SPAM SMS DETECTION

Sachin Gurusamy
Department of Computer Engineering
San Jose State University
San Jose, United states
sachin.guruswamy@sjsu.edu

Neha Parakh
Department of Computer Engineering
San Jose State University
San Jose, United states
neha.parakh@sjsu.edu

Yugeshwari Brahmankar
Department of Computer Engineering
San Jose State University
San Jose, United states
yugeshwari.brahmankar@sjsu.edu

Naveen Ravipati
Department of Computer Engineering
San Jose State University
San Jose, United states
naveen.ravipati@sjsu.edu

***Abstract*— In recent years with the advent of technology, popularity of mobile phones observed to be increased and became most necessary device for everyone. With the rapid growth of mobile phone devices, users started using short message service (SMS) for communication as well. Most of the advertising companies adopted SMS as an effective marketing and advertising tool because of its easy usage and cost-effective factor. Popularity of SMS gives hackers chances to steal user's private information. In this project, we are going to propose some machine learning approaches to detect spam SMS. Dataset is taken from UCI Machine Learning Repository and classification is done using Support vector machine, Naïve Bayes and XGBoost classifiers. In our solution we mostly focused on feature extraction from raw data and data preprocessing to achieve best accuracy.**

Keywords— Spam SMS, feature extraction, data preprocessing, Support Vector Machine, Naïve Bayes, XGBoost.

I. INTRODUCTION

According to statistics, global mobile phone population percentage was 62.9% in 2016. After looking at growing usage of mobile devices worldwide, it has forecasted to reach up to 75% by 2020. Mobile phones are one of the most powerful communication devices which let us connect over thousands of miles. Since utilization of mobile phone has become commonplace, users also started responding to other modes of communication such as voice calling, text messaging, video calling etc. Due to unlimited SMS packs facility, text messaging has gained more popularity over others.

SMS is a text communication platform that allows mobile phone users to exchange short text messages usually less than 160 characters. This is the reason many companies started using SMS as an effective marketing and advertising tool which also gave more chances for hackers to steal personal information and cheat mobile users. Furthermore, traditional SMS is not only used for texting among people, but it is also used as an alternative authentication method, usually as a second-way token (mainly in mobile banking, one-time password delivery, etc). However, cost and ease factors made SMS more popular, but it also gave birth to spam SMS. SMS Spam is showing growth, and in 2012 in parts of Asia up to 30% of text messages was spam. In middle east, some of the carriers themselves are responsible for sending out marketing text messages. Additionally, SMS Spam is particularly more irritating than email spams, since in some countries they contribute to a cost for the receiver as well.

Unlike emails, which have a variety of large datasets available, real databases for SMS spams are very limited. Small length of the text messages results into a smaller number of features extraction that can be used for classification. Additionally, text messages are full of abbreviations and have much less informal language that what one would expect from emails, may leads to underperform in their classification. Also spam SMS increasing social and security risk to customer. These challenges along with limited availability of spam detection software motivated us to work on spam text messages detection technique.

II. RELATED WORK AND BACKGROUND

For spam SMS text messages detection there are many techniques available these days like using android apps to block spam messages, using machine learning classification algorithms to filter out spam SMS, etc. In this section, we will review different approaches implemented by authors for spam SMS detection.

In [2], the author Houshmand Shirani-Mehr, mostly focused on feature extraction technique to achieve highest accuracy. Features like counting number of digits, length of message, presence of email id or link, count of numbers, number of dollar sign and flags determines length of message. Author used algorithms like multinomial NB, SVM, KNN, Random forest and decision tree. Author [3] extracted some new features such as presence of dots, special symbols, emoticons, lowercase words, uppercase words etc.

Author [4] developed a classification model using high performance filters. Bogofilter- popular open source Bayesian spam filter, DMC (Dynamic Markov Compression)- adaptive method based on the compression; Orthogonal Sparse

Bigrams with confidence Factor, LR, SVM etc. A vector representation technique using character bi-grams and tri-grams and word bi-grams is also given in this paper.

