Predictive Analysis: Machine Learning Models for URL Classification

A Major Qualifying Project

Submitted to the Faculty of

Worcester Polytechnic Institute in partial

fulfillment of the requirements for the

Degree in Bachelor of Science in

Computer Science

By

Robert A. Dwan Jr.

Alex M. Tavares

Date: 10/9/2019

Sponsoring Organization:

MIT Lincoln Laboratories

Project Advisor:

Professor George Heineman, Major Advisor

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Air Force.

Table of Contents

Table of Figures

Table of Tables

Abstract

The rise of cybercrime has motivated the need for improved early detection and prediction mechanisms to prevent cyber-attacks from causing damage to unsuspecting victims. We developed and analyzed various machine learning algorithms to tackle one approach for early detection, URL classification. Unlike previous research, which focused on binary classification, our approach focuses on classifying URLs to their likely attack category. Through testing and evaluation, we found that ensemble methods perform the best with our optimal feature set, producing accuracies as high as 95%.

Acknowledgements

Professor George Heineman (*WPI*) - Faculty advisor to the project, provided advice on software structure and design.

Leslie Shing (*MIT/LL*) - Advisor and sponsor to project, provided advice on the direction of the project.

Kimberly Holmgren (*MIT/LL)* - Advisor to project, provided resources for data sources and machine learning.

Executive Summary

More people than ever before have access to the Internet. Of the billions of Internet users, there are some who take advantage of and exploit others, known as cyber criminals. These attackers are hard to track, and their attacks can be complex and sophisticated. One avenue of attack is through the misuse of Uniform Resource Locators (URL).

The goal for this project is to develop sensors for publicly available data sources to detect for indications of techniques or traces left behind by an attacker during their planning and/or reconnaissance activities in order to predict for cyber-attacks that may be targeted against an organization. More specifically, we developed one particular sensor to classify URLs to their likely attack method type based on opensource data. We tested several machine learning algorithms and ensemble methods in order to identify the optimal model, hyperparameters, and feature set to classify the URLs.

Our development process included data gathering, feature extraction, and algorithm implementation. We gathered data from several sources including: PhishTank, URLhaus, and Alexa Top 1 Million. The selected features were based on previous research for URL classification. We implemented the following algorithms using the *Scikit-Learn* Python library: Random Forest, Logistic Regression, Support Vector Machine (SVM) with a linear kernel, and SVM with a Radial Basis Function kernel. Random Forest is an ensemble method, while the others are single classifiers. Ensemble methods use the decisions from several classifiers to improve predictive performance.

After developing our code, we tested features. We implemented two types of features: lexical and host-based. Lexical features were those gathered from the textual characteristics of the URL, and host-based features were those gathered from the network information related to

the host domain. We evaluated the full lexical feature set but were unable to test the host-based features due to time constraints and the fact that many URLs in our data set were no longer active. Through tests for feature independence (Chi-Squared and ANOVA F-value), we found the class labels to be dependent on all of the features in the lexical feature set. This tells us that the value of the feature has an effect on the class label, therefore it can be an indicator of class type. Features being independent show that the value of the feature has no effect on the class label, therefore it cannot be used as an indicator of class type. Through our feature independence testing, we determined the full lexical feature set to be the best choice.

We also analyzed the performance of the algorithms. Random Forest consistently had the highest accuracy and the lowest false positive rate. Along with testing for accuracy, we performed tests to optimize for the best parameter values for the various algorithms. Random Forest performed best with a parameter value of 40 decision trees.

The results from running these algorithms revealed phishing and malware URLs were often mislabeled as the other. Random Forest had the most success differentiating these two classes, while the complicated boundaries between the class types limited the success of the single classifiers. Therefore, we implemented 2 more ensemble methods using *Scikit-Learn*.

We implemented the Extra Trees and AdaBoost algorithms. They performed well, with accuracy scores comparable to Random Forest. In addition to testing for accuracy, we ran tests to optimize for the ensemble method parameters. The ideal parameters for the Extra Trees algorithm were a minimum sample split of 6 and the number of estimators equal to 91. The ideal parameters for the AdaBoost algorithm were a learning rate of 1 and the number of estimators equal to 66. We found that all of these methods produced accuracies above 90%.

To build on this research we recommend delving deeper into semi-supervised algorithms, to test their performance. There is substantially more unlabeled URL data than labeled URL data and semi-supervised algorithms can take advantage of that. These algorithms would be able to train on massive datasets and generate models that can better handle real world internet traffic. We also recommend expanding the variety of features beyond lexical and host-based. Due to the risk of compromising our systems, we were unable to use content-based features. One final recommendation is to look into modifying existing supervised machine learning algorithms to develop a stronger algorithm for URL classification.

1. Introduction

Advancements in technology have led society to shift towards a greater online presence. With more people and businesses online, it is difficult to protect the private information that is stored on the internet and our computers. Technology is constantly evolving and changing, so security measures must also adapt to continue to protect users. Cybersecurity is a field in computer science with the goal of creating secure systems and securing existing systems from cyber-attacks.

Cybercrime is the fastest growing crime, with an estimated $6 trillion annual cost to individuals and organizations by 2021. This cost includes, but is not limited to, damage to infrastructure and data, theft, fraud, and lost productivity [1]. With 1.9 billion websites and more than 4 billion Internet users it is becoming increasingly difficult to monitor this criminal activity. This has led to an increase in cybersecurity spending as many companies and individuals try to protect themselves [1].

Cyber-attacks are defined as “any attempt to expose, alter, disable, destroy, steal or gain unauthorized access to or make unauthorized use of an asset” [2]. An asset is something of worth to an organization or individual. As new software is developed there is the potential for new vulnerabilities that can be exploited. Combating this issue and preventing attacks requires better monitoring of systems and networks. This has led to research into preventing potential attacks by monitoring publicly available data to find traces of cyber-criminal activity.

There are many different publicly available data sources available that require monitoring. Our project focused on developing one of the sensors to detect for indications of techniques or traces left behind by an attacker. The resulting data will be used as part of a larger

analysis pipeline to build an improved awareness of the existing Internet threats. This paper aims to look at one of those threats: malicious Uniform Resource Locators (URL).

Malicious URLs exist in all facets of the Internet and any user can come across them. URL links are embedded in emails, appear on web sites, and are posted on social media sites, among other areas of the Internet. It is challenging for the average user to distinguish between legitimate and illegitimate URLs. This paper focuses specifically on classifying URLs to their likely cyber-attack category. We will research the characteristics of malicious URLs which distinguish them from normal ones. We will then develop a program that will learn from these characteristics to classify URLs, using machine learning.

2. Background

Malicious URLs make up one third of all publicly accessible URLs [3]. With the vast quantity and increased sophistication of malicious URLs online, it is becoming increasingly difficult to distinguish between legitimate and illegitimate URLs. Artificial intelligence, more specifically machine learning, is a field of study that has yielded highly accurate classifiers to address this issue [4,5, 6]. Past research has explored a variety of algorithms and feature sets to determine effective machine learning models to classify URLs as malicious or benign and have produced binary classifiers with over 90% accuracy [4,5,6]. There is still much work to be done with URL classification and many researchers aim to improve the accuracy of existing classifiers.

2.1 Cyber Attacks Using Malicious URLs

A URL is an address that corresponds with a web page. The structure of a normal URL can be seen in Figure 1. Malicious URLs are illegitimate Internet addresses used by cyber criminals to take advantage of users who visit the page [5]. Cyber criminals trick unsuspecting users to click these URLs to retrieve personal information, for financial gain, or to download malware. Common cyber-attack categories include social engineering and malware distribution [7].

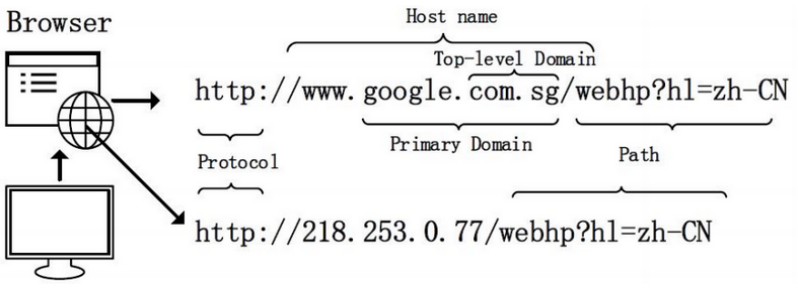

Figure 1. URL Structure [3]

2.1.1 Social Engineering

Social engineering encompasses a variety of techniques, where the attacker imitates legitimate sites and email addresses to retrieve private or personal information. The most notorious and well-known attacks in this category are phishing attacks [8]. Phishing methods mimic legitimate URLs or websites to coerce users into unknowingly divulging their personal information. Attackers stimulate their victim's emotions such as curiosity and fear, which tricks the user into clicking the URL [9].

Phishing attacks are not limited to gaining one individual's data, they can also be used to gain access to an entire organization's data. For example, Presbyterian Healthcare in New Mexico was the victim of a phishing attack on May 9, 2018. Employees from the company fell victim to a phishing email that gave the attackers access to their accounts. The attackers were able to gather healthcare plan data from 183,370 patients including patient names, dates of birth, and social security numbers. The company did not know that they had been a victim of an attack until June 26, 2018, over one month later [10]. This example shows how cybercrimes can affect a large number of people. Had the emails been filtered for malicious URLs the attack could have been prevented.

2.1.2 Malware Distribution

The term malware comes from the combination of the words malicious and software. Malware is a general term used to describe any software developed for the purpose of damaging, disrupting, or gaining access to another user's system. Popular examples of malware include:

- Ransomware – The victim is locked out of their system until a ransom is paid to the attacker.
- Spyware – Software that allows the attacker to see what the victim is doing on their computer [11].

Ransomware is increasing in frequency and can cost companies and the government a significant amount of money to resolve. In 2016, there was an estimated ransomware attack every 40 seconds. With this frequency and the cost per attack, global ransomware is predicted to cost $11.5 billion in 2019 [1].

In August of 2019, there was a significant ransomware attack on a hospital in Aberdeen, Washington. Grays Harbor Community Hospital and the Grays Harbor Medical Group, consisting of eight clinics, were attacked and patient information was locked. The cyber criminals held 85,000 patient records hostage for the Bitcoin equivalent of $1 million. While this attack was ongoing, the clinics involved were forced to keep records on paper which resulted in delays for appointments. It is thought that this attack started with a phishing email which triggered the malware download. This could have been prevented if the URL was detected earlier [12]. Because of the risks associated with malware it is important to be able to identify malicious URLs to prevent attacks and keep users' information safe.

2.1.3 Other Types of Cyber Attacks

Another category of malicious URLs are botnet command and control (C&C) server URLs. These servers are the command centers for botnets. Botnets are a network of systems that have been infected with malware distributed from the C&C server. The C&C servers are able to communicate with these systems, steal information, and control the computers to use to achieve their objectives [13]. Botnets can be used for different types of cyber-attacks. They can be used to steal information, hold the victim for ransom, and execute Distributed Denial of Service (DDoS) attacks. A DDoS attack is used to flood the target with Internet traffic causing the target to lose availability of their system [13, 14]. Detecting URLs related to existing and potential botnets can prevent future DDoS and other attacks by shutting down or preventing communication between the systems and server.

