can be ham for others, this is a very tricky situation as building personalized system would require data to be not encoded to maintain the meaning of the emails. As discussed in the paper the existing datasets are restrictive in the sense that the tokens/words in the messages are encoded/hashed and the features are restricted to statistical analysis only, as there is no raw form. The Enron spam-ham dataset on the other hand provides the raw form of emails and can be used to build more features to extract information relevant to the problem.

The focus of the dataset is on Enron employees having large mail boxes. These include the following employees' mailboxes as provided by Bekkerman [3]: - farmer-d, kaminski-v, kitchen-l, williams-w3, beck-s, and lokay-m.
Spam messages gathered from 4 different sources were also incorporated, namely: -

1) The SpamAssassin corpus
2) The Honeypot projects
3) The spam collection of Bruce Guenter (BG) [2]
4) Collection of Spam by Georgios Paliouras (GP) [1]

The Metsis et al. spam-ham corpus is a compilation of the above-mentioned four spam sources and ham messages from 6 different employee mailboxes. The corpus is divided in 6 datasets outlined in Table 3.1, each corresponding to genuine ham messages from a separate employee and spam messages drawn from the aforementioned sources. All the datasets in Table 3.1 are well randomized and reproduced with Ham, spam time periods (month and year in between which the messages were received) as described in [1]. The ham-spam ratio used among three of the resulting benchmark datasets was about 3:1, though for the other three, the ham-spam ratio was inverted to 1:3 as depicted in figure 3.1. On average, the total number of messages in each dataset is within the range of five to six thousand. The characteristics of the six datasets are summarized in Table 3.1.

Enron Dataset	Ham + Spam	Ham: Spam	Ham, Spam Periods
Enron 1	farmer_d + GP	3672: 1500	[12/99, 1/02], [12/03, 9/05]
Enron 2	kaminiski_v + SH	4361: 1496	[12/99, 5/01], [5/01, 9/05]
Enron 3	kitchen_l + BG	4012: 1500	[2/01, 2/02], [8/04, 7/05]
Enron 4	william_w + GP	1500: 4500	[4/01, 2/02], [12/03, 9/05]
Enron 5	beck_s + SH	1500: 3675	[1/00, 5/01], [5/01, 9/05]
Enron 6	lokay_m + BG	1500: 4500	[6/00, 3/02], [8/04, 7/05]

Table 3.1: Original Metsis et al. Enron corpus Ham: Spam ratio

The reproduction of the data as discussed in Table 3.1 has been done with a slight variation in the project with respect to the study in [1] as follows:

- Let S and H be the number of spam and ham messages in each Enron data set.
- The set was ordered using this rule: the lower of the two counts i.e. size(S) or size(H) is used as the proxy for selecting the fixed slots within which the other one (the one with higher counts) is inserted without replacement.
- The choice of inserting them was varied from 0 to 7 messages with a skewed probability distribution of 0.025, 0.15, 0.25, 0.25, 0.15, 0.10, 0.05, 0.025, respectively.

For example, consider the Enron-1 dataset which has S=1500 and H=3672 messages. The lower of the two counts is for spam messages. So, the spam messages were placed in the rows of a data-frame sorted according to the dates of their arrival. Then, the ham messages were picked at random while maintaining the date order at which they arrived based on the probability distribution earlier mentioned and inserted in between the spam messages. This created a simulation where the ham messages were closely packed with fewer instances of spams in between them. The same was done for Enron 2 and 3 datasets as they fall under this caveat of logic.

Similarly, the same process was repeated for cases (Enron 4 to 5) where the ham count was less than the spam count, but this time spam messages were chosen and inserted in between ham messages, creating a simulation where a mailbox has fewer instances of ham messaging and plagued by spams. Figure 3.1 represents the fluctuations of ham: spam ratio per 100 mails batch in the test phase during simulation. The ratio of 3:1 for the first three original Enron datasets (i.e. Enron 1- Enron 3) are kept intact as mentioned in [1] and 1:3 is kept for the last 3 datasets (i.e. Enron 3- Enron 6).

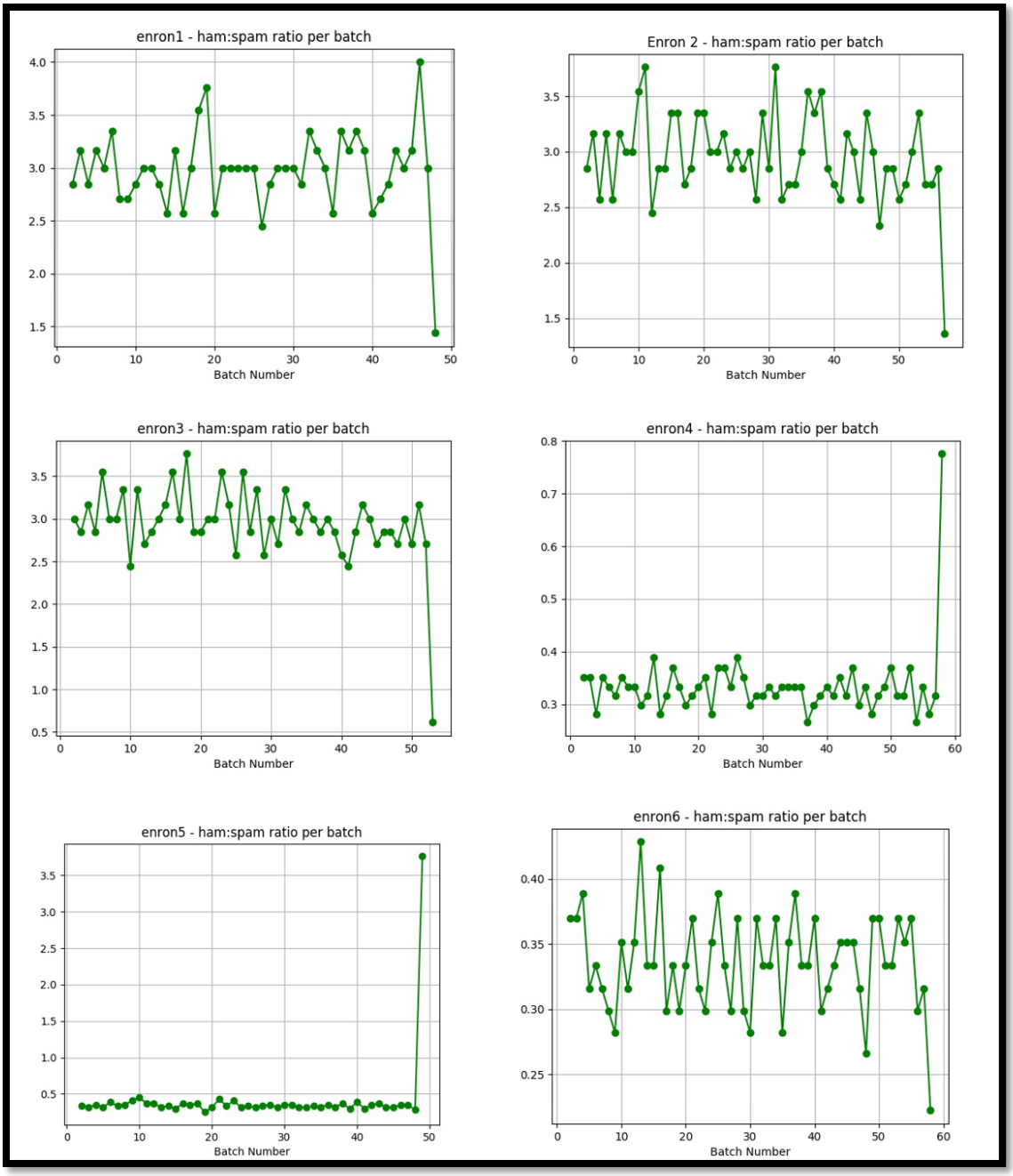


Figure 3.1: Fluctuations of ham: spam ratio per 100 mails batch in the test phase during simulation.

The following are the **pre-processing steps** which were followed for the six datasets in Table 3.1:

Firstly, all the messages sent by the mailbox owners were removed by checking the owner's address which appeared in the 'To:', 'Cc:', or 'Bcc:' fields of the email.
Secondly, for simplification purpose, all HTML tags and headers of the messages were removed, to keep the subject and the body only.

Thirdly, the spam messages that were written in non-Latin character sets were removed since the ham messages in the dataset were written in Latin characters, this may have led to easy identification of non-Latin spam messages.

