URL DETECTION USING MACHINE LEARNING

Himanshu Dhingra1(2110993793) Harshit Behl2 (2110