2.2 Non-Machine Learning Approaches

A frequently used method for malicious URL detection is blacklisting. Blacklists are lists of known malicious URLs that can be used to check if a URL is already known to be malicious. Blacklists need to be constantly updated as new sites are discovered; they cannot protect against unknown URLs. Previous research has looked into creating a predictive blacklist specific for users on a network. This algorithm uses user-defined data and determines the likelihood that a specific network or user would be attacked in the future. Using this information, a final blacklist specific for each user is produced. This algorithm produced high attacker hit-rates, good "new attacker" predictions, and stability for the future [15].

There are existing commercial products on the market that focus on the problem of malicious URLs. For example, WebAdvisor from McAfee is a browser plugin that attempts to

protect users from malware and phishing attacks as the user browses the Internet [16]. Another example of a commercial product is the SafeLink feature on Microsoft Outlook. This feature checks to see if any links or emails contain phishing attacks, malware, or viruses. If links or emails are deemed suspicious or malicious then the users are unable to click on the link [17]. These products rely on existing blacklists, user feedback, and proprietary research done by the companies. This reactive approach cannot keep up with the volume of new attacks being generated, therefore a more proactive solution, such as machine learning, is required.

2.3 Machine Learning

Artificial intelligence (AI) is the study of intelligent machines that are able to perceive the world around them and make decisions based on that input. Machine learning is one of the many subfields of artificial intelligence that exist today. In a famous quote by Arthur Samuel, an early pioneer in the field, he said machine learning gives "computers the ability to learn without being explicitly programmed" [18]. The foundations of machine learning can be traced back to 1950, when Alan Turning developed a test to determine if a computer had real intelligence called the "Turing Test". To pass the test the computer must deceive a human into thinking it is also human. In 1952, Arthur Samuel wrote the first computer learning program. The program played checkers and it improved after every game it played [19]. More than 50 years since the field first emerged, we have seen huge advancements in machine learning and AI. In March 2017, the company OpenAI reported that the AI agents they created, developed a new language to achieve their goals more efficiently. Also, soon after that, Facebook reported that AI agents they developed were able to negotiate and lie [20].

Machine learning gives computers the ability to learn on their own. These programs are not developed to perform specific tasks but to learn about data and the patterns that exist in the data. An overview of the machine learning process can be found in Figure 2. Machine learning can be broken down into three main categories: supervised, semi-supervised, and unsupervised.

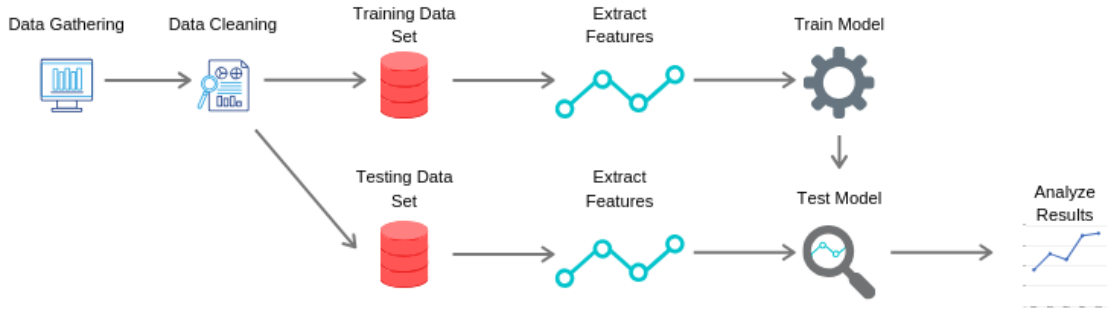

Figure 2. Overview of the Machine Learning Process

Supervised machine learning models are trained on labeled data, where the model knows the input and desired output. Using this data, the model develops a function to describe the relationship between the input and output data. This function can then be used with new inputs to predict the desired output. A popular use for supervised learning is classification problems. Classification is a machine learning approach where the model learns from input data to classify and categorize new data. An example of a classification problem is image recognition, such as classifying an image as containing a car or not containing a car [22, 23].

Unsupervised machine learning models take input data with no output data. These methods detect patterns within the data, which can be useful when experts do not know what they are looking for or where output data is unavailable. This type of learning is used mainly for clustering and association. An example of a clustering problem is customer segmentation. In this problem, an organization would discover clusters of customers within their customer base that

may not have been obvious to them previously. This gives the organization more information about their customers, aiding in advertising and sales. An example of an association problem is market basket analysis. An organization would analyze the contents of customers' shopping cart to determine associations between products. This can be used to discover the correlation between products, which can aid in designing the layout of stores, sales, and marketing [22, 23].

Semi-supervised learning sits between the previous two. It is particularly useful in scenarios where labeled data is hard to get. The labeled data is used to help the algorithm find patterns rather than model relationships. This learning type is mainly used for classification and clustering, as described above [22, 23].

Our problem focuses on classification of data, URLs, with known outputs, attack category; therefore, we will research and test existing classification algorithms. There are several classification algorithms that have been used for malicious URL classification such as Random Forest, Support Vector Machine (SVM), and Logistic Regression [4,5,6].

All machine learning models use features—properties or attributes of the data—extracted from the input data sets to create their models. In the context of URL classification, there are three feature types: host-based, lexical, and content-based. Host-based features are those that define the identity, location, and other network information about the host. Lexical features are textual properties obtained from the URL itself. Lastly, content-based features come from the web pages linked to the URLs themselves. Content-based features require a more in-depth analysis of the content and are more computationally expensive. They also present an inherent risk as our systems could become compromised during exploration of the web pages related to the URLs we are trying to classify [3,5]. The content-based feature set falls outside the scope of our research, due to the associated risks and greater time requirement.

3. Methodology

The necessary steps to construct an accurate URL classifier began with gathering a representative data set for training and testing the models. Next, we developed supervised machine learning models using several algorithms. Once the models were implemented, we trained and tested them using different feature sets. We then evaluated these models and feature sets to find areas for improvement. Finally, using the performance evaluations of our feature sets and models, we iterated over our tests to develop more accurate classifiers.

3.1 Data Gathering

The first step to acquire a representative data set was to gather data. To collect this data set, we used several open-source databases and sites. The data comes from 5 different attack categories: normal, phishing, malware, ransomware, and botnet. We required representative data from each of the 5 categories.

Our normal data came from two sources: Canadian Institute for Cybersecurity (CICS) and research done by Frantisek Strasak [17, 18]. The CICS obtained data by passing URLs from Alexa Top websites into a Heritrix web crawler to extract the URLs. Once the URLs were extracted and duplicates were removed, the data set was left with 35,300 URLs classified as normal [17]. Frantisek Strasak recorded his web traffic for 3 days while accessing secure sites in the Alexa Top 1000. He created several packet capture files from this web traffic. He was able to verify that the sites visited in the capture files were normal after scanning his computer for malware [18]. We used his capture files to extract all the URLs. We then removed duplicates and added them to our data set. The malicious URLs came from four different sources; each is a

blacklist for the specific threat type. Table 1 shows the source and date retrieved for the data gathered.

Class	Source	Count	Date Retrieved	Description
Phishing	PhishTank [19]	21,979	July 30, 2019 - August 20, 2019	A blacklist containing phishing URLs
Malware	Abuse.ch, URLhaus [20]	217,818	May 22, 2019 - August 20, 2019	A blacklist containing malware URLs
Ransomware	Abuse.ch, Ransomware Tracker [21]	1,903	May 16, 2019 - August 20, 2019	A blacklist containing ransomware URLs
BotnetC&C	CyberCrime [22]	16,292	August 20, 2019	A blacklist containing botnet URLs

Table 1. Malicious Data Gathered

Our data gathering effort produced significantly more malicious data than normal data. As a result, the training and testing sets were created by varying ratios of normal to malicious URLs to allow for a more realistic distribution. Even with this measure in place, there is still a possibility that our training and testing data set is skewed and does not represent realistic traffic. The normal data gathered from CICS and Frantisek Strasak is assumed to be composed of completely normal data, with no malicious URLs, and consist of a representative sample of the normal URL population.

3.2 Algorithms

We identified several supervised machine learning algorithms in our background research that performed well in URL classification. Much of this previous research was on binary URL

classification, but we worked towards creating a multiclass URL classifier. A binary classifier categorizes data into two separate classes, in this case classifying URLs as normal or malicious. A multiclass classifier categorizes data into three or more classes, we aim to classify URLs as normal, phishing, malware, ransomware, and botnet. The benefits of a multiclass classifier are it provides more context about the URL and potential attacks. This can be greatly beneficial for planning for and preventing attacks detected over public data sources. The algorithms we implemented were: Support Vector Machine (SVM), Logistic Regression, and Random Forest.

3.2.1 Support Vector Machine

A support vector machine is a discriminative classifier that, given labeled training data, produces an optimal hyperplane to classify new data. SVMs use kernels which are functions that transform data into a higher dimensionality space. This is essential for classification in complex, non-linear data sets, such as the data generated from URL feature extraction. This allows the model to generate a mapping function that separates the data points into their respective classes. We tested two different kernels in our models: the linear kernel and the Radial Basis Function (RBF) kernel. The value of gamma defines the distance between the boundary data points and the separation line; this parameter was tuned to increase the accuracy of the model. A consideration when tuning this parameter is that a high gamma value can lead to a more complicated decision boundary and over-fitting [4, 23, 24].

3.2.2 Logistic Regression

Logistic regression determines the likelihood an input belongs to a specific class. This is done by using the logit function. The logit function is defined as:

$$logit(p) = log(\frac{p}{1-p}), where\ p\ is\ the\ probability\ of\ classification$$

This gives a value between 0 and 1, representing the probability that an inputted data point belongs to a class. The closer the value is to one, the more likely an input belongs to a class. This probability is then used to fit a line used for predicting new data. The model learns by altering coefficients representing the line to fit the data. As the model learns, the coefficients change to produce the maximum likelihood of predicting the correct category [25, 26].

After training, the model uses the final coefficients to predict new data. The predictions use a classification threshold to determine which category the data point belongs to. For example, if the threshold for a binary classifier is set at 0.5, any value 0.5 or greater will be classified as normal, and any value below 0.5 will be classified as non-normal. Logistic regression is traditionally a binary classifier and we are implementing a multiclass classifier; therefore we used the multinomial version of the algorithm. This method works similarly to the binary version, except it uses multiple one-vs-all binary classifiers. The one-vs-all method compares one class type against the rest; for example, normal vs not normal would classify the data as either normal or not [25, 26]. Another example would be, phishing vs not phishing. The algorithm would then combine the binary classifiers generated from each type into one model. When the model is run, each binary classifier is run on the input and the class with the highest probability is the selected classification.

3.2.3 Random Forest

The Random Forest algorithm is a classification algorithm. That works as follows:

1. The data set is randomly divided into ‘L’ subsets with ‘k’ entries each. This is done with replacement, meaning subsets can contain the same entries. This method of data sampling is called bootstrapping.
2. Each subset is then used to train a decision tree. A decision tree works by having several splits where the data is separated based on a feature. The feature is randomly selected from all the features of the data. The value of the feature dictates which direction down the tree it will travel.
3. After training the trees, new inputs go through all of the trees to get a prediction. For classification, the final prediction is based on a majority vote from all of the trees.

