Machine Learning Model Generation for Fraudulent Email Detection

CIS 5930 Data Mining

Zhixi Lin, Linwei Jiang, Haiping Yuan

I. Abstract

This paper aims to discuss the results we obtained with our fraudulent email detection model trained using several different machine learning algorithms as well as different approaches we took to improve the accuracy and performance of our model. In this paper, we will be referring to normal/non-fraudulent emails as type 0, commercial emails as type 1, and fraudulent/phishing emails as type 2. Initially, we trained our model using the Support Vector Machine (SVM) classifier with Radial Basis Function (RBF) kernel and obtained an average f-score of 65.9%. Then, we used Logistic Regression (LR) to train our model and obtained an average F-score of 77%. Since the average accuracy for the two variations of the model were still way below our expectations, we decided to make use of various optimization methods on our dataset and data pre-processing as well as introduce more modeling algorithms in an attempt to find the one with the best results. In the end, the average f-score after the optimization (use of oversampling methods and NLP) was 92.57% for SVM, 98.43% for LR, 98.74% for Random Forest (RF), and around 79.49% to 93.17% for the different variations of the Naive Bayes classifier.

II. Introduction

1. Intuition:

In modern society, Email is considered one of the most essential means of communication, especially for formal occasions. However, as there are no restrictions to sending emails if the email address is already known, spam emails are prone to exist with the purpose of mass-sending useless information or commercial advertisements, or in certain cases, with the malicious intention of committing fraudulent activities or phishing. As most of our modern spam filters classified all these types of emails as a single category of spam, we believe such categorization is insufficient as people may want to take different actions against the former harmless spam and the latter harmful spam, with the potential damage that can be caused by the latter. Thus, this has formed the purpose of our project of classifying emails as Type 0, normal emails, Type 1, harmless spam emails, and Type 2, harmful spam emails that are fraudulent or phishing.

2. Approaches:

With the purpose of classifying emails, we will be using Machine Learning models to do so. Specifically, we will be approaching this problem by (1) selecting appropriate input data for training, (2) using appropriate pre-processing methodologies to effectively convert the text into trainable features, and (3) using a variety of models to train the input data to obtain the most effective model.

The first part of selecting an appropriate dataset is quite difficult as there are few datasets that contain the classification we wanted, and most of the available datasets consist of only classification between normal email and spam email. After two weeks of searching, we are only able to allocate a single small dataset that contains only 159 rows of data fully satisfying our purpose. Thus, to address this issue, we have decided to label the spam category from the spam/ham dataset into Type 1 and Type 2 manually, and use techniques to deal with the huge imbalance between classes.

To tackle the second part, after researching for quite some time, we have decided to first use a simple count vectorization method to convert the input text into a vector of word occurrence frequency. However, by only applying this single pre-processing step, the outcome is not desirable, thus we have eventually decided on using Natural Language Processing (NLP). This has proven to be extremely effective, but the sheer time spent on processing the data is extremely long. Thus, to address this issue, we have implemented a simpler version of NLP, which is a lot faster with a similar performance as NLP.

For the third part, as we are wishing to obtain a more comprehensive result, we have used four models altogether. First would be the Support Vector Machine with a gaussian kernel, as we have a lot more experience with this model throughout our study. The second one would be logistic regression, where we believe this model will provide us with a fairly comprehensive result and the model itself is quite simple. The third one would be the Random Forest algorithm which inherits the advantage of the decision tree. This is a collection of decision trees where each tree makes its own independent prediction, which can then be averaged to obtain the final result. Due to this, the predictions made from the Random Forest algorithm generally have high accuracy. Lastly, we have picked the Naive Bayes model because it offers quick and efficient classification algorithms that allow us to obtain a "baseline" accuracy that we can use to compare with other algorithms used in this project to determine which one is best suited for our model.

III. Literature Survey

In the research of Akinyelu & Adewumi, they deemed the best fraud detection system should be dynamic as fraudsters are always updating their mode of phishing to escape discovery. Since the traditional way of filtering emails was hard to update, machine learning algorithms are recommended [1]. In the research, the author claimed that random forest is the best approach with 99.7% accuracy on the best eight feature classification [1].