Along with the Enron-Spam datasets provided in pre-processed form, the ham and spam messages in raw form were also provided by Mestis et al., although the authors did not use such data in the experiment. These raw messages contained raw ham messages from employees' mailboxes: - farmer-d, kaminski-v, kitchen-l, williams-w3, beck-s, and lokay-m, and contained raw spam messages from three spam sources named BG, GP, SH. All these raw messages were pre-processed to bring them in the same shape as the original datasets. We extended the original spam-ham corpus by Mestis et al., by creating 6 more datasets Enron 7 to 12 from the raw data. These will allow testing the models on different datasets, compared to the initial datasets the models were trained and tested on initially. Furthermore, we shuffled the combination of spam sources and ham mails in these new datasets among all 6 users. Table 3.2 summarizes the characteristics of the six new datasets.

Enron Dataset	**Ham + Spam**	**Ham: Spam**	**Ham, Spam Periods**
Enron 7	farmer_d + BG	3669: 1500	[12/99, 1/02], [8/04, 7/05]
Enron 8	kaminiski_v + GP	4363: 1500	[12/99, 5/01], [12/03, 9/05]
Enron 9	lokay_m + SH	2364: 1500	[6/00, 3/02], [5/01, 9/05]
Enron 10	william_w + BG	1500: 1664	[4/01, 2/02], [8/04, 7/05]
Enron 11	beck_s + GP	1500: 4500	[1/00, 5/01], [12/03, 9/05]
Enron 12	kitchen_l + SH	1500: 4500	[2/01, 2/02], [5/01, 9/05]

Table 3.2: Enron 7 to Enron 12 Ham: Spam ratio

CHAPTER 4 – MODEL BUILDING

4.1 EXPLORATORY DATA ANALYSIS (EDA)

Exploratory data analysis or EDA is an approach of analyzing the datasets to summarize their important characteristics. EDA helps in exploration of data, and helps bringing important insights about the data, which further helps in building a better machine learning model. It also helps in feature selection as the exploration shows important features in the datasets. Observing the graphs can further help us reveal the hidden pattern in the dataset, which helps in feature building. Below are some of insights we got while doing EDA:

- The Enron dataset used by the authors of [1] have 17,171 spam mails and 16,545 ham mails.

- The paper [1] uses the absence or presence (tf-idf values) of the top (500, 1000, and 3000) attributes from the mail corpus to build the list of features. As a result, many tokens that could have a potential information to classify a mail as spam or ham were missed out.

- As in my study, I am using the non-linear models, which are well known to perform better when provided with extra information gained from the same data as compared to the linear models. I decided to work around these missed out information and build some new features to further enhance the classification.

- For finding out these meaningful features I looked at the basic structure of how a mail was differentiated into spam and ham. We narrowed down to these four features namely:
  # of Lines in a mail,
  # of Tokens in mail,
  # of punctuations usage,
  # of single character in a mail.

- As evident from the distribution curve shown in Fig 4.1, spam mails tend to have more of these rare tokens. Thus, we can see a hidden pattern from here which can be easily captured by the non-linear models. It can be further supported from the fact that when we plot the feature importance curve, the top features are not the tokens but the above listed features as shown in Fig 4.2.

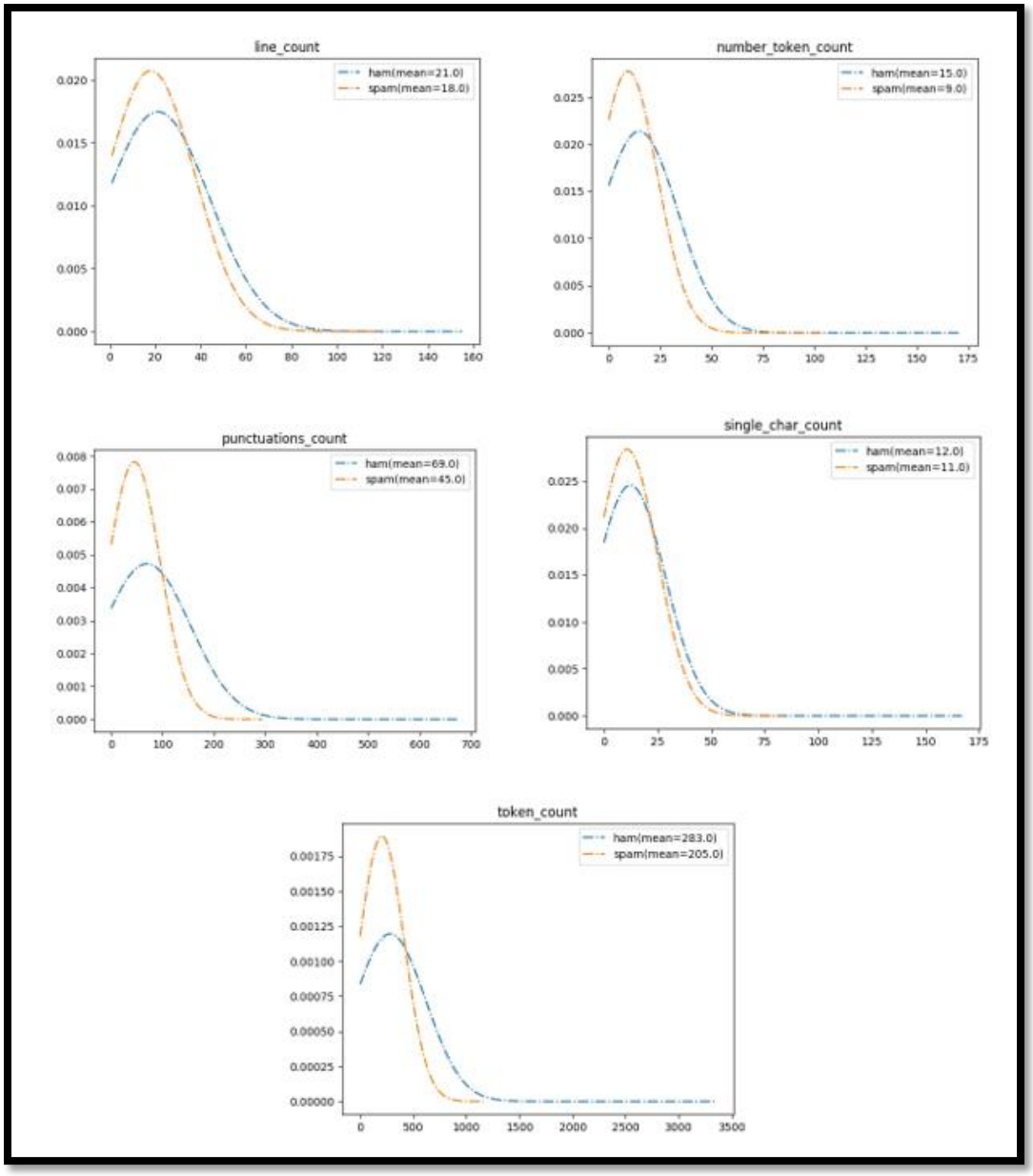


Figure 4.1: The frequency plots of rare tokens which were neglected in the study by Metsis et al. [1].