This algorithm improves on a single decision tree and creates robustness of the model as it prevents overfitting the data and is able to make splits on randomly chosen features, as opposed to using only the best features to split the data [27, 28]. Random Forest is a bagging type ensemble method. Ensemble methods combine the decisions from other algorithms to give a less biased, more accurate prediction [29]. In particular, bagging methods run several algorithms in parallel and aggregate their results to create a prediction. An overview of bagging can be found in Figure 3.

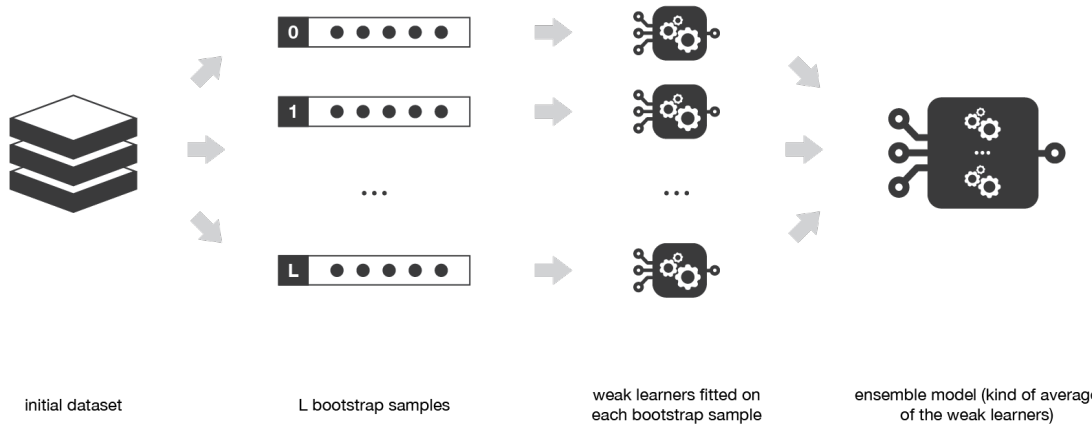

Figure 3. Bagging Method Overview [29]

3.3 Feature Extraction

Our background research provided us with a wealth of features and feature sets to test. The features we focused on for our implementation were lexical and host-based. Previous research shows that these features are sufficient to create an accurate classifier. Content-based features may provide value but present risk to the integrity of our systems due to potentially malicious web page content related to the URLs; therefore they were not evaluated. We implemented 32 features in total, 29 of them being lexical. The features can be found in Table 2.

Features	Description of Feature	Type	Returns
Length of URL	The number of characters in the URL.	Lexical	Count
Length of hostname	The number of characters in the hostname.	Lexical	Count
Length of path	The number of characters in the path.	Lexical	Count
Number of . In URL	The number of periods in URL.	Lexical	Count
Number of @ in URL	The number of @ symbols in URL.	Lexical	Count
Count % in url	The number of % symbols in URL.	Lexical	Count
Count _ in path	The number of underscores in path.	Lexical	Count
Count & in path	The number of ampersands in path.	Lexical	Count
Check - in hostname	Check - symbol in hostname.	Lexical	Boolean
Number of . in hostname	The number of periods in the hostname	Lexical	Count
Count - in path	The number of - symbols in the path.	Lexical	Count
Count / in path	The number of / symbols in the path.	Lexical	Count
Count = in path	The number of = symbols in the path.	Lexical	Count
Count ; in path	The number of semi-colons in the path.	Lexical	Count
Count , in path	The number of commas in the path.	Lexical	Count
Count . in path	The number of periods in the path.	Lexical	Count
Params in URL	If there are parameters in the URL.	Lexical	Boolean
Queries in URL	If there are queries in the URL.	Lexical	Boolean
Fragments in URL	If there are fragments in the URL.	Lexical	Boolean
Entropy of hostname	The calculated entropy of the hostname.	Lexical	Float
Check for Non Standard port	Checks if URL connects to host server through non standard port. The standard ports for http and https are 80, 443, and 8080.	Lexical	Boolean
Check Alexa Top 1 Million	Checks to see if domain name is in the Alexa Top 1 Million.	Lexical	Boolean
Check for punycode	Checks for presence of punycode in the URL.	Lexical	Boolean
Check sub-domains	Checks the sub-domains of URL for names in the Alexa Top 1 Million.	Lexical	Boolean
Digits in hostname	Check for digits in hostname.	Lexical	Boolean
IP based hostname	Looks for IP address in hostname.	Lexical	Boolean
Check TLD	Checks if the Top Level Domain of URL is common.	Lexical	Boolean
Username/Password in URL	If username or password is in URL.	Lexical	Boolean
Check protocol	If the URL protocol is https or not.	Lexical	Boolean
IP address location	The country the IP address comes from	Host Based	String
Address Registry	The country the host is registered in.	Host Based	String
Days Registered	The number of days the host has been registered for.	Host Based	Count

Table 2. Preliminary Feature List

3.3.1 Lexical Features

The lexical features are text-based characteristics of the URL. We split the URL into its protocol, host name and path. From there we analyzed the textual features in each. We used the Python libraries *tldextract* and *urllib* to parse the URLs and extract features. The lexical features we implemented were based on features described in previous research [2, 4, 30, 31].

3.3.2 Host-Based Features

Host-based features are composed of the network information about the URL host. We used a combination of features identified from our background research. The extracted features

include: the IP address location, the registered country of the host, and the amount of time the host has been registered. We used the Python package *ipwhois* and *socket* to get the host information [2, 4]. The full feature list can be found in Table 2. The last three features in the list are the three host-based features. These features are useful because they can identify URLs with hosts located in suspicious areas and identify inconsistencies between the hosts and where they are registered. Also, malicious URLs tend to be registered more recently, therefore the length of time for domain registration can be a good indicator for detecting malicious URLs [2,4].

3.4 Development

We first determined the set of existing tools and libraries appropriate for our use case. Then, we implemented and trained several models using a training data set. After the models were trained, they were tested with a test data set to determine the models' performance. Finally, we optimized the parameters and train/test data set ratios to maximize the accuracy of the models.

3.4.1 Tools Used

We decided to use Python as our coding language because it is useful for processing large amounts of data and has readily available open source machine learning libraries. We assessed and selected a suitable machine learning library.

The three main Python libraries for machine learning are: *PyTorch*, *TensorFlow*, and *Scikit-Learn*. *Scikit-Learn* is an easy to use Python library that comes with out-of-the-box algorithm implementations. *Scikit-Learn* is more of a general-purpose machine learning library that includes implementations of many classic algorithms. *TensorFlow* and *PyTorch* are deep learning frameworks. They are more flexible and allow for the integration of custom code. We

decided to use *Scikit-Learn* for the beginning implementations of our models because it contained models for all the aforementioned algorithms. *TensorFlow* and *PyTorch* are excellent alternative libraries to *Scikit-Learn*, but due to our algorithms of choice and the ease of use we selected *Scikit-Learn*. Nonetheless, it is possible to replicate what we have implemented using models from *TensorFlow* and *PyTorch*.

We also needed a way to extract the features we discussed previously. Thus, we created our own tool. Our tool takes a URL as input and returns a numerical array containing values for the features previously mentioned

3.4.2 Training

We split the training data into normal and malicious URLs and varied the split ratios of these two categories. We trained using a 50/50, 60/40, 70/30, and 80/20 normal/malicious splits. We trained the model on each of these split ratios in order to find the optimal training split between normal and malicious data that would produce the best performing model given a real scenario.

3.4.3 Testing and Evaluating

We used several methods to test and evaluate the models. We evaluated the performance of the features, along with an evaluation of the models' performance. We used built-in *Scikit-Learn* functions as well as some other mathematical tools to test the effectiveness of our features. We used two methods to test the relationship between the feature variables and classes: a chi-square test and ANOVA F-Value test.

The chi-square test is used to test for independence of categorical features. A chi-squared value is calculated for each categorical feature. If the chi-score is greater than or equal to the threshold value, then the feature affects the URL class. Otherwise, if the chi-score is less than the

threshold, the feature is most likely not useful. The ANOVA F-Value test is similar to the chi-square test in that it is a test for independence, except it is used for numerical values. Similarly, if the F-Value is greater than or equal to the threshold value, then the feature affects the URL class, and vice versa. We also used a heatmap plot from the Python library, *Seaborn*, to visualize the correlation between features. The heatmap plot required a correlation matrix which was generated using the Python library, *Pandas*. These techniques reduced the complexity of the data, which led to faster and more accurate classification.

To analyze the performance of our models we examined several metrics. We first looked at the overall accuracy of the model. The accuracy is a percent-value based on the number of true positives over the total number of predictions. A true positive is a correctly classified URL. Table 3 describes true positives in the case of a 'Normal' URL.

	Predicted Class	Actual Class
True Positive	Normal	Normal
True Negative	Not Normal (e.g. phishing, malware)	Not Normal (e.g. phishing, malware)
False Positive	Normal	Not Normal (e.g. phishing, malware)
False Negative	Not Normal (e.g. phishing, malware)	Normal

Table 3. Example of True Positives and Negatives, and False Positives and Negatives

We also examined the confusion matrices to calculate the number of false positives and the number of false negatives for specific classifications, also defined in Table 3. This information gave insight into how well the model can classify new data, based on the training data. We also looked at the time it took to train and test the models. Although our focus was on creating more accurate models, the speed of training is an additional factor to consider.

Once we determined the accuracy of the model, we began improving upon it. We used three methods to improve the model: modifying the tuning parameter, changing the dataset volume, and changing the ratio in the training dataset. The specific tuning parameter differed for each algorithm. For example, in the Random Forest algorithm we varied the number of decision trees produced, and in the SVM algorithm we varied the value of gamma. Lastly, we tested the models using various ratios of malicious to normal URL data in order to determine the optimal ratio of training data that would produce the most accurate model.

3.4.4 Iterate

Upon evaluation of our preliminary results, we found that the algorithms had difficulty discerning between phishing and malware URLs. This led us to explore additional features as well as additional algorithms.

After looking at the results of our feature evaluation and results of the features used in previous research, we added 2 features and changed 4 features. The two features we added were: 'Number ~ in URL' and 'Number # in URL'. The features we changed can be found in Table 4. We made the changes to these features to increase the amount of textual information they provide about a URL. In addition, we made changes to the names of many features to add more consistency to the feature names. In total we implemented 34 features, 31 lexical and 3 host-based. The final list of features can be found in Table 5.