Natural language processing (NLP) is applicable for data pre-processing with SVM models [2]. Using NLP and machine learning is an effective way to catch phishing emails [3]. In Saleem's research of classifying non-phishing and phishing emails after the data process, the random forest got 98% accuracy, while the decision tree and Naïve Bayesian in gaussian got the highest accuracy [3]. Ferdjouni considered the machine learning pipeline in NLP includes: text processing including tokenization, clean and normalization, SKLearn Pipeline, and Evaluation [4].

Naive Bayes can predict the possibility of future events based on previous events, making the Naive Bayes classifier more advantageous in word prediction and text processing [8]. Sanghi claimed that the critical part to increase the performance of machine learning is finding the features from words and transforming the text into numbers [9]. He applied the multinomial Naive Bayes algorithm into NLP processed features that enabled binary prediction for the text attribute.

IV. Methodology

1. SVM

A simple algorithm that's used for classification and regression. This algorithm uses an N-dimensional space to find a hyperplane that distinctly classifies data points into its own class. Depending on the number of features, the number of dimensions of the hyperplane will also increase [5]. Through the variety of kernels that we can choose from, we will be using the Gaussian Kernel due to the number of features of our model is quite large and can fluctuate, making the type of relation between features extremely complicated, thus it would not be suitable to use either linear kernel or polynomial kernel which can lead to underfitting or overfitting. As the Gaussian Kernel uses euclidian distance to identify margin, making it quite flexible and could potentially perform well in our circumstances.

2. Logistic regression

Logistic regression is a statistical method that analyzes a dataset to determine whether or not there exist independent variables that determine an outcome. It offers a way to deal with multi-class problems through its multinomial variant that deals with three or more potential types. This is perfect for our model because we have three types of email to classify. Moreover, logistic regression has the advantages of being easy to implement, interpret, and fast.

3. Random Forest

Random forest is a supervised learning method that ensemble multiple decision trees to increase the score of prediction. Decision tree is a tree-shaped model with leaves, nodes, and branches. Node stands for the class feature of data and the features split into binary or multiple classes like tree node splitting tree branches. Before the data points are categorized into a label (leaves), nodes in different features will keep branching until leaves generate. [6]Therefore, a random forest, as same as a decision tree, conducts classification or regression by following a set of if/else conditions. To overcome the issue of random sampling in the decision tree algorithm, the random forest algorithm uses the out-of-bagging method to achieve higher accuracy.

4. Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the naive assumption of conditional independence between every pair of features given the values of the class variable. For our model, we employed four different Naive Bayes classifiers: Gaussian Naive Bayes, Multinomial Naive Bayes, Complement Naive Bayes, and Bernoulli Naive Bayes. Depending on the different types of Naive Bayes classifiers, the assumption of the distribution of the conditional independence differs causing the resulting score to also differ. In addition, Naive Bayes classification algorithms are fast, efficient, and easy to use. As a result of these characteristics, we decided that Naive Bayes would serve as a good classification algorithm to help us obtain the baseline accuracy range of our model. [7]

V. Implementation

1. Data gathering

With the purpose of classifying email into categories of Type 0, Type 1, and Type 2, we need to allocate datasets that contain these three categories or more detailed categories that can be merged into these three. However, as splitting spam emails into two types or more detailed classification is not a popular topic, there is rarely any dataset that contains the classification we wanted. After approximately two weeks, we are able to allocate a few datasets that do contain the classification we can utilize, but some of the emails contain contents that are either company-specific or for a specific setting such as casual conversation only, thus they are not representative that we can use. In the end, we are only able to allocate a dataset that contains 159 rows with 25% of normal email, classified as Type 0, 25% of Commercial Spam, classified as Type 1, and 50% of Fraud/Phishing emails, which can be classified as type 2. However, the amount of data is still way too small for a proper training session, thus we have decided to find some other dataset that contains phishing emails as our Type 2 emails to merge with the current dataset, and utilize the spam/ham dataset to use the ham emails as our Type 0. For the lack of Type 1 emails, we have manually labeled the spam emails in the Spam/Ham dataset for a total of 194 Type 1 emails.

