A MACHINE LEARNING BASED WEB SERVICE FOR MALICIOUS URL DETECTION IN A BROWSER

by

Hafiz Muhammad Junaid Khan

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

Electrical and Computer Engineering Department

Hammond, Indiana

December 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Quamar Niyaz, Committee Chair
Department of Electrical and Computer Engineering

Dr. Vijay Devabhaktuni, Committee Co-Chair
Department of Electrical and Computer Engineering

Dr. Xiaoli Yang, Committee Member
Department of Electrical and Computer Engineering

Dr. Sidike Paheding, Committee Member
Department of Electrical and Computer Engineering

Approved by:
Dr.Vijay Devabhaktuni

ACKNOWLEDGMENTS

I would like to express my sincere gratitude, appreciation message of thanks to my thesis advisor Dr. Quamar Niyaz for providing valuable guidance, mentoring, and encouragement for carrying out this research work. The door to Dr. Niyaz was always open whenever I needed some assistance or had questions for my research. I took several courses offered by Dr. Niyaz especially Big Data and Network Security that cleared too many aspects of this research work.

I would like to thank Dr. Vijay Devabhaktuni for his extensive support and help in providing valuable comments, support, and guidance throughout the graduate program and research. He always inspired me through his past research experience and guided me in the right direction all the time. I would acknowledge efforts from Dr. Xiaoli Yang for her outstanding guidance from start till the end of my research work. I would also like to thank Dr. Sidike Paheding for his insightful comments towards my work. Special thanks to the Department of Electrical and Computer Engineering for financially supporting me through the teaching assistantship.

Finally, I would like to express my deep gratitude to my parents and my wife for their enduring support and endless encouragement throughout the process of research and writing this thesis. This accomplishment would not have been possible without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Malicious URLs pose serious cybersecurity threats to the Internet users. It is critical to detect malicious URLs so that they could be blocked from user access. In the past few years, several techniques have been proposed to differentiate malicious URLs from benign ones with the help of machine learning. Machine learning algorithms learn trends and patterns in a dataset and use them to identify any anomalies. In this work, we attempt to find generic features for detecting malicious URLs by analyzing two publicly available malicious URL datasets. In order to achieve this task, we identify a list of substantial features that can be used to classify all types of malicious URLs. Then, we select the most significant lexical features by using Chi-Square and ANOVA based statistical tests. The effectiveness of these feature sets is then tested by using a combination of single and ensemble machine learning algorithms. We build a machine learning based real-time malicious URL detection system as a web service to detect malicious URLs in a browser. We implement a chrome extension that intercepts a browser's URL requests and sends them to web service for analysis. We implement the web service as well that classifies a URL as benign or malicious using the saved ML model. We also evaluate the performance of our web service to test whether the service is scalable.

1. INTRODUCTION

The Internet as we know today has grown rapidly in recent years. It has become an essential part of our daily lives in today's digital world. It is the way to interact and aggregate abundance of information from different sectors around the globe including government, industry, academia, and private organizations. The Internet is still in continuous growth with the penetration of smartphones. Because of the growth and technological developments, Internet users can get information from any part of the globe. As a result, they are being exposed to information from different known or unknown sources that lead to various adverse phenomena such as cybercrimes and cyberbullies.

1.1 Background and Motivation

According to Internet World Status, the number of Internet users has enormously increased from 558 million in 2002 to 4.4 billion in 2019, which is approximately 58.6% of today's world population. As shown in Figure 1.1 that Asia alone has 55% of the world population and contributed to 54.2% of the Internet growth with around 2.36 billion users alone. The same source explained that Internet usage has grown by 1157% worldwide since 2000, whereas North America has the largest proportion of the Internet users of 89.4% holding 4.7% of the world population surpasses Europe by 1.7%, which has 10.7% world population [1].

Another report from "We Are Social" reveals that Internet users are growing by an average of more than one million every day [2]. The key factor of this increasing growth is the current usage of mobile devices and social media platforms. According to the report, there are 5.11 billion mobile users, 4.39 billion Internet users, and 3.48 billion social media users as of 2019. Internet user's growth has accelerated by 366 million new users with a rate of more than 11 users per second compared to the last year. One factor that contributed to this increasing growth of the Internet users is the use of social media platforms. The worldwide social media users have grown to almost 3.5 billion at the start of 2019.

Figure 1.2 shows Internet world penetration rate that indicates that each continent has progressive Internet penetration rate, which resulted from the popularity of social media platforms and simplicity of web applications. The unprecedented growth of websites, social media and kinds of information available through them have attracted cybercriminals to perform malicious activities. These activities have become so frequent in the last few years that resulted in a huge increase of malicious websites with the continuous change of malware development and deployment scenarios. According to Forrester's report, 95% of the data breaches are from three popular industries: government, retail, and technology [45]. They are popular because they contain the top-level personal identification information (PII).

WORLD INTERNET USAGE AND POPULATION STATISTICS
2019 Mid-Year Estimates

World Regions	Population (2019 Est.)	Population % of World	Internet Users 30 June 2019	Penetration Rate (% Pop.)	Growth 2000-2019	Internet World %
Africa	1,320,038,716	17.1 %	522,809,480	39.6 %	11,481 %	11.5 %
Asia	4,241,972,790	55.0 %	2,300,469,859	54.2 %	1,913 %	50.7 %
Europe	829,173,007	10.7 %	727,559,682	87.7 %	592 %	16.0 %
Latin America / Caribbean	658,345,826	8.5 %	453,702,292	68.9 %	2,411 %	10.0 %
Middle East	258,356,867	3.3 %	175,502,589	67.9 %	5,243 %	3.9 %
North America	366,496,802	4.7 %	327,568,628	89.4 %	203 %	7.2 %
Oceania / Australia	41,839,201	0.5 %	28,636,278	68.4 %	276 %	0.6 %
WORLD TOTAL	7,716,223,209	100.0 %	4,536,248,808	58.8 %	1,157 %	100.0 %

Figure 1.1. World Internet Usage and Population Statistics [1]

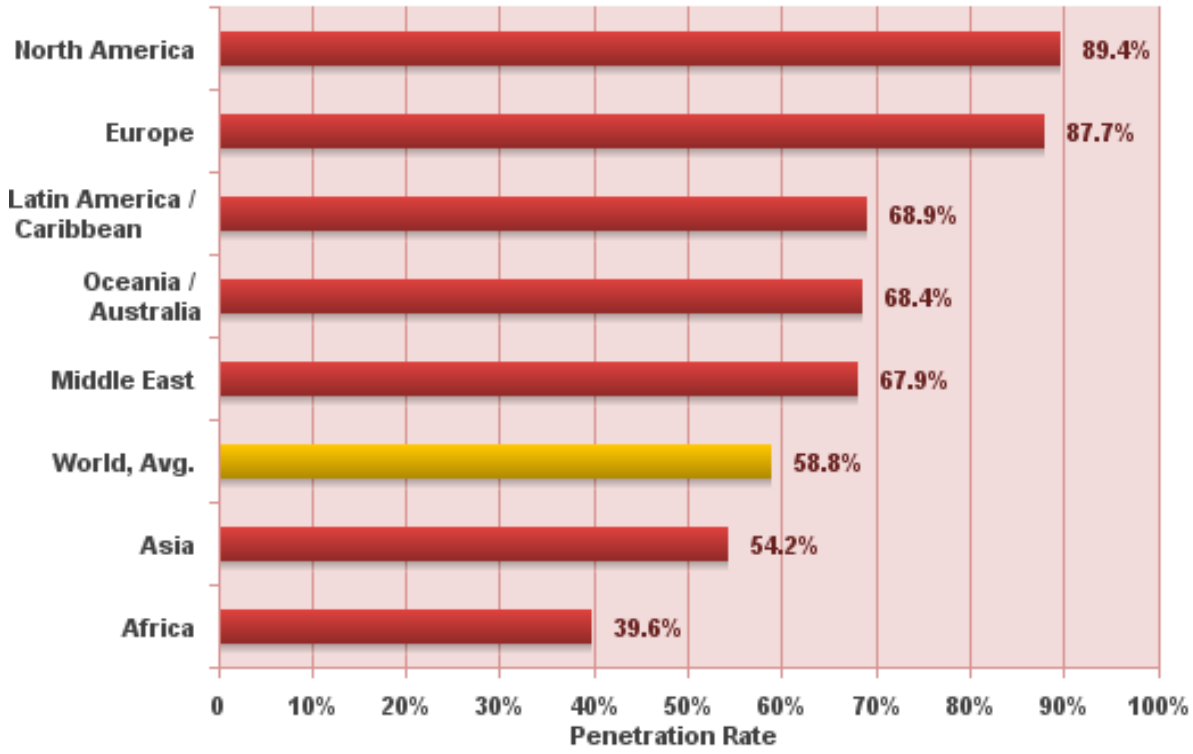


Figure 1.2. Internet World Penetration Rates by Geography [1]

Clark School at the University of Maryland reported that there is a constant adversarial attack on computers with Internet access every 39 seconds on an average [3]. Due to this continuous rate of cyberattacks, most of the small and medium scale businesses are exposed to security risks and data breaches as they do not usually consider all the security risks and are under the radar of adversaries. According to Kelser Corporation, 65% of the cyber-attacks are aimed at small and medium-scale businesses [46]. Healthcare, education, and energy sectors are the prime targets for cyber adversaries. For example, the patients' PII data at healthcare units can be used for insurance fraud or identity theft. Education sector including colleges and universities hold PPI ranging from social security numbers to addresses to bank information. Similarly, the energy sector uses technology and communication devices that can be hacked to put the economy of a country at risk.

Another sector that is expanding rapidly along with security risks is the Internet of Things (IoT). It includes interconnected networks of small devices and home appliances such as connected

security devices, cameras, alarms, and cars. IoT has made many things simpler and easier. It is estimated by Symantec Internet Security that around 200 billion devices will be connected by the end of 2020 [4]. As these devices are interconnected and they communicate with each other via Internet, there is always a risk of data breach and network attacks. Other factors that make these devices more vulnerable to cyber-attacks are insufficient authentication and authorization, insecure web interface, network access, mobile interface, and cloud interface.

Malicious websites are designed and created by cybercriminals to access or manipulate user information in an unauthorized manner. According to HelpNet Security, 40% of the malicious URLs were found in good domains where the attackers attempt to run scripts by injecting their malicious code in those legitimate domains [35]. The attack happens when the victim visits those legitimate websites containing suspicious activities such as drive-by-download, malware, spamming, and phishing.

1.2 Types of Malicious URLs

A malicious URL is designed to perform malicious activities such as scams, theft, attack, and fraud. There are several ways to deceive the victims using phishing, spamming, spoofing, malware, and website defacement.

1. Malware: Malware word is a concatenation of two words: malicious and software [13]. Its purpose is to run a piece of code in victims' computer to obtain unauthorized access or infect files. Virus, Worm, Spyware, Trojan, and Ransomware are different types of malware. They are commonly considered to be associated with computer software or file system, but they can be used to contaminate corporate websites and cloud systems. Website malware is used in defacement to replaces the original content of a website by any message or activity causing threat to the organization and its consumers. Malvertising is another trick used by cybercriminals to replace the original content of advertising with their ads. These ads could redirect website visitors to some blacklisted domains. According to SiteLock Website Security, websites experience an average of 58 attacks per day and search engines are blacklisting only 17% of the infected websites [5].

2. Spoofing: Spoofing is another trick used by cybercriminals to let the victim believe that they are communicating with a trusted source [14]. The most common spoofing techniques are website and email spoofing. In website spoofing, the attacker usually replicates the exact design and functionality of legitimate websites in their servers. The goal of these websites is to get victim's personal information such as username, password, and credit/debit card information by making them believe that they are visiting a trusted website. Email spoofing is one of the most common types of attacks as cybercriminals usually send emails along with malicious attachments such as malicious files. According to CAIDA study Internet users and organizations encounter 30,000 spoofing attacks every day [6]. Another report by Proofpoint shows that on an average organization were targeted by 18.5 attacks per quarter in 2017 however this average has been increased to 28 in the first quarter of 2018 [47]. The same source explains another significant 25% increase in email fraud attacks in organizations.
3. Phishing: Phishing is commonly achieved through deceptive emails to gain personal information. Unlike spoofing, phishing emails usually provide links to a fraudulent website that requires end-user information and other sensitive data [15]. According to Webroot 1.5 million new phishing are created each month [36]. Another report from Retruster states that phishing attempts have grown to 65% in the last year whereas 76% of the businesses are being affected by phishing attacks when compared to last year [37].
4. Defacement: Website defacement involves altering the original content or appearance of a website with malicious content [38]. Hackers usually break into the web server and replace the hosted website with their malicious one. This could cause phishing, code injection and cross-site scripting. Common targets can be government and corporate websites.

