Project Milestone 3

ISQS 5342 Big Data Security

Texas Tech University

04/21/2024

Emily Spector

James Parker

Jonathan Busch

Part 1: Observations on Data Patterns

Before working on any artifacts for our data, we first decided to perform some exploratory data analysis on our two datasets. This is an important process in data analysis and offered insights into our data that proved useful for building machine learning models, performing text analysis, and performing input analysis.

Phishing URL Visualizations

We first wished to look at one of the most basic features of a URL, its length. As can be seen in Figure 1, the distributions of the length of phishing versus non-phishing URLs are slightly different.

A graph of a graph of a number of url

Description automatically generated with medium confidence

Figure 1: Histogram of UrlLength and CLASS\_LABEL

From the figure, we can see that phishing URLs (denoted by CLASS\_LABEL = 1 and the color red) are typically shorter than non-phishing URLs (denoted by CLASS\_LABEL = 0 and the color blue). This could be an important finding, as quickly checking the length of a URL could be a simple and effective way to check if a URL was phishing or not.

Next, due to the length and large number of features of the Phishing URL dataset, our goal was to decide which features were most important in determining if a URL was phishing or non-phishing. To do this, we found the correlation between all features and the outcome variable, CLASS\_LABEL. The top 5 highest correlated variables with CLASS\_LABEL were PctExtNullSelfRedirectHyperlinksRT, FrequentDomainNameMismatch, NumDash, SubmitInfoToEmail, PctNullSelfRedirectHyperlinks. We chose to visualize the last of these variables, PctNullSelfRedirectHyperlinks.

A graph of a graph

Description automatically generated with medium confidence

Figure 2: Histogram of PctNullSelfRedirectHyperlinks and CLASS\_LABEL

As can be seen in Figure 2, nearly all the non-phishing links had less than 50% of hyperlinks fields containing an empty value, a self-redirect value such as “#”, the URL of the current webpage, or some abnormal value such as “file://E:/”. In other words, non-phishing links typically have less than half of their hyperlinks missing or pointing to unusual places. On the other hand, nearly 25% of all phishing URLs *did* have an empty value, a self-redirect value, the URL of the current webpage, or some abnormal value.

Spam Email Visualizations

For the spam email dataset, we were most curious about which words appeared the most in each class. By identifying and analyzing the most used words in both spam and non-spam emails, email classification systems, such as the one we produced (discussed in part 2) can better understand the patterns and characteristics unique to spam/non-spam emails. Ultimately, the goal is to improve email security, reduce the risk of phishing attacks, and enhance user experience by ensuring that important messages reach their intended recipients while unwanted or malicious emails are appropriately flagged and managed.

Figures 3 and 4 below show the top 20 words in spam and non-spam emails, respectively. We learned three important things from doing this text visualization. The first was that the most common words were words like “the”, “to”, and “for” which lack discriminatory power and are not specific to any topic or category. This meant that when building our artifact for email classification we needed to perform some more advanced text analysis techniques. For example, we used TF-IDF (as seen in figure 6) or Term Frequency-Inverse Document Frequency. Using more advanced techniques would allow us to better predict if an email is spam or non-spam. Second, we learned that many of the most frequent terms/words were words containing information about the email metadata. For example, words like “ESMTP”, “Received:”, and the dates the email were sent appeared in every email. Again, this told us that we needed robust text analysis measures to ensure that our artifact was effective. Lastly, many of the words that were most frequent turned out to be HTML characters, such as “>”, “<”, or “id”. Once again, this just showed that much of the information that could be gathered from the text of the emails needed to be filtered through text analysis.

A graph of red bars

Description automatically generated

Figure 3: Bar Plot of Top-20 Words in Spam Emails

A graph of a number of emails

Description automatically generated

Figure 4: Bar Plot of Top-20 Words in Non-Spam Emails

A graph of a graph

Description automatically generated with medium confidence

Figure 5: Bar Plot of Top-20 Words in All Emails

Part 2: Artifact Selection

The three artifacts that we created include text analysis of the email spam dataset and classification plus input analysis of selected variable for the phishing link dataset.