Paper [5] claims, SVM is best suited for text categorization because it uses overfitting protection method. It provided both theoretical and empirical evidence that SVMs are very well suited for text categorization. The theoretical analysis concludes that SVMs acknowledge the particular properties of text: (a) high dimensional feature spaces, (b) few irrelevant features (dense concept vector), and (c) sparse instance vectors.

In [6], author has applied Naïve Bayes algorithm because of its computational efficiency and good prediction performance and highly sensitive to feature selection. This paper presents two feature evaluation metrics for the Naïve Bayesian classifier applied on multi-class text datasets: Multi-class Odds Ratio (MOR), and Class Discriminating Measure (CDM).

III. Pre-processing and Feature Extraction

A. Data Pre-Processing

For the analysis of the data, special characters, numerals in the message are removed initially. This extracted message is then tokenized by splitting it into an array of words. The effect of abbreviations and misspellings in the messages are ignored. Every sentence contains words which are most frequently used in English, these words are called stop words which are being removed from the array of words extracted from the raw message. From the whole dataset, we will calculate the frequency of each word except the stop words in the data set and the word with the least frequency commonly known rare words will also be removed from the array leading to a clean processed data.

B. Feature model

Our Learning model takes 7 different feature sets as the input, which are as follows:

- The output of the preprocessed data needs to be converted to an array of numbers with each value inside the array indicating the frequency of occurrence of each word which forms the final version of the first feature.
- Count of the number of $ signs that are found in each message
- The length of the message. The intuition behind entering the length of the message as a feature is that the cost of sending a text message is the same if it is contained below 160 characters, so marketers would prefer to use most of the space available to them as if it doesn't exceed the limit.
- Count of the number of digits in the message
- Number of numbers in the message.
- Set this feature flag to 1 when there is a website link or email in the message.
- Set this feature flag to 1 when if there is an emoticon in the message. This is due to the intuition that this will be most unlikely to be used by the marketers.

These different set of features needs to be concatenated with the array of 1st feature to make a complete feature array which will be split into two sets named training and testing sets. Where we train the algorithms with the training set and check for accuracy and MSE using the test set.

C. Dataset

As mentioned, our dataset is a comma separated value file, in which each line consists of messages followed by its type for example ham/spam. We use a dataset 5574 text messages from UCI Machine Learning Repository collected in 2012 [1]. It consists of 425 spam messages manually extracted from the Grumbletext Web site (a UK forum in which cell phone users make public claims about SMS spam). Features extraction is done on raw data, followed by data pre-processing. Processed data is then used in various machine learning classifiers such as Support Vector machine, Naïve Bayes and XGBoost and their performances compared.

IV. Model Algorithms

A. Naïve Bayes

In text classification, two models of Naïve Bayes classifiers are most frequently used: Multi-Variate Bernouli Event Model and the Multinomial Event Model. These two models use the Bayes rule to classify a document. It assumes that all attributes (i.e., features) of the examples are independent of each other given the context of the class, i.e., an independence assumption Given a document d_j, the probability of each class C is calculated as [8]

$$p(c|d_j) = \frac{p(d_j|c)p(c)}{p(d_j)} = \frac{p(d_j|c)p(c)}{p(d_j|c)p(c) + p(d_j|\bar{c})p(\bar{c})} = \frac{\frac{p(d_j|c)}{p(d_j|\bar{c})} \cdot p(c)}{\frac{p(d_j|c)}{p(d_j|\bar{c})} \cdot p(c) + p(\bar{c})}. \quad (1)$$

Now, let us assume a new function Z_{jc},

$$z_{jc} = \log \frac{p(d_j|c)}{p(d_j|\bar{c})}, \quad (2)$$

(1) can be rewritten as

$$P(c|d_j) = \frac{e^{z_{jc}} \cdot p(c)}{e^{z_{jc}} \cdot p(c) + p(\bar{c})}. \tag{3}$$

Now, using (3) we can get the posterior probability $p(c|dj)$ by calculating z_{jc}.

