Bayesian Machine Learning on Email Spam Classification

K. V. Huynh, T. B. Khondowe, S. Yang

ABSTRACT

Nowadays, spam content can be quite frustrating. Agencies and companies are always developing and finding new machine learning approaches that would protect customers, employees, or clients from dangerous spam content. In this research, we compared the efficiency of different methods using the Naive Bayesian approach. Our research focuses on the accuracy score Bernoulli Multivariate Model and Multinomial Bayesian Model. Since Multinomial is the more famous method due to its higher accuracy score, which is also backed up in this paper, we propose altering the Multinomial method with a threshold parameter.

1 INTRODUCTION

For decades, institutions and individuals across the globe have depended on electronic mail for conveniently exchanging information. While emails have undeniable utility, large volumes of unsolicited email - spam email - have become a familiar universal problem. Spam email costs businesses and individuals in America billions of dollars every year. Fortunately, there are algorithmic solutions to this problem. In this paper, we analyzed an artificial intelligence that filters spam email. Specifically, we tested two implementations of the Bayesian model for email spam detection with the aim of choosing one of these.

In this project we compared two different implementations of the Bayesian model for email spam filters. Our focus is answering a two-part question: how do the multivariate and multinomial implementations of the Bayesian model compare and how does the best of the two perform against other programs available in the real world? Metrics tested for include speed and accuracy given multiple email datasets.

After that, we built an functional program based on the faster version of the Bayesian model. We will eventually compare the modified program to other email spam filters. The Gmail spam filter would be a good benchmark, so we used that. Our concern is the programs' rate of successfully identifying spam emails as well as how quickly they get that done. We may use other metrics as well but the most critical for us are accuracy and speed - the efficiency of the programs.

Between 2003 and 2015, the volume of unsolicited emails fell to a 12-year low. Volumes of spam email fell below 50%, according to a report from the Kaspersky lab. The anti-virus software developer company Symantec also reported the halving of spam email by 46.4%. This reduction in spam traffic was largely due to the decline in the number of major botnets responsible for sending billions of spam emails. Kaspersky detected between 3 and 6 million malicious spam emails. In the same year, the spam email volume was reported to be constant. Kaspersky also noticed an escalation in the number of spam emails that had ransomware, malware, malicious macros and JavaScript in the last month of 2015. That trend continued to the point where, by March 2016, that number had quadrupled. In March 2016, the volume of spam emails discovered by Kaspersky Lab was 22,890,956. In 2016, spam messages accounted for 56.87% of email traffic worldwide and the most prevalent types of spam emails related to healthcare and dating. On a global scale, spam results in the unproductive use of resources on Simple Mail Transfer Protocol (SMTP) servers. These servers have to process a massive amount of unsolicited emails.[2]

1.1 Available Methods of classifying spam

Many big technology company right now are using different Machine Learning method to implement then email spam filtering. One of the most famous company is Google with their 99% accurate spam filtering. Based on [2], "Statistics from Google revealed that between 50-70 percent of emails that Gmail receives are unsolicited mail. Google's detection models have also incorporated tools called Google Safe Browsing for identifying websites that have malicious URLs."

Over the years, academicians and researchers have proposed numerous email spam classification techniques which have been successful in classifying data into groups. Some of the methods include probabilistic, decision tree, artificial immune system, support vector machine (SVM),artificial neural networks (ANN), and case-based technique. In literature, it has been shown that it is possible to use these classification methods for spam mail filtering by using content-based filtering techniques. The classifiers will identify certain features, usually keywords frequently utilized in spam emails in order to identify the spam. The frequency of each word in the emails ascertain the probabilities for each characteristic in the email, after which it is measured against the threshold value. Email messages that exceed the threshold value are classified as spam. ANN is a non-linear model that imitates biological neural network functions. The model consists of simple processing components called neurons and carries out its computational operations by processing information [2]. Neural networks have been used to classify unwanted emails as spam by making use of content-based filtering. These methods decide the properties by either computing the rate of occurrence of keywords or patterns in the email messages. Neural Network algorithms that are used in email filtering attain average classification performance. Some of the most popular spam email classification algorithms are Multilayer Perceptron Neural Net-works (MLPNNs) and Radial Base Function Neural Networks (RBFNN).Researchers used MLPNN as a classifier for spam filtering but not many of them used RBFNN for classification [2].