Old	New
Count _ in path	Number _ in URL
Count & in path	Number & in URL
Check - in hostname	Number - in hostname
Digits in hostname	Number digits in hostname

Table 4. Features Changed

Features	Description of Feature	Type	Returns
Length of URL	The number of characters in the URL.	Lexical	Count
Length of hostname	The number of characters in the hostname.	Lexical	Count
Length of path	The number of characters in the path.	Lexical	Count
Number . in URL	The number of periods in URL.	Lexical	Count
Number @ in URL	The number of @ symbols in URL.	Lexical	Count
Number % in URL	The number of % symbols in URL.	Lexical	Count
Number _ in URL	The number of underscores in URL.	Lexical	Count
Number ~ in URL	The number of ~ in URL.	Lexical	Count
Number & in URL	The number of ampersands in URL.	Lexical	Count
Number # in URL	The number of # symbols in URL.	Lexical	Count
Number - in hostname	The number of - symbol in hostname.	Lexical	Count
Number . in hostname	The number of periods in the hostname	Lexical	Count
Number - in path	The number of - symbols in the path.	Lexical	Count
Number / in path	The number of / symbols in the path.	Lexical	Count
Number = in path	The number of = symbols in the path.	Lexical	Count
Number ; in path	The number of semi-colons in the path.	Lexical	Count
Number , in path	The number of commas in the path.	Lexical	Count
Number . in path	The number of periods in the path.	Lexical	Count
Params in URL	If there are parameters in the URL.	Lexical	Boolean
Queries in URL	If there are queries in the URL.	Lexical	Boolean
Fragments in URL	If there are fragments in the URL.	Lexical	Boolean
Entropy of hostname	The calculated entropy of the hostname.	Lexical	Float
Check for Non Standard port	Checks if URL connects to host server through non standard port. The standard ports for http and https are 80, 443, and 8080.	Lexical	Boolean
Check Alexa Top 1 Million	Checks to see if domain name is in the Alexa Top 1 Million.	Lexical	Boolean
Check for punycode	Checks for presence of punycode in the URL.	Lexical	Boolean
Check sub-domains	Checks the sub-domains of URL for names in the Alexa Top 1 Million.	Lexical	Boolean
Number digits in hostname	The number of digits in the hostname.	Lexical	Count
IP based hostname	Looks for IP address in hostname.	Lexical	Boolean
Check TLD	Checks if the Top Level Domain of URL is common.	Lexical	Boolean
Username/Password in URL	If username or password is in URL.	Lexical	Boolean
Check protocol	If the URL protocol is https or not.	Lexical	Boolean
IP address location	The country the IP address comes from	Host Based	String
Address Registry	The country the host is registered in.	Host Based	String
Days Registered	The number of days the host has been registered for.	Host Based	Count

Table 5. Final Feature List

We observed that many of the URLs belonged to multiple classes; this was the case for many of the malware and phishing URLs. Thus we implemented a tagging method, which tagged the URL with class labels above a certain threshold. This allowed URLs to have multiple classifications. Also, we implemented several ensemble methods based on the success of the previous algorithms and the complicated decision boundaries presented by the data.

For the tagging algorithm, we used the functions already built into the models to return class probabilities instead of a single class prediction. Using these probabilities, the URLs were tagged as belonging to a particular class if the class probability for the URL is above a configurable threshold. This allowed URLs to have multiple classifications. The output of the tagging was then output to a file for visual verification. A correct prediction was one that contained the true

label in the list of tags. We also calculated the false positive and false negative rates of each run. A false positive is any normal URL that was given a malicious tag. A false negative is any malicious URL that was given a normal tag. The tagging method was tested on the 4 training ratios previously mentioned.

We implemented a boosting and an additional bagging ensemble method. Boosting methods run algorithms sequentially, with each model learning from the previous one. An overview of boosting can be found in Figure 4. The bagging algorithm we implemented is called the Extra Trees classifier and it uses a decision tree model as its underlying classifier. The boosting algorithm we implemented is the AdaBoost classifier and it uses an Extra Trees model as its underlying classifier. Both algorithms were optimized for accuracy and tested on the 4 training ratios previously used. The AdaBoost algorithm was optimized for the following parameters: n_estimators and learning_rate. The Extra Trees algorithm was optimized for the following parameters: n_estimators and min_samples_split.

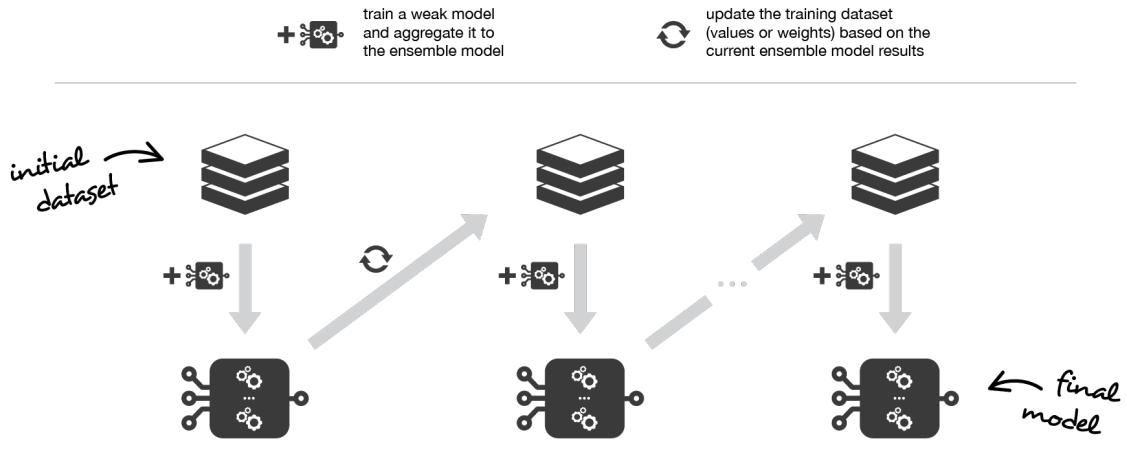

Figure 4. Boosting Overview [29]

4. Results

4.1 Iteration One

4.1.1 Features

From the preliminary list of 32 features, we analyzed the 29 lexical features implemented using three methods: a chi-squared test for categorical features, a calculation of ANOVA F-values for numerical features, and a correlation heatmap. The results of the chi-squared test can be found in Table 6, the computed ANOVA F-values can be found in Table 7, and the correlations between the features can be found in Figure 5. Due to many of the malicious URLs in our dataset being inactive and time constraints with this project, we were unable to run an analysis on the 3 host-based features implemented.

4.1.1.1 Chi-squared test

The null hypothesis for the chi-squared test is that the features are independent of the class labels. With a significance level of 95%, any feature that had a p-value below 0.05 could reject the null hypothesis. Rejecting the null hypothesis means that the class labels are dependent on that feature. Our analysis of the lexical features revealed that all but one categorical feature rejected the null hypothesis, which means that the value of the feature had an effect on the class label. The feature “Check TLD” (TLD = Top Level Domain) had a chi-score value below the threshold and p-value below 0.05, which means it could not reject the null hypothesis; therefore we cannot say whether or not the value of that feature affects the class label.

Features	Chi-Score	P-Value
Params in URL	291.5194003	7.31E-62
Queries in URL	8488.84037	0
Fragments in URL	848.9911149	1.87E-182
Check for Non Standard port	1036.61199	4.15E-223
Check Alexa Top 1 Million	36269.36334	0
Check for punycode	169.8805246	1.11E-35
Check sub-domains	2729.653052	0
- in hostname	6521.523552	0
Digits in hostname	10524.18791	0
IP based hostname	20225.85635	0
Check TLD	9.013484167	0.060763323
Username/Password in URL	1756.960445	0
Check protocol	1690271.492	0

Table 6. Chi-Squared Test Results

4.1.1.2 ANOVA F-Values

We looked at the results of the ANOVA F-test with the same hypothesis and a 95% confidence level. All of the numerical features we implemented had a p-value below 0.05. Therefore, with 95% confidence we can say the class labels are dependent on all the numerical features, which means the value of the feature affected the class label.

Features	F-Value	P-Value
Length of URL	6355.742229	0
Number of . in URL	72.49164612	1.91E-61
Number of @ in URL	104.7265241	3.40E-89
Count % in URL	512.9904889	0
Entropy of hostname	3079.634139	0
Length of hostname	2412.265795	0
Count . in hostname	1953.511942	0
Length of path	5508.004089	0
Count - in path	5364.662328	0
Count / in path	1922.416499	0
Count = in path	103.0353527	9.72E-88
Count ; in path	4.434759704	0.001388553
Count , in path	16.55564592	1.44E-13
Count _ in path	268.7054323	2.99E-230
Count . in path	394.0376352	0
Count & in path	10.09240177	3.64E-08

Table 7. ANOVA F-Value Test

4.1.1.3 Correlation Heat map

The correlation heatmap in Figure 5 shows us how heavily correlated features are to one another. This tells us which features are redundant and add unnecessary complexity to the feature set. There are a few features such as number of '.' in URL and number of '.' in hostname, as well as the count of '/' in path and the length of the path. These features are highly correlated, which means they may be expressing some of the same information about the URL. The other features are not highly correlated meaning these features are providing helpful information to the model.

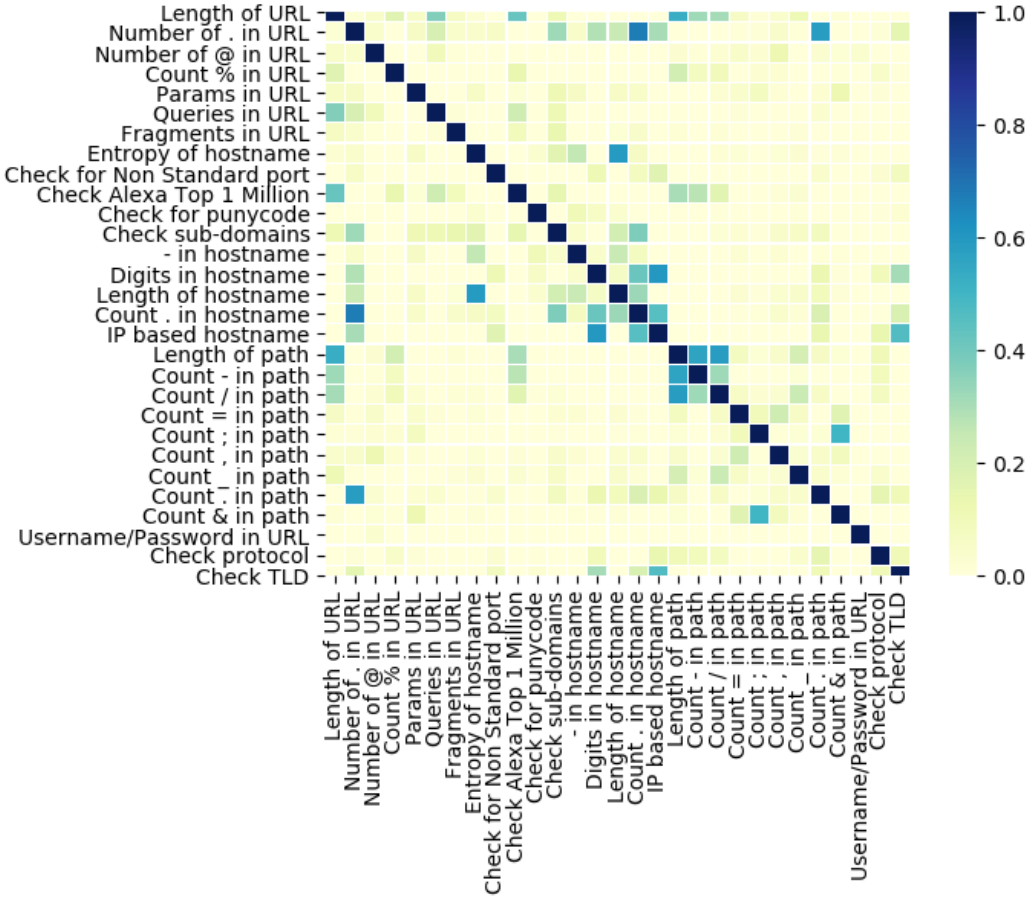

Figure 5. Correlation Heat Map

4.1.2 Algorithm Performance

4.1.2.1 Model Parameter Optimization

With the best feature set identified as the full lexical feature set, we began to tune certain parameters for each algorithm to find the highest accuracy. The parameters tuned were the number of decision trees in the Random Forest, the maximum number of training iterations for both Logistic Regression and SVM linear, and the value of gamma for SVM-RBF. All tuning tests were run on a training set with 50% normal URLs and 50% malicious URLs and a testing set containing 70% normal URLs and 30% malicious URLs.