Using this, [9] arranged the naive Bayes text classification models according to derivation strategy for z_{jc}. They designated the pure naive Bayes classifier as the multivariate Bernoulli model, and the unigram language model classifier as the multinomial model. In the multivariate model, a document is considered as a $|V|$-dimensional vector $D=(w1, w2,\cdots w|V|)$ which is the result of $|V|$ independent Bernoulli trials, where $|V|$ is the vocabulary size and w_k is a binary variable representing the occurrence or nonoccurrence of the *k*th word in the vocabulary. The problem with this approach is that in this the frequency of occurrence of the words is ignored.

Contrary to the multivariate Bernoulli model, the multinomial model treats a document as an ordered sequence of word occurrences, with each word occurrence as an independent trial. In other words, a document is drawn from a multinomial distribution of words. In the multinomial model, z_{jc} is computed as follows [9]:

$$z_{jc} = log\frac{p(d_j|c)}{p(d_j|\bar{c})} = log\frac{P(|d_j|)|d_j|!\prod_{i=1}^{|V|}\frac{p(w_i|c)^{tf_{ij}}}{tf_{ij}!}}{P(|d_j|)|d_j|!\prod_{i=1}^{|V|}\frac{p(w_i|\bar{c})^{tf_{ij}}}{tf_{ij}!}} = \sum_{i=1}^{|V|} tf_{ij} \cdot log\frac{p(w_i|c)}{p(w_i|\bar{c})}, \tag{4}$$

Here, tf_{ij} represents the number of t_i in the document d_j.

In (4), the parameters $p(w_i|c)$ and $p(w_i|\bar{c})$ are estimated with Laplacian smoothing as follows:

$$p(w_i|c) = \frac{\theta + \sum_{j=1}^{|\mathcal{D}|} tf_{ij}P(y_j = c|d_j)}{\theta \cdot |V| + \sum_{k=1}^{|V|}\sum_{j=1}^{|\mathcal{D}|} tf_{kj}P(y_j = c|d_j)}, \tag{5}$$

$$p(w_i|\bar{c}) = \frac{\theta + \sum_{j=1}^{|\mathcal{D}|} tf_{ij}P(y_j \neq c|d_j)}{\theta \cdot |V| + \sum_{k=1}^{|V|}\sum_{j=1}^{|\mathcal{D}|} tf_{kj}P(y_j \neq c|d_j)}. \tag{6}$$

In (6), the value of θ is determined empirically. Multinomial method discards the order of occurrence of words but considers the frequency of occurrence of each word in the document.

B. *Support Vector Machines*

Support vector machine examines the data, classifies the data based on a hyper plane. A separating hyper plane is written as:

$$W*X + b = 0 \tag{7}$$

Where $W = \{w_1, w_2, w_3, \ldots\ldots, w_n\}$ is defined as the weight vector. b is defined as bias. The minimum distance of the separating hyper plane is $1/||w||$. SVM works well for the text categorization [5]. 1) SVMs use overfitting protection, which does not necessarily depend on the number of features, they have the protection to handle these large features. 2) Since we used bag of words technique, there are few irrelevant features which can be discarded. If we use any aggressive approach to reduce the dimension of the input space it can only result in loss of information. 3) Input vectors are sparse, which contains only few entries not zero. Kivinen et al. [10] gave both theoretical and empirical evidence for the mistake bound model that additive algorithms, which have a similar inductive bias like SVMs, are well suited for problems with dense concepts and sparse instances.

C. *Ensemble Methods*

Ensemble methods is a machine learning technique that combines several models to produce one final predictive model. Ensemble methods are built on the decision trees but instead of relying on just one decision tree and hoping to predict the correct decision at each split, Ensemble Methods takes into consideration several outcomes from the decision trees and then makes a prediction based on the aggregated results of the sampled decision trees. Hence, Ensemble modeling is a powerful way to improve the performance of the model.