1.2 Why Naive Bayesian?

A Naive Bayes (NB) classifier simply apply Bayes' theorem on the classification of each email. NB is based on a strong assumption that the words included in the email are independent of each other

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. Andre, S. Koenig (eds.), July 2018, Stockholm, Sweden

https://doi.org/doi

[2]. NB for email spam filtering because of its simplicity, ease of implementation and quick convergence compared to conditional models such as logistic regression. It Needs less training data. Naive Bayes algorithm is very scalable. No bottleneck is created by increase in the number of predictors and discrete unit of information. NB can be used to solve both classification problems involving two or more classes. It can be used to make forecasting that is subject to or involving probability variation. The different versions of the Naive Bayesian can effectively manage continuous and discrete data (multinomial, multivariate). NB algorithms are not susceptible to irrelevant features. Naive Bayes algorithm is predominantly famous in business-related and open-source spam filters [2]. This is because apart from the ad vantages listed above, NB needs little training time or speedy assessment to detect and filter email spam. NB filters need training that can be offered by the earlier set of non-spam and spam messages [2]. It keeps the record of the changes that take place in each word that occurs in legitimate, illegitimate messages, and in both. NB can be applied to spam messages in diverse datasets having different features and attributes. And this is what we did for our project — we used an SMS and an email data.

2 RELATED WORK

Filtering spam emails has always been a critical issue to resolve, and many academic institutions and organizations have attempted to develop or use existing classification algorithms to filter spam emails. Such algorithms are support vector machines, rules-based filtering, challenge filtering, and case-based filtering. However, much research has been done on Naive Bayes Classifiers due to the algorithms' simplicity and high accuracy. These qualities make Naive Bayes Classifiers the best option for filtering out spam emails. For instance, in a 2017 research on spam filtering, the researchers wanted to implement Naive Bayes for spam filtering due to its simplicity[1]. The researchers first noted the frequency of each word in their data of emails, then used Bayes' Theorem to determine the probability of an email being spam or not. After their method was finished, they tested it and found that the accuracy of their method was over 80% for two distinct test databases.[6] Similar results were also found in a study done at Jadavpur University in India. The research developed a Naive Bayesian Classifier to filter out spam emails which resulted in an accuracy of 94.16%. [5]

2.1 Probabilistic Approaches

While there are many classical approaches to a classification problem, the probabilistic approach is still one of the best current approaches balancing between the accuracy and complexity of an algorithm. For instance, if we want to classify a text to be either spam or not spam, a simple logistic regression will be a good choice in the case that we want to prioritized the time and lower the complexity level; however, the method will not capture everything that will be presented in the data set and usually result in an atrocious outcome. On the other hand, a deep learning approach or neural network would elevate the accuracy of the outcome, but at the same time, it will be extremely time-consuming.

That is why probabilistic machine learning has increased in classifications problems nowadays. Instead of trying to capture everything presented in the dataset, the method will calculate the uncertainty of such a prediction and decision. As Zoubin Ghahraman mentioned, "Fortunately, the probabilistic approach to modelling is conceptually very simple: probability distributions are used to represent all the uncertain unobserved quantities in a model (including structural, parametric and noise-related) and how they relate to the data. Then the basic rules of probability theory are used to infer the unobserved quantities given the observed data. Learning from data occurs through the transformation of the prior probability distributions (defined before observing the data), into posterior distributions (after observing data)" [3]. This is called Bayesian machine learning.

2.2 Bayesian Machine Learning

The Bayesian classification exemplifies a supervised learning technique and at the same time a statistical technique for classification. It acts as a fundamental probabilistic model and let us seize ambiguity about the model in an ethical way by influencing the probabilities of the results. It is used to provide solution to analytical and predictive problems. Bayesian classification is named after Thomas Bayes (1702–1761), who proposed the algorithm. The classification offers practical learning algorithms and previous knowledge and experimental data can be merged.Bayesian Classification offers a beneficial viewpoint for comprehending and appraising several learning algorithms. It computes exact likelihoods for postulation and it is robust to noise in input data. A Naive Bayes classifier is a straightforward probabilistic classifier that is founded on Bayes theorem with sound assumptions that are independent in nature.