Text analysis was determined to be the best artifact for the Spam Assassin dataset as generating a data frame of the frequency of word use in emails would allow us to determine if different word distributions are emblematic of determining whether an email is spam. Determining spam is valuable for protecting the industry of education as spammed emails are a major vector for unleashing different types of cyberattacks such as phishing links, ransomware, general malware, and even various scams. If spam emails can be properly identified, they can be removed from inboxes before they even reach the intended victim. The data required for this analysis is a series of emails labeled as either spam or not. For these purposes we found the Spam Assassin dataset which included emails with their given text and a label perfect for our purposes. We decided on using Python with the pandas and scikit-learn packages. The step for building this artifact begins with importing the necessary packages. Then the data is split into test and training sets. We then add the features necessary to generate our vectorizer for use in text analysis. One the vectorizer is created a model is then generated using the random search best estimator. Then we predict the test data based on the model generated from the training data. We then report the accuracy, precision and recall scores of the model. Finally, we use a for loop to obtain the most frequent words within the emails and their associated frequency count. The alternative approaches we are considering for our text mining included using R instead of Python. We also considered using other Python packages for text mining. We settled on using Scikit-learn due to its ease of use.

A screenshot of a computer program

Description automatically generatedFigure 6: Importing of Packages and Creation of Features

A screen shot of a computer program

Description automatically generatedFigure 7: Feature creation, Random Forest Classifier, and Model Fitting

For our other two artifacts, we used our phishing link dataset for classification and input analysis of phishing links. Classifying phishing links would benefit the industry of education as it would provide another vector for protecting the cybersecurity of students and faculty. With a classification artifact of phishing links any phishing links can be correctly identified, within an acceptable margin of error, when they are sent to university networks. Any content containing these identified phishing links can then be promptly removed thus protecting universities. Furthermore, input analysis of the distributions of key variables can further protect students and faculty as this would allow another form of analysis to determine whether a link is dangerous or not.

Building the classification artifact included these following steps. We began by determining which software/packages we wanted to use for this artifact. We decided on using Python with the pandas, numpy, matplotlib, and scikit-learn packages. The first step for building this artifact was to import the required packages mentioned above. We then read in the data using read\_csv and then split the data into independent and dependent variables. We chose to drop the “Httpsinhostname” variable from our independent variables as it only included the Boolean value 0 for the entire column. The data was then split into both training and test sets with a test size of 20% of the data. A random forest estimator was then used and fitted to the training data. Finally, the model was used to predict the classification of phishing links within the test data. We obtained an accuracy of 98.45% with our model. Further details on this artifact will be given within the next section. Alternative approaches to achieve this classification goal could include using a different type of model. For example, we considered using K-nearest-neighbor and a logistic regression for our model.

A screen shot of a computer program

Description automatically generatedFigure 8: All Code for Random Forest Classifier for Phishing Links

For the input analysis artifact, we chose to use R instead of Python due to our greater experience in performing input analysis within R. We also decided to focus our analysis on the URL length variable for both spam and non-spam rows of data. The first step is to read in the data using the read.csv function and import the fitdistrplus package. An acf and pacf graph was then created on the URL length variable. Various distributions were then checked to see how well they fit to the data. For the legit websites we found that the negative binomial distribution was best with a size of 6.25 and a mu of 72.75. A chi-squared test was then performed on the fitted distribution, and we failed to reject the null hypothesis that this data was a negative binomial. The phishing links best distribution was also negative binomial, but a chi-squared test returned a p-value of near zero rejecting the null hypothesis that the phishing links URL length distribution was in fact negative binomial. We did consider doing this analysis in Python but as stated before we chose to use R instead due to our experience with R for input analysis.

A screenshot of a computer program

Description automatically generated

Figure 9: Code for Input Analysis

Part 3: Artifact Presentation

We chose Option 1 to analyze our data, and as stated in Part 2, decided on three artifacts. First, we performed text analysis of the email spam dataset.

For the email spam dataset, we utilized python for text analysis. We split our dataset into training and testing sets, using the Stratified Shuffle Split technique, which ensures that the proportion of spam and ham mail remains consistent between both sets. We chose to reserve 20% of our data for testing and 80% for training. The following figure shows our data partitioning.