1.3 Machine Learning in Cybersecurity

Machine learning has improved dramatically over the past two decades in several domains. It has been using in computer vision, speech recognition, natural language processing, and robotics. Modern research shows that it is progressing very fast and capable to learn and trainthe model parameters using past data and can predict or classify new data based on its trained metrices. With its ability to detect patterns in data through, it has been increasingly used to uncover cyber threats by automatically predicting them before making any havoc.

Microsoft's Windows defender advanced threat protection is a cybersecurity platform by Microsoft which is used for prevention protection, breach detection, and automated investigation. It stopped Trojan malware in early 2018 which was an attempt to install malicious cryptocurrency miners on hundreds of thousands of computers [39]. It restricted this attack with the help of numerous layers of machine learning algorithmic structure that detect and prevent perceived attacks. It utilizes cloud AI with numerous levels of machine learning algorithms. In the same way Chronicle [40], a company owned by Alphabet is a cybersecurity company that analyzes a large amount of data and uses machine learning for threat detection and malicious pattern recognition. SQRRL [41], acquired by Amazon, has designed a platform that searched across the network traffic to find code which can escape the safety measures in place. It uses machine learning to create action maps that acts as a visual representation of a computer network.

Machine learning can be more effective when classification or prediction needed in real-time by reducing the amount of time spent on routine tasks. It can make a task automated, more simple, proactive, less expensive and far more effective and reliable. It helps to learn the pattern by using previous rich data which represents many potential scenarios. According to [32], security devices that provides real-time alerts and events based on signature-based anomaly detection has some limitations. It may not detect unknown attacks or malicious activity and detection and protection mechanism is typically limited and not shared. This was a common problem for organizations on how to detect and identify a zero-day attack. These types of attacks are more challenging to detect as they are slow and low. In these scenarios, machine learning methods show promising results with the capability to adapt to new trends and respond to new techniques and attacks while continuing to address the known attacks. James et al. in their work shows the leverage of using machine learning when compared to other traditional methods [32].

1.4 Thesis Outline

The thesis is organized in the following manner:

- In Chapter 1, we discussed the problem and motivation of this study.
- In Chapter 2, we provide literature survey for the previous work done related to this domain.

- Chapter 3 discusses our methodology to identify generic features. We also discuss the performance results with generic features.
- In Chapter 4, we discuss the real-time URL detection system and how we achieve and classify URL with high throughput.
- Chapter 5 concludes the thesis and provides an insight for future work.

2. LITERATURE SURVEY

In this chapter, we discuss the previous and existing works on malicious URL detection with different approaches. Mohammad et al. have used lexical features for malicious URL detection [5]. The authors obtained features from raw URL dataset without performing any network calls or host information. They mainly used lexical features from URL due to speed, and lightweight computation. They classified different types of attacks and created a separate feature for each type of attack. A total of 79 features were extracted from the UNB dataset with a total of 1,65,366 records. They used correlation-based feature selection (CFS) and information gain for feature selection. A few machine learning algorithms such as KNN, Random Forest, and C4.5 were used to classify the URLs. In addition, they also discussed different obfuscation techniques as an extension to previous work by using AttributeSelection with InfoGainAttributeEval as an attribute evaluator. They covered a wide range of attacks (phishing. malware, spam, defacement) being performed by maliciousURLs. They reported 97% accuracy for multi-class classification with Random Forest, whereas 99% accuracy for binary classification.

Faeze Asdaghi developed an effective way of discovering web spam through a feature selection technique called Smart-BT [9]. They created four-set features named as content-based, link-based, and transformed link-based, and All-Features. They proposed a backward elimination approach using IBA (Index of Balance Accuracy) values of chi-square, information gain, along with the gain ratio to identify the useful features and their impact on various ML algorithms. The major purpose of IBA is to introduce a weight measure that shows better classification in an unbalanced dataset, especially in the minority class as they contain low data samples and information. They have analyzed the impact of dimensionality reduction on classification accuracy of an unbalanced dataset by using the WEBSPAM-UK2007 dataset with a predefined set of features and studied the impact of dimensionality reduction on increasing classification performance. They have used several machine learning algorithms with different IBA values, features set and methods and provided a detailed comparison report. They reported improvement in the classification results with Smart-BT compared to the well-known feature selection techniques such as Ranker, Forward Selection, and Genetic Algorithm. Their method also shows efficiency in low dimension dataset by selecting the near-optimal features.

Hung et al. developed URLNet, a Convolutional Neural-Network (CNN) based deep learning framework that uses characters and words of the URL string to capture the semantic information to classify malicious and benign URLs [10]. Their work showed a promising way of URL Detection through deep learning. They discussed the limitation of features obtained by using bag-of-words and statistical features like the length of different segments in URL. They used CNN to get useful structural information for the URLs with two different datasets generated by characters and words of URLs. Word Level CNN is like character level CNN except that the convolutional operators are applied to words. The URLs dataset was collected from VirusTotal. They created the features set using the entire training corpus with all the unique words as a dictionary. This method gives another way of classifying malicious URLs by catching several semantic information through the URLNet which existing methods based on bag-of-words features could not. It offers a significant jump in AUC over baseline.

Ram et al. [11] are only considering phishing URLs and found a way to identify them using machine learning. They detected phishing attacks by using four different categories of features in their work. They used lexical, keyword, reputation, and search engine-based features on seven different ML classification algorithms. They reported an accuracy of 99.4% with 138 features while maintaining 0.5% of the false-positive and false-negative rate. Random forest classifier showed best results in most of the experiments. Apart from good results, there are some limitations to their work. They have used only one kind of dataset collected from PhishTank which may not contain the overall variation of phishing URLs. Second, search engine-based features will require some time to gather information as they require network calls whereas reputation-based features depend on blacklists and other historical statistics provided by third parties.

Another successful attempt of detecting malicious URLs through domain and word segmentation was done by Wei Wang [12]. Their work shows the importance of word segmented features and how it improves the detection capability by emphasizing the most used words in malicious domains. In [12], authors created seven different range of lexical features based on characters, basics, top-level domains, log-likelihood, words, and their different combinations. Three experiments using balanced data, filtered cellular data, and unfiltered cellular data, were performed on seven combinations of feature set using Logistic regression with lasso penalty. 10-

fold cross-validation was performed on training data and the largest value of the penalty parameter that gave Area Under the curve of one standard error was selected. To measure the effectiveness, area under the curve (AUC) and misclassification rate (MCR) were used. In their work, models which have used word segmentation significantly decreases the misclassification rates and increases the AUC rates by 10% compared to other models that did not use word segmentation. Their model also showed interpretable results that show which set of words attracts the victim to malicious sites.

Guolin Tan designed MalFilter to detect malicious URLs in real-time [42]. Their work explores the malicious URL detection system by considering large-scale real-time networks. They have used server-based features, user-based features, URL based features and referral-based features. They have implemented three tiers of modular filtering system which comprises packet parser, training, and filtering. Packet parser is responsible for feature generation by parsing the header files of network packets. The training module performed best when to be used with non-linear classifiers such as Adaboost. The filtering module is implemented on Spark on a cluster of 26 nodes responsible for training and classification. Benning and suspicious URLs are separated with the help of the filtering module. The proposed system effectively reduces the load to an average of 28.99% while achieving the recall rate of approximately 90%.

Kurt Thomas designed a real-time URL spam filtering scalable system capable to detect and differentiate between e-mails and twitter spam [43]. System flow of monarch comprises different modules including URL aggregation, feature collection, feature extraction, and classification. URL aggregation module aggregates URLs from two sources for training and testing purposes. Feature extraction module visits each URL through the Firefox web browser to collect the content including HTML, page links, JavaScript activities, and popup windows. Feature extraction module is responsible for converting raw features into features understood by the classification engine such as tokenizing URLs into binary features and HTML content conversion into a bag of words. The classification module is responsible for detecting and classifying the injected URLs. Two main algorithms were used for classification called Logistic Regression with L1-regularization and Stochastic Gradient Descent. To handle real-time traffic and scalability, Hadoop distribute file system and spark is used on Amazon EC2 double extra-large instances. The

results reveal that Monarch on cloud infrastructure can achieve a throughput of 638,000 URLs per day with an overall accuracy of 91% with 0.87% of false positives.

Our approach of detecting malicious URLs is different as we attempt to identify generic features using different datasets. Machine learning algorithms either single or ensemble learners learn parameters by learning the patterns in the dataset. We have identified generic features that can work in different dataset holding different trends and patterns. Previous work by other researchers have used different datasets but did not focus on generic features. Similarly, our approach to real-time detection is different as it includes the complete cycle of request and response architecture where the source that generated the request waits for the response back from the server. Previous work on real-time detection has not mentioned the request-response architecture in detail as they were more focused on traffic flow and network filtering in real-time. For this purpose, we implemented and analyzed the request-response web service calls on client-server architecture with the help of Google Chrome Extension.

3. IDENTIFYING GENERIC FEATURES

In order to find information on the Internet, we need to have an address or pointer to locate the information. The address points to a computer hosting information that can be physical or virtual. The IP address or hostname has been used widely as an address to those computers across the Internet. It is the mean of getting information and is used to identify the location of the resource. In contrast, URL contains other information along with the IP address or hostname such as protocol to be used, path, optional fragment identifier. Through URLs, we can visit a plethora of information by accessing different websites. URLs are also used in web services including application or cloud technology for data messaging through Internet. Web services have been widely used in the corporate sector nowadays because of the rapid improvements in architectural design and scaling.

3.1 Overview of URL

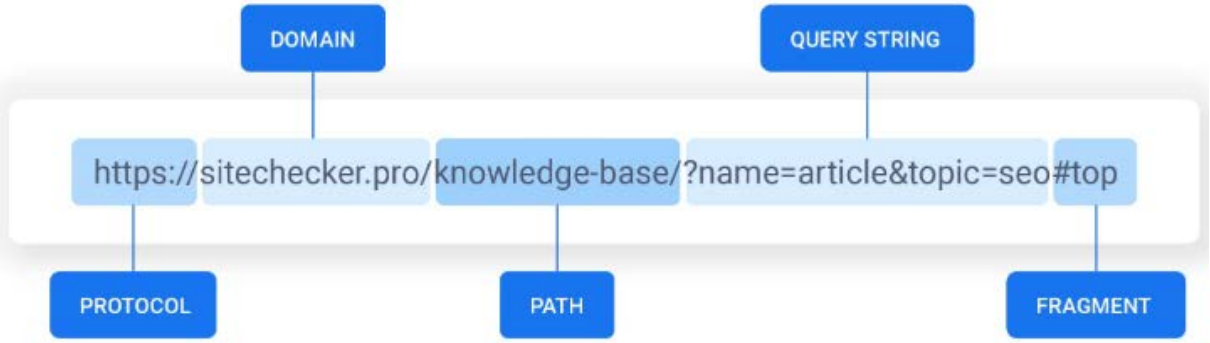


Figure 3.1. Uniform Resource Locator (URL) [33]

URL stands for Uniform Resource Locator also known as web address. It is a reference to a web resource that specifies resource location in a remote server. The resource could be a web page, text file, email, images, and database access. A URL has two main components known as protocol identifier and resource path. The protocol indicates which protocol is being used. The resource path consists of the IP address or domain/hostname. The protocol and resource can be identified in Fig 3.1. Path in URL consists of a sequence of path segments separated by a forward slash (/). The slashes in the URL are used to separate directory or filenames. URL may contain a

question mark (?) which represents an optional query for passing non-hierarchical data. Its syntax is not well defined, but by convention is most often a sequence of attribute-value pairs separated by a delimiter.

In this work, we focus on identifying the most significant features that can give enough information to detect malicious URLs. Fig. 3.2 shows the detailed workflow for the model development of our malicious URL detection system. It consists of three stages. The first stage deals with data processing, along with feature creation and extraction. We use different statistical methods to identify the most significant features. The second stage is the optimization and tuning of various ML algorithms. This stage involves choosing a set of optimal hyper-parameters for each algorithm. Finally, a majority voting classifier is used to classify the URL into benign or malicious in the third stage. In the following subsections, we provide the details of our approach.