There are two most commonly used Ensembled learning techniques: 1) Bootstrap Aggregation (Bagging) and 2) Boosting. Bootstrapping is a statistical technique in which the given data set is split into random sub-samples, in which each sample is picked from the dataset with replacement. Model is trained on each of the sample. Then, the most frequently occurring hypothesis is chosen as the predictor. Bootstrap Aggregation is a general procedure that is applied to reduce the high variance of decision trees. Boosting is an ensemble method by which the model predictions are improved for any given learning algorithm. The idea is that, in boosting each subsequent layer learns from the predecessor model's error and tries to minimize the error.

Boosting is again split into three types: 1) AdaBoost (Adaptive Boosting) 2) Gradient Boosting and, 3) XGBoost (Extreme Gradient Boosting). As discussed, in the definition of boosting, minimizing the error in previous stages happens through gradient descent algorithm in the gradient boosting method. This process happens sequentially as it involves minimizing the previous error. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Gradient boosting technique is generally very slow because of the process happening sequentially. The process in XGBoost is optimized by parallelization of tree construction using all the CPU cores, cache optimization and distributed computing.

V. Results And Analysis

Content based filtering was used to segregate ham and spam messages. The data was passed to the algorithm by fusing the messages and the features in vector form to the algorithms and the accuracy of each of them is calculated. The metric which we used to calculate the accuracy is confusion matrix. Each algorithm was provided with the appropriate input and the efficiency was calculated for each of them by taking the error calculation into consideration.

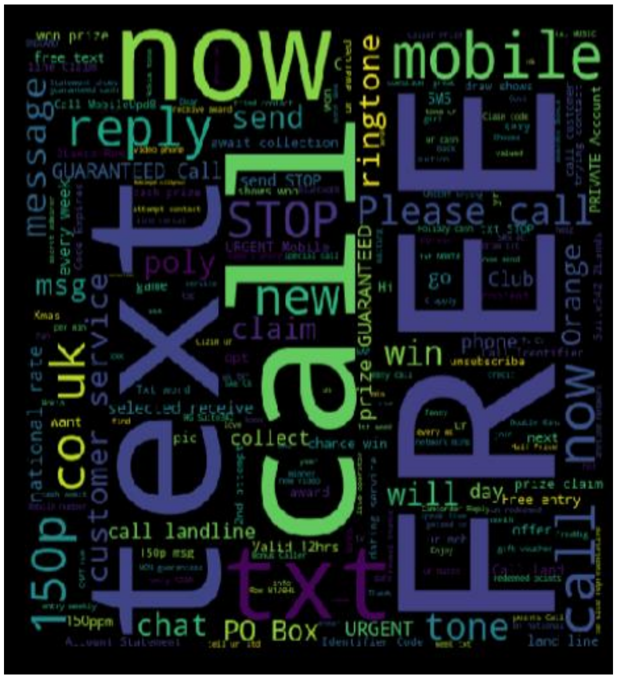

Fig. 1. Most Frequently Occurring Spam messages

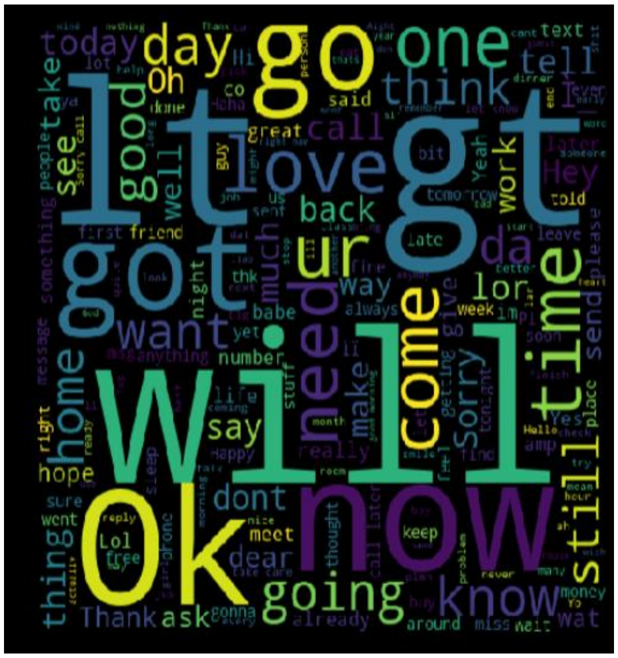

Fig. 2. Most Frequently Occurring Ham messages

For implementation of Naïve Bayes, we use the sklearn library and pass the data in the vector form after data processing to a multinomial model. The training set is fit and the values are predicted for the test dataset. The accuracy score was calculated as the difference between the predicted and the actual values followed by plotting of the confusion matrix and its normalized form as shown in the Fig 3 and Fig 4.