Bayesian Machine Learning is a probabilistic machine learning approach. As Zoubin Ghahraman mentioned, "Fortunately, the probabilistic approach to modelling is conceptually very simple: probability distributions are used to represent all the uncertain unobserved quantities in a model (including structural, parametric and noise-related) and how they relate to the data. Then the basic rules of probability theory are used to infer the unobserved quantities given the observed data. Learning from data occurs through the transformation of the prior probability distributions (defined before observing the data), into posterior distributions (after observing data)" [3]. This is how Zoubin defined Bayesian.

The main question here is how does the algorithm work? The core of Bayesian Machine Learning is the Bayes' Theorem (rules) itself. In general, The formula for Bayes rules could be presented as follow

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

The purpose of Bayes' theorem is to calculates the probability of an event A occurring given that event B has already occurred. To further apply this theorem into classification problem, Zoubin Ghahraman presented the following formula

$$P(\theta|D, m) = \frac{P(D|\theta, m)P(\theta|m)}{P(D|m)} [3]$$

From Zoubin's note, we know that θ is the unknown parameters of the dataset, D is the data set and m contains all the class we

are considered for the model. We also obtain the information that $P(D|\theta, m)$ represents the chance of that θ is inside m, while $P(\theta|m)$ is the prior probability and $P(\theta|D, m)$ is the posterior probability given data set D [3].

2.3 Naive Bayesian Approach

Naive Bayes is a subfield of Bayesian Machine Learning. The "naive"-ness of this algorithm stems from the assumption that these features are independent from each other and the assumption of the distribution of the data. Most Naive Bayes classifiers differ in their assumption of the distribution and how they identify features in the data. This simple formula may be used for classification. However, most implementations of Naive Bayes take the form of a more efficient version such as the Bernoulli Multivariate and the Multinomial Naive Bayes.

2.4 Bernoulli Multivariate Model

A document sample (email in this case) e_i is presented as a vector [vector form] $\vec{v_j}$ where each entry represents a word where w_{jk} = 0 or 1 depending on if that word appears in the document (return 1) or not (return 0).

We assume that one documents classification will not affect whether or not another document's classification. At the same time, we also assume that the appearance of one word will not affect the other words' appearance or in other word, their appearance are independent. In each case, we define s to denote the spam class. Furthermore, $\bar{s}$ will denote the ham class or "not spam" class in this situation.

The pseudocode below is the adapted version of what was presented on the Natural Language Processing webpage of Stanford [4]. Instead of classifying multiple classes like what they demonstrate inside the e-book, we presents the pseudo just to classify only two class, "spam" and "not spam".

Algorithm 1: Bernouli Multivariate event model

```
1  Function Training(document D):
2      W ← ExtractVocabulary(D)
3      NumberOfSamples ←
         TotalNumberOfDocumentsForTraining
4      for c ∈ {s, s̄} do
5          NumberOfDocs_c ← Group(c, D)
6          ratiosOfDocs[c] ← NumberOfDocs_c / NumberOfSamples
7          for each t ∈ W do
8              NumberOfDocs_ct ←
                 DocsInClassHasTerm(D, c, t)
9              P[t][c] = (NumberOfDocs_ct + 1) / (NumberOfDocs_c + 2)
10         end
11     end
12 Function ApplyBernoulli({s, s̄}, W, ratiosOfDocs[{s, s̄}], P,
     an email e):
13     WordsInEmail ← ExtractWords(e)
14     PredictScore ← 1
15     for class ∈ {s, s̄} do
16         PredictScore ←
             PredictScore × ratiosOfDocs[class]
17         for t ∈ WordsInEmail do
18             if t ∈ W then
19                 PredictScore[class] ←
                     PredictScore × P[t][class]
20             end
21             if t ∉ W then
22                 PredictScore[class] ←
                     PredictScore × (1 − P[t][class])
23             end
24         end
25     end
26     if PredictScore[s] ≥ PredictScore[s̄] then
27         return s
28     else
29         return s̄
30     end
```

To train a model using Bernoulli Model, the formula in line 9 is the core of the whole training process. In this formula, we used Laplace Smoothing (the reason why we add 1 and 2 at the end of both denominator and divisors) to ensure that we will not get a probability of either 0 or 1 which may affect the *PredictScore* in the *ApplyBernoulli*. In the apply state, we start the process by extract words from the email and put it inside a vector (no replicas). After that, we move on by comparing each entries that was stored in *WordsInEmail* with the vocabulary inside the set of vocabulary of the training set. Line 18-23 of the pseudocode is how the Bernoulli model was applied. The return class will be based on the comparisons of the two *PredictScore*'s.