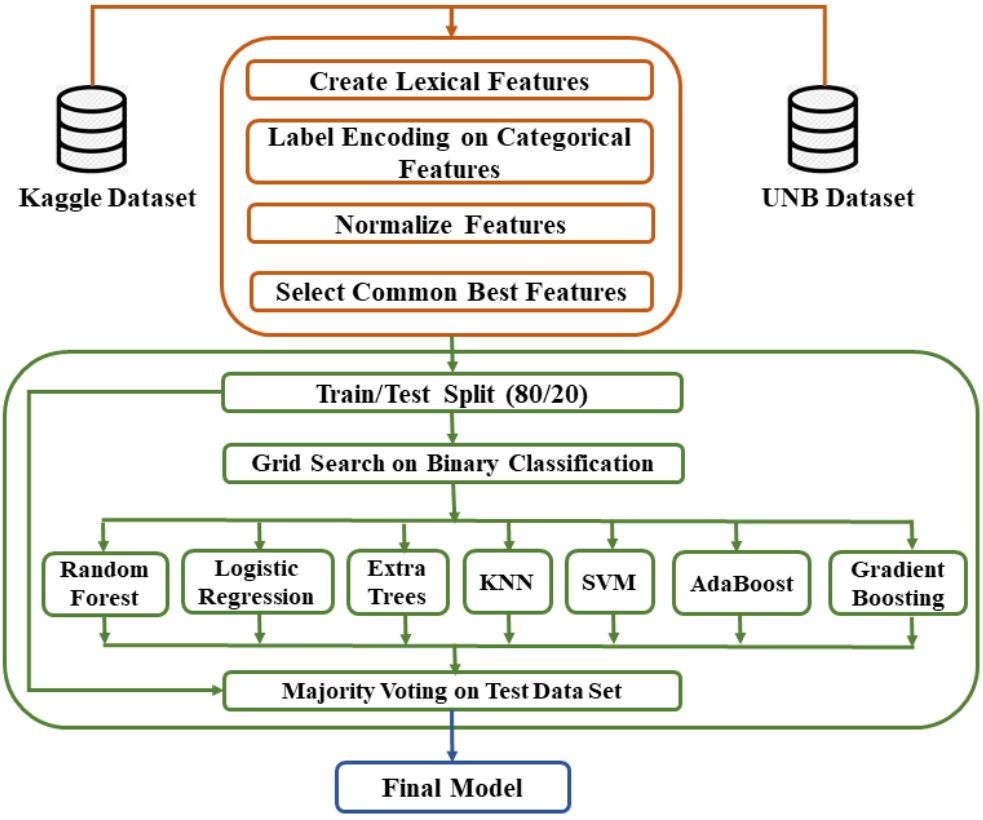


Figure 3.2. Workflow of Malicious URL detection Model

3.2 Dataset Description

We have used two publicly available datasets in our work. The first one is released by Mammun et al. [18], and the second one is available through the well-known online data science community platform, Kaggle [19]. We refer them as UNB and Kaggle datasets, respectively. In UNB dataset, the authors selected 1,65,366 URLs. There are 35,378 benign, 12,000 spam, 9,965 phishing, 11,566 malware, and 96,457 defacement URLs in the dataset. They collected benign URLs from Alexa Top websites by removing the duplicate and domain-only URLs, whereas malicious URLs were collected from OpenPhish, DNS-BH, Zone-H, and WEBSPAM-UK2007. The Kaggle dataset contains 4,20,464 URLs in which 82% are benign, and 18% are malicious URLs. The dataset includes URLs from Phishtank, JWSPAMSPY, DNS-BH, and Majestic.

3.3 Feature Engineering

We created 106 lexical features for the URL classification by referring [18], [21], and [22]. There are 41 word-based features, 36 count-based features for alphanumeric characters, URL entropy, domain, host, path, parameters query and remaining 29 features includes special character count. Word-based features represent top words used by attackers for obfuscation and defacements like secure, webscr, login, ebaysiapi, signin, banking, confirm, and signon. Although authors in [5] considered the presence of alphanumeric characters as features, we have counted the frequency of each character that appears in the URL. Our derived features can be found at appendix A.

We have used two scoring functions for feature selection, Chi-square [16] and ANOVA [17]. Chi-square is a statistical method that provides two types of statistical tests called “goodness of fit” and “test for independence.” In our case, we have used the latter one, which compares two variables and checks if they are related. If the target variable is independent of the feature, that feature can be discarded. ANOVA is known as Analysis of Variance. In statistics, the variance represents the data spread out as how far does any value varies from the mean value of the distribution. ANOVA can determine whether the mean of a certain group is different or the same with the help of F-Scores. F-Scores are the statistical F-test and represent the ratio of mean squares.

3.3.1 Chi-Square

Chi-Square test is statistically used to measure the goodness of fit and independence test [16]. The goodness of fit is used to know in a frequency how many cases fell into one category whereas the test of independence shows if there is any relation between the variables. The test of independence shows the relationship between the independent variables/features with the dependent variable or response feature. Through Chi-Square, we can check which features set are highly correlated and dependent on the response. Higher values of Chi-square show the independence of the hypothesis is incorrect and there is a high dependency between the label/response class and the tested feature. Any two variables can be observed by getting the observed count and expected count. Below equation 3.1 shows such deviation between the variables.

$$Xc^2 = \sum_{i=1}^{m} \frac{(O_i - E_i)}{E_i}$$

Equation 3.1 : Chi- Square equation [16]

3.3.2 ANOVA

ANOVA is a popular statistical method to analyze the variance in the dataset [17]. In machine learning, variance analysis gives information metrics through which we can determine whether the feature does a good job of accounting for variation in dependent variables. Analysis of variance in features or group can be explained as

$$ANOVA = \frac{\sum_{i=1}^{k}[\sum_{j=1}^{n_j} X_{ij}]}{\sum n_i}$$

Equation 3.2 : ANOVA equation [17]

Where k represents the number of features and n represents i_{th} sample data of the dataset. F-values are used for variable ranking and can be applied sequentially to all the variables in order to discriminate according to the classes. ANOVA is based on F-test to estimate the degree of linear dependence between variables.

Figure 3.2 and 3.3 show the feature importance scores using both chi-Square and ANOVA on Kaggle and UNB datasets. We selected top 60 features from both datasets that show the highest feature score.

3.3.3 Finalizing Common Features

With the help of the above Chi-square & ANOVA feature scoring techniques, we finalize the most common significant features in the datasets. We applied scoring techniques one by one on each dataset to find top 60 significant features set. Then, we identified common features in all the top 60 significant features sets. We found 47 features that were available in all the sets. Figure 3.4 and 3.5 shows the flow of the feature selection model whereas Table 3.1 lists finalized features after the feature selection process. Out of these 47 features, Features 1-4 show the presence of specific words in a URL. Feature 5-16 represents the frequency of specific symbols found in the URL. Feature 17-19 contain path, host, and URL length. The URL paths refers to the exact location of the file or asset, whereas the host represents the name or address of the webserver. Feature 20 checks the presence of any number. Feature 21 checks the URL path extension. Path extension is used to check if the URL is trying to access or download any file with a specific extension, such as “.exe”, and “.zip.” Feature 22-46 represents the frequency of alphanumeric characters. Feature 47 shows the entropy of query parameters. A query parameter is attached at the end of the URL that links to a specific action or file depending on the data being passed to the server. We have used Shannon’s Entropy which gives the information produced by the stochastic source of data. We then develop our ML-based malicious URL detection model with the help of these 47 selected features.