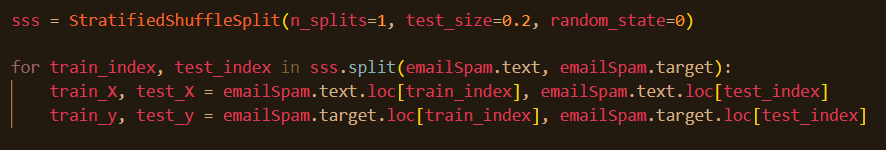


Figure 10: Training and Testing sets

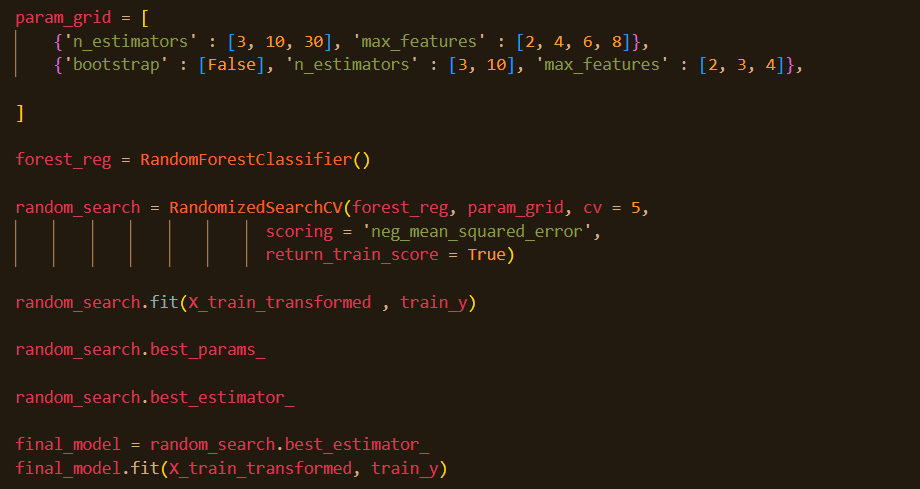


Figure 12: Model Training

Our final model was created by first tuning parameters using RandomizedSearchCV, which finds the optimal combination of parameters for our model which we then fit to our training data. This classification model is the last step to complete our text analysis. We evaluated our model based on accuracy, precision, recall, and a confusion matrix.

|  |  |
| --- | --- |
| **Evaluation of Text Analysis** | |
| **Method** | **Score** |
| Accuracy | 0.99 |
| Precision | 0.99 |
| Recall | 0.99 |

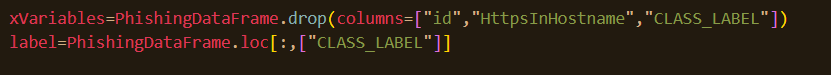
Table 1: Text Analysis Evaluation Scores

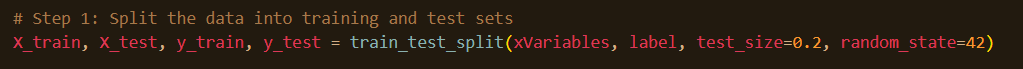
|  |  |
| --- | --- |
| 779 | 2 |
| 9 | 370 |

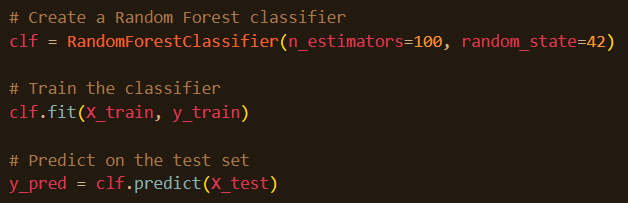
Table 2: Confusion Matrix of Text Analysis

The model handled testing data very well, scoring 99% for all three metrics of performance. and only incorrectly labeled 11 instances of our testing data. This shows promise for real world implementation, especially in the use of places of higher education having a goal of reducing spam mail for students and faculty.

Our next two artifacts were classification and input analysis of a selected variable for the phishing link dataset. For classification, we went with Random Forest Classification in python.

Figure 13: Separating variables for training

Figure 14: Creating training and test sets

Figure 15: Training and Testing of Model

The model was then created and trained with our training data before evaluating it on, again, accuracy, precision, and recall.

|  |  |
| --- | --- |
| **Evaluation of Phishing Classification** | |
| **Method** | **Score** |
| Accuracy | 0.98 |
| Precision | 0.98 |
| Recall | 0.98 |

Table 3: Phishing Classification Model Scores

|  |  |
| --- | --- |
| 974 | 14 |
| 17 | 995 |

Table 4: Confusion Matrix of Phishing Classification

The model for phishing links scored 98% for the same metrics as we used to evaluate our text analysis of spam mail. This is an extremely high score for a classification model, and once again, would be found to be especially useful in detecting illegitimate websites, one of the leading cybersecurity concerns within the education industry.

Our next artifact was performing input analysis in R. In reference to Figure 1, we noticed the UrlLength for phishing and non-phishing sites seemed like they could be distributed slightly differently. We aimed to estimate the distribution of that variable for each class to quantitatively compare the two.