2.4.1 Advantage of Multivariate Model. Bernoulli Multivariate model is a really fast algorithms. At the same time, Bernoulli Multivariate Model also capable of solving multi-class classification problem. It is especially useful for categorical input.

2.4.2 Disadvantage of Multivariate Model. This disadvantage of Multivariate model is that it have one huge assumptions about the predictors (or features) are mutually independent. The assumption is actually not really realistic because in English, there are a lot of common terms and idioms that we cannot separate word by word since the meaning will be change. This fact create a boundary for Multivariate Model to be actually implied in real-world use cases. At the same time, since the Multivariate model only keep track of the presentation of the word, sometimes, the algorithms assign zero probability to a features whose words was not in the data set. That why there need to be some smoothing methods like Laplace smoothing to assure that we will not get any probability of 1 or 0.

2.5 Multinomial Model

Multinomial Bayes is a powerful Bayesian machine learning algorithm popularly used in text classification. Unlike other classifiers, it assumes that the position of a word does not affect whether or not the email is spam. The latter assumption allows us to implement the bag of words model for determining the probabilities we need prior to calculating the classification probability. With the bag

of words model, Multinomial Bayes takes into consideration the occurrences of each word rather than just taking into account what words appear. Similar to other Naive Bayes classifiers, it assumes that all words are mutually independent from each other, and takes advantage of Bayes' Formula as its basis for classification. In our case, we are computing the probability of whether or not an email is spam given the content of the email. To find that probability, we evaluate the following equation,

$$P(S|w_1, w_2, w_3, \ldots, w_n) = P(S)\frac{P(w_1, w_2, w_3, \ldots, w_n|S)}{P(w_1, w_2, w_3, \ldots, w_n)}$$

Where S is the event when the email is a spam, and $w_1, w_2, w_3, \ldots w_n$ are the words that appeared in the email with repeats with n as the number of words in the email. We'll now define the other probabilities in the equation. The probability of an email being spam is defined as P(S) and calculated using

$$P(S) = \frac{s}{N}$$

Where s is the number of spam emails in the training set and N is the number of emails in the training set. To define $P(w_1, w_2, w_3, \ldots w_n)$, we recall that Multinomial Bayes assumes all words are mutually independent. Hence, $P(w_1, w_2, w_3, \ldots, w_n) = 1$. The same assumption can be used to define

$$P(w_1, w_2, w_3, \ldots, w_n|S) = P(w1|S)P(w2|S)P(w3|S)\ldots P(w_n|S)$$

$$P(w_i|S) = \frac{\text{Number of times } w_i \text{ appears in spam emails}}{\text{Number of words in spam emails}}$$

Now we can derive the equation

$$P(S|w_1, w_2, w_3, \ldots, w_n) = P(S)\prod_{i=0}^{n} P(w_i)$$

2.5.1 Advantage of Multinomial Model. It is easy to implement as you only have to calculate probability. You can use this algorithm on both continuous and discrete data. It is simple and can be used for predicting real-time applications.

It is highly scalable and can easily handle large datasets.

2.5.2 Disadvantage of Multinomial Model. The prediction accuracy of this algorithm is lower than the other probability algorithms.

It is not suitable for regression. Naive Bayes algorithm is only used for textual data classification and cannot be used to predict numeric values.

2.6 Time complexity and Space Complexity of Naive Bayesian

Multivariate Model and Multinomial Model are using different estimation methods. While the Multivariate Model trying to check the present of a term t inside the document class $c = \{s, \bar{s}\}$. However, given the number of training examples, N and the dimensionality of the features, d, Naive Bayesian classifiers are $O(nd)$. All the algorithm does is compute either the presentation or frequency of each feature value, d_i in each class. That's a pretty decent time complexity. The space complexity is similar.

3 PERFORMANCE TESTING

3.1 Data Preparation

In order to use the data with our code, reconfiguration of the dataset was done. The dataset from SpamAssassin that we aquired contains a column of the emails and another column identifying the emails as spam or not. Fortunately, this SpamAssassin dataset works with our code. We also have another dataset that contains SMS messages with a classification on whether or not the message is spam. We wanted to test this method with a SMS dataset, but first we needed to reformat it to work with the code for the multivariate method. The SMS dataset has a column with its messages and another column filled with 1's and 0's, indicating whether or not a message was spam. We had to change the 1 value to "spam" and the 0 value to "ham." We manually changed the SMS dataset using spreadsheets.