For Random Forest, we started with 20 decision trees and stopped at 70, with increments of 5 each step. After running these tests, the run with 40 decision trees had the highest accuracy with 94.6% accuracy. Figure 6 shows how the other number of trees performed. All runs with more than 40 decision trees stayed around the same accuracy, with no improvement. We determined 40 trees to be the optimal amount to maximize accuracy and performance.

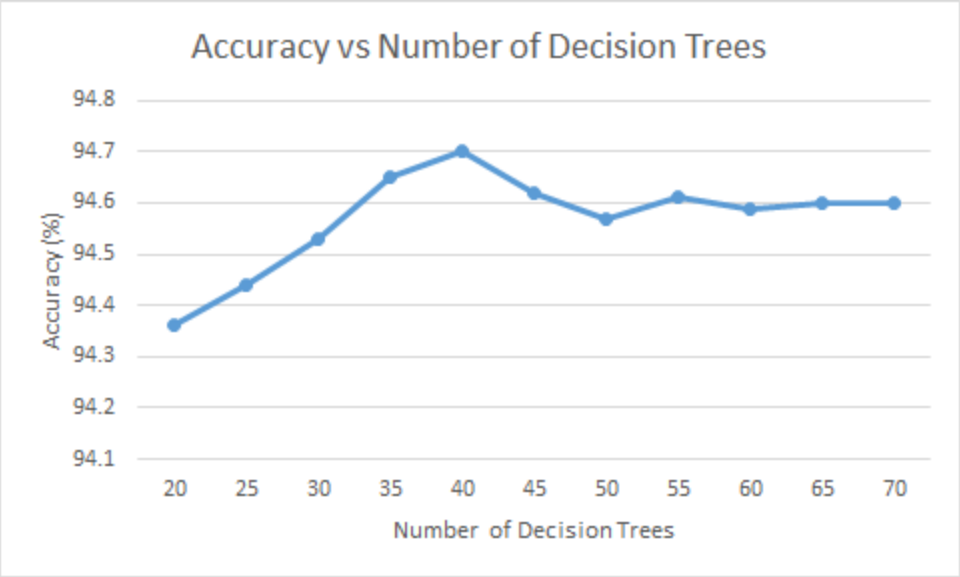

Figure 6. Accuracy with Varying Number of Decision Trees

For logistic regression, we started with 1,000 as the maximum number of iterations and increased the number to 20,000 with increments of 1,000. After running these tests, 10,000 iterations produced the highest accuracy of 88.33%. Figure 7 shows how the other number of iterations performed. All runs with more than 10,000 iterations stayed at the same accuracy, with no improvement. We determined 10,000 iterations to be the ideal number of iterations. With greater than 10,000 iterations, the algorithm was prone to overfitting, leading to lower accuracy.

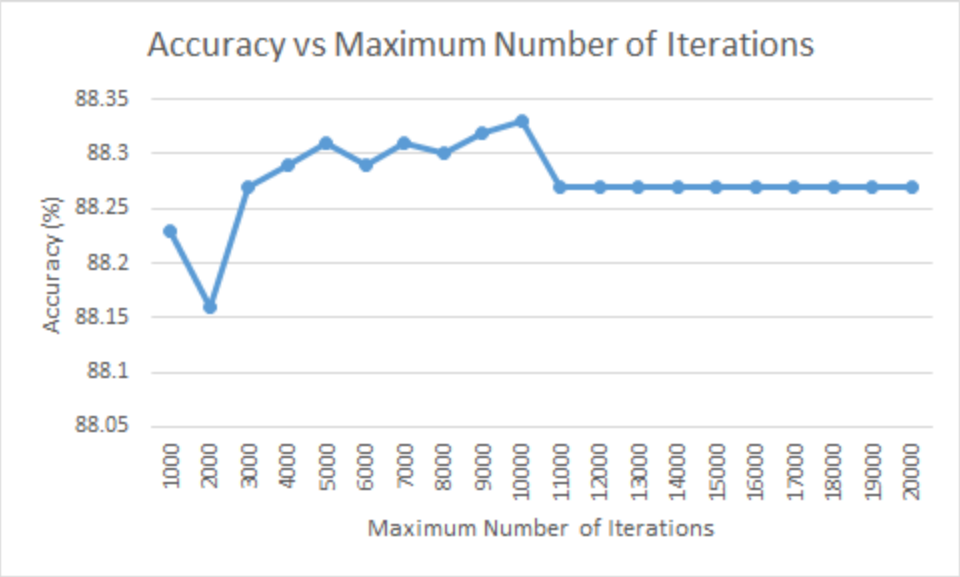

Figure 7. Accuracy with Varying Number of Iterations

For SVM linear, we started with 1,000 as the maximum number of iterations and increased the number to 20,000, with iterations of 1,000. We followed the same numbers as logistic regression to start. This led to all iterations having the same accuracy. We then shifted the starting value to 100 and the maximum value to 1,000, stepping by 100. This again produced the same accuracy for all results. We then shifted one more time to a minimum of 10 and a maximum of 100, stepping by 10. After running this test, 30 iterations produced the highest accuracy of 87.80%. Figure 8 shows how the other number of iterations performed. The model converged at 30 iterations, and every additional iteration after that produced the same results. We determined 30 to be the best number of iterations.

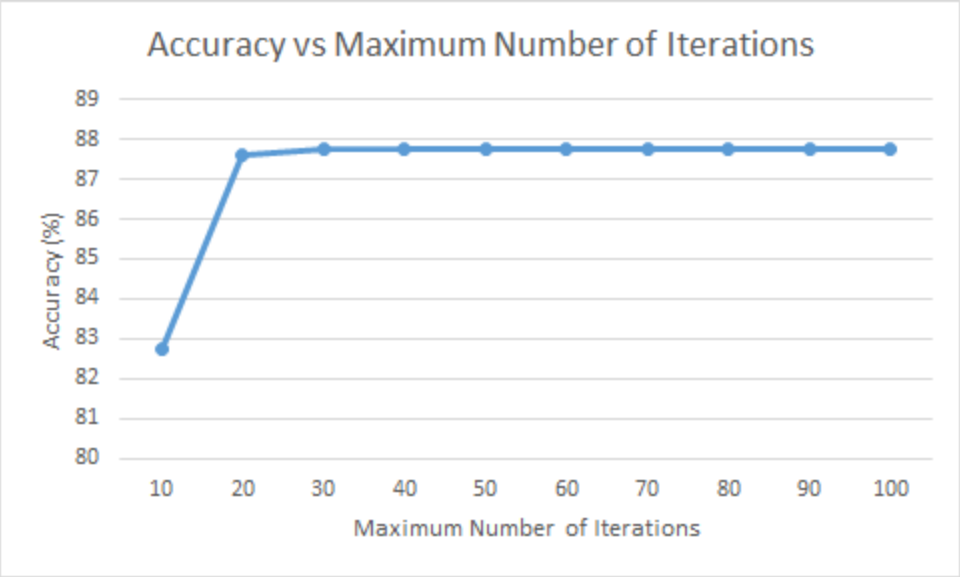

Figure 8. Accuracy with Varying Number of Iterations

Finally, for the SVM-RBF model we varied the gamma value. We tested with gamma values that ranged from 1-10 and 0-1. For values between 1 and 10 we incremented gamma by 1 for each run. This did not yield high accuracies as most were in the 10-20% range. The test for values between 0 and 1, stepping by 0.1, showed significantly higher accuracies. The best value was 0.9 and recorded an accuracy of 69.8%. Figure 9 shows the values for the other values between 0 and 1.

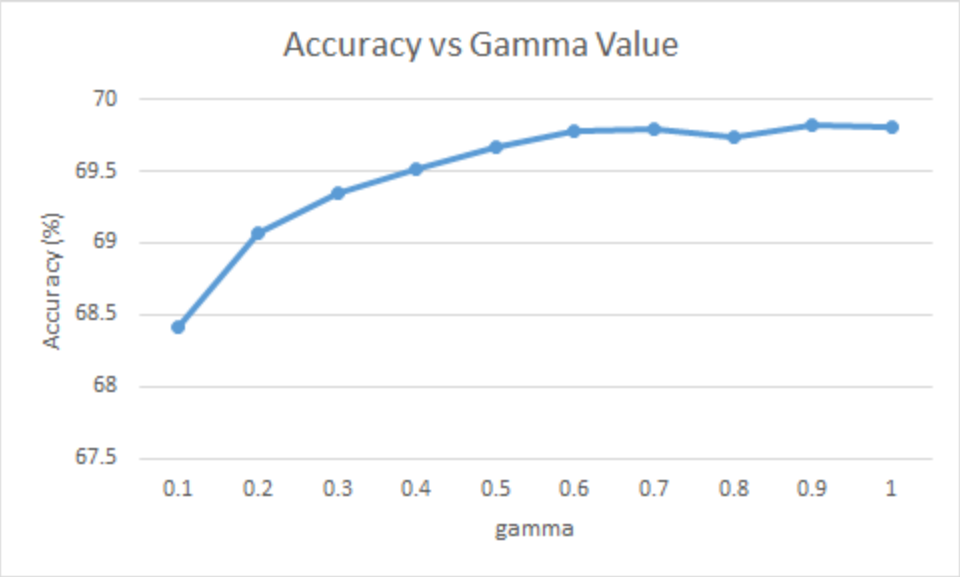

Figure 9. Accuracy with Varying Value for Gamma

4.1.2.2 Training Ratios

Next, we determined the ideal ratio of malicious to benign URL samples required in our training data. We created 4 data sets which we used to train the models: one with 50% normal and 50% malicious, one with 60% normal and 40% malicious, one with 70% normal and 30% malicious, and finally one with 80% normal and 20% malicious. After training our models with each of these data set ratios, all models were tested against the same data set used for testing the different parameters.

Our results showed that the ratio of 60% normal to 40% malicious was the best training set ratio as it produced the highest accuracy for each algorithm. The results from the testing are depicted in Figure 10.