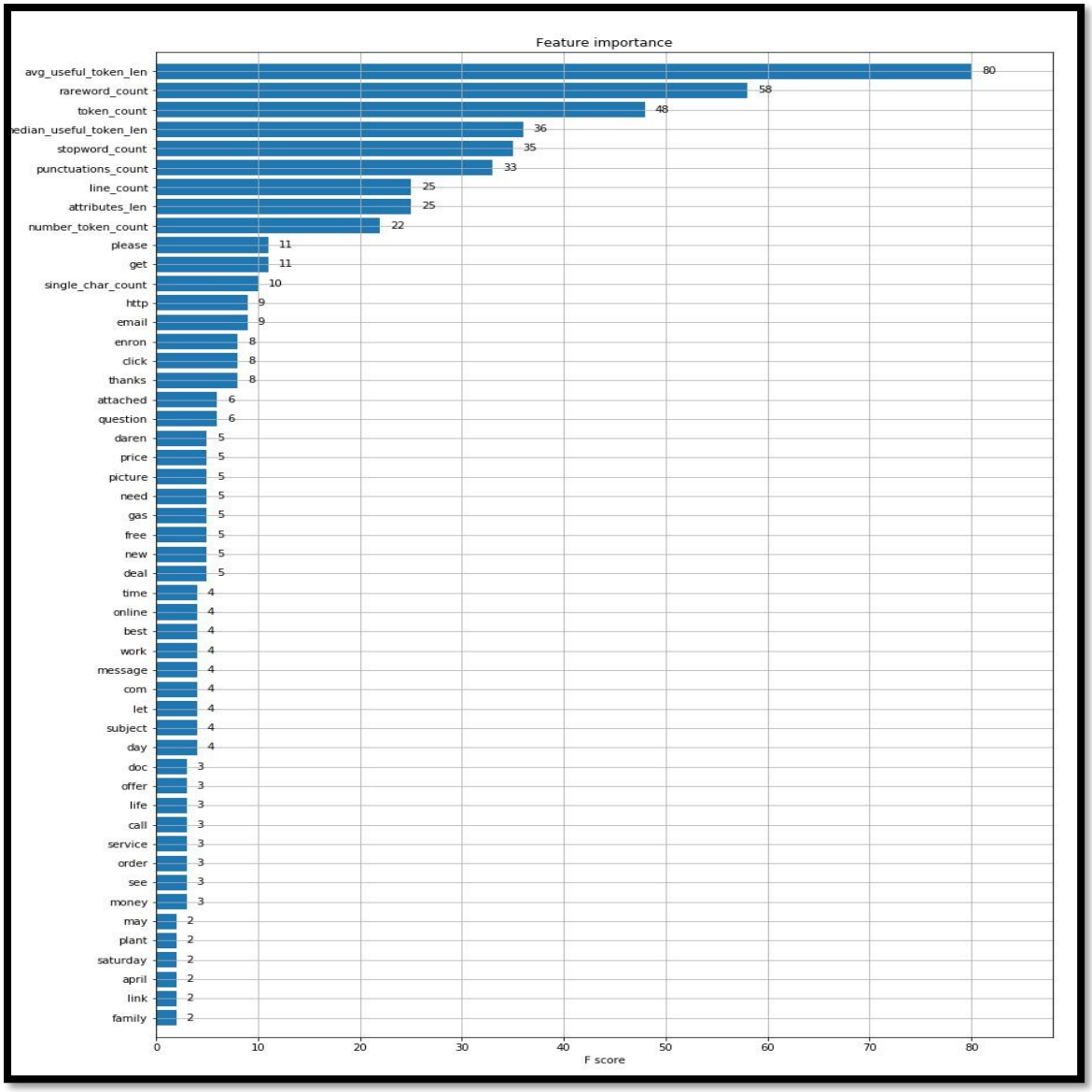


Figure 4.2: Plot showing the feature importance for the Enron 1 Dataset using the xgb model

- This led us to discover some extra features as used in the standard NLP processes, such as stop word count, stemmed tokens and lemmatized tokens.
- For finding out the top tokens whose absence or presence with tf-idf values that would be used as the attributes, I set a minimum threshold of 100 and maximum of 4,000. About 47,195 unique ham words and 110,961 unique spam words from the original Metsis et al. datasets were discovered as shown in Figures 4.3 and 4.4, respectively.

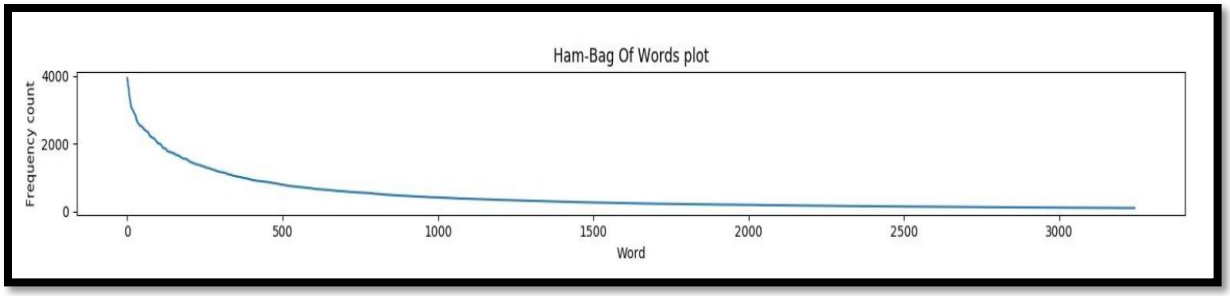


Figure 4.3: Ham word Frequency count

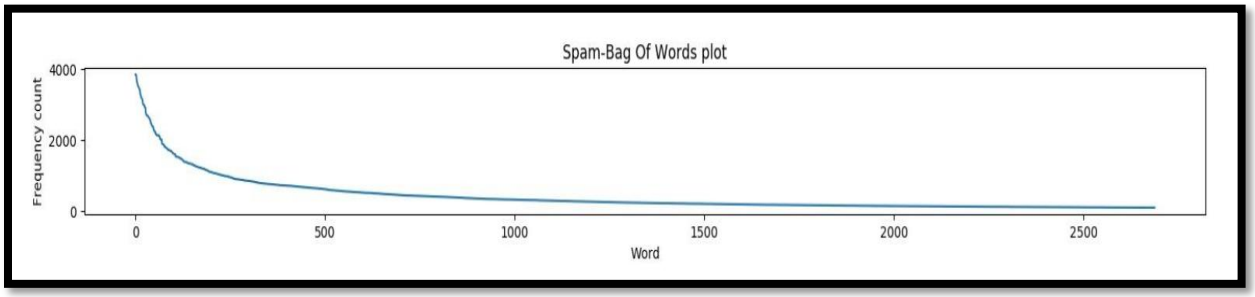


Figure 4.4: Spam word frequency count.

4.2 FEATURE ENGINEERING

Feature Engineering is the process of extracting tokens from the sentences to create features that make sense to machine learning algorithms. In the project, we go through the list of tokens which are high-frequency words in the emails and show relation to spam and ham dataset. The contribution of the project consists of exploring a novel usage of rare tokens that Metsis et al. [1] have discarded in their work; specifically, the tokens/words whose occurrence is very low were discarded. My approach consists of using the number of "garbage" words contained in the messages, as many spam messages have a lot of rare words (often common words being misspelled and flagged as rare).

Following are the steps involved in the extraction of features from a body (message content) of email:

- **Tokenization:** This process involves the conversion of sentences to words. NLTK, the Natural Language ToolKit, is one of the most used NLP libraries which helps in tokenization of sentences.
- Removal of unnecessary punctuation, tags.
- Removing the stop words, such as "the", "is" etc. from the bag of tokens that do not have specific semantic.
- **Stemming:** This process involves the reduction to root by removing inflection through dropping unnecessary characters, usually a suffix.
- **Lemmatization:** This is one of the most popular approaches to move inflection, which is done by determining parts of speech and reducing the word to a common base root word. NLTK can be used for all sort of tasks from stemming, tagging, parsing and beyond.

Figure 4.5 summarizes the process followed in the feature engineering to extract the features from the data and train and build the model around it. This includes the steps followed from the extraction of data to the feature engineering and selection of lemma tokens (tokens which are lemmatized to help reduce words to a common base root word). Figure 4.6 shows the list of unique tokens after stemming and lemmatization. Figure 4.7 shows the useful token count from spam and ham email.

After token engineering the dataset is split in 80:20 ratio to train and test. The recall, accuracy, and specificity are noted, and the best learning non-linear algorithm is then tested on a real-world scenario, which in our case is the newly built datasets as shown in table 3.2.