Figure 3.3. Feature Score from UNB Dataset

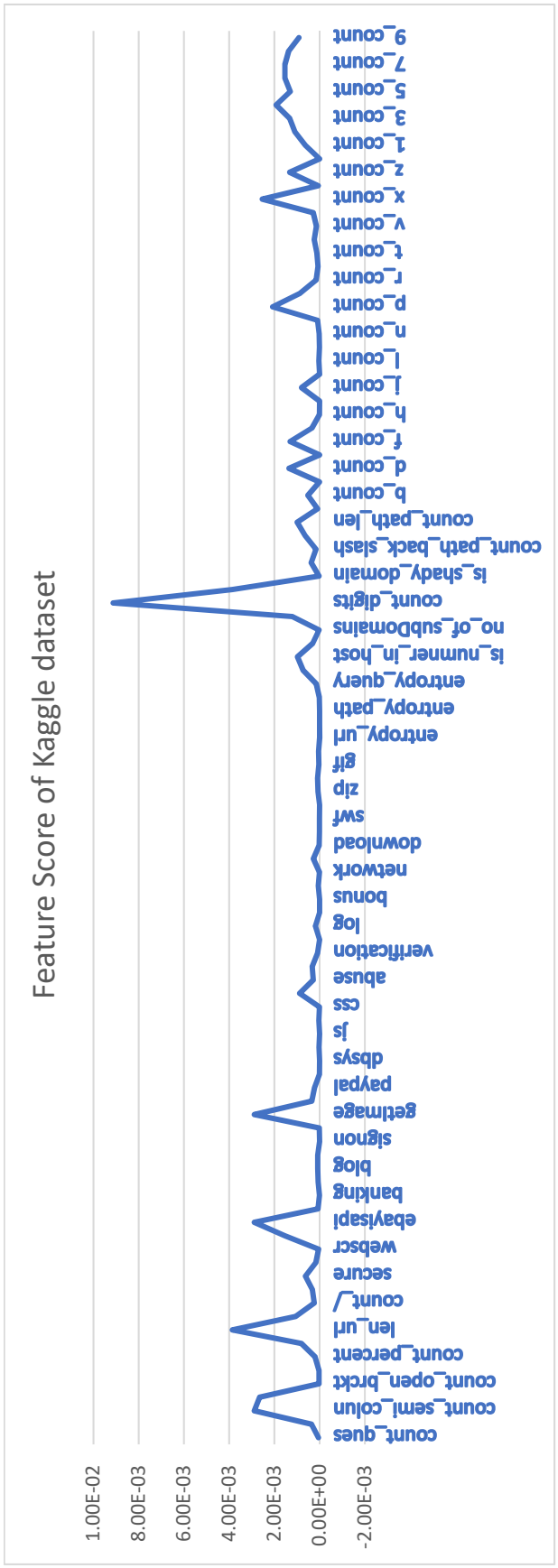


Figure 3.4. Feature Score from Kaggle Dataset

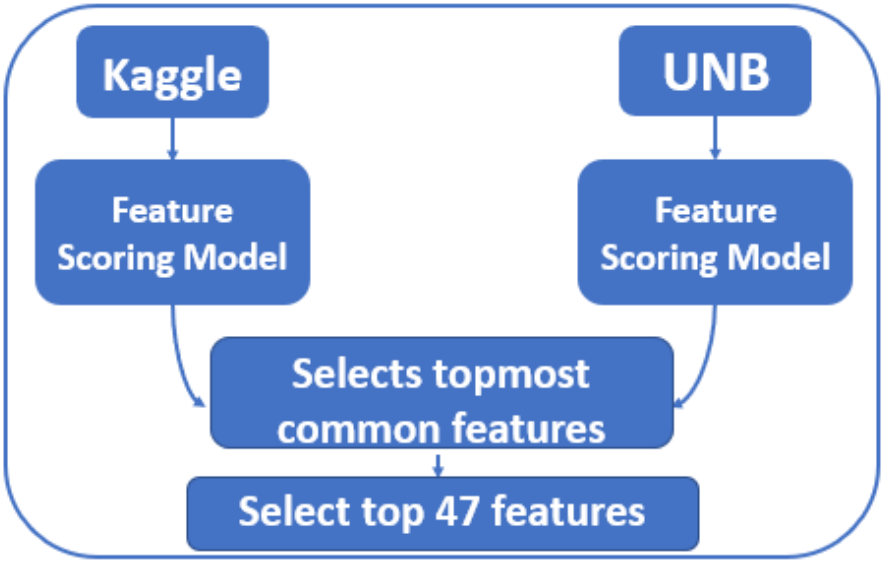


Figure 3.5. Feature Selection Model

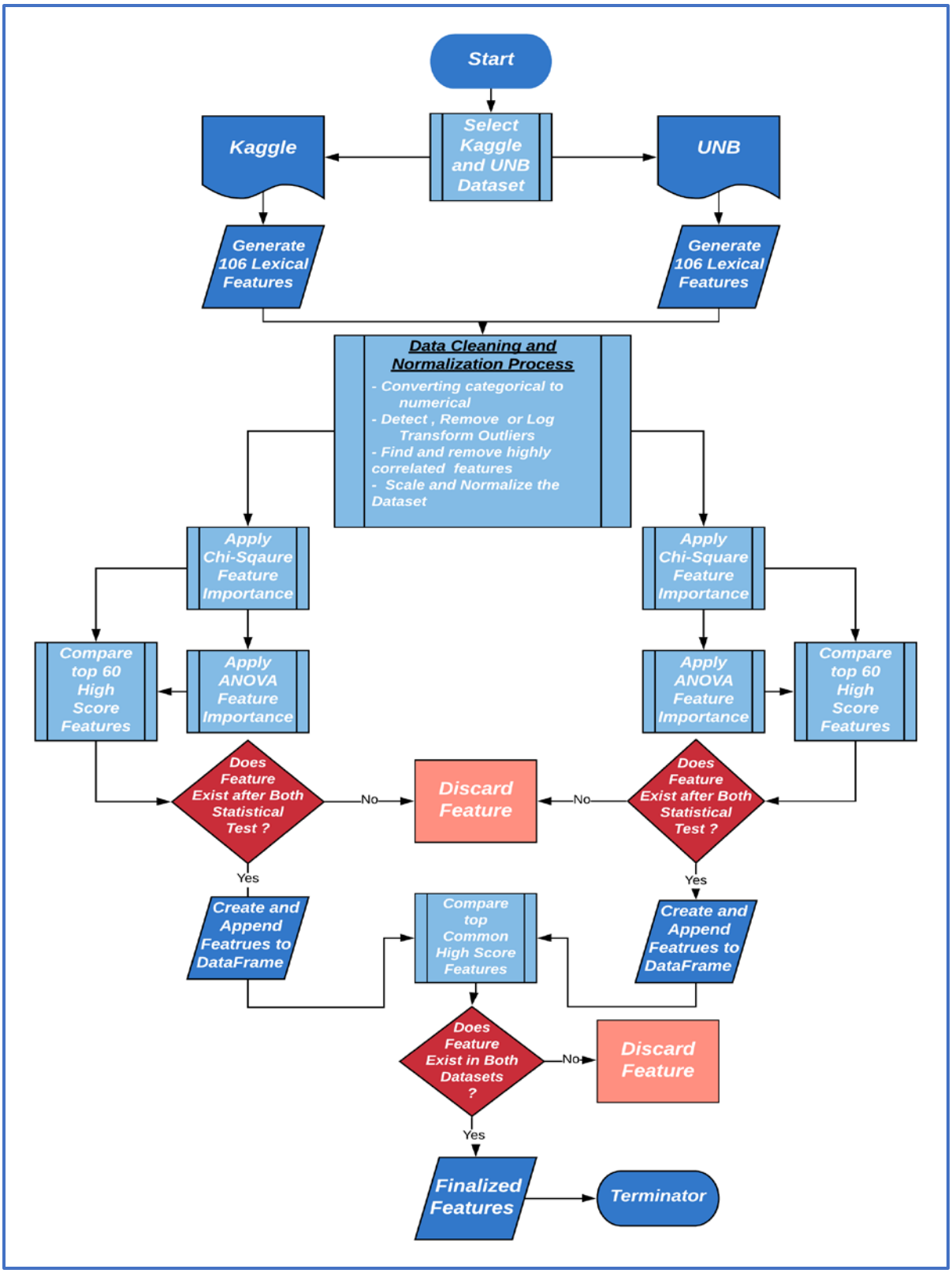


Figure 3.6. Steps involved in Feature Selection Model

Table 3.1. Finalized Selected Features

#	**Feature Name**	**Description**
1	Ebayisapi	Check word presence
2	getImage	Check word presence
3	Jpg	Check word presence
4	Log	Check word presence
5	Count_&	Count ‘&’ symbol
6	Count_/	Count ‘/’ symbol
7	Count_@	Count ‘@’ symbol
8	Count_dash	Count ‘- ‘symbol
9	Count_digits	Count total digits in URL
10	Count_equals	Count ‘=’ symbol
11	Count_letters	Count total alphabetical letters
12	Count_path_back_slash	Count back slashes in URL path
13	Count_path_dot	Count dots in URL path
14	Count_question	Count ‘?’ symbol
15	Count_semi_column	Count ‘;’ symbol
16	Count_symbols	Count total symbols in the URL
17	Path_length	Length of the URL path
18	Host_length	Length of the host name in URL
19	URL_length	Length of the URL
20	Is_number_in_host	Check digits in the host name
21	Path extension	Extension of the URL path
22	Freq_0	Frequency of 0
23	Freq_2	Frequency of 2
24	Freq_3	Frequency of 3
25	Freq_4	Frequency of 4
26	Freq_5	Frequency of 5
27	Freq_6	Frequency of 6
28	Freq_7	Frequency of 7

Table 3.1. continued

29	Freq_8	Frequency of 8
30	Freq_9	Frequency of 9
31	Freq_b	Frequency of b
32	Freq_d	Frequency of d
33	Freq_f	Frequency of f
34	Freq_g	Frequency of g
35	Freq_j	Frequency of i
36	Freq_l	Frequency of l
37	Freq_o	Frequency of o
38	Freq_p	Frequency of p
39	Freq_r	Frequency of r
40	Freq_s	Frequency of s
41	Freq_t	Frequency of t
42	Freq_u	Frequency of u
43	Freq_w	Frequency of w
44	Freq_x	Frequency of x
45	Freq_y	Frequency of y
46	Freq_z	Frequency of z
47	Entropy_query	Entropy of query parameters

3.4 Single & Ensemble Machine Learning (ML) Algorithms

To develop the malicious URL detection model, we have used various machine learning algorithms including Logistic Regression (LR), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and ensemble learning algorithms. Fig 3.6 shows the detailed workflow for the model development of our malicious URL detection system. After identifying the common features, we split the data and used grid search for different machine learning algorithms for best optimal accuracy. Later, we applied the majority voting technique to get the best result from majority votes.

3.4.1 Single Machine Learning Algorithms

K-Nearest Neighbor (KNN)

KNN can be used for classification and regression [25]. It is a non-parametric method that uses distance as a metric to classify by a plurality vote of its K neighbors. There are several ways to perform KNN. The three most popular methods for KNN are brute force, ball tree, and k-d tree. We have used all three algorithms in our case. Brute force is a tedious and time-consuming way of calculating the distance of each data sample in the data set whereas k-d and ball tree uses a tree data structure to further optimization.

Support Vector Machine (SVM)

A support vector machine is a supervised learning model used for classification and regression [26]. It classifies with the help of separating hyperplane between different groups of data. In SVM, data points can be treated as a p-dimensional vector and tries to find (p-1) dimensional hyperplane called linear classifier. Optimal results can be achieved by getting a hyperplane which has the largest separation called margin between the two classes. There are different kernel tricks in SVM which is used to transform the dataset from a lower dimension to a higher dimension in order to find the optimal hyperplane. We have used 4 different kernel tricks [27] such as Nystroem approx, Fourier approx, linear SVM and RBF kernel tricks. Fig 3.7 and 3.8 classification accuracy tested on different SVM kernels and among all RBF kernel shows the best results for both datasets.

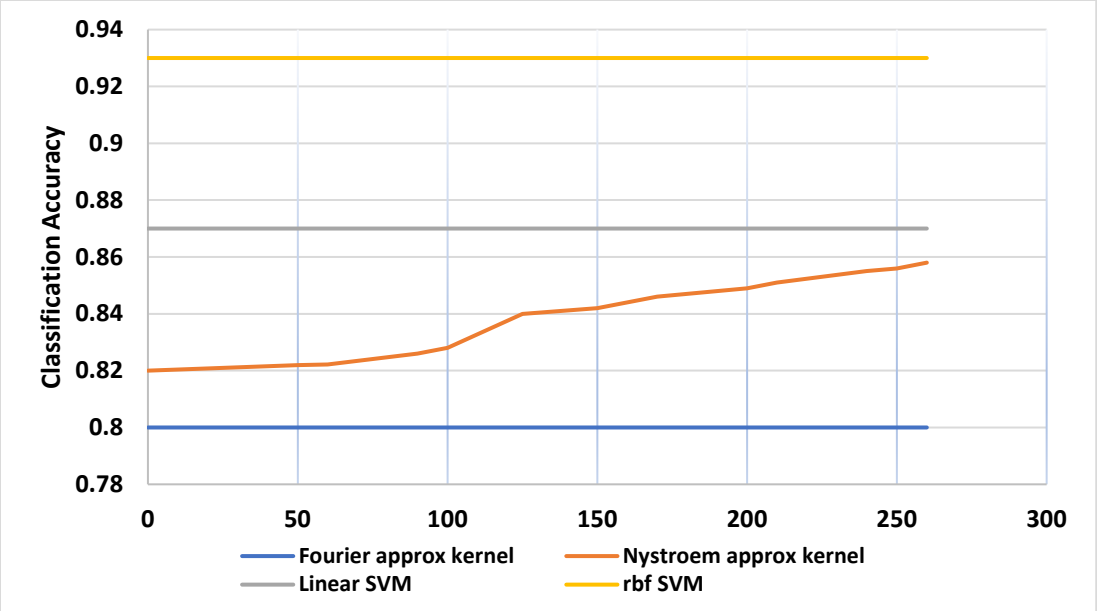


Figure 3.7. Different kernels of SVM using Kaggle dataset

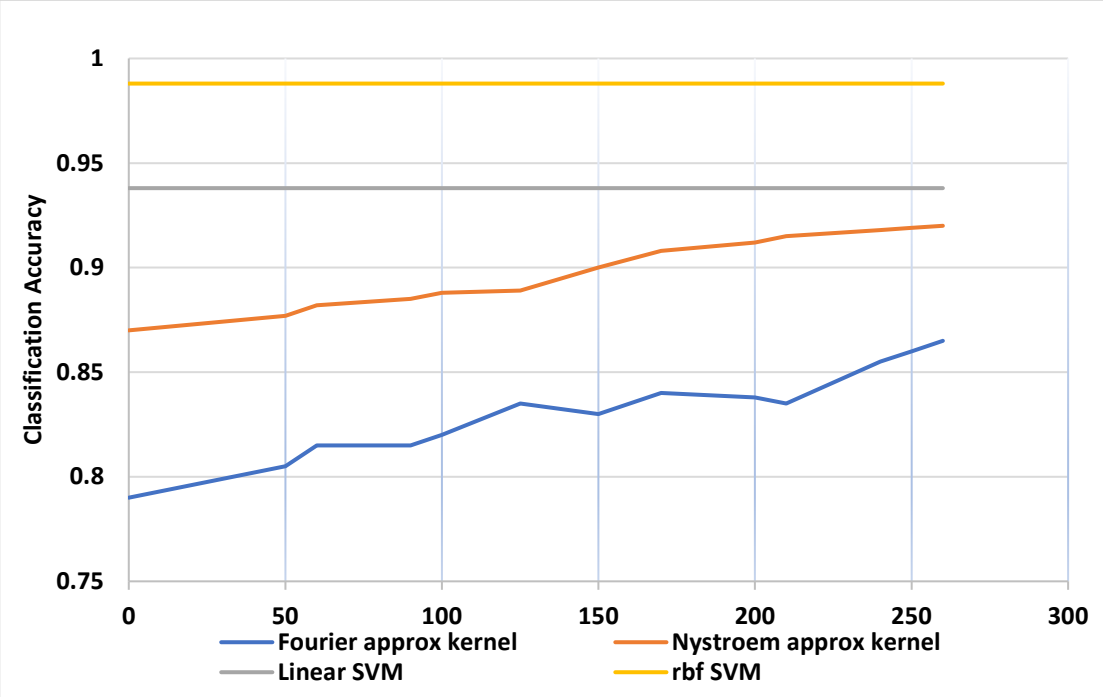


Figure 3.8. Different kernels of SVM using UNB dataset

Logistic Regression

Logistic regression is a famous ML algorithm which is very close to linear regression [48]. The term logistic is taken from the "Logit Function". It predicts the outcome that can only have two values. It produces a logistic curve which is limited to value 0 and 1. Logit Function is used to get the values within the range of 0 and 1. Logistic regression is best used for nonlinear decision boundaries. Equation 3.3 shows the formula where the curve is constructed using the natural logarithm of the odds of target variable.

$$p = \frac{1}{(1 + e^{-(b_o + b_1 x_1 + \cdots + b_n x_n)})}$$

Equation 3.3. Logistic Regression [48]

3.4.2 Ensemble Machine Learning Algorithms

Ensemble learning is done by combining base learners of different machine algorithms. It combines several machine learning techniques to decrease variance and bias to improve classification performance [29]. In our work, we have used Bagging, Random Forest, Adaboost and Gradient Boosting.

Bagging

Bagging, an acronym for bootstrap aggregation, is used for stabilizing the accuracy by reducing the variance [44]. It is subclass of ensemble learning and can be used in both classification and regression. It consists of several weak learners trained in parallel and combines the individual results through the deterministic averaging method. It improves the stability of algorithm by improving accuracy and reducing variance.

Random Forest

A random forest algorithm can be used for classification as well as regression and consists of a multitude of decision trees. These decision trees work for a different random subset of a dataset and make predictions according to that specific dataset. The final prediction is done by obtaining the prediction of all individual trees and predict a class that gets most votes and such ensemble of decision trees is called Random Forest. In this way, the random forest can be grown very deep and

might learn irregular patterns that can overfit the training dataset. That is why averaging multiple decision trees give the results by reducing variance.

AdaBoost

AdaBoost short for "Adaptive Boosting" learn the weak learners sequentially in an adaptive way [31]. It tries to correct the errors of the previous stage. Each stage in boosting depends on the previous one iteratively. Unlike bagging that had each model run independently and then aggregates the output at the end, boosting works in team where each model run, dictates what features the next model will focus on.

Extra-Trees

This algorithm is like a random forest that uses a decision tree at the core. The main difference between Random forest and Extra Trees lies in the fact that, instead of computing optimal feature for the split, a random value is selected for the split.