3.2 Bernoulli Multivariate

3.2.1 Testing. Python was chosen as the language for implementation due to its simplicity and vast amount of libraries. The machine learning library scikit-learn was utilized for this project because of its built-in function implementing the bernoulli multivariate model. Then, the bernoulli function called BernoulliNB() was executed with 70 percent of the dataset as the training set and the remaining dataset for testing. The accuracy score was then calculated using by testing the model with the test dataset and checking the what proportion of the test emails were correctly classified. The program also prints out the time it takes for the program to finish. This process was performed on both datasets and both accuracy scores were calculated and compared.

3.2.2 Testing Summary. After testing, we found that the accuracy for classifying spam emails for the SMS dataset is 97.78%. The runtime for this program was 2.5 seconds. When running for the SpamAssassin dataset we got an accuracy score of 88.22%. The runtime recorded is 10.77 seconds.

3.3 Multinomial Model

3.3.1 Testing. Python was also chosen as the language for running this implementation. The machine learning library scikit-learn also has a function that implements the multivariate method. A similar process to the multivariate method, as explained above, for testing was performed for the multinomial method. Again, the datasets were loaded in using the pandas library. The function BernoulliNB was executed with the same proportion. The training set consist of 70% of the dataset, and the testing dataset is the remaining 30%. The accuracy score and runtime for both datasets were computed and recorded.

3.3.2 Testing Summary. After testing, we found that the accuracy for classifying spam text for the SMS dataset is 98.86% and 99.11% for the SpamAssassin dataset. When testing the SpamAssassin dataset using the multinomial function from scikit-learn, the program took 9.29 seconds to complete. Not surprisingly, the test for the SMS dataset took 1.96 seconds to complete since the SMS dataset is smaller than the SpamAssassin dataset.

3.4 Comparison

Our results, as shown in Figure 1, show that the multinomial method does better than the multivariate algorithm. Not only is the accuracy score better, the runtime that was recorded means that multinomial runs slightly faster than multivariate.

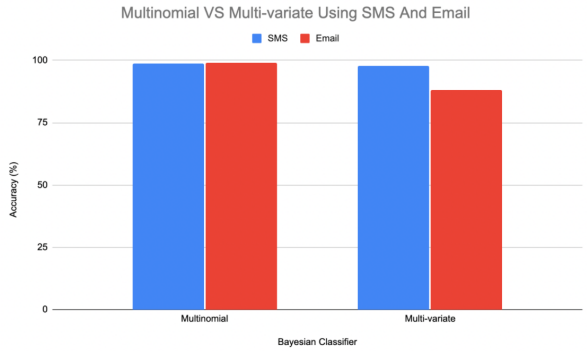

Figure 1: Accuracy score for each method

After testing both of the model on both of the data set and analyze all the aspects, there are some fascinating conclusion that we drew out The first thing to point is that the nature of the two data set are different. In general, SMS content tend to be more informal as well as shorter than the Email data set. Multinomial Event Model actually worked well on the predicting both SMS and Email. The fact that Multinomial Even Model can have a exceptionally high accuracy in Email data set implies that Multinomial can works well despite the nature of the data set. On the other hand, Multivariate Model shown lower accuracy score when testing on the Email data set and tend to works better with SMS data set. There is one obvious reason is that since the Multivariate model does not keep track of the frequency, sometimes, if a set of words has too high impact score, the algorithm will generally believe that whenever this set of words appear, the email will be classified as that category.

4 OUR IMPLEMENTATION

Multinomial is an implementation that considers the frequency of the each words. However, what would happen if we only consider the words that appears the most by putting a threshold that prevents all probabilities lower than the threshold from being apart of the classification calculation. We denote this as the following equation:

$$P(S|w_1, w_2, w_3, \ldots, w_n) = P(S) \prod_{i=0}^{n} P(w_i)$$

$$\text{where } P(w_i) \geq threshold$$

For the foundation of our implementation, we are adapting code created by Alex Olteau on the KDnugget blog post. Olteau's goal was to implement the multinomial model from scratch. This means that there was no use of libraries that contain a function for multinomial classification. Olteau's program also used the SMS dataset for their testing with an accuracy score of 98.74%, and so we used the SMS dataset for testing our implementation to Olteau's code.