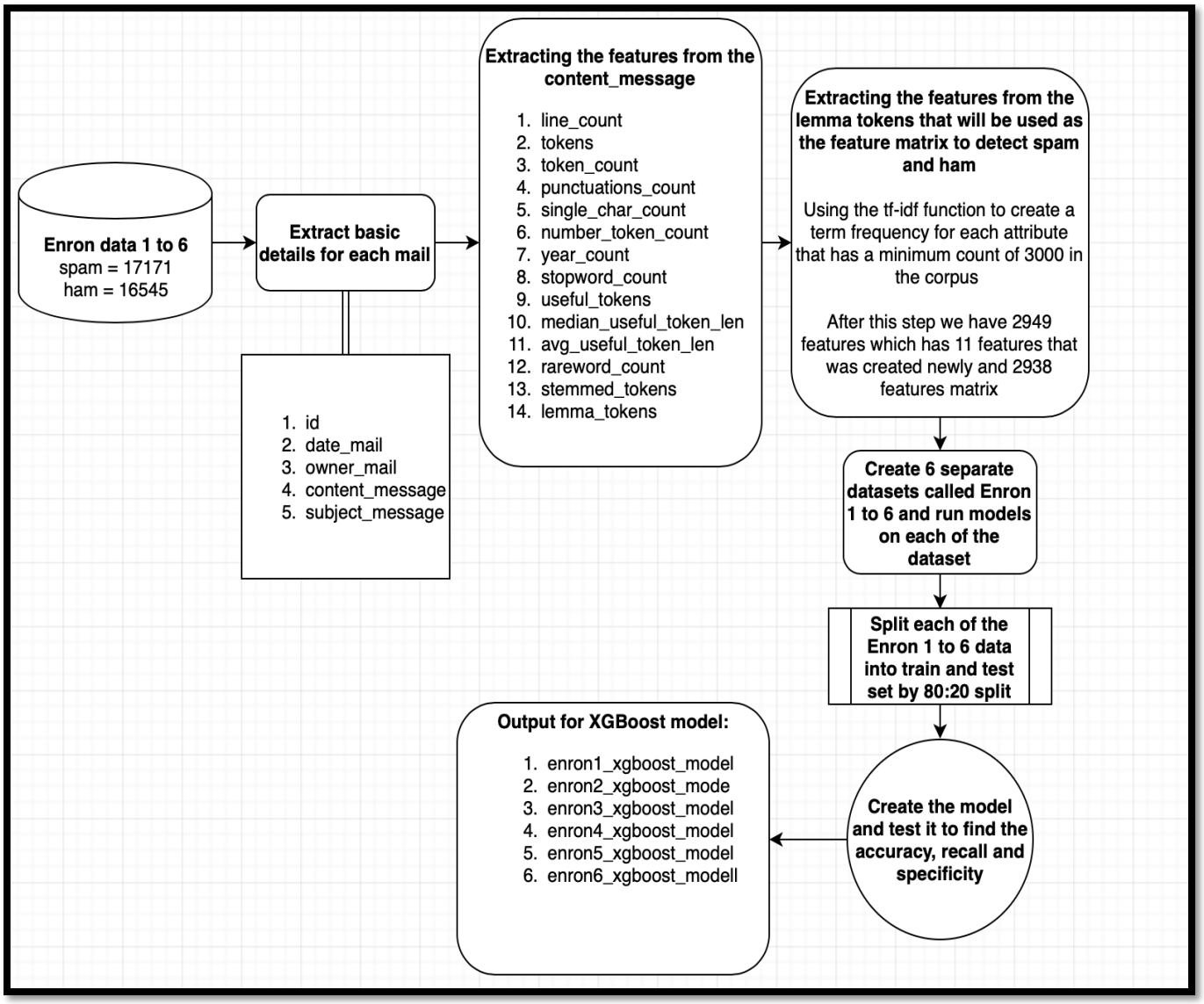


Figure 4.5: Feature Engineering and Model Building process flow diagram

Figure 4.6: Overview of tokens created after lemmatization.

Figure 4.7: Useful token count from spam and ham email.

Two types of features were proposed by Metsis et al. in [1]: features where the words/tokens are presented as Boolean values indicating the presence or not of the token, and features that are based on the token counts.

In this project, a set of new features are introduced as follows:

- ***Line_count:*** the number of lines in the mail received
- ***Token_count:*** the number of tokens (words, punctuations & numbers) in the mail
- ***Punctuations_count:*** the number of punctuations in the mail
- ***Single_char_count:*** the number of single character strings in the mail
- ***Number_token_count:*** the number of numeric strings in the mail
- ***Year_count:*** the number of years mentioned in the mail between 1970 to 2050
- ***Stopword_count:*** the number of stop words (corpus taken from nltk library) contained in the mail
- ***Useful_tokens:*** the useful tokens made by taking the intersection of the tokens contained in the mail with the top 2938 tokens (by frequency occurrence count)
- ***Median_useful_token_len:*** the median of the length of the characters present in the useful tokens in the mail
- ***Avg_useful_token_len:*** the average of the length of the characters present in the useful tokens in the mail
- ***Rareword_count:*** the number of rare words in the mail, which is essentially the number of tokens (Token_count) minus the number of useful tokens (Useful_token)

- ***Stemmed_tokens:*** the useful tokens after following the process of stemming using the PorterStemmer library
- ***Lemma_tokens:*** the useful tokens after following the process of lemmatization using the WordNetLemmatizer library
- ***Attributes:*** the useful tokens used in the original feature model proposed by Metsis et al. [1]. In addition, we use in our feature a slightly different set of tokens that consist of lemma tokens and must have a frequency count of greater than 220 (statistically found based on the data at hand).
- ***Attributes_len:*** the number of attributes that are present in the mail. The attributes are essentially a list of tokens that are used to create a one-hot encoding features matrix and stitched back to the data with values filled using the term frequency- inverse document frequency (tf-idf) metric calculated with the help of the python sklearn TfidfVectorizer library.

Only the numeric features from the features created in figure 4.2 will be used during the model building. The process involved in the mapping of these features to real valued vectors is known as feature extraction. One of the most known and simple technique to do so is known as Bag of Words. In **Bag of Words (BOW)** we make a list of unique words in the corpus, which is also known as vocabulary. We represent the email text as a vector with each word represented as either 1 for being present in the text and 0 for being absent from the vocabulary. The most popular approach is using **Term Frequency Inverse Document Frequency (TF-IDF)** technique.

- Term Frequency (TF) = (Number of times term *t* appears in a document)/(Number of terms in a document)
- Inverse Document Frequency (IDF)= log(N/n) where N is the Number of documents and *n* is the Number of documents a term *t* has appeared in.
- The IDF of rare word is high and that of frequent word is low.

The rationale behind creating new features (as shown in figure 4.2) other than the attributes **tf-idf** from the mails are mainly centered around two main reasons.

Firstly, the selection of top tokens considered for the tf-idf calculation may not be able to capture the most common words used in spam as it is highly data specific. Hence, we decided to add new features, such as punctuation counts, which help in capturing a very common property of spam mails notorious for using dollar signs. Another instance is the use of simple words in spam mails to make it easy for the reader to understand the theme; for this we developed the rare word counts which considers the diversity of words used by humans in normal messaging or hams.

Secondly, as we decided to compare the Naive Bayes models with the non-linear models like Decision trees, boosting algorithms and neural networks, it is important that more information about the system is provided to the models as it is a well-known fact that non-linear models are excellent at extracting complex/hidden knowledge from the data.

The plot for feature importance shown in Figure 4.2 gives enough evidence for the first rationale we pointed out i.e. some information is definitely gained by the model when the aforementioned attributes were added. Also, it is noticeable that the attribute - average_useful_token_length has a very high f-score meaning this attribute had the maximum splits or it contained the most information to decide whether it's a spam or ham.

The frequency plots in figure 4.1 also show that the occurrence of punctuations in the mails have a pattern. More punctuations in a mail mean it is probably a spam. As evident from the frequency curve, which clearly shows a higher hump compared to the ham curve. More area under a frequency curve means a greater number of mails are contained in that feature. This makes it obvious that the extra features, which we have included is of utmost important and are neglected in the study by the authors of the paper [1], as they have only considered the absence or presence of top 3000 attributes that seemed important based on the frequency of occurrence in the mails. The use of more features in this project is obvious from the fact of using more powerful non-linear algorithms, that work better with extra features extracted from the data.

CHAPTER 5 – EXPERIMENTAL EVALUATION

5.1 EVALUATION METHOD

Once the model is built, the next step is the evaluation of the model. The primary objective of the evaluation in this study was to reproduce a scenario in which a new user of customized learning based anti-spam filter faces: the user starts with a small amount of training messages and retrains the filter as new messages arrive. The cross-validation experiments are generally adopted to evaluate the performance of learning algorithms. In this project, we have also used this technique and according to the research, we found that **there is significant variation between incremental retraining and evaluation from the cross-validation experiments.**

Different factors justify the choice of the evaluation technique, including the following:

- The size variation of the training set;
- The growing use of revolutionary tricks by spam senders to evade detection;
- Over discrete time intervals, the variation in the proportion of spam to ham messages leading to difficult prior estimation;
- A Topic shift of spam messages over time.

To apply the incremental mechanism to the different datasets involved in the Enron spam-ham corpus, we need to arrange the order of messages of each dataset in the same way that it preserves the original messages at the time of arrival i.e., each spam message must be preceded by all spam messages that arrived earlier, and the same applies to ham messages.