2. Initial Implementation and Iteratively Optimization

   a. *Initial Test Run with Simple Gaussian SVM model*

To get started with our actual implementation step, we have decided to use a simple model and basic data processing methodology to see where we should head for the next step. For the pre-processing method, we have appended the subject with text and used the Count Vectorization mentioned in the introduction to convert our input text into a vector of word frequency as our features and convert the label into 0 to 2 corresponding to our Type 0 to 2. At this time, we only have the dataset with 159 rows, which is way too small. As for the training, we used a simple SVM model with Gaussian Kernel. We have gotten an accuracy score of 0.5208, and an F-score of 0.3917 serving as a more representative score as our dataset is uneven. This poor score is expected as the dataset is way too small, and the pre-processing of input data is very simple.

b. Dataset Size Enhancement

To increase the amount of phishing and commercial email available to use for our model, Irene went over the spam/ham dataset and labeled over 700 spam emails as either Type 1 or Type 2 emails. Then, we used the datasets available in our GitHub repository to train an SVM and LR model and obtained the average accuracy for the SVM model which was around 65% while the f-score was 66%, and the average accuracy for the LR model which was around 75% while the f-score was 77%.

c. NLP

At this point, we have identified another issue with our dataset, which is the use of capitalization. Our major source of Type 0 and some Type 1 belongs to a dataset that capitalized all the subjects and text, and as our CountVectorization is case sensitive, this will give a false factor to our model fitting. Thus, with removing this trait in mind, while improving the quality of features, we have decided to employ a Natural Language Processing methodology. By using the built-in functions provided by the python library Natural Language Toolkit (ntlk), we were able to tokenize each string within our email dataset by converting each string into its lowercase form, removing punctuation, and splitting them up into words. Then, we removed every stop word and normalized the data through lemmatization. Once the text was successfully processed, we used the CountVectorizer function from sklearn's feature_extraction.text library to convert the text we have into a matrix of token counts to allow our text data to be easily used by the machine learning algorithms we've chosen for classification [4]. After employing this NLP algorithm, we have gotten an f-score slightly less than before, which is expected. However, with this method, each test run was around 3-5 minutes long due to how time intensive the NLP process can take on larger datasets. As such, we decided to replace the NLP process with a more simplified version of the NLP that simply goes through each string within our dataset and converts each of them to lowercase as well as replacing the punctuations within each string with spaces. This allows us to speed up the process of training and running our model while maintaining similar or in some cases, better scores.

d. Logistic Regression

So far the only means to identify the effectiveness of our model is through our SVM modeling score, which is first insufficient to evaluate the data pre-processing progress, and second, an alternative model could identify the performance from a different perspective. Thus, we have added a logistic regression in addition to SVM. This has proven to be quite a good decision as we see a significant improvement over the F-score (on average 8%) compared to SVM.

e. Oversampling

When we finished data processing, we found out that we had a pretty big class imbalance between our datasets. We had around 3672 emails of Type 0, 194 emails of Type 1, and 693 emails of Type 2. As such, the imbalance classes would create a significant issue during classification for the minority class, such as Type 1, where the accuracy will certainly be lower than the other two classes. As such, we

decided to employ oversampling methods on our dataset to prevent such cases from arising. More specifically, we used random oversampling and expanded smaller data classes through random selection until their size was equal to the bigger data classes. With that, we will be able to obtain similar but more accurate accuracy scores from our model for every class. Of course, this came with the drawback of possibly over-emphasizing certain features from the minority class, and we have also added the option for users to decide whether to turn on the oversampling.