5-fold cross validation was also used to validate the results which gave us a final accuracy of 98.58%.

I Results of 5-fold cross validation for Naïve Bayes

```
[0.98393574 0.97991968 0.98661312 0.98793566 0.97989276]
```

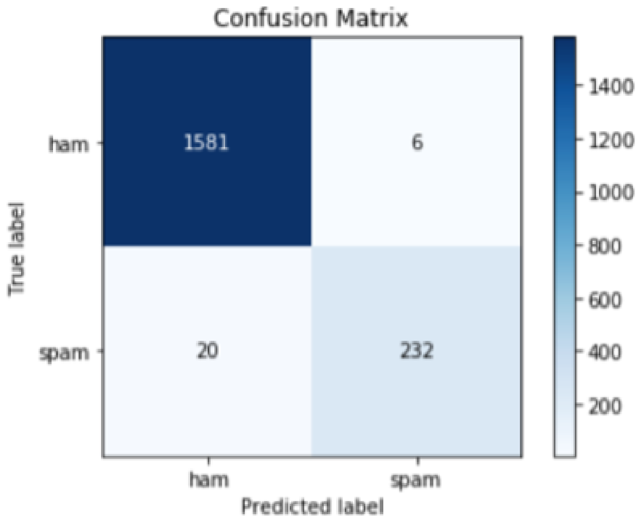

Fig. 3. Confusion Matrix for Naïve Bayes

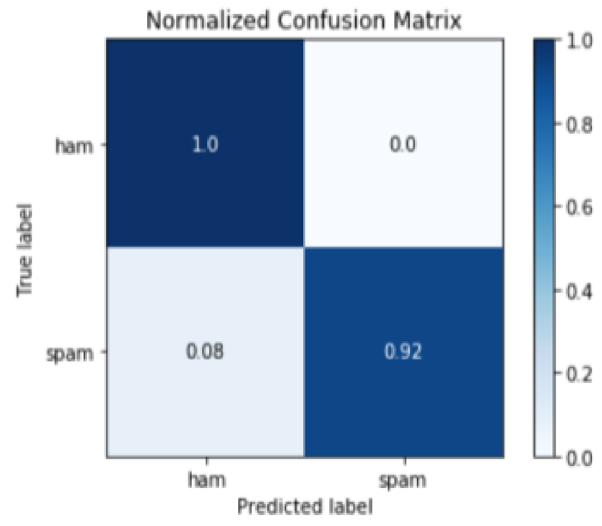

Fig. 4. Normalized Confusion Matrix for Naïve Bayes

The implementation of SVM begins with returning the "best fit" hyperplane that classifies the given data using the RBF kernel. It is followed by predicting the values and calculating the accuracy score. 5-fold cross validation is performed to improve the results and the two confusion matrices were plotted.

Results of 5-fold cross validation for SVM

```
[0.95582329 0.95850067 0.95180723 0.94235925 0.94906166]
```

The accuracy obtained was 95.8%.