Voting Classifier

Voting Classifier [30] also known as majority rule, itself is not a classifier or an algorithm but rather a wrapper for a set of different other classifiers that work in parallel and uses the majority

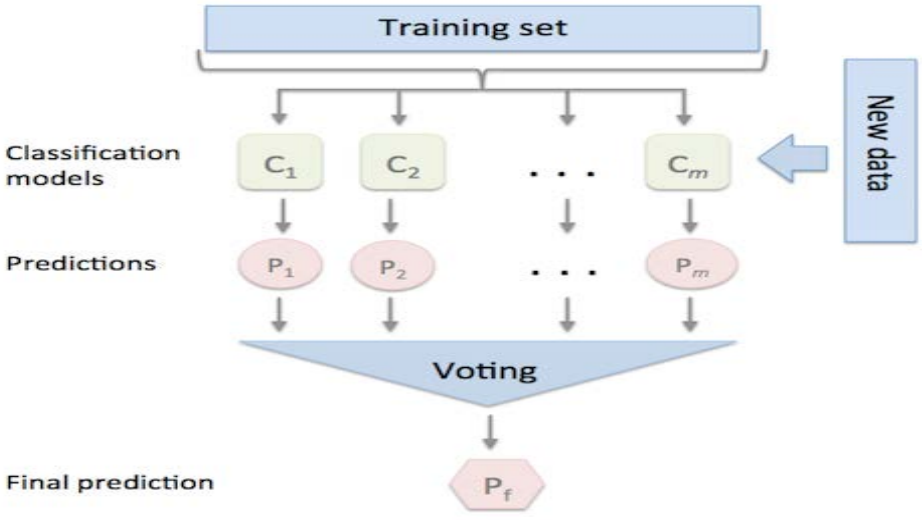


Figure 3.9. Voting Classifier [34]

vote according to several strategies. It combines several algorithms and usually gives more accurate results than any individual algorithm. It implements hard and soft voting. Hard voting is about predicting the class which has higher frequency among the classification models whereas soft voting is about predicting the class labels by averaging the individual classifier class probabilities. Fig 3.9 shows the generalized view of the internal working of voting classifier.

3.5 Results

To get good results, we have used an optimization technique called Hyperparameter optimization. It is about choosing the optimal hyperparameters for a learning algorithm. There are different approached Hyperparameter optimization such as Grid Search, Random search, Bayesian, Gradient-based and Population-based optimization.

3.5.1 K-fold Cross Validation

The machine learning model can give biased results if training data lies or cover a certain distribution of sample data of the population. To avoid such scenarios validation techniques in Machine learning can be convenient to use as it gives an error rate close to the true error of the population. We have used the K-fold cross-validation technique for algorithm training.

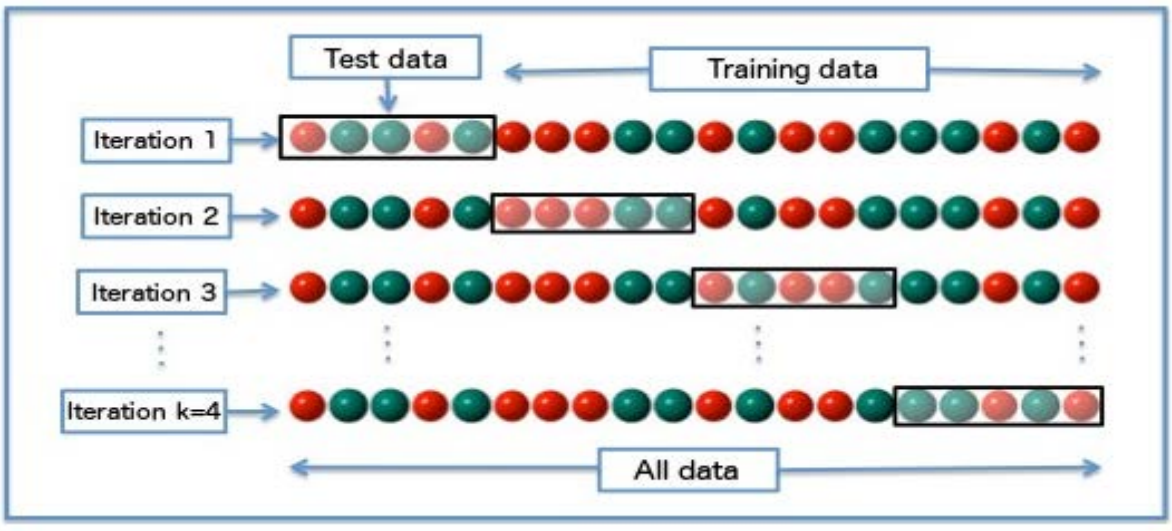


Figure 3.10. K-fold cross validation with k = 4 [24]

K-fold splits the data into k parts and fed those data set into k different models. Each model is trained on to (K-1) parts and tested on the not included model.

3.5.2 Grid Search

Grid search is the most used and traditional way of performing hyperparameter optimization. It is a brute force way of selecting a manually specified subset of the hyperparameter space of an algorithm. In our case, we performed a grid search on all the above-mentioned algorithms. Model hyperparameters [23] are the features of any ML algorithms which cannot be estimated from the data as it must be set before fitting the training data into the algorithm. On the other hand, model parameters are different than hyperparameters as it shows the internal characteristics of the model and its value can be estimated from the training data.

We tuned and optimized each algorithm with the help of grid search and cross-validation. The performance metrics that we used for evaluating the models are confusion matrix, accuracy, precision, recall, and F1-Score. The brief description for each of them are as follows:

- ***Confusion Matrix*:** It contains the information for actual and predicted results. The model performance can be evaluated with the help of this matrix which shows the positive and negative values for actual and predicted classes. It should be noted that we have considered malicious URL as positive class and benign URL as negative class.
- ***True Positive (TP):*** The number of observations that are predicted as positive (malicious) and are positive (malicious) in actual.
- ***False Positive (FP):*** The number of observations that are predicted positive (malicious) but are negative (benign) in actual.
- ***True Negative (TN):*** The number of observations that are predicted as negative (benign) and are negative (benign) in actual.
- ***False Negative (FN):*** The number of observations that are predicted negative (benign) but are positive (malicious) in actual.

- ***Accuracy:*** It is the ratio of correctly predicted observations to the total number of observations.

$$Accuracy = \frac{Correctly\ predicted\ observations}{Total\ observations}$$

$$= \frac{TP + TN}{TP + FP + TN + FN}$$

- ***Precision*****:** This metric is a good performance metric when the cost of false-positive is high and given as:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

- ***Recall*****:** This metric is a good performance metric when the cost of false negative is high and given as:

$$Recall\ = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

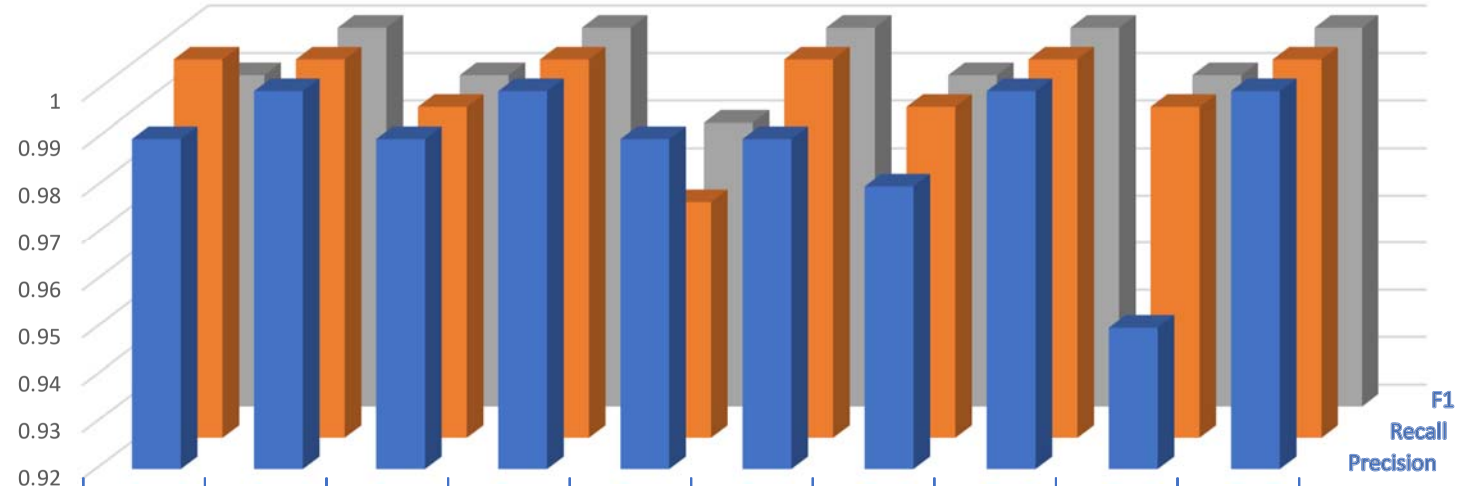
- ***F1-score*****:** This metric is holistic evaluation of precision and recall. It is the harmonic mean of precision and recall.

$$Recall\ = \frac{2 * Precision * Recall}{Precision + Recall}$$

Table 3.2 shows the classification accuracy for all the algorithms after they were optimized. Voting classifier is the final classifier that combines the outcomes for different algorithms with their best parameters. Fig 3.11 and 3.12 shows the performance metrics for UNB and Kaggle datasets for five best algorithms.

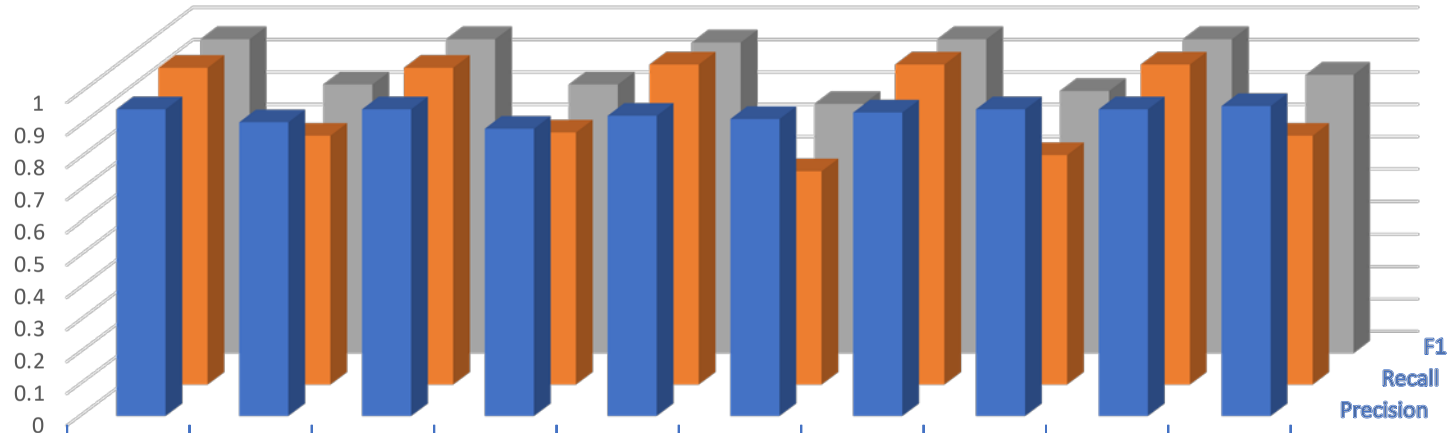
The accuracy, precision, and recall for the UNB dataset are slightly better than [5] with the generic features set. The overall precision, recall, and F1-Score for both the datasets are better than [15], however, their feature selection methods and datasets are different than our work. Table 3.3 shows the confusion matrix for the Voting classifier reported on UNB and Kaggle test datasets.

The classifier achieved an accuracy of 99.72% and 95.37% for UNB and Kaggle datasets, respectively. The false-positive rate and false-negative rate for the UNB dataset were 0.1% and 0.7%, and 3.97% and 4.73% for the Kaggle dataset. The weighted accuracy for the classifier is 96.60% along with 2.88% false-positive and 3.60% false-negative rate. Andrew et al. [11] on the other hand are classifying phishing attacks with four different features set categories by using 138 features, whereas we are covering all malicious URLs with 47 features. The results of this work have been published in [49].



	Benign	Malicious	Benign	Malicious	Benign	Malicious	Benign	Malicious	Benign	Malicious
	Gradient Boosting		KNN		SVM(RBF)		Random Forest		Voting Classifier	
Precision	0.99	1	0.99	1	0.99	0.99	0.98	1	0.95	1
Recall	1	1	0.99	1	0.97	1	0.99	1	0.99	1
F1	0.99	1	0.99	1	0.98	1	0.99	1	0.99	1

Figure 3.11. Performance metrics for UNB dataset with different classifiers



	Benign	Malicious	Benign	Malicious	Benign	Malicious	Benign	Malicious	Benign	Malicious
	Gradient Boosting		KNN		SVM(RBF)		Random Forest		Voting Classifier	
Precision	0.95	0.91	0.95	0.89	0.93	0.92	0.94	0.95	0.95	0.96
Recall	0.98	0.77	0.98	0.78	0.99	0.66	0.99	0.71	0.99	0.77
F1	0.97	0.83	0.97	0.83	0.96	0.77	0.97	0.81	0.97	0.86

Figure 3.12. Performance metrics for Kaggle dataset with different classifiers

Table 3.2. Classification accuracy of UNB and Kaggle datasets

Dataset	Algorithm	Accuracy (in %)
UNB	KNN	99.59
	SVM (RBF)	99.29
	Logistic Regression	94.04
	Adaboost	97.85
	Gradient Boosting	99.68
	Extra Trees	98.96
	Random Forest	99.49
	Voting Classifier	**99.72**
Kaggle	KNN	94.31
	SVM (RBF)	92.99
	Logistic Regression	87.21
	Adaboost	90.34
	Gradient Boosting	94.44
	Extra Trees	88.00
	Random Forest	94.22
	Voting Classifier	**95.37**

Table 3.3. Confusion Matrix for Voting Classifier on UNB and Kaggle datasets

			Predicted	
			Benign	Malicious
UNB	**Actual**	Benign	6917	37
		Malicious	51	26069
Kaggle	**Actual**	Benign	62651	478
		Malicious	3414	11550

4. REAL TIME MALICIOUS URL DETECTION

The previous chapter provided the generic features selection for malicious URL detection using two different datasets. . In this chapter, we discuss the implementation of detection model into real-time capable of handling thousands of end-user requests. We design a client-server architecture to handle the load and analyze the performance in different scenarios.