Figure 16: Sub-setting Data

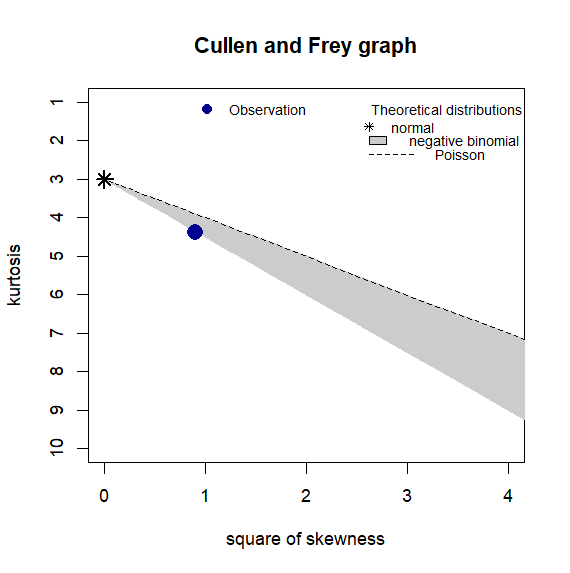


Figure 17: Cullen and Frey Graph for Legitimate Websites

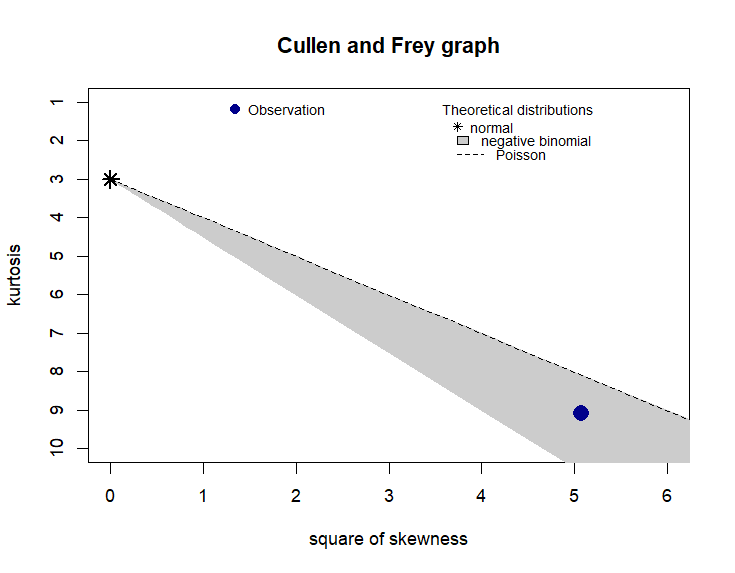
Based on Figure 17, we were led to believe that the distribution for UrlLength of legitimate websites was either negative binomial or Poisson. We went on to use the fitdist() function for both distributions.

|  |  |  |
| --- | --- | --- |
| **Distribution** | **Log-Likelihood** | **P-Value** |
| Negative Binomial | -23872 | 0.1623 |
| Poisson | -46409 | N/A |

Table 5: Distributions (Legitimate)

To evaluate the input analysis, we look at the p-value. Since it is greater than 0.05, we fail to reject the null hypothesis that UrlLength for legitimate websites has a negative binomial distribution.

For the phishing websites, we repeated the same steps, now with the other subset of data.

Figure 18: Cullen and Frey Graph for Phishing Websites

|  |  |  |
| --- | --- | --- |
| **Distribution** | **Log-Likelihood** | **P-Value** |
| Negative Binomial | -23766 | 0 |
| Poisson | -52561 | N/A |

Table 6: Distributions (Phishing)

Again, we assessed the distributions and negative binomial was the best fit. Although it was the best fit, we have a p-value of 0, so we must reject the null hypothesis that the distribution of UrlLength for phishing websites is negative binomial.

By acknowledging these distributions, we can gain valuable insights into some key differences between legitimate and phishing websites. Legitimate websites tend to prioritize usability while phishing websites tend to prioritize deception, which can include the manipulation of URL structures to hide their true intentions, which can then cause the distribution of URL lengths to lack discernible patterns. It is a key observation to realize that legitimate URLs follow a distribution pattern while phishing URLs do not. Understanding these differences helps in identifying potential threats and the nature of phishing website structures.

Part 4: Group roles and signatures

James Parker-Part 1-33%

Emily Spector-Part 3-33%

Jonathan Busch-Part 2-33%