f. Adding More Models

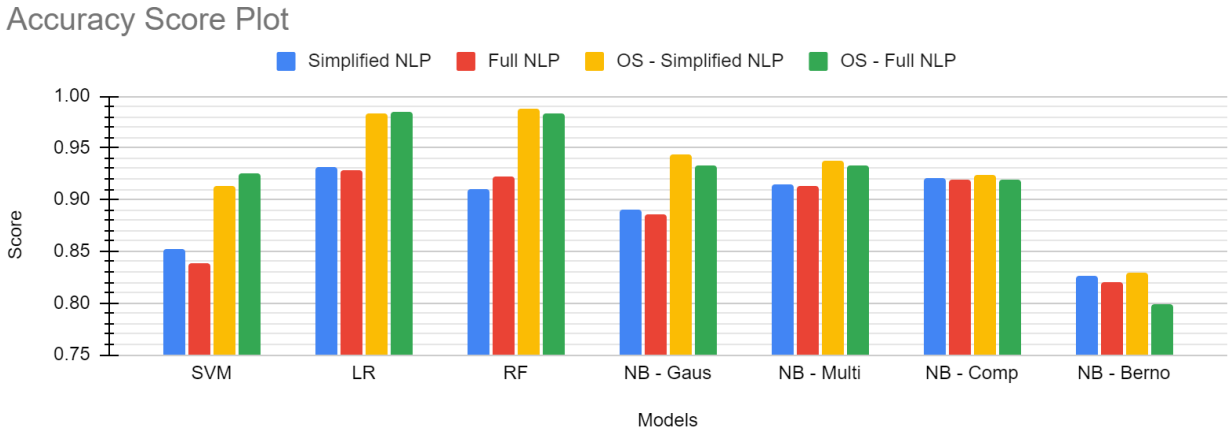
All of our previous implementation steps are focused on optimizing the dataset so that the algorithms we used to train our model will yield better results. Once we were done with optimizing the dataset, we wanted to include more data mining algorithms so that we can have more results to compare with and more easily determine which algorithm is the best fit for our model. In addition to LR and SVM which we were already using, we also added the Random Forest and Naive Bayes classification to help train our model. The reason we chose these two would be that Random Forest provides high accuracy compared to other classification models and it can handle big datasets well, and Naive Bayes offers many different variations we can use for classification and it is both a quick and efficient model that we can easily implement for results.

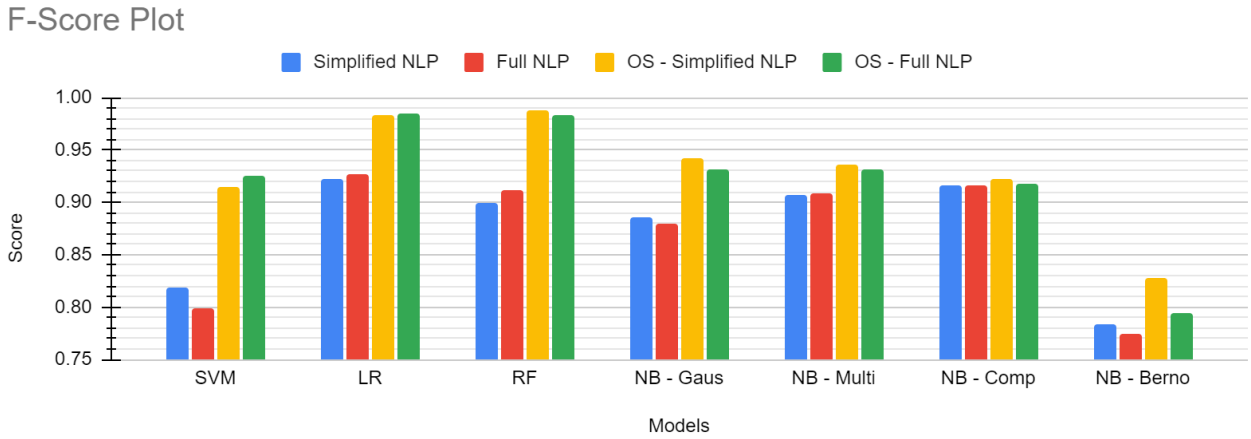
3. Final Model and Modularization

With all the above implementations, we believe we could start to obtain scores for different models and methods. Thus, we first started cleaning up our code while adding a sufficient amount of comments to describe the purpose of each section of the code. Then, we modularized every method and modeling algorithm into function calls and grouped a class to host different methods of our email classification, ranging from data intake, data pre-processing, and modeling. We were also able to build the main program with command line options control that we can input parameters when running the code, which allows the user to choose the NLP option, test size, random seed, oversampling feature, and what model to run. With all these complete, we are able to identify any issues with our code easily, and most importantly, run tests with ease.