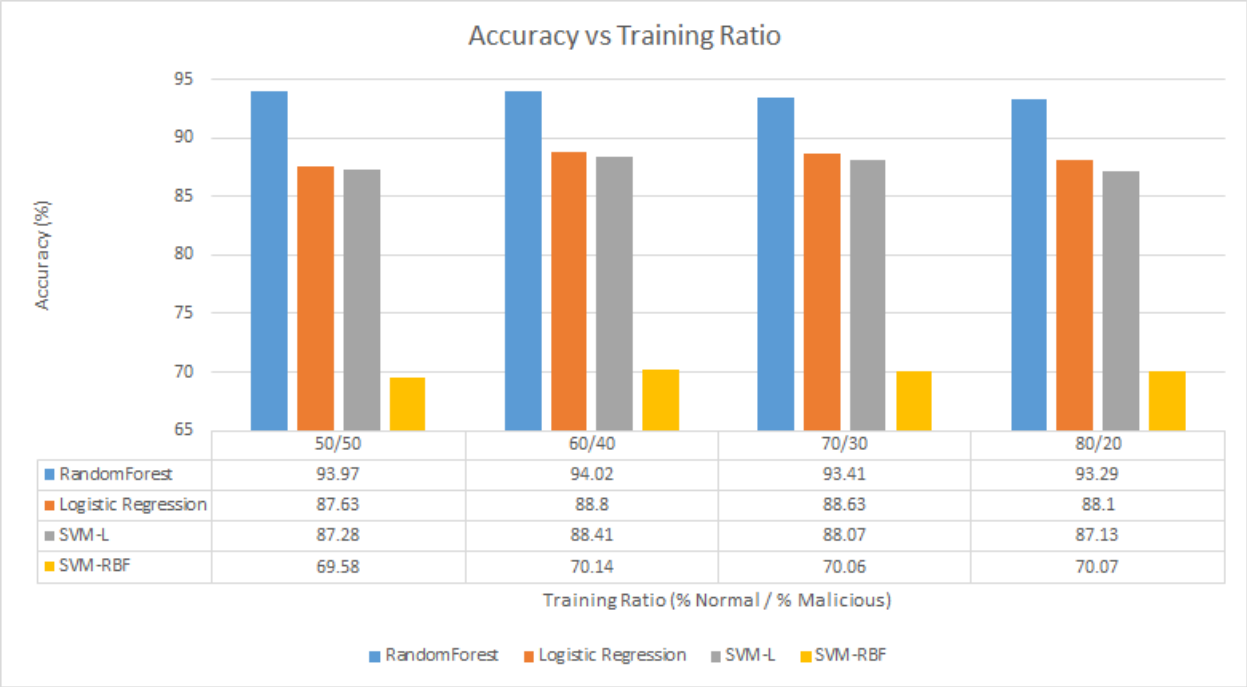

Figure 10. Accuracy with Different Training Ratio

Random Forest produced the highest accuracies across all training ratios, with accuracies all above 93%. The lowest accuracy for the Random Forest algorithm was obtained using the 80/20 split training method, while the lowest accuracy for the Logistic Regression algorithm was obtained when the 50/50 training method was used. The 80/20 training method also yielded the lowest accuracy for the SVM-L algorithm, however the lowest accuracy for the SVM-RBF was obtained using the 50/50 split training method. These results indicate that while the 60/40 training method yields the highest accuracies in malicious URL identification, the 80/20 and the 50/50 training methods yielded the lowest accuracies.

4.2 Iteration Two

For iteration two we looked more into ensemble algorithms as well a method for tagging. We also added features to our feature list and evaluated them. Our goal for iteration two was to improve upon the accuracies from iteration one test new methods.

4.2.1 Features

The new feature set contained 31 lexical and 3 host-based features. We ran a chi-squared test on the 11 categorical-lexical features in the set and calculated ANOVA F-values for the 20 numerical-lexical features in the set. The results of the chi-squared test can be found in Table 8 and the results of the ANOVA-F-Value test can be found in Table 9. The correlation heat map between all the features in the list can be found in Figure 11.

4.2.1.1 Chi-Squared Test

In the previous chi-squared test, the feature 'Username/Password in URL' was considered significant and could reject the null hypothesis at a 95% confidence level. This is not the case for the feature 'Username/Password in URL' in the second test. The other 10 features have very low p-values and can reject the null hypothesis. The p-value of the 'Username/Password in URL'

Features	Chi-Score	P-Value
Params in URL	291.5194003	7.31E-62
Queries in URL	8488.84037	0
Fragments in URL	848.9911149	1.87E-182
Check for Non Standard port	1036.61199	4.15E-223
Check Alexa Top 1 Million	12804.85726	0
Check for punycode	169.8805246	1.11E-35
Check sub-domains	1854.51143	0
IP based hostname	20225.85635	0
Check TLD	5852.198686	0

Table 8. New Chi-Squared Test Results

feature is still low enough to reject the null hypothesis at the 90% confidence level, so we did not take it out of our feature set.

4.2.1.2 ANOVA F-values

The results of the ANOVA F-value test indicate with 95% confidence that the class labels are dependent on all the numerical features. All of the numerical features have very low p-values, which also indicates they are strong features.

Features	F-Value	P-Value
Length of URL	6355.742229	0
Length of hostname	2412.265795	0
Length of path	5508.004089	0
Number . in URL	72.49164612	1.91E-61
Number @ in URL	104.7265241	3.40E-89
Number % in URL	512.9904889	0
Number _ in URL	658.5511411	0
Number ~ in URL	41.66978914	5.74E-35
Number & in URL	1357.719698	0
Number # in URL	37.48283806	2.21E-31
Number - in hostname	1914.056009	0
Number . in hostname	1953.511942	0
Number - in path	5364.662328	0
Number / in path	1922.416499	0
Number = in path	103.0353527	9.72E-88
Number ; in path	4.434759704	0.001388553
Number , in path	16.55564592	1.44E-13
Number . in path	394.0376352	0
Entropy of hostname	3079.634139	0
Number digits in hostname	5797.688167	0

Table 9. New ANOVA F-value

4.2.1.3 Correlation Heat Map

The heat map shows fewer highly correlated features than previous results. One exception is the high correlation between ‘Fragments in URL’ and ‘Number of # in URL’. These two features could be indicative of the same information since fragments often begin with the ‘#’ symbol. We remove ‘Number of # in URL’ as they both indicated the same information about the fragment.

There are a few similar cases in the heatmap but most of the other features do not show a high correlation with one another.

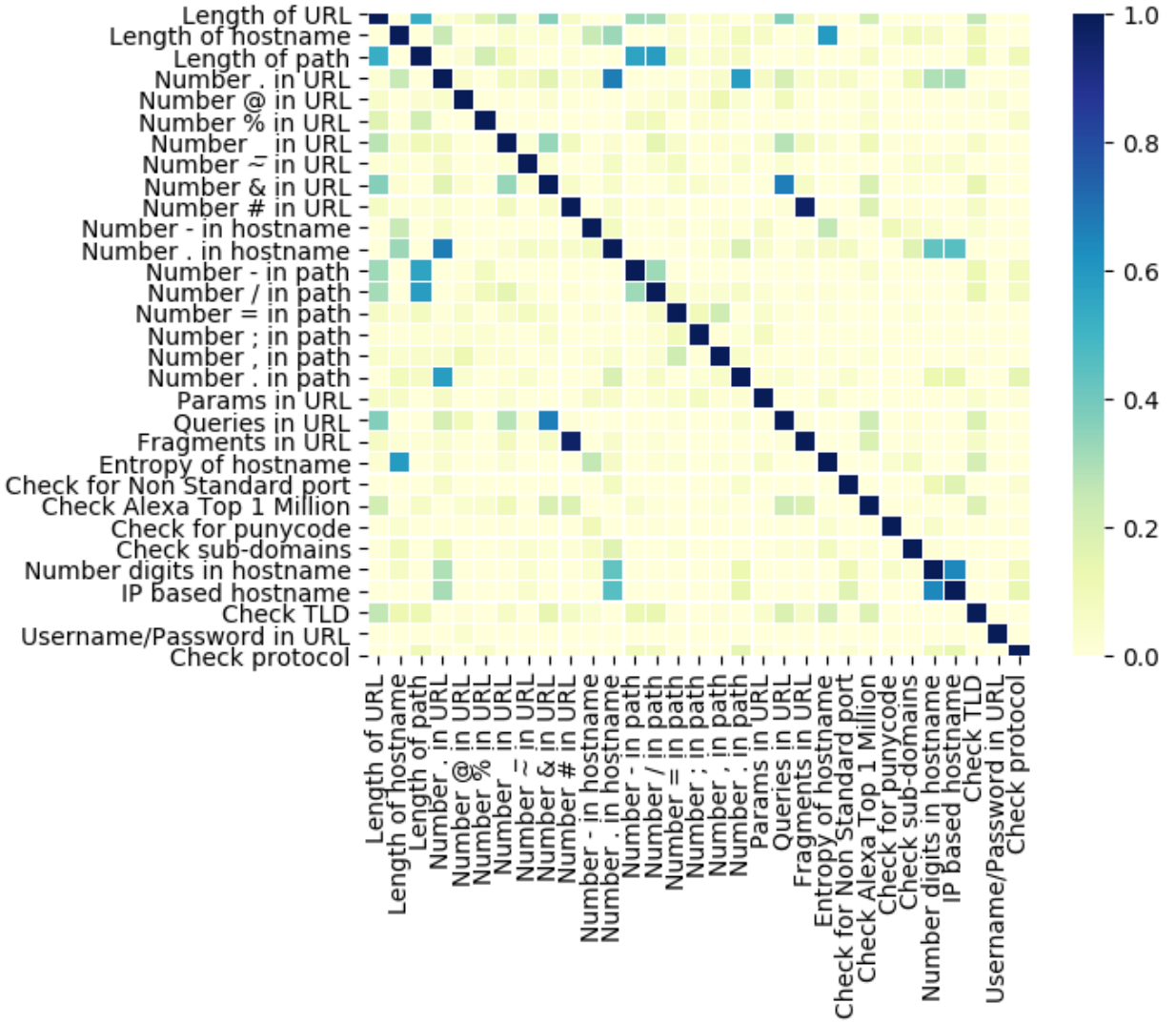

Figure 11. New Correlation Heat Map

4.2.2 Algorithm Performance

4.2.2.1 Tagging

We tested the tagging method using the Random Forest algorithm, since it had the highest accuracy among the algorithms in our previous results. We used a 60% Normal / 40% Malicious split and 4 different threshold values. The results of this test and the threshold values used can be found in Figure 12 and Figure 13. In Figure 13, a false positive refers to a 'Normal' URL that was given a malicious tag (e.g. 'malware', 'phish', etc.). A false negative is a malicious URL that was given a 'Normal' tag. The false positive rates were calculated by counting the number of false positives and dividing by the number of URLs. The false negative rate was calculated in the same way but using a count of false negatives. The tagging algorithm produced very high accuracies. With lower threshold values producing higher accuracies. The higher accuracies came with a trade-off as lower thresholds led to higher false positive and negative rates.