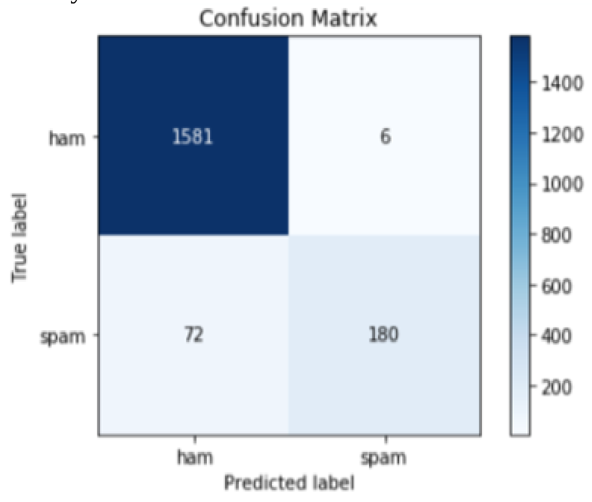

Fig. 5. Confusion Matrix for SVM

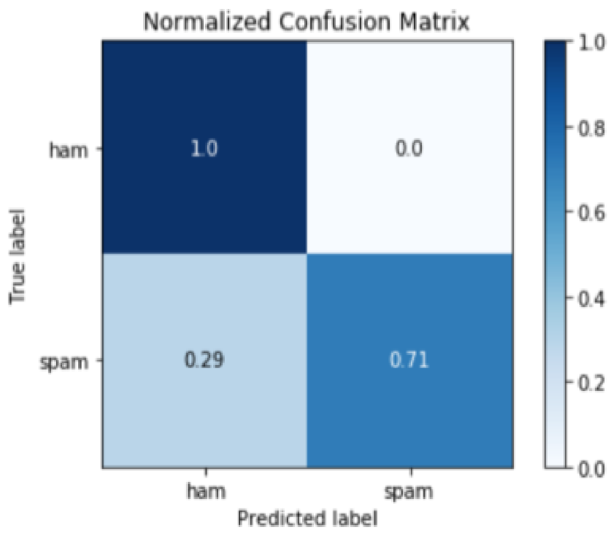

Fig. 6. Normalized Confusion Matrix for SVM

The XGBOOST algorithm was implemented by using the XGB Classifier that classified the data provided into ham and spam. The accuracy score was calculated by computing the difference between the actual values and the predicted values and was plotted in the form of confusion matrix as shown in the Fig 7 and Fig 8.

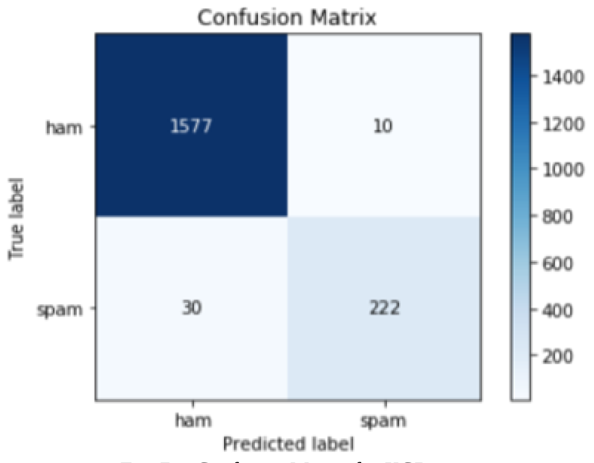

Fig. 7. Confusion Matrix for XGBoost

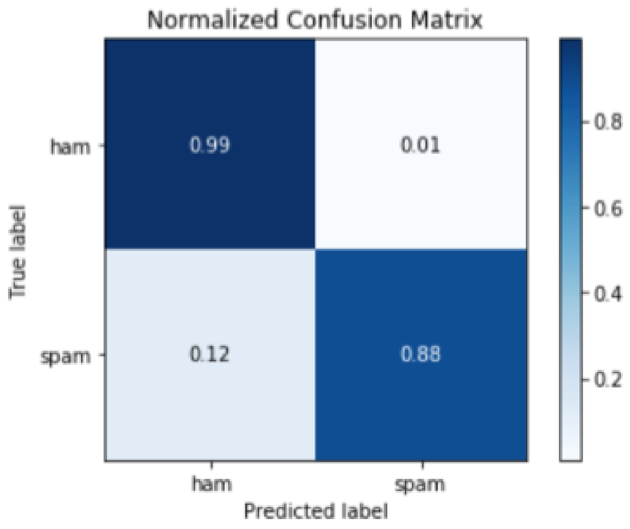

Fig. 8. Normalized Confusion Matrix for XGBoost

The accuracy obtained from XGBoost was 98% and was obtained without any form of cross validation as XGBoost has inbuilt property of the same.