VI. Final Results

The two plots below depict the average accuracy score and F-score of our model after it is trained by the different classification algorithms we chose to use for this project.





VII. Conclusion

We will start by analyzing the first plot for the Accuracy score in the Results section. As Accuracy is meaningless with unbalance between class rows, we will only concentrate on Oversampled (OS) results. Based on the scoring here, we see our top performers are LR and RF models with almost indistinguishable differences between Simplified NLP and Full NLP, while the scores are above 98% in both cases. Then we will move to our more representative graph of F-Score that gives equal weights for accuracy between classes, which is meaningful even for unbalanced classes. The first noticeable result would be for most models, the F-score differed drastically between regular and oversampled datasets, with NB - Complement as the only exception. As for the highest scores, the LR and RF will again outperform all other models, with their oversampled F-score of above 98%, making them quite a promising choice to distinguish emails. However, as oversampling came with the drawback of possibly emphasizing unimportant features on the smaller classes, we have made this feature optional for users to decide. If oversampling is not used, I would recommend using either LR model with its highest F-score in both Simplified NLP and Full NLP of above 92%, and NB-Comp with its consistent performance compared to the oversampled dataset and relatively good F-score of above 91%. Lastly, as for whether or

not to use Full NLP, we believe simplified NLP is preferable as overall there are only slight differences between the performance of Full NLP and Simplified NLP.

References

[1]Akinyelu Andronicus A, Adewumi Aderemi O. Classification of Phishing Email Using Random Forest Machine Learning Technique. Journal of Applied Mathematics. vol. 2014, Article ID 425731. 6 pages. 2014.DOI: https://doi.org/10.1155/2014/425731

[2]Kumar A, M C Jyotir, V. G. Díaz. A novel hybrid approach of SVM combined with NLP and probabilistic neural network for email phishing. International Journal of Electrical and Computer Engineering, 10(1). 2020. P 486-493. DOI: https://doi.org/10.11591/ijece.v10i1.pp486-493

[3]Babar M Saleem. The P-Fryer: Using Machine Learning and Classification to Effectively Detect Phishing Emails. Marymount University. Ann Arbor.2021. ProQuest, DOI: https://www.proquest.com/dissertations-theses/p-fryer-using-machine-learning-classification/docview/2572551978/se-2.

[4]Meriem Ferdjouni. Email Classification Using Natural Language Processing (NLP). Retrieved Nov. 20th 2022 from https://medium.datadriveninvestor.com/email-classification-using-natural-language-processing-nlp-ee3573bc79f7

[5] Ajay Yadav. SUPPORT VECTOR MACHINES(SVM). Towards Data Science. Oct. 2018. Retrieved Nov. 20th 2022 from https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

[6] Prince Yadav. Decision Tree in Machine Learning. Towards Data Science.Nov 13, 2018. Retrieved Nov. 26th 2022 from https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96

[7] 1.9. Naive Bayes. Scikit-learn. Retrieved Nov. 18th 2022 from https://scikit-learn.org/stable/modules/naive_bayes.html

[8]Email Spam Classifier Using Naive Bayes. Shubham Kumar Raj. May 17 2020. Retrieved Nov. 26th 2022 from https://medium.com/analytics-vidhya/email-spam-classifier-using-naive-bayes-a51b8c6290d4

[9] Applying Multinomial Naive Bayes to NLP Problems. Darshika Sanghi. GeeksforGeeks.Aug. 05 2021. Retrieved Nov. 26th 2022 from https://www.geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/