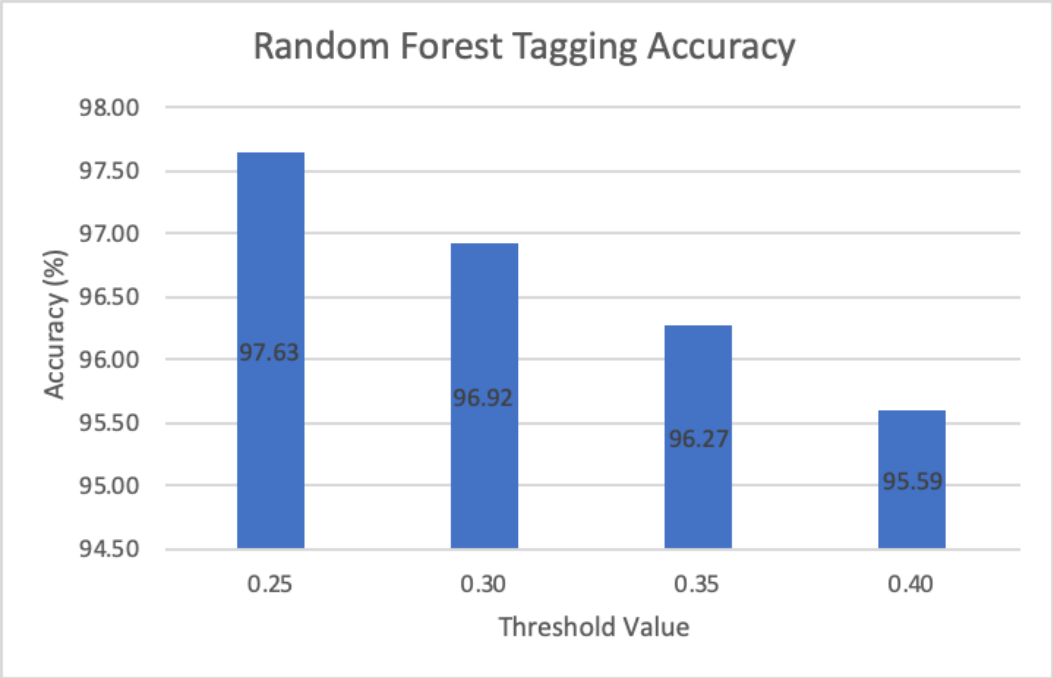

Figure 12. Tagging Accuracy Results

Figure 13. False Positive/Negative Rates

4.2.2.2 Algorithms Parameters

We performed a 2-dimensional optimization for the Extra Trees and AdaBoost algorithms on a 50% Normal / 50% Malicious split. The results of the AdaBoost optimization can be found in Figure 14 and Figure 15. In Figure 14, darker blue represents a higher accuracy. For this test, we varied the number of estimators from 40 to 80, stepping by 10. We varied the learning rate from 1 to 2, stepping by 1. Based on this result we chose to look at the number of estimators in more detail. In Figure 15, using a learning rate of 1, we varied the number of estimators from 65 to 75 and stepped by 1. These results showed the ideal parameters for the AdaBoost algorithm which were a learning rate of 1 and the number of estimators equal to 66.

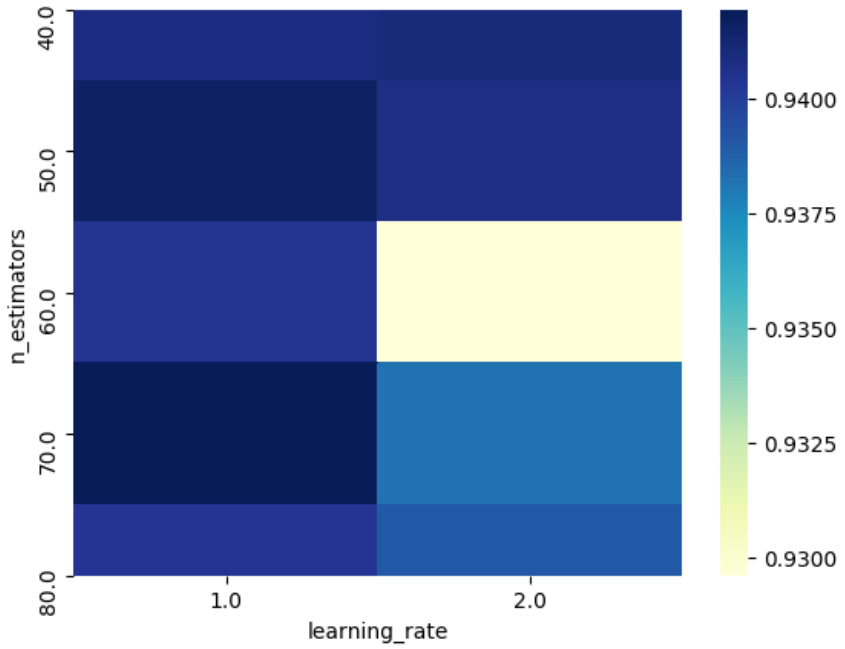

Figure 14. AdaBoost Optimization Heat Map

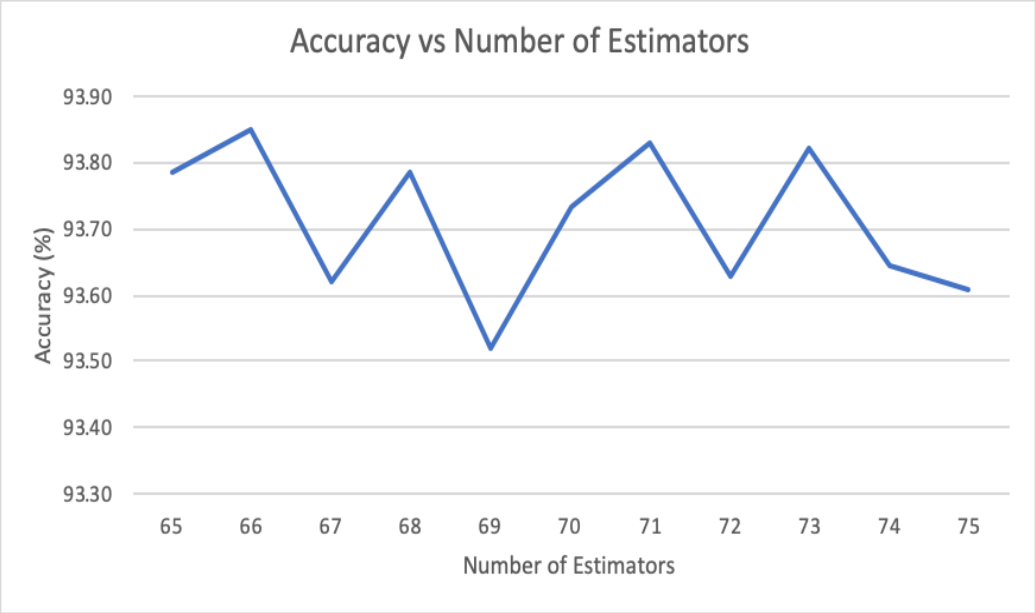

Figure 15. AdaBoost Optimization Line Chart

Next we optimized the Extra Trees algorithm. In Figure 16, we varied the number of estimators from 40 to 90, stepping by 5. We varied the number of minimum sample splits from 2 to 11, stepping by 1. Based on these results, we decided to look at the number of estimators in more detail. In Figure 17, we used a minimum sample split of 6 and varied the number of estimators from 70 to 100, stepping by 1. These results showed that the ideal parameters for the Extra Trees algorithm were a minimum sample split of 6 and the number of estimators equal to 91.

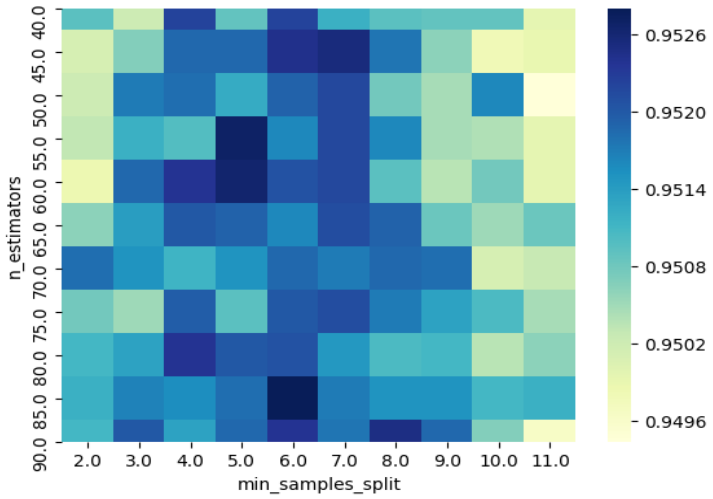

Figure 16. Extra Trees Optimization Heat Map

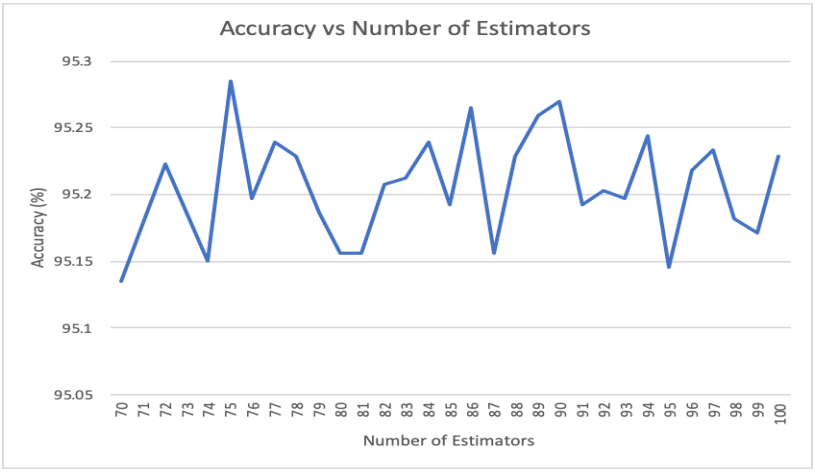

Figure 17. Extra Trees Optimization Line Chart

4.2.2.3 Training Ratios

Next, we tested the algorithms against the 4 different training ratios we used previously. The results of this test can be found in Figure 18. Similar to our previous results, the 60/40 split yielded the highest accuracies. The Extra Trees classifier had the highest accuracy of 95.10%. The 80/20 split had the lowest accuracies for Random Forest and Extra Trees. The 50/50 split

had the lowest accuracy for AdaBoost. These results indicate that the 60/40 training method yields the highest accuracies in malicious URL identification.