Our analysis showed us that we get the best results from Naïve Bayes which was 98.58% because it performs well on small dataset. Also, it gave the best results because the features were independent. Moreover, it was simpler and faster when compared to other algorithms. The difference between the raining set and the test set error were very close to each other in Naïve Bayes. XGBoost was used to improve the robustness and generalization when compared to other methods. Moreover, the presence of internal parameters for cross validation saves a lot of computation time and makes the implementation simpler.

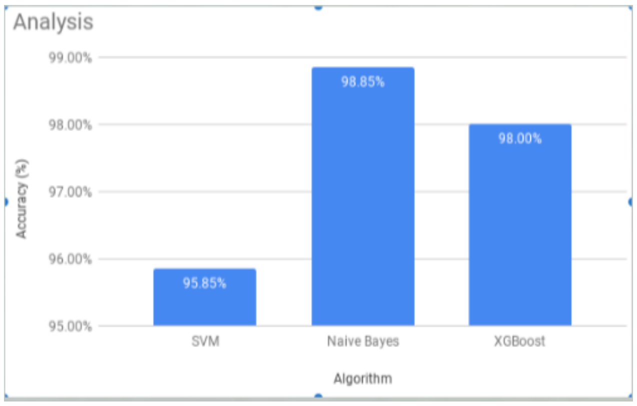

Fig. 9. Overall accuracy graph of the 3 models.

VI. Conclusion

The accuracy of all the models were high due to the addition of appropriate features like the presence or absence of emoticon, the length of the message and presence of hyperlinks etc. The best accuracy was obtained by the Naive Bayes model which was 98.58%. The second best was provided by XGBoost which was close to 98% and the third was SVM which gave an accuracy of 95.8% which might be due to overfitting or high variance.

Acknowledgment

We would like to take this opportunity to extend our gratitude to Professor Wencen Wu who has been a constant support and guidance by helping us with the concepts and implementation of the code.

References

[1] SMS Spam Collection Data Set from UCI Machine Learning Repository,http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection

[2] Houshmand Shirani-Mehr, hshirani@stanford.edu, SMS Spam Detection using Machine Learning Approach, 2013

[3] Neelam Choudhary and Ankit Kumar Jain, Computer Engineering Department, National Institute of Technology, Kurukshetra, Haryana, India. 'Towards Filtering of SMS Spam Messages Using Machine Learning Based Technique'.

[4] Cormack, G. V., Hidalgo, J. M. G., & Sanz, ´ E. P. (2007). Feature engineering for mobile (sms) spam filtering. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '07, ACM, New York, NY, USA, pp. 871–872.

[5] Thorsten Joachims, Universit• at Dortmund, Informatik LS8, Baroper Str. 301, 44221 Dortmund, Germany, Text Categorization with Support Vector Machines: Learning with Many Relevant Features

[6] Jingnian Chen a,b,*, Houkuan Huang a, Shengfeng Tian a, Youli Qu a, a School of Computer and Information Technology, Beijing Jiaotong University, Beijing 100044, China b Department of Information and Computing Science, Shandong University of Finance, Jinan, Shandong, 250014, China,' Feature selection for text classification with Naïve Bayes'

[7] Dino Isa, Lam Hong Lee, V.P. Kallimani, and R. RajKumar, Text Document Preprocessing with the Bayes Formula for Classification Using the Support Vector Machine, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 20, NO. 9, SEPTEMBER 2008

[8] Kim, S.-B., Han, K.-S., Rim, H.-C. and Myaeng, S. (2006), 'Some effective techniques for naive bayes text classification', IEEE Transactions on Knowledge and Data Engineering 18(11), 1457–1466.

[9] Yang Song, Aleksander Kołcz, C. Lee Giles, "Better Naive Bayes classification for high-precision spam detection", *Software: Practice and Experience*, vol. 39, pp. 1003, 2009.

[10] J. Kivinen, M. Warmuth, and P. Auer. The perceptron algorithm vs. winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. In Conference on Computational Learning Theory, 1995.