The variation of ham ratio over time must also be emulated. It is important to note that the spam and ham messages of each dataset are from different time periods. So, we cannot simply use the message dates. To achieve this, the following algorithm was used by Metsis et al. [1] for each dataset:

1. Let S be the set of spam messages of the dataset and H be the set of ham messages of the dataset.
2. Arrange the order of H messages according to time arrival.
3. Insert |S| spam slots between the ordered messages of H by |S| independent random draws from {1, . . ., |H|} with replacement (|S| is the cardinality of S). If the outcome of a draw is i, a new spam slot is inserted after the i-th ham message. A ham message may thus be followed by several slots.
4. Fill the spam slots with the messages of S, by iteratively filling the earliest empty spam slot with the oldest message of S that has not been placed to a slot.

The messages actual dates are discarded, and it is assumed that for each dataset, message arrival is in the same order as the algorithm produced. In the whole dataset, the ham-spam ratio was observed to be 2.45.

For every ordered dataset, the implementation of incremental retraining and evaluation procedure was as follows:

1. In order to preserve the order of arrival, messages were split in the sequence of batches $b_1, \ldots, b_l$ of k adjacent messages each. Batch b_l may consist of less than k messages.
2. For i = 1 to l − 1, filter is trained on the messages of batches 1, . . ., i, and tested on the messages of b_i+1.

5.2 EVALUATION METRICS

We used standard pattern recognition performance metrics in our evaluation, including true positive, true negative, false positive, false negative, recall, accuracy, and specificity. The complement of spam classification rate is known as spam recall and complement of ham misclassification rate is known as ham recall.

Spam recall is defined as the percentage of spam messages which were recognized accurately by the filter, whereas ham recall is defined as the percentage of ham messages which managed to pass the filter. I will be focusing mostly on specificity as for spam and ham detection, the specificity is of utmost importance. Furthermore, to explore different operating parameter values and investigate tradeoff, we used ROC curves to present the performance measures.

5.3 PERFORMANCE RESULTS BASED ON THE ENRON SPAM-HAM CORPUS

Linear Algorithms: Tables 5.1 to 5.5 summarize the performance measures obtained for the linear algorithms considered in the project based on the original Enron spam-ham corpus.

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	291	705	26	6	96.793003	97.979798	96.443228
enron2	1496	4329	1196	3463	300	866	295	859	7	5	98.886033	98.333333	99.191686
enron3	1500	4002	1200	3201	300	801	289	577	224	11	78.584392	96.333333	72.034956
enron4	4458	1500	3566	1200	892	300	862	287	13	30	96.311819	96.636771	95.666667
enron5	3675	1500	2940	1200	735	300	725	296	4	10	98.552124	98.639456	98.666667
enron6	4499	1499	3599	1199	900	300	870	295	5	30	97.002498	96.666667	98.333333

Table 5.1: Evaluation Performances for the Gaussian Naive Bayes model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	269	633	98	28	87.657920	90.572391	86.593707
enron2	1496	4329	1196	3463	300	866	242	584	282	58	70.779777	80.666667	67.436490
enron3	1500	4002	1200	3201	300	801	276	335	466	24	55.444646	92.000000	41.822722
enron4	4458	1500	3566	1200	892	300	796	235	65	96	86.420788	89.237668	78.333333
enron5	3675	1500	2940	1200	735	300	682	255	45	53	90.444015	92.789116	85.000000
enron6	4499	1499	3599	1199	900	300	763	224	76	137	82.181515	84.777778	74.666667

Table 5.2: Evaluation Performances for the Multinomial Naive Bayes model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	260	698	33	37	93.100097	87.542088	95.485636
enron2	1496	4329	1196	3463	300	866	300	836	30	0	97.343616	100.000000	96.535797
enron3	1500	4002	1200	3201	300	801	296	569	232	4	78.493648	98.666667	71.036205
enron4	4458	1500	3566	1200	892	300	831	275	25	61	92.707460	93.161435	91.666667
enron5	3675	1500	2940	1200	735	300	733	275	25	2	97.297297	99.727891	91.666667
enron6	4499	1499	3599	1199	900	300	896	271	29	4	97.169026	99.555556	90.333333

Table 5.3: Evaluation Performances for the Bernoulli Naive Bayes model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	145	680	51	152	80.174927	48.821549	93.023256
enron2	1496	4329	1196	3463	300	866	29	855	11	271	75.749786	9.666667	98.729792
enron3	1500	4002	1200	3201	300	801	139	770	31	161	82.486388	46.333333	96.129838
enron4	4458	1500	3566	1200	892	300	858	173	127	34	86.420788	96.188341	57.666667
enron5	3675	1500	2940	1200	735	300	683	43	257	52	70.077220	92.925170	14.333333
enron6	4499	1499	3599	1199	900	300	863	57	243	37	76.602831	95.888889	19.000000

Table 5.4: Evaluation Performances for the Logistic Regression model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	248	708	23	49	92.905734	83.501684	96.853625
enron2	1496	4329	1196	3463	300	866	224	861	5	76	92.973436	74.666667	99.422633
enron3	1500	4002	1200	3201	300	801	238	787	14	62	93.012704	79.333333	98.252185
enron4	4458	1500	3566	1200	892	300	852	268	32	40	93.880972	95.515695	89.333333
enron5	3675	1500	2940	1200	735	300	628	289	11	107	88.513514	85.442177	96.333333
enron6	4499	1499	3599	1199	900	300	878	187	113	22	88.676103	97.555556	62.333333

Table 5.5: Evaluation Performances for the linear SVM model over the original Enron Spam-Ham corpus

Non Linear Models: Tables 5.6 to 5.12 summarize the performance measures obtained for the non linear algorithms considered in the project.

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	214	699	32	83	88.726919	72.053872	95.622435
enron2	1496	4329	1196	3463	300	866	106	826	40	194	79.862896	35.333333	95.381062
enron3	1500	4002	1200	3201	300	801	196	766	35	104	87.295826	65.333333	95.630462
enron4	4458	1500	3566	1200	892	300	848	242	58	44	91.366303	95.067265	80.666667
enron5	3675	1500	2940	1200	735	300	665	146	154	70	78.281853	90.476190	48.666667
enron6	4499	1499	3599	1199	900	300	839	134	166	61	81.015820	93.222222	44.666667

Table 5.6: Evaluation Performances for the Gaussian Neural network model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	190	702	29	107	86.686103	63.973064	96.032832
enron2	1496	4329	1196	3463	300	866	116	842	24	184	82.090831	38.666667	97.228637
enron3	1500	4002	1200	3201	300	801	187	784	17	113	88.112523	62.333333	97.877653
enron4	4458	1500	3566	1200	892	300	888	130	170	4	85.331098	99.551570	43.333333
enron5	3675	1500	2940	1200	735	300	688	186	114	47	84.362934	93.605442	62.000000
enron6	4499	1499	3599	1199	900	300	847	161	139	53	83.930058	94.111111	53.666667

Table 5.7: Evaluation Performances for the SVM non-linear model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	288	720	11	9	97.959184	96.969697	98.495212
enron2	1496	4329	1196	3463	300	866	284	865	1	16	98.457584	94.666667	99.884527
enron3	1500	4002	1200	3201	300	801	280	800	1	20	98.003630	93.333333	99.875156
enron4	4458	1500	3566	1200	892	300	892	269	31	0	97.317687	100.000000	89.666667
enron5	3675	1500	2940	1200	735	300	733	292	8	2	98.938224	99.727891	97.333333
enron6	4499	1499	3599	1199	900	300	900	275	25	0	97.835137	100.000000	91.666667

Table 5.8: Evaluation Performances for the random forest model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	264	703	28	33	93.974733	88.888889	96.169631
enron2	1496	4329	1196	3463	300	866	276	845	21	24	96.058269	92.000000	97.575058
enron3	1500	4002	1200	3201	300	801	269	787	14	31	95.825771	89.666667	98.252185
enron4	4458	1500	3566	1200	892	300	876	270	30	16	96.060352	98.206278	90.000000
enron5	3675	1500	2940	1200	735	300	718	277	23	17	96.042471	97.687075	92.333333
enron6	4499	1499	3599	1199	900	300	869	265	35	31	94.421316	96.555556	88.333333

Table 5.9: Evaluation Performances for the decision tree model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	289	723	8	8	98.347911	97.306397	98.905609
enron2	1496	4329	1196	3463	300	866	295	857	9	5	98.714653	98.333333	98.960739
enron3	1500	4002	1200	3201	300	801	287	795	6	13	98.185118	95.666667	99.250936
enron4	4458	1500	3566	1200	892	300	888	280	20	4	97.904443	99.551570	93.333333
enron5	3675	1500	2940	1200	735	300	731	287	13	4	98.262548	99.455782	95.666667
enron6	4499	1499	3599	1199	900	300	893	276	24	7	97.335554	99.222222	92.000000