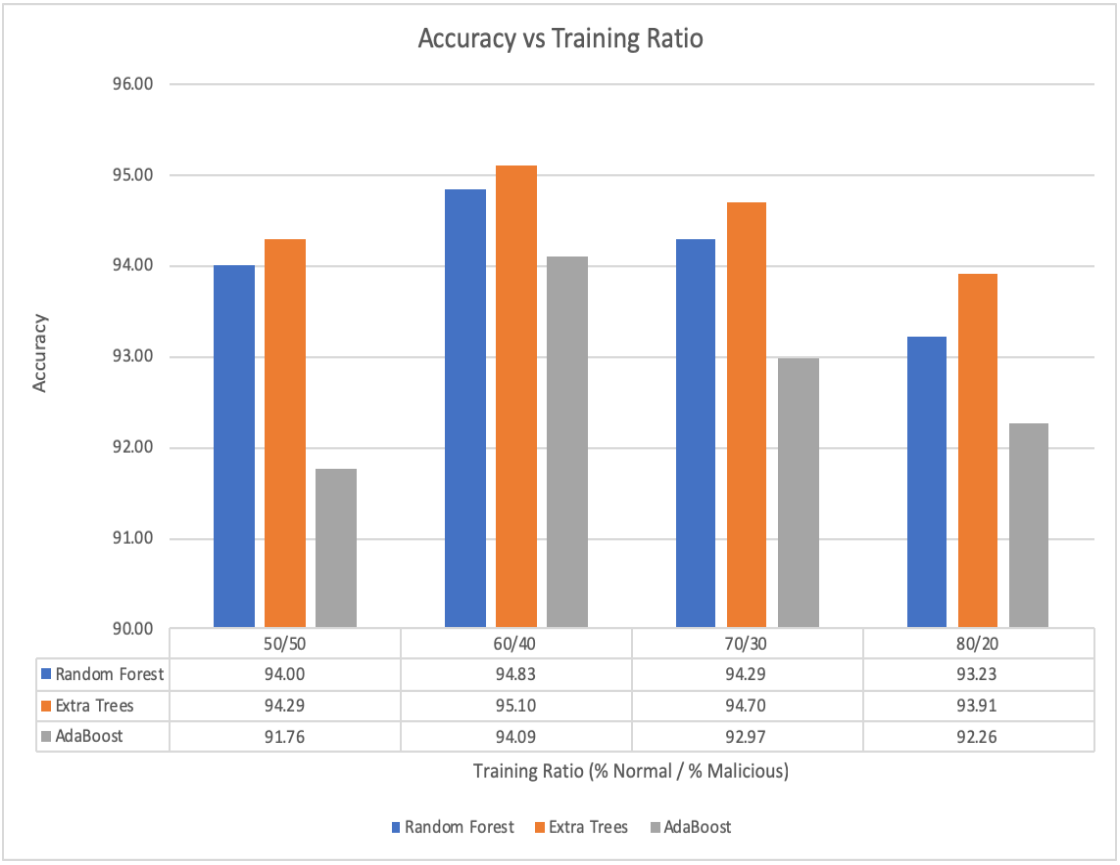

Figure 18. Ensemble Method Accuracies

5. Discussion

The major goals of the project were to identify the optimal feature set to use for URL classification, as well as test and evaluate the performance of multiple algorithms. We did this through an iterative process. In our first iteration, we implemented 29 lexical features and the best performing algorithms identified in previous research. The preliminary results from this first iteration showed that Random Forest was the best performing algorithm. It also showed that the full lexical feature set performed the best. In the second iteration, with the success of Random Forest, we implemented several other ensemble methods (i.e. bagging, boosting). These other ensemble methods performed similarly to Random Forest. Our findings showed that ensemble methods performed the best for this classification problem and that our full feature set gave the best performance.

Our findings showed that ensemble algorithms achieve a high accuracy using our full feature set. The ensemble algorithms performed with higher accuracy because they are better suited for multi-classification problems. Other algorithms tested, such as SVM linear and Logistic Regression, are more useful for binary classification. Another benefit of the ensemble algorithms is that they are less prone to overfitting and bias. Instead of relying on one algorithm to produce the best fit for the data, ensemble methods use multiple algorithms and the average of their results to generate predictions. Due to the nature of the data, we found that there was a lot of overlap between the classes of URLs. This leads to a more complex decision boundary which can be difficult to produce using one algorithm. We suggest that future research pursue ensemble type algorithms for similar multi-class problems.

5.1 Limitations

There were several limitations in our research process. One limitation was the short amount of time for the development and analysis of algorithms (i.e. 8 weeks). There were also limitations throughout our development process. We describe those limitations in the following sections.

5.1.1 Data

Although we were thorough in our data gathering and creating our testing and training sets, there are several limitations with our data. One limitation is that URLs labeled as normal may potentially not be normal. We assume that the normal URLs identified as normal by others' publications is correct. There is also the potential that URLs once labeled as normal have since been compromised. Another limitation is that our data may not be representative of all the categories. For each malicious URL category, data was collected from a single source, which may not be representative of all types of URLs in that particular category. Due to limited literature about ratios of malicious URLs in real web traffic, our training and testing data sets may not reflect real web traffic. This means our results may vary when applied to real world traffic. These limitations may have caused variance of our results and findings.

5.1.2 Features

Our results are entirely based on lexical features. Although we implemented and tested three host-based features, we were unable to evaluate them due to the fact that many of the URLs in our data set were no longer active. With our training data sets consisting of more than 50,000 URLs and the limited time we had to complete the project, the timeout was a huge problem. Whenever the feature extractor came across a URL that was no longer active it would take 25

seconds for the function to timeout. This issue can be fixed in future research with the gathering of more recent URLs that are still active.

5.1.3 Algorithms

Algorithmic limitations include the tools used to generate the models and the data used to train and test the models. We used the libraries from *Scikit-Learn* for the implementation of our algorithms. Since we did not develop or test our own implementations, we trust the developers at *Scikit-Learn* to develop well tested and reliable code. Also, the limitations mentioned for the data sets apply to the algorithm performance. Although the algorithms performed well in our tests, applying these models to real world web traffic may yield different results.

5.2 Future Work

Many of the limitations described in the previous section can be improved in future research. Labeled data sets of URLs are crucial, but difficult to find. Further testing of algorithms using more realistic web traffic and a broader sample of URLs could lead to improved models.

There is more research that can be done with features. Host-based and content-based features may provide more context when classifying a URL. There is a substantial risk that comes with generating content-based features because the process involves downloading the contents of websites which could contain malicious software. If this risk is managed correctly, the information gathered using these types of features could greatly improve model performance.

Another major area for improvement in future research is the models used. We focused our research on popular algorithms found in the *Scikit-Learn* library—mainly supervised algorithms. Future work could look into semi-supervised classification algorithms and how neural-networks

could be used to better classify URLs. Also, future research in this field could focus on researching and developing new supervised algorithms and ensemble methods to tackle this type of problem.

6. Conclusion

Cybercrime is on the rise as society shifts to a more online presence. Thus there is a need to detect cyber-attacks early to prevent damage to unsuspecting victims. We developed and analyzed machine learning algorithms to tackle one approach for early detection—URL classification. We gathered data from 5 different categories: normal, phishing, malware, ransomware, and botnet C&C. Using characteristics identified in previous research, we developed a comprehensive feature set made up of 31 lexical and 3 host-based features. Though we were limited to evaluating the lexical features, we found that all of our lexical features were relevant. Through development and testing of several algorithms, we discovered that ensemble algorithm methods performed the best with our set of lexical features. In particular, the algorithms Extra Trees and Random forest performed exceptionally well with accuracy. This work is one step in the right direction by allowing URLs to be accurately classified enabling early detection and prevention of cyber-attacks.

References

[1] Morgan, S. (2019). *2019 official annual cybercrime report* Herjavec Group.

[2] Sahoo, D., Liu, C., & Hoi, S. C. H. (2017). *Malicious URL detection using machine learning: A survey* Retrieved from https://www.openaire.eu/search/publication?articleId=od_____18::28e41a0b8b48aee3824dfa74f6fbcf9d

[3] Ma, J., Saul, L., Savage, S., & Voelker, G. (Jun 14, 2009). Identifying suspicious URLs. Paper presented at the 681-688. doi:10.1145/1553374.1553462 Retrieved from http://dl.acm.org/citation.cfm?id=1553462

[4] Ma, J., Saul, L., Savage, S., & Voelker, G. (Jun 28, 2009). Beyond blacklists. Paper presented at the 1245-1254. doi:10.1145/1557019.1557153 Retrieved from http://dl.acm.org/citation.cfm?id=1557153

[5] Patgiri, R., Katari, H., Kumar, R., & Sharma, D. (2019). Empirical study on malicious URL detection using machine learning. Paper presented at the *International Conference on Distributed Computing and Internet Technology, , 11319* Retrieved from https://doi.org/10.1007/978-3-030-05366-6_31

[6] Paganini, P. (2019). The most common social engineering attacks. Retrieved from https://resources.infosecinstitute.com/common-social-engineering-attacks/#gref

[7] What is phishing? Retrieved from https://www.phishing.org/what-is-phishing

[8] Davis, J.183,000 patients impacted by presbyterian health phishing attack. Retrieved from https://healthitsecurity.com/news/183000-patients-impacted-by-presbyterian-health-phishing-attack

[9] What is malware? malware defined, explained, and explored. Retrieved from https://www.forcepoint.com/cyber-edu/malware

[10] Barker, D. (2019, Aug 13,). Records of 85,000 involved in hospital attack. *The Daily World* Retrieved from https://www.thedailyworld.com/news/records-of-85000-involved-in-hospital-hack/

[11] Zhang, J., Porras, P., & Ullrich, J. (2008). Highly predictive blacklisting. *Usenix,* Retrieved from https://www.usenix.org/legacy/events/sec08/tech/full_papers/zhang/zhang.pdf

[12] McAfee.McAfee WebAdvisor. Retrieved from https://www.mcafee.com/consumer/en-us/store/m0/catalog/mwad_528/mcafee-web-advisor.html

[13] Peswani, S. (2018). Safelinks protection outlook - can you and should you disable it? Retrieved from https://www.thewindowsclub.com/safelinks-protection-outlook

[14] Bhatnagar, N. (2017). Getting started - machine learning. Retrieved from http://www.shanklab.com/machine-learning-getting-started/

[15] Fumo, D. (2017). Types of machine learning algorithms you should know. Retrieved from https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

[16] Brownlee, J. (2015). Basic concepts in machine learning. Retrieved from https://machinelearningmastery.com/basic-concepts-in-machine-learning/

[17] Canadian Institute for Cybersecurity. (2016). URL dataset (ISCX-URL-2016). Retrieved from https://www.unb.ca/cic/datasets/url-2016.html

[18] Strasak, F.Normal datasets. Retrieved from https://www.stratosphereips.org/datasets-normal

[19] PhishTank.PhishTank. Retrieved from http://phishtank.org/

[20] URLhaus.URLhaus database. Retrieved from https://urlhaus.abuse.ch/browse/

[21] Ransomware Tracker.Blocklist. Retrieved from https://ransomwaretracker.abuse.ch/blocklist/

[22] CyberCrime.CyberCrime. Retrieved from http://cybercrime-tracker.net/

[23] Bhattacharyya, S. (2018). Support vector machine: Kernel trick; mercer's theorem . Retrieved from https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d

[24] Patel, S. (2017). Chapter 2 : SVM (support vector machine) — theory. Retrieved from https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[25] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* John Wiley & Sons.

[26] Agrawal, A. (2017, March 31,). Logistic regression. simplified. Retrieved from https://medium.com/data-science-group-iitr/logistic-regression-simplified-9b4efe801389

[27] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32. doi:1010933404324

[28] Liaw, A., & Wiener, M. (2001). Classification and regression by RandomForest. Retrieved from https://www.researchgate.net/profile/Andy_Liaw/publication/228451484_Classification_and_Regression_by_RandomForest/links/53fb24cc0cf20a45497047ab/Classification-and-Regression-by-RandomForest.pdf

[29] Rocca, J. (2019). Ensemble methods: Bagging, boosting and stacking. Retrieved from https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

[30] Sahingoz, O. K., Buber, E., Demir, O., & Diri, B. (2019). Machine learning based phishing detection from URLs. Expert Systems with Applications, 117, 345-357. doi:10.1016/j.eswa.2018.09.029

[31] Chiew, K. L., Tan, C. L., Wong, K., Yong, K. S. C., & Tiong, W. K. (2019). A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Information Sciences, 484, 153-166. doi:10.1016/j.ins.2019.01.064