4.1 Overview & Architecture

Designing architecture is the most significant part when it comes to scalability and handling a huge number of incoming requests. Suitable and sustainable architecture can help organizations to achieve their desired goals with the capability to handle huge loads. Architectural patterns are like software design patterns but have a wider scope. Design patterns for software development proved to be very effective for the development of any software, however, design patterns for applications that leverage new sources and types of big data are still needed.

Implementing real-time machine learning model can be a bottleneck when it comes to processing thousands of requests per second. We designed an architectural pattern that could detect any malicious URLs in real-time. Fig 4.1 shows a high-level overview of our real-time malicious URL detection system. This pattern requires clients to add google chrome extension in their chrome browser. Stepwise process of detection system is discussed below:

1. User visits any website, google chrome extension intercepts and halts that specific incoming request.
2. Chrome browser extension generates separate web service request along with target URL content as payload to a remote web server.
3. Web server checks the URL results in database and preform classification accordingly.
4. Web server sends the response back to the client browser extension.
5. Browser extension checks if URL is malicious or benign. It blocks the URL if its malicious otherwise it allows the client to access to the requested URL using internet.

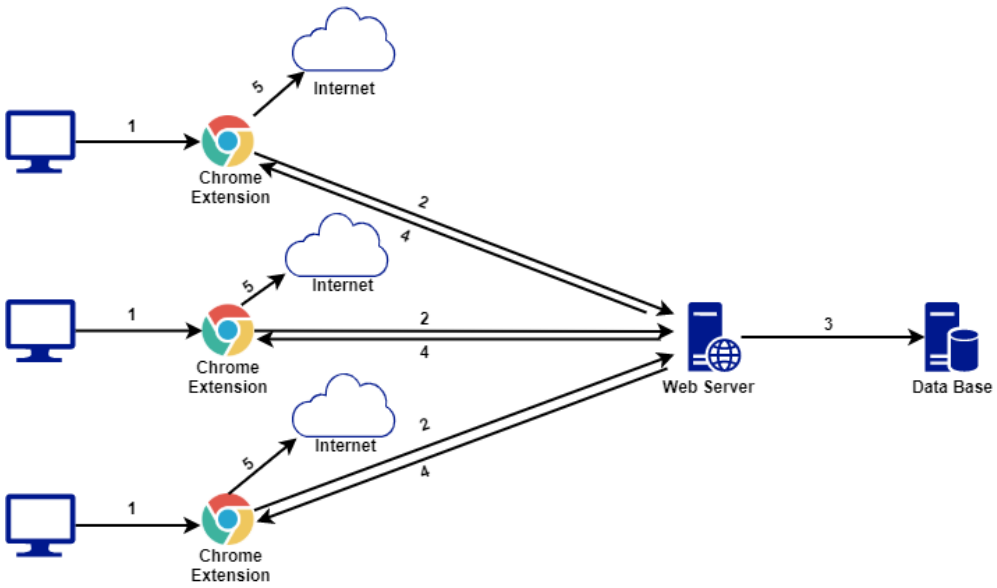


Figure 4.1. Client server architecture

4.2 Implementation of Client-Server Architecture

A single node client-server architecture is implemented to handle all HTTP requests from the client in real-time. In order to achieve this task, we have done a series of tests with different throughput and load variations. Table 1 and 2 show the hardware specification and software/libraries used for single node server.

Table 4.1. Hardware specification of a system that runs the server VM

Processor	Intel(R) Core (TM) i7-6900 @ 3.20 GHz
Installed RAM	40 GB
Operating System	Linux, Ubuntu 64 bit

Table 4.2. Software level configuration

Package/Tool	**Version/Name**
Web Server	Django
Web Server	Japronto
Client	Google Chrome Plugin

Table 4.2. continued

Python Version	3.7.1
Sckit-Learn	0.20.1
Pandas	0.23.4
NumPy	1.15.4
Matplotlib	3.0.2
Database	SQLite
Load Testing Tool	Apache JMeter

Pseudocode 1. Client - Server Architecture

```
httpRequestListener(request) {

      url = request.getUrl
      dbCon = getSingletonDBConnection()
      domain = getDomain(url)
      urlDetail = dbCon.gerUrl(domain)
      isMalicious = true

      If (urlDetail != null)
          isMalicious = urlDetail.getClassificationResult()
      Else
          dataset = createLexicalFeatures(url)
          normalizedData = normalizedDataset(dataset)
          result = prediction.makePrediction(normalizedData)
          saveResultsinDB(url,domain,result)
          isMalicious = result
      Endif

      return isMalicious

}
```

4.2.1 Google Chrome Plugin as Browser Extension

In order to develop the whole end to end system, we have developed browser extension for Google chrome. Browser extensions are software that can be embedded with the web browser to accomplish certain desired tasks. They are separate modules and are different from browser plugins. Browser plugins are always executable whereas extensions are usually just the source code. The popular two browser plugin examples are Adobe Flash Player and Java virtual machine also called Applets. On the other hand, browser extension does not contain object code or

executables. Extensions are lightweight and are limited to make changes at the browser level. It has the capability to intercept any request initiated from the browser. We have used it to intercept and verify all the request that has been generated from google chrome browser in real-time. Those intercepted request along with the URLs are then passed to the cloud-based web server for classification.

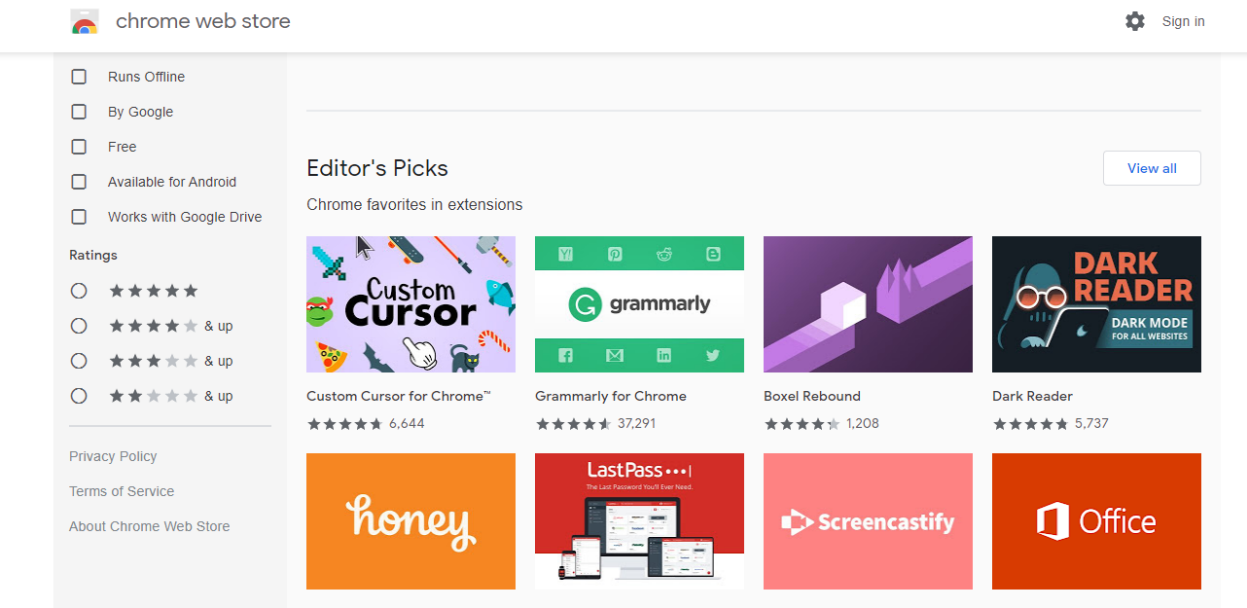


Figure 4.2. Chrome Extensions from Chrome Web Store

Pseudocode 2. Chrome Extension Interceptor

```
listenerOnBeforeRequest(dtls) {

      isMalicious = true
      url = dtls.visitingUrl
      isMalicious = cloudApiCall(url)
      blockUrl = false
      If (isMalicious = True)
          blockUrl = true
      Else
          blockUrl = false
      EndIf

      return blockUrl

}
```

4.2.2 Apache JMeter

Apache JMeter is an open-source Java-based desktop application that is used to perform load testing functional behavior and performance measurement [51]. We have used JMeter to analyze and measure the performance of our web server in single node mode and distributed cluster mode. JMeter allocates concurrent and real-time sampling of different functions by a separate thread group. Through JMeter, we can tune the throughput of web request and analyze the latency of web server from thousand to millions of requests per second concurrently.

4.2.3 Python and Related Libraries

Python is an interpreted high-level language which is dynamically typed, and garbage collected. It supports procedural, functional and object-oriented programming. We have used python3 for our project. Since it is open-sourced, we can find a lot of libraries and tools for it. We have used the Sckit-learn [50] library for machine learning classification. It is a free software machine learning library for Python. Other libraries such as NumPy and Matplotlib have also played an important role to achieve our task. NumPy is used to sustain large multi-dimensional arrays and matrices along with the large collection of high-level mathematical functions to operate these arrays. Matplotlib is a plotting library that provides object-oriented API for embedding plots into the application.

4.2.4 Django as a Webserver

A web server is server software that is capable to handle the worldwide web client request. It contains website and processes client requests over HTTP [52]. Django is a high-level python-based web framework that has inbuilt web server based on Web Server Gateway Interface (WSGI). WSGI is a calling convention to incoming requests to web frameworks and application written in python programming language.

4.2.5 Japronto as a Web Server

Japronto is a micro-framework which can handle the synchronous and asynchronous request [53]. It is scalable, lightweight and even faster than NodeJS. It is written in C language to take advantage of modern CPUs.

4.2.6 Performance Evaluation and Optimization

We have considered two scenarios for load testing of a single node web server. These two scenarios are based on the real-time detection mechanism at the server end-side.

I. **Scenario I**: we assume that there are no records in the database for previously classified URLs.

II. **Scenario II**: We assume that URLs have already been classified and stored in the database along with their labels.

Pseudocode 1 shows the implementation of URL classification. It first checks if the incoming URL is stored in database and then predicts in case if it's not there in database otherwise it returns the previous calculated result from database.

Django webserver

By considering both scenarios as discussed, we got the same throughput of 120 concurrent requests per second as shown in Fig 4. Figure 4.4 and 4.5 show the response time of all the requests which are less than 500ms.