Table 5.10: Evaluation Performances for the extreme gradient boosting model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	282	721	10	15	97.473275	94.949495	98.632011
enron2	1496	4329	1196	3463	300	866	289	858	8	11	98.286204	96.333333	99.076212
enron3	1500	4002	1200	3201	300	801	290	796	5	10	98.548094	96.666667	99.375780
enron4	4458	1500	3566	1200	892	300	890	275	25	2	97.652976	99.775785	91.666667
enron5	3675	1500	2940	1200	735	300	729	287	13	6	98.069498	99.183673	95.666667
enron6	4499	1499	3599	1199	900	300	897	269	31	3	97.085762	99.666667	89.666667

Table 5.11: Evaluation Performances for the light gradient boosting model over the original Enron Spam-Ham corpus

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
enron1	1481	3653	1184	2922	297	731	293	720	11	4	98.445092	98.653199	98.495212
enron2	1496	4329	1196	3463	300	866	293	858	8	7	98.628963	97.666667	99.076212
enron3	1500	4002	1200	3201	300	801	288	799	2	12	98.638838	96.000000	99.750312
enron4	4458	1500	3566	1200	892	300	889	269	31	3	97.066220	99.663677	89.666667
enron5	3675	1500	2940	1200	735	300	734	273	27	1	97.200772	99.863946	91.000000
enron6	4499	1499	3599	1199	900	300	898	252	48	2	95.753539	99.777778	84.000000

Table 5.12: Evaluation Performances for the cat gradient boosting model over the original Enron Spam-Ham corpus

Discussion: The obtained results show that **xgboost** model performed the best in terms of specificity, accuracy and recall metrics. Furthermore, the processing time including the learning time is best for the xgboost model. On average xgboost achieves for 96.35% specificity, 98.12% accuracy, and 98.25% recall, and processing time of 10 minutes approximatively.

Because xgboost model gives us the best results among all the algorithms, including both linear and nonlinear ones, we evaluate subsequently the xgboost model using incremental retraining and evaluation. Figure 5.1 and Table 5.13 show the visual representation of spam and ham recall per 100 batch, and the spam and ham recall percentage in tabular form using incremental training for the xgboost based on the original Enron spam-ham corpus, respectively.

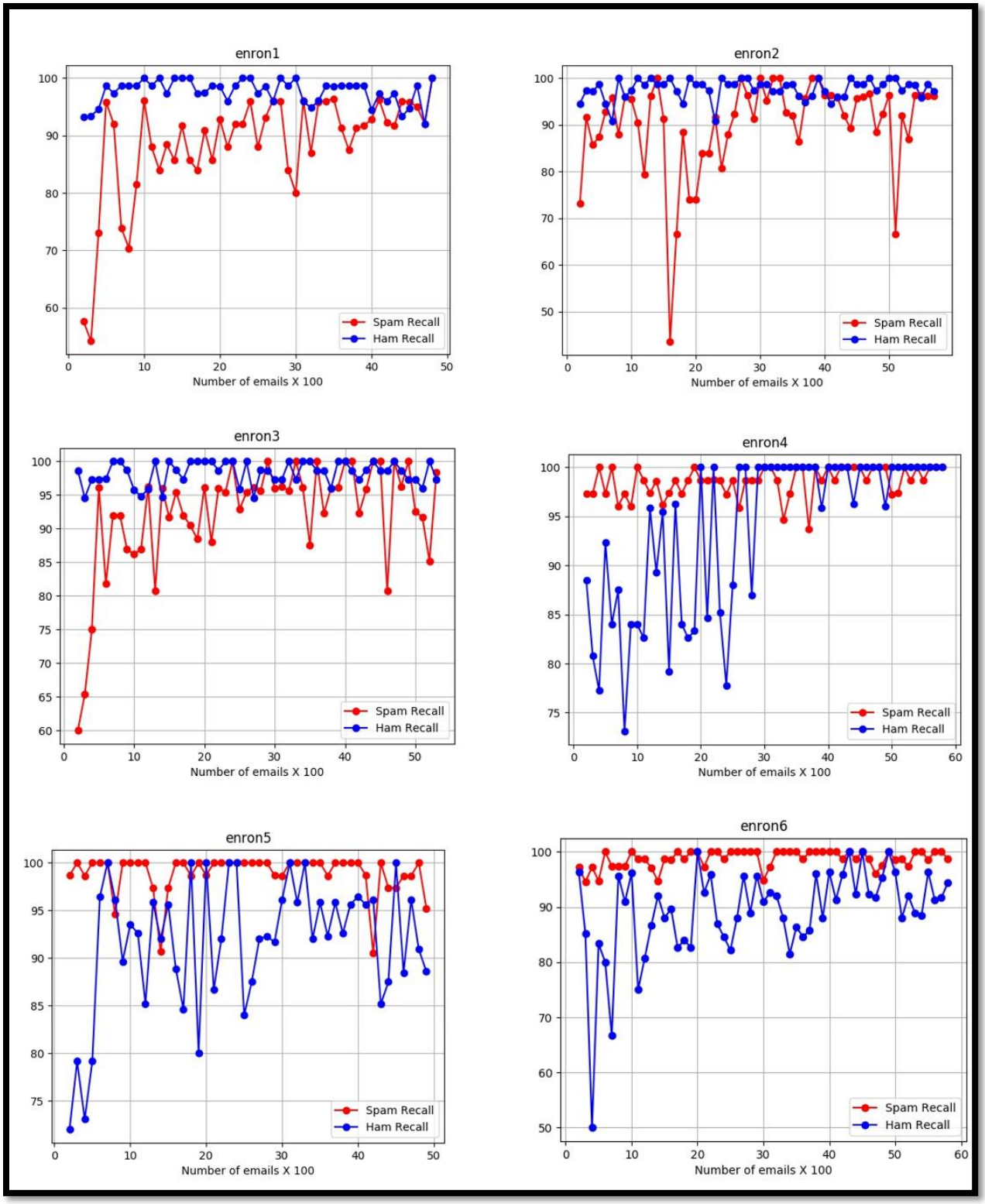


Figure 5.1: The spam and ham recall percentage using incremental training on original Enron dataset trained over xgboost model.

Enron Dataset	**Spam Recall (%)**	**Ham Recall (%)**
Enron 1	88.1	97.6
Enron 2	89.8	97.7
Enron 3	91.9	98.2
Enron 4	98.5	93.5
Enron 5	98.9	91.2
Enron 6	98.8	89.1

Table 5.13: Average recall and specificity for the incremental retraining performed by xgboost on original Enron corpus.

Figure 5.2 depicts the performances obtained for the xgboost on the Enron spam-ham corpus through ROC curves. ROC or Receiver Operator Curve is an important evaluation metric for classification problems like this. It shows how well the machine learning model accurately predicts the dataset. It shows the sensitivity of classifier by plotting the true positive rate and false positive rate. For the outstanding classifiers, the area under the curve or AUC will be close to 1. The Higher the curve, the better is the prediction.

The ROC curves shown in figure 5.2 are magnified in the range of (0,0.02) on x axis to show the minute error details in xgboost model we trained for our study. As the area under the curve is very close to 1.0 and with specificity and sensitivity close to 1, the curve showing a complete range of (0,1) on y axis with error details is impossible. So, we have magnified the curve to give a better view of the error rate which can be seen with uneven lines in the graph. This shows how well the xgboost model is performing that we need to magnify about 10 times as compared to ROC curves in [1], where the authors show a range of (0,0.2) on x axis for Naive Bayes models.

Being a non-linear algorithm, the xgboost model learned very well from the extra set of features and tokens which gave us better performance than any of the Naïve Bayes model shown by Metsis et al [1]. This proves that non-linear models work best for spam and ham detection and have advantages in lesser processing time.