Due to the large amount of words in the SMS dataset, $P(W|S)$ are very small values, and so a small thresholds are needed to get non-trivial results. For example, if we were to have a threshold of 0.5, no sample in the training dataset will be filtered out since all frequencies for each word aren't bigger then 0.5. In fact, the word that appears the most is "to", with only a frequency of 0.023. This issue was solved by setting the threshold as a value of a specifically chosen word's frequency. First, frequency of each word was sorted in ascending order. Then, a position was chosen of our choice. The position that we chose, was more or less a guess. Further study in choosing the position will need to be done. However, this paper does not cover such problem. After choosing the position, the frequency corresponding to the word in the chosen position is used as the threshold. Finally, with the threshold found, we prevent all words with a frequency lower than the threshold from being calculated into the multinomial equation. A for loop was added to find the results for multiple thresholds. The first iteration begins with the threshold value that we founded in our chosen position. For all following iterations, threshold increases by 0.0001.

4.1 Results from our implementation

As shown in Figure 2, the accuracy of multinomial model decreases as the threshold increases. The position that we choose was 7000 out of the 7,783 unique words found in the dataset. This chosen position give us an initial threshold is 0.0001741174422 with an accuracy score of 18.671%. With this threshold, we filtered out 6,948 probabilities from our multinomial calculation. When the threshold increases, the accuracy score decreases more and more. However, this makes sense because as the threshold gets higher, we filter more and more word frequencies.

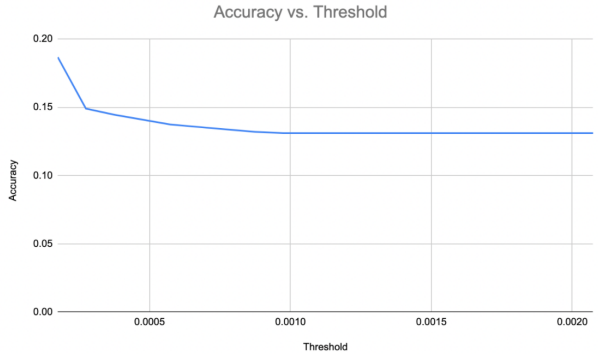

Figure 2: Accuracy score for different thresholds

As mentioned by Stanford about using Feature Extraction could actually improve the performance of the algorithms in general [4]. However, in our case, the result seems to be dramatically decreased whenever we take out features. There are two possibles explanation. The first one is because the data set is always clean and not contain any words that will not impact the accuracy score. At the same time, the problem of having too many features that have almost the same prior probability (or ratio) is also impact our implementation. The second way of explaining this result is because the threshold

value is incorrect. As far as we know, we are still unable to figure out a formula to find the correct threshold that could actually result in a better accuracy.

5 UNRESOLVED PROBLEM - FUTURE WORK

The implementation is working based on the idea of feature extraction [4]. We have tested the implementation using multiple data set as well as multiple threshold but there are some problems need to be resolved.

5.1 Problem of picking threshold

Although we have tested with multiple thresholds, we are still wondering how could we pick a threshold that could boost up the performance of the multinomial event model. The main issue of picking a threshold is that in a large data set, especially the training data set, there will be a lot of features (or words) that was stored. This creates the problem having too many features with probability close to 0. As a consequence, when we increase the threshold with a small amount 10^{-5}, the accuracy dramatically decreases.

Based on the idea of feature extraction from Stanford [4], removing some of the features out of the list could actually resolve the problem of high biases - high variance and improve the general performance of the Multinomial model. However, when we tested it, the accuracy tended to decrease over time. We believe this largely because of the way we pick the threshold.

5.2 Proposed approach

As we point out, the problem of picking the correct threshold is our issues in this research. As of now, we have figured there are some alternatives ways to deal with this issues.

5.2.1 Using a portion of the data. In our approach, we filter out the probability as well as the features and store them inside a dictionary using Python. After that, we sorted the dictionary by the value of the key. Instead of picking threshold, one of the possible alternative way is to pick a percentage, denoted x, between 0% and 100%. Then, we extract x% of the dictionary and used it for training. This method probably will ease out the result of picking threshold since the number we are working now is larger and it is easier to control.