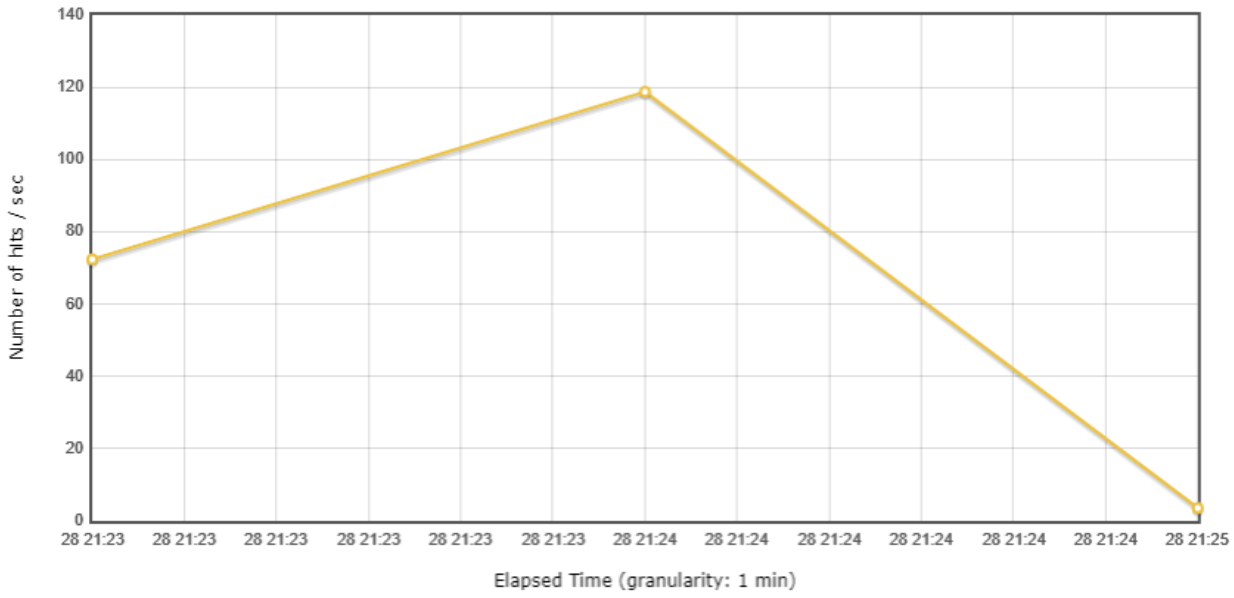


Figure 4.3. Number of hits/sec plot

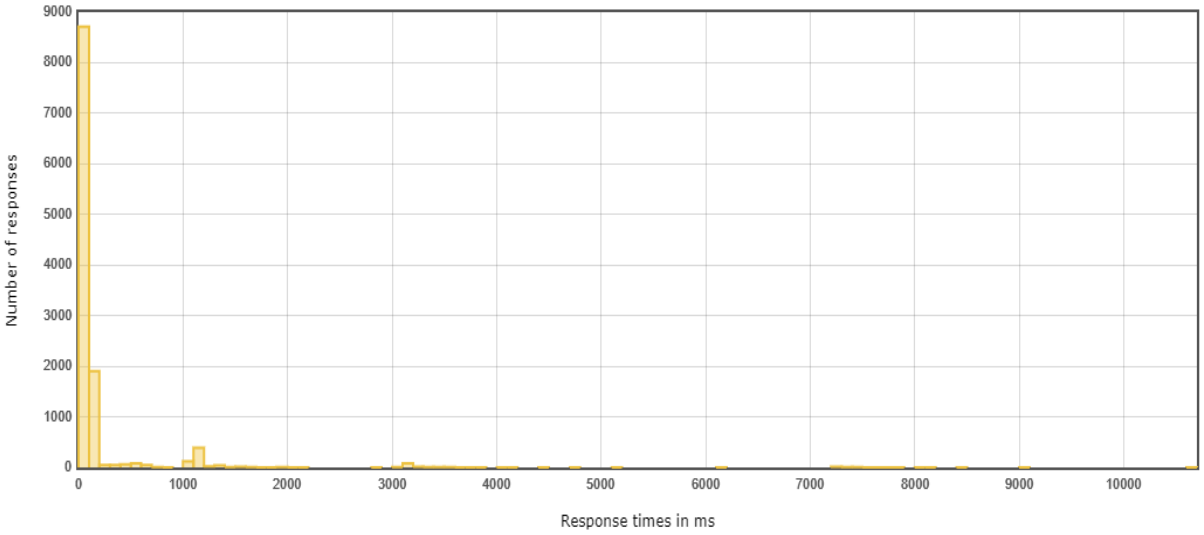


Figure 4.4. Response time in millisecond

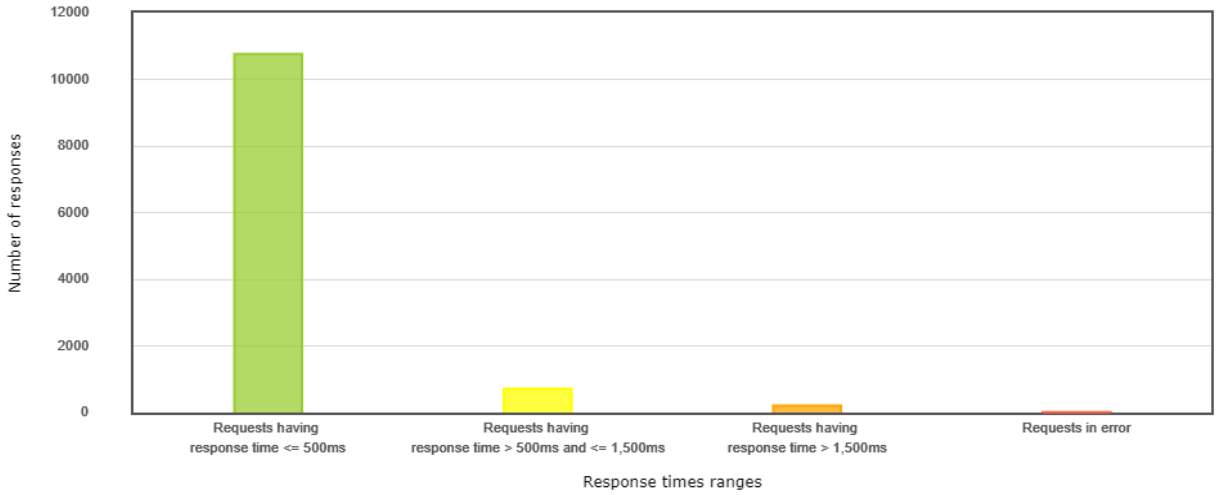


Figure 4.5. Summary of response time in different times ranges

Japronto webserver

We perform load testing on Japronto by considering the two scenarios as discussed earlier. We got a maximum throughput with the second scenario where the classification results have already been stored into the database previously. This scenario assumes that the user had already visited a URL. Japronto was able to reach up to the 7438 hits per second. As from figure 4.6 and 4.8, the average response time is 4.11 millisecond with a total of 864710 request samples whereas

Figure 4.7 shows the number of transactions within the span of 2 seconds and Figure 4.9 shows the response time.

Requests	Executions			Response Times (ms)						Throughput	Network (KB/sec)	
Label	#Samples	KO	Error %	Average	Min	Max	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**864710**	**3**	**0.00%**	**4.11**	**0**	**580**	**7.00**	**8.00**	**10.00**	**7155.83**	**642.96**	**1879.80**
HTTP Request	864710	3	0.00%	4.11	0	580	7.00	8.00	10.00	7155.83	642.96	1879.80

Figure 4.6. Load test summary on Japronto with Database

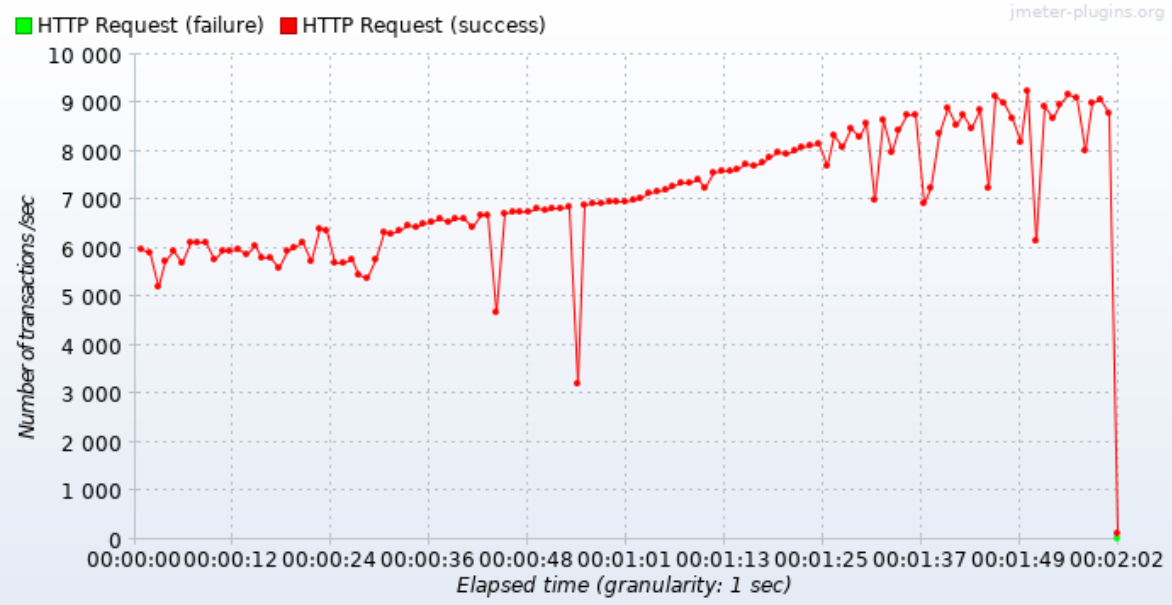


Figure 4.7. Number of Transactions/secs on Japronto with Database

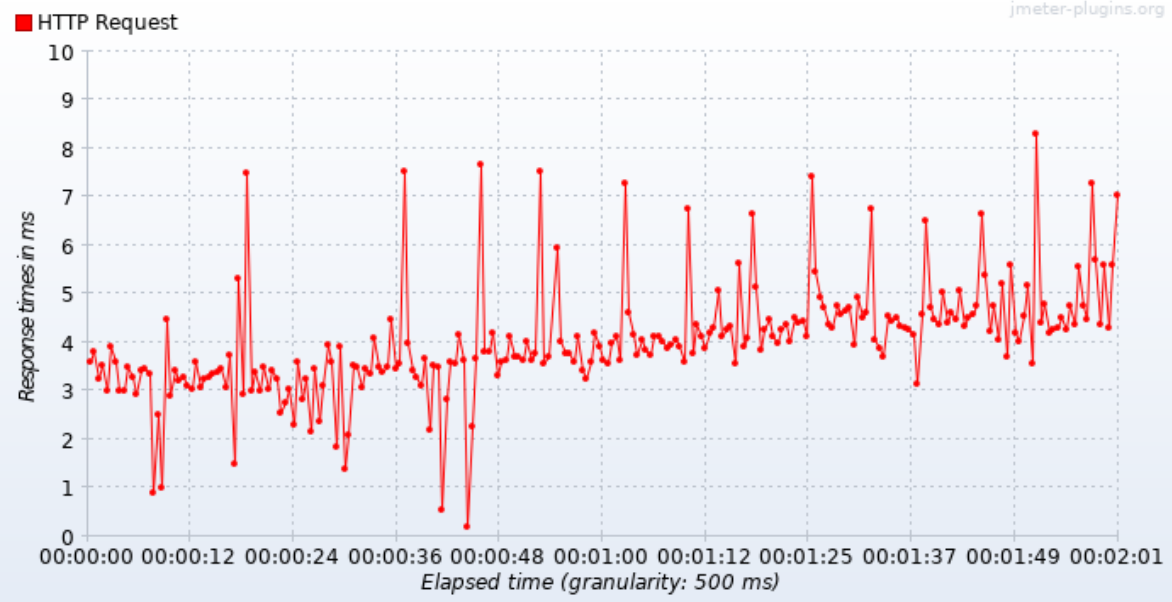


Figure 4.8. Response times in millisecond on Japronto with Database

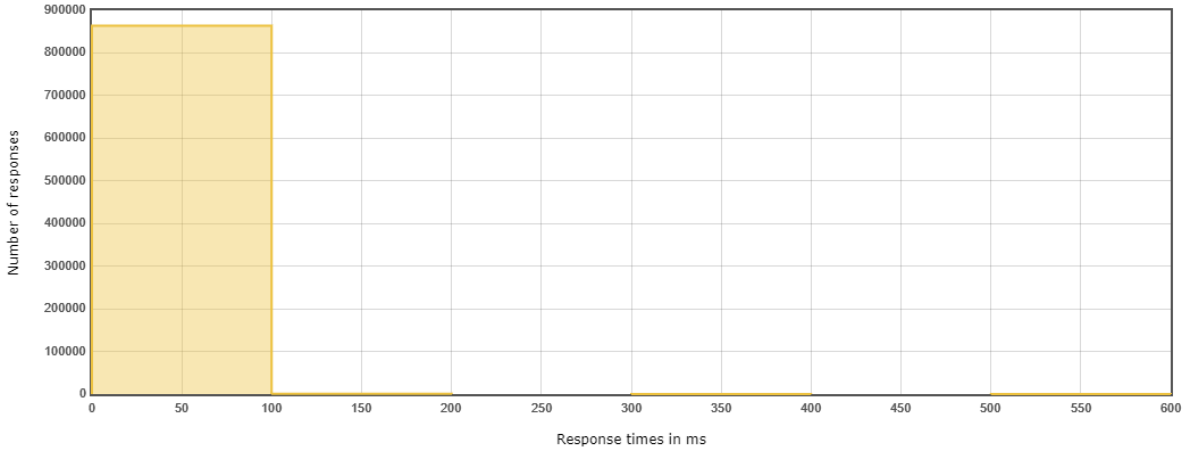


Figure 4.9. Histogram of Response time on Japronto with Database

5. CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In this work, we have been able to detect malicious URLs in real-time using machine learning via web service calls. We applied different statistical techniques, known as Chi-Square and ANOVA, to identify the most significant lexical features by using different datasets. We have used a combination of different machine learning algorithms including single and ensemble machine learning algorithms. We finalized 47 most significant features out of 106 that has the potential to identify any malicious URLs accurately and precisely with an average low false-positive rate of 2.88% and an accuracy of 96.6% for two different datasets.