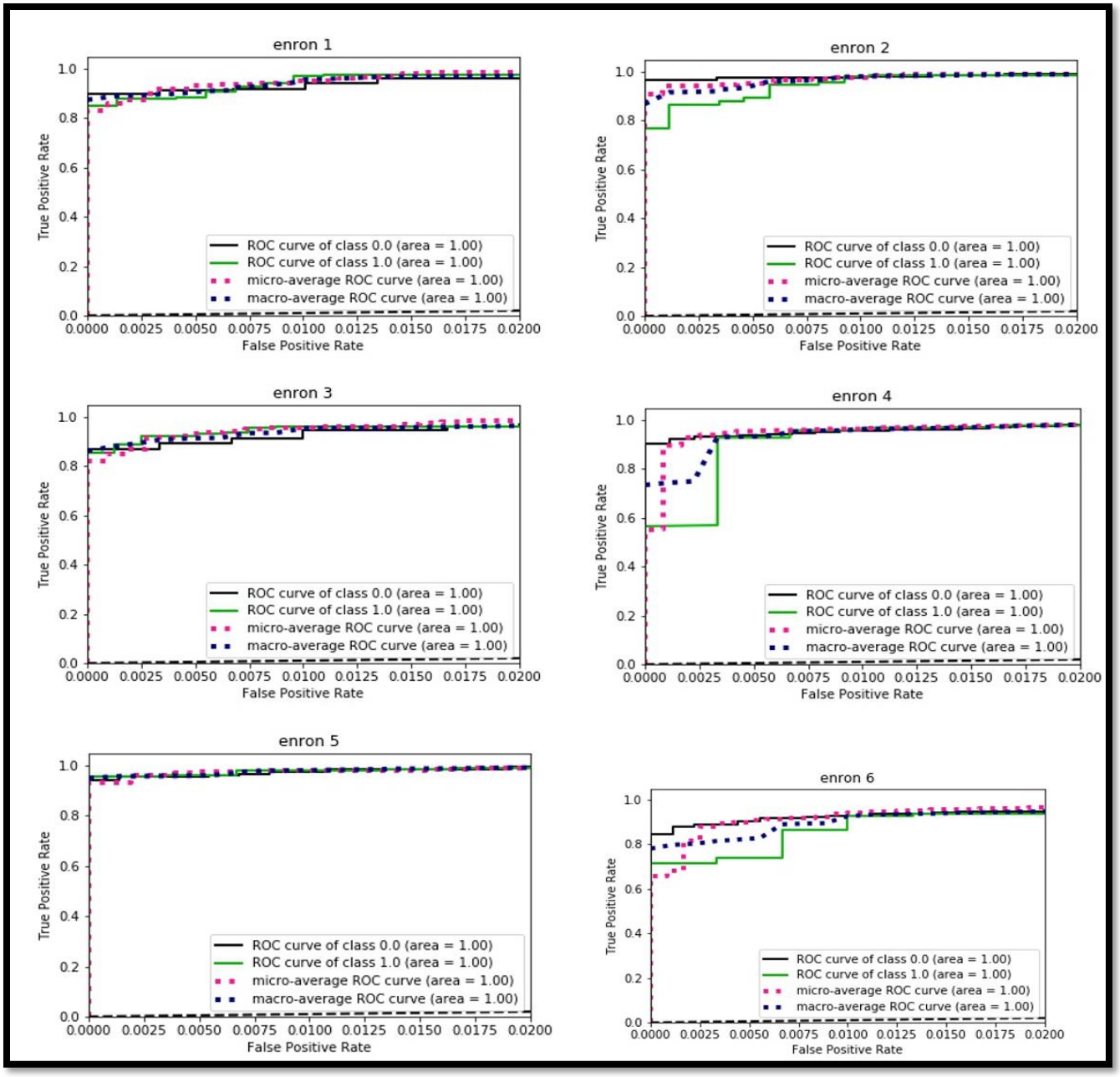


Figure 5.2: ROC curve representing the xgboost model tested on Orignal Enron Corpus

5.4 PERFORMANCE RESULTS BASED ON EXTENDED CORPUS

Using the extended corpus, the xgboost model trained earlier is used in this study to test the accuracy with the same batch processing of 100 mails but without any continuous training. I considered that the optimal/desired accuracy have been achieved using the 6 earlier Enron datasets so I wanted to use these 6 new datasets as fresh datasets with no prior knowledge of the mails. So, I did not boost our model with

each iteration of 100 mails. Table 5.14 shows the performance measures from testing on the extended Enron datasets the previously trained xgboost model from all users in the Original Enron Corpus. This shows the perfect picture of how well the model performs on new unseen data. The specificity, accuracy and recall are all above 90 and shows good results.

Enron_dataset	spam	ham	train_spam	train_ham	test_spam	test_ham	tp	tn	fp	fn	acc	recall	specificity
Enron7	1330	3662	1064	2929	266	733	250	712	21	16	96.200000	93.984962	97.135061
Enron8	1290	4356	1032	3484	258	872	250	866	6	8	98.673740	96.899225	99.311927
Enron9	1387	2347	1109	1877	278	470	259	456	14	19	95.460614	93.165468	97.021277
Enron10	1466	1498	1172	1198	294	300	284	283	17	10	95.294118	96.598639	94.333333
Enron11	3878	1499	3102	1199	776	300	767	281	19	9	97.307335	98.840206	93.666667
Enron12	4192	1493	3353	1194	839	299	835	285	14	4	98.331870	99.523242	95.317726

Table 5.14: Performance obtained for xgboost when using the extended corpus for testing

The other main reason for the creation of the Extended Enron datasets was to show the user centered learning which means the model treats the mails from a specific user as training set and learns over time. This way the model works only best for the user specific ham and spam mails.

To investigate the above assertion, we created different xgboost models trained with one particular user initially (enron1_xgboost, enron2_xgboost, …, enron6_xgboost) and ran each of these models on all of the Extended Enron datasets. Tables 5.15 to 5.20 show the performance measures of different enron_xgboost models tested on the Extended Enron Corpus. It can be seen that the models trained on the original spam-ham corpus (i.e. Enron 1 to 6 datasets) perform poorly with very less specificity on the newly created datasets (Enron 7 to 12) except where the owner of the dataset is same both in the training and testing, which proves that the model created here is a customized spam-ham detection model. So, the model is not going to work that well for emails that belong to someone else except where the spam mails are identical for other users as well.

Enron_dataset	model_name	spam	ham	tp	tn	fp	fn	acc	recall	specificity
Enron7	enron1_xgboost	1330	3662	1229	3316	346	101	91.045673	92.406015	90.551611
Enron7	enron2_xgboost	1330	3662	1068	2994	668	262	81.370192	80.300752	81.758602
Enron7	enron3_xgboost	1330	3662	877	2604	1058	453	69.731571	65.939850	71.108684
Enron7	enron4_xgboost	1330	3662	1324	1575	2087	6	58.072917	99.548872	43.009285
Enron7	enron5_xgboost	1330	3662	1226	2436	1226	104	73.357372	92.180451	66.521027
Enron7	enron6_xgboost	1330	3662	1140	2583	1079	190	74.579327	85.714286	70.535227

Table 5.15: Accuracy and Recall Metrix for Enron 07 using xg-boost training on previous datasets

Enron_dataset	model_name	spam	ham	tp	tn	fp	fn	acc	recall	specificity
Enron8	enron1_xgboost	1290	4356	1250	3198	1158	40	78.781438	96.899225	73.415978
Enron8	enron2_xgboost	1290	4356	1161	4193	163	129	94.828197	90.000000	96.258035
Enron8	enron3_xgboost	1290	4356	1238	3372	984	52	81.650726	95.968992	77.410468
Enron8	enron4_xgboost	1290	4356	1288	2260	2096	2	62.840949	99.844961	51.882461
Enron8	enron5_xgboost	1290	4356	1275	3024	1332	15	76.142402	98.837209	69.421488
Enron8	enron6_xgboost	1290	4356	1251	2868	1488	39	72.954304	96.976744	65.840220

Table 5.16: Accuracy and Recall Metrix for Enron 08 using xg-boost learning on previous datasets

Enron_dataset	model_name	spam	ham	tp	tn	fp	fn	acc	recall	specificity
Enron9	enron1_xgboost	1387	2347	1302	1735	612	85	81.333690	93.871665	73.924159
Enron9	enron2_xgboost	1387	2347	1237	1847	500	150	82.592394	89.185292	78.696208
Enron9	enron3_xgboost	1387	2347	1143	1894	453	244	81.333690	82.408075	80.698764
Enron9	enron4_xgboost	1387	2347	1383	1127	1220	4	67.220139	99.711608	48.018747
Enron9	enron5_xgboost	1387	2347	1348	1392	955	39	73.379754	97.188176	59.309757
Enron9	enron6_xgboost	1387	2347	1292	1946	401	95	86.716658	93.150685	82.914359