5.2.2 Using Information Gain to extract key features. In this case, we actually find out the key features that give us the most information. To define information gain, we would want to say that it is a measurement of the frequency of a word in a particular class. For example, in spam email, the words "sales" or "sale" appeared a lot. The motivation of using this approach is to actually remove some of the useless like pronouns, conjunctions, etc.

One of the famous ways to calculate this is the chi-square. In this method, we sort the words by calculating the score of the whole list of vocabulary using the chi-square function and take the top x of the list. We then stored the whole list into the set and use that for our training so that the algorithm will only pick up the words from the improved list.

5.2.3 Making a list of common words and remove from the vocabulary. While analysing the data set, we see that there are a lot of common words such as pronouns, conjunctions, etc. that actually not affect the classification but they are still presented in the training set. One of proposed future work is to actually remove these words. This actions might actually improve the impact score of the key or important words and actually eradicate some of the high impact scores that these vocabulary have.

5.2.4 Case sensitivity. Most of our implementation as well as online implementation right now have the independent features assumption. This reduces the case sensitivity and how the program actually works in the real-world case. In the future, we would like to implement a Naive Bayesian model with some modification in the assumption. For instance, instead of using a set of single words, we would like to implement our program in a way that it would take into the concern about idioms and expression.

5.2.5 More options for Naive Bayes. Although we have thoroughly worked and analyzed the two most famous Naive Bayesian techniques, we also want to explore some of the different methods. As of right now, we also want to know about Gaussian Naive Bayes, Complement Naive Bayes, Categorical Naive Bayes, and Out-of-core naive Bayes model fitting. At the same time, we also want to explore the spam filtering using different approach like neural networking, random forest, etc.

6 FINAL CONCLUSION

Probabilistic Machine Learning, especially Bayesian Machine Learning is an active research area. The application of the methods is used widely in a variety of applications including text classification. In our research, we figure out that Naive Bayesian model will especially efficient when it comes to the problem of uncertainty. For instance, in email classification problem, usually, the predict score of both spam and not-spam email will be compare before the algorithms give out the final result. The uncertainty is playing a major role in how Bernoulli Multivariate Model and Multinomial Model since the core of these two algorithms is just about the transformation of prior probability and uncertainty to posterior probability and uncertainty. However, the main issues with both the methods that we are analyzing is it have a really low case sensitivity. The main reasons is that there are two assumptions that are not realistic and actually could impact how it works in the real-world. At the same time, the real-world use case may be significantly different because the data set will be not as clean as our testing data set.

Our work with Bayesian classifiers in this research is the evidence of how accurate the tools could be. Both Naive Bayes implementations (Multivariate and Multinomial) scored at least 97% accuracy with the SMS dataset which had at least 7000 messages. When the email dataset was used, the Multivariate implementation's accuracy was 88% while the Multinomial was 98%. The higher accuracy score for the multinomial method encouraged us to further improve it. In our attempt to improve the Naive Bayes Multinomial classifier by using threshold, we found that that approach drastically worsens results. Even though the threshold method uses the most frequent feature instances, because these account for a very small percentage of the feature set, the overall accuracy of the algorithm plummets.

REFERENCES

[1] Hanif Bhuiyan, Akm Ashiquzzaman, Tamanna Islam Juthi, Suzit Biswas, and Jinat Ara. 2018. A Survey of Existing E-Mail Spam Filtering Methods Considering Machine Learning Techniques. *Global Journal of Computer Science and Technology; C Software and Data Engineering* 18 (2018).

[2] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi EmmanuelAjibuwa. 2019. Machine learning for email spam filtering: review, approaches and open-research problems. *ScienceDirect* (2019). https://doi.org/10.1016/j.heliyon.2019.e01802

[3] Zoubin Ghahramani. 2015. Probabilistic machine learning and artificial intelligence. *Nature* (2015).

[4] Christopher D. Manning, Prabhakar Raghavan, and Himrich Schutlze. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

[5] S. Roy, A. Patra, S.Sau, K.Mandal, , and S. Kunar. 2013. An Efficient Spam Filtering Technique for Email Account. *American Journal of Engineering Research* 2 (2013), 63–73.

[6] Nurul Fitriah Rusland, Norfaradilla Wahid1, Shahreen Kasim, and Hanayanti Hafit. 2017. Analysis of Naïve Bayes Algorithm for Email Spam Filtering across Multiple Datasets. *International Research and Innovation Summit* (2017). https://doi.org/10.1088/1757-899X/226/1/012091