We further implemented a client-server based real-time detection mechanism that relies on HTTP communication protocol over web service calls which uses google chrome along with browser plugin as a client and sends a request to the server. We have analyzed the performance measure using client-server architecture by considering Japronto as HTTP webserver.

5.2 Future Work

Our work can further be optimized and improved in several ways. The following are a few considerations for upgrading this work which can improve its functionality

- In our work, we considered two datasets and have tested generic features on them. Although these two datasets contain a different variation of malicious URLs, we can still further verify our derived generic features by using a different dataset from different sources.
- To make classification faster and time-efficient, the lexical feature-based model is always preferable. However, lexical based features for URL detection can miss malware and malicious activities that are embedded on legit websites without knowing the site owner. For this reason, further consideration is to use dynamic web content/source-based features, and network-based features. By providing these features, it will be able to detect embedded malicious code inside web pages.

- Another worth consideration is about online learning. In our case, we considered already provided past data as training data for our model. It can be further updated in a way where machine learning algorithms can update their weight matrices by considering the past stream of classified data as training data that works in rounds. For each round, the algorithm can predict with its model along with prediction suffer loss.

REFERENCES

[1] "Internet World Status: Usage of Internet and Population Status," *"https: //www.internetworldstats.com/stats.htm Accessed July 20, 2019"*

[2] "We are social: Digital 2019, Global Internet Use Accelerates" *"https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates"*

[3] "Hacker Attacks every 39 Second,""*https://www.securitymagazine.com/articles/87787-hackers-attack-every-39-seconds"*

[4] Symantec Internet Security Thread Report *"https://www.symantec.com/security-center/threat-report"*

[5] SiteLock Website Security *"https://www.sitelock.com/blog/website-security-insider-q2-2018/"*

[6] CAIDA study "a Macroscopic Characterization of the DoS Ecosystem" *"https://www.caida.org/publications/papers/2017/millions_targets_under_attack/millions_targets_under_attack.pdf"*

[7] Email fraud continue to rise by Proof Point *"https://www.proofpoint.com/us/corporate-blog/post/email-fraud-continues-rise-number-attacks-grew-36-q2"*

[8] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious urls using lexical analysis," in Network and System Security, J. Chen, V. Piuri, C. Su, and M. Yung, Eds. Cham: Springer International Publishing, 2016, pp. 467–482

[9] Faeze Asdaghi, "An effective feature selection method for web spam detection " *"https://www.sciencedirect.com/science/article/abs/pii/S095070511830621X"*

[10] H. Le, Q. Pham, D. Sahoo, and S. C. H. Hoi, "Urlnet: Learning a URL representation with deep learning for malicious URL detection," CoRR, vol. abs/1802.03162, 2018. [Online]. Available: *"http://arxiv.org/abs/1802.03162"*

[11] R. B. Basnet, A. H. Sung, and Q. Liu, "Learning to detect phishing URLs."

[12] W. Wang and K. Shirley, "Breaking bad: Detecting malicious domains using word segmentation," arXiv preprint arXiv:1506.04111, 2015

[13] Takesh Yogi "Investigation and analysis of malware on websites" *"https://ieeexplore.ieee.org/abstract/document/5623567"*

[14] Stephen A.C "Spoofing and Anti-Spoofing Measures" *"http://php.iai.heig-vd.ch/~lzo/biomed/refs/Spoofing and Anti-Spoofing Measures - 2002_Schuckers.pdf"*

[15] Jason Hong, "The current State of Phishing Attacks" *"http://php.iai.heig-vd.ch/~lzo/biomed/refs/Spoofing and Anti-Spoofing%20Measures - 2002_Schuckers.pdf"*

[16] Chi-Square Statistics *"https://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.html"*

[17] ANOVA *"https://en.wikipedia.org/wiki/Analysis_of_variance"*

[18] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, *"Detecting malicious urls using lexical analysis"* in Network and System Security, J. Chen, V. Piuri, C. Su, and M. Yung, Eds. Cham: Springer International Publishing, 2016, pp. 467–482.

[19] "Malicious & non-malicious url," https://www.kaggle.com/antonyj453/datasets

[20] M Junaid Khan "Complete 106 feature set" *"https://docs.google.com/spreadsheets/d/1fO_giLLsmxU47cxrDVJlIgmZojwmWmyW36MFCh4qMvY/edit#gid=0"*

[21] D. Patil and J. Patil, "*Feature-based malicious url and attack type detection using multi-class classification,"* The ISC International Journal of Information Security, vol. 10, no. 2, pp. 141–162, 2018. [Online]. Available*: http://www.isecure-journal.com/article 63041.html*

[22] W. Wang and K. Shirley, *"Breaking bad: Detecting malicious domains using word segmentation"* arXiv preprint arXiv:1506.04111, 2015

[23] Hyperparameters in Machine Learning Model *"https://en.wikipedia.org/wiki/Hyperparameter_optimization"*

[24] K-fold Cross Validation – "*https://towardsdatascience.com/5-reasons-why-you-should-use-cross-validation-in-your-data-science-project-8163311a1e79*"

[25] K Nearest Neighbors *"http://scholarpedia.org/article/K-nearest_neighbor"*

[26] Support Vector Machine *"http://scholarpedia.org/article/Support_vector_clustering"*

[27] Kernel Tricks in SVM *"https://en.wikipedia.org/wiki/Kernel_method"*

[28] Logistic Regression *"https://en.wikipedia.org/wiki/Logistic_regression"*

[29] Robi Polikar "Ensemble learning," *"https://link.springer.com/chapter/10.1007/978-1-4419-9326-7_1"*

[30] Majority Rule *"https://en.wikipedia.org/wiki/Majority_rule"*

[31] AdaBoost *"https://en.wikipedia.org/wiki/AdaBoost"*

[32] James B Fraley, “The Promise of Machine Learning in Cybersecurity”

[33] What is URL *“https://sitechecker.pro/what-is-url/”*

[34] Voting Classifer in Detail http://rasbt.github.io/mlxtend/user_guide/classifier/EnsembleVoteClassifier/

[35] Report on Malicious URLs in good domain by Help Net Security, *“ https://www.helpnetsecurity.com/2019/03/01/malicious-urls-good-domains/”*

[36] Threat Report by WebRoot , *“https://www-cdn.webroot.com/9315/2354/6488/2018-Webroot-Threat-Report_US-ONLINE.pdf”*

[37] 2019 Phishing Statistics and Email Fraud Statistics by Retruster, *“https://retruster.com/blog/2019-phishing-and-email-fraud-statistics.html”*

[38] Jason Andress, in Cyber Warfare, 2011, *“https://www.sciencedirect.com/topics/computer-science/web-site-defacement ”*

[39] How artificial intelligence stopped an Emotet outbreak, *“https://www.microsoft.com/security/blog/2018/02/14/how-artificial-intelligence-stopped-an-emotet-outbreak/”*

[40] Chronicle, Cybersecurity based company, https://chronicle.security/

[41] SQRRL, security and threat detection, *“https://searchaws.techtarget.com/news/252433932/AWS-snaps-up-Sqrrl-to-strengthen-threat-detection-analytics”*

[42] Guolin Tan, Peng Zhang, Qingyun Liu “MalFilter: A lightweight rea-time malicious URL filtering system in large scale networks”

[43] Kurt Thomas, Justin Ma, “Design and evaluation of a real-time URL spam filtering service”

[44] J.R. Quinlan, “Bagging, Boosting and C4.5”, *http://www.cs.ecu.edu/~dingq/CSCI6905/readings/BaggingBoosting.pdf*

[45] Forester Report on Data Breach, *“https://www.forrester.com/search?tmtxt=data breach#”, “https://www.techrepublic.com/article/forrester-what-can-we-learn-from-a-disastrous-year-of-hacks-and-breaches/”*

[46] KelserCorp press release on “Defend Forward” against cyber attacks “*https://www.kelsercorp.com/blog/press-release-kelser-enables-mid-size-companies-to-defend-forward-against-cyber-attacks*”

[47] Proof point on email spoofing *"https://www.proofpoint.com/sites/default/files/pfpt-us-tr-email-fraud-yir-180212.pdf"*

[48] Logistic Regression *"https://en.wikipedia.org/wiki/Logistic_regression"*

[49] Hafiz M Junaid Khan, Quamar Niyaz, Vijay Devabhaktuni, Sile Guo, and Umair Shaikh, *"Identifying generic features for malicious URL detection",* IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, October 2019

[50] Scikit-learn machine learning library *"https://scikit-learn.org/stable/"*

[51] Apache JMeter *"https://jmeter.apache.org/"*

[52] Django Framework *"https://www.djangoproject.com/"*

[53] Japronto Server *"https://github.com/squeaky-pl/japronto"*

APPENDIX

Total Number of Generated Features

#	Feature Name	Description
1	count_ques	Count Question Mark (?)
2	count_equals	Count Equals (=)
3	count_semi_colun	Count Semi Colun (;)
4	count_dash	Count Dash (-)
5	count_open_brckt	Count Opening Bracket (
6	count_close_brckt	Count Closing Bracket)
7	count_percent	Count Percent Symbol (%)
8	count_dots	Count Dots (.)
9	len_url	Calculate Length of whole URL
10	count_&	Count AmpersandSymbol (&)
11	count_/	Count BackSlash (/)
12	count_path_len	Calculate Length URL Path
13	count_@	Count At the Rate Symbol (@)
14	Secure	Check Presence
15	account	Check Presence
16	Webscr	Check Presence
17	Login	Check Presence
18	ebayisapi	Check Presence
19	Signin	Check Presence
20	banking	Check Presence
21	confirm	Check Presence
22	Blog	Check Presence
23	Logon	Check Presence
24	signon	Check Presence
25	viewer	Check Presence

26	getImage	Check Presence
27	plugins	Check Presence
28	paypal	Check Presence
29	Order	Check Presence
30	Dbsys	Check Presence
31	Config	Check Presence
32	Order	Check Presence
33	Js	Check Presence
34	payment	Check Presence
35	css	Check Presence
36	admin	Check Presence
37	abuse	Check Presence
38	update	Check Presence
39	verification	Check Presence
40	shopping	Check Presence
41	Log	Check Presence
42	access	Check Presence
43	bonus	Check Presence
44	click	Check Presence
45	network	Check Presence
46	pay	Check Presence
47	download	Check Presence
48	Jar	Check Presence
49	swf	Check Presence
50	Cgi	Check Presence
51	Zip	Check Presence
52	Jpg	Check Presence
53	Gif	Check Presence
54	redirect	Check Presence
55	isIpAddress	Check Presence of IP address

56	is_numner_in_host	Check Numeric Number presence in URL
57	host_length	Length of host
58	no_of_subDomains	Count Number of Sub-Domains
59	count_letter	Total count of letters in URL
60	count_digits	Total count of digits in URL
61	count_symbols	Total count of symbols in URL
62	entropy_url	Entropy of URL
63	entropy_host	Entropy of Host
64	entropy_path	Entropy of URL Path
65	entropy_params	Entropy of URL Parameters
66	entropy_query	Entropy of Query Parameters
67	Tld	Extract Top Level Domain
68	pathExtension	Extract URL Path Extension
69	count_path_back_slash	Count total back slash in URLPath
70	count_path_dot	Count total Dots in URL Path
71	A	Frequency of Alphabet "a"
72	B	Frequency of Alphabet "b"
73	C	Frequency of Alphabet "c"
74	D	Frequency of Alphabet "d"
75	E	Frequency of Alphabet "e"
76	F	Frequency of Alphabet "f"
77	G	Frequency of Alphabet "g"
78	H	Frequency of Alphabet "h"
79	I	Frequency of Alphabet "i"
80	J	Frequency of Alphabet "j"
81	K	Frequency of Alphabet "k"
82	L	Frequency of Alphabet "l"
83	M	Frequency of Alphabet "m"
84	N	Frequency of Alphabet "n"

85	O	Frequency of Alphabet "o"
86	P	Frequency of Alphabet "q"
87	Q	Frequency of Alphabet "q"
88	R	Frequency of Alphabet "r"
89	S	Frequency of Alphabet "s"
90	T	Frequency of Alphabet "t"
91	U	Frequency of Alphabet "u"
92	V	Frequency of Alphabet "v"
93	W	Frequency of Alphabet "w"
94	X	Frequency of Alphabet "x"
95	Y	Frequency of Alphabet "y"
96	Z	Frequency of Alphabet "z"
97	0	Frequency of Alphabet "0"
98	1	Frequency of Alphabet "1"
99	2	Frequency of Alphabet "2"
100	3	Frequency of Alphabet "3"
101	4	Frequency of Alphabet "4"
102	5	Frequency of Alphabet "5"
103	6	Frequency of Alphabet "6"
104	7	Frequency of Alphabet "7"
105	8	Frequency of Alphabet "8"
106	9	Frequency of Alphabet "9"