Table 5.17: Accuracy and Recall Metrix for Enron 09 using xg-boost learning on previous datasets

Enron_dataset	model_name	spam	ham	tp	tn	fp	fn	acc	recall	specificity
Enron10	enron1_xgboost	1466	1498	1355	605	893	111	66.126856	92.428377	40.387183
Enron10	enron2_xgboost	1466	1498	1185	553	945	281	58.636977	80.832196	36.915888
Enron10	enron3_xgboost	1466	1498	969	640	858	497	54.284750	66.098226	42.723632
Enron10	enron4_xgboost	1466	1498	1460	1333	165	6	94.230769	99.590723	88.985314
Enron10	enron5_xgboost	1466	1498	1348	407	1091	118	59.210526	91.950887	27.169559
Enron10	enron6_xgboost	1466	1498	1255	412	1086	211	56.241565	85.607094	27.503338

Table 5.18: Accuracy and Recall Metrix for Enron 10 using xg-boost learning on previous datasets

Enron_dataset	model_name	spam	ham	tp	tn	fp	fn	acc	recall	specificity
Enron11	enron1_xgboost	3878	1499	3767	1178	321	111	91.965780	97.137700	78.585724
Enron11	enron2_xgboost	3878	1499	3418	1315	184	460	88.023061	88.138216	87.725150
Enron11	enron3_xgboost	3878	1499	3676	1269	230	202	91.965780	94.791129	84.656438
Enron11	enron4_xgboost	3878	1499	3870	767	732	8	86.237679	99.793708	51.167445
Enron11	enron5_xgboost	3878	1499	3824	1351	148	54	96.243258	98.607530	90.126751
Enron11	enron6_xgboost	3878	1499	3748	1074	425	130	89.678259	96.647757	71.647765

Table 5.19: Accuracy and Recall Metrix for Enron 11 using xg-boost learning on previous datasets

Enron_dataset	model_name	spam	ham	tp	tn	fp	fn	acc	recall	specificity
Enron12	enron1_xgboost	4192	1493	3927	1103	390	265	88.478452	93.678435	73.878098
Enron12	enron2_xgboost	4192	1493	3700	1141	352	492	85.153914	88.263359	76.423309
Enron12	enron3_xgboost	4192	1493	3497	1409	84	695	86.297274	83.420802	94.373744
Enron12	enron4_xgboost	4192	1493	4175	790	703	17	87.335092	99.594466	52.913597
Enron12	enron5_xgboost	4192	1493	4063	936	557	129	87.933157	96.922710	62.692565
Enron12	enron6_xgboost	4192	1493	3940	1036	457	252	87.528584	93.988550	69.390489

Table 5.20: Accuracy and Recall Metrix for Enron 12 using xg-boost learning on previous datasets

The Enron 7 dataset contains the mails from user farmer_d and the model that performs well is enron1_xgboost, which was trained on the mails belonging to farmer_d with the highest accuracy of 91.04 %, recall of 92.4% and specificity of 90.55%. The rest of the models do not perform that well on the Enron 7 dataset in all the measurement parameters, which can be explained by the fact these models were trained on other users' email samples. Hence, as indicated by Metsis et al [1], the model enron1_xgboost that was trained specifically on the mails of user farmer_d can be called as the *"best customized model"* for the user farmer_d.

Similar results were observed for the remaining datasets Enron 8 to Enron 12 where the *"best customized model"* belonged to the model whose trained data belonged to the same owner. **More precisely, the specificity (or ham recall) is the highest in cases where the owner of the mails matches the models training set.**

Table 5.21 supports the observation which proves our hypothesis that we have successfully created a customized model from the given data that will perform well only when the model for a given user is trained on ham mails from the same user.

Test Dataset (Ham mail owner)	**Best customized model (Trained on ham mails of owner)**	**Specificity % (Ham recall)**
Enron 7 (**farmer_d**)	Enron_1_xgboost (**farmer_d**)	90.55
Enron 8 (**kaminiski_v**)	Enron_2_xgboost (**kaminiski_v**)	96.25
Enron 9 (**lokay_m)**	Enron_6_xgboost (**lokay_m**)	82.91
Enron 10 (**william_w)**	Enron_4_xgboost (**william_w**)	88.98
Enron 11 (**beck_s)**	Enron_5_xgboost (**beck_s**)	90.12
Enron 12 (**kitchen_l)**	Enron_3_xgboost (**kitchen_l**)	94.37

Table 5.21: Observation of Experiment performed

Another very interesting observation is that the spam recall is the highest for the Enron_4_xgboost model for all the test datasets Enron 7 to 12 as shown in table 5.22, which means the spam mails of the user **GP turns out to be the most generalized spam mail collection,** which helped the models in categorizing even spam mails of the users BG and SH. A possible reason for this would be the way in which these mails were categorized as spams. The mails collected under the SpamAssasin corpus, Honeypot project, and Bruce Gunter mails seem to be a narrow spam data, because an algorithm was used to collect SpamAssasin and Honeypot project, which would have narrowed the focus of spam detection.

Test Dataset (Spam owner)	**False negatives (Wrong spam label)**	**Spam Recall (%)**
Enron 7 (**BG**)	6	99.54
Enron 8 (**GP**)	2	99.84
Enron 9 (**SH)**	4	99.71
Enron 10 (**BG)**	6	99.59
Enron 11 (**GP)**	8	99.79
Enron 12 (**SH)**	17	99.59

Table 5.22: False Negatives using the enron_4_xgboost model

CHAPTER 6 - CONCLUSION

6.1 SUMMARY

The study conducted in the current project shows enough evidence that non-linear models outperform the linear models, and more specifically Naive Bayes models, by a significant margin. The nonlinearity captures more information from the mails to significantly classify spams from hams. The reasons to select extreme gradient boosting model are two-fold, which cover both the advantages provided by Naive Bayes model. Firstly, the model is able to perform well on a wide variety of mails with sensitivity and specificity close to 1. Secondly, the time to perform the xgboost modelling and analysis is also at par with the Naive Bayes modelling time performance.

The latter part of study showed how well the xgboost model performed on the unseen data i.e Extended Enron Corpus. The model when ran among each user helped us prove that the model performs best for the particular user only and not for everyone except when the spam mails are the same among different users. This kind of spam detector filters are used in organizations where the model needs to be user specific and there is no appetite to train the model on every user due to privacy issues.

6.2 FUTURE WORK

We have explored several nonlinear models but there is still room for further improvement using more complex models, such as deep neural networks or CNN models using the help of word2vec transformations that take into account the relationship of words in the mails. In the current report, we have explored the presence and absence of words and frequency variance to supply information into the model to classify the mails. This leaves a huge scope of work in the sense that the CNN or the deep neural networks can exploit not only the presence or absence of words but the relationship that exists between the word usage in mails to capture that extra information to segregate the mails into spam and hams.

CHAPTER 7 - REFERENCES

[1] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam Filtering with Naïve Bayes – Which Naïve Bayes?". Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006), Mountain View, California, USA, 2006.

[2] http://untroubled.org/spam/

[3] R. Beckermann, A. McCallum, and G. Huang. Automatic categorization of email into folders: benchmark experiments on Enron and SRI corpora. Technical report IR-418, University of Massachusetts Amherst, 2004.

[4] S. Hershkop and S. Stolfo. Combining email models for false positive reduction. In 11th ACM SIGKDD Conference, pages 98–107, Chicago, Illinois, 2005.

[5] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, and P. Stamatopoulos, "Stacking classifiers for anti-spam filtering of e-mail. In Conference on Empirical Methods in Natural Language Processing", pages 44–50, Carnegie Mellon University, Pittsburgh, PA, 2001.

[6] M. Wang, Y. He, and M. Jiang, "Text Categorization of Enron Email Corpus Based on Information Bottleneck and Maximal Entropy", Proceedings in IEEE 10th International Conference on Signal Processing (ICSP 2010), China, 2010.

[7] I. Androutsopoulos, J. Koutsias, K. Chandrinos, and C. Spyropoulos. An experimental comparison of Naive Bayesian and keyword-based anti-spam filtering with encrypted personal e-mail messages. In 23rd ACM SIGIR Conference, pages 160–167, Athens, Greece, 2000.

[8] I. Androutsopoulos, G. Paliouras, and E. Michelakis. Learning to filter unsolicited commercial e-mail. technical report 2004/2, NCSR "Demokritos", 2004.

[9] Dataset download from site: http://www2.aueb.gr/users/ion/data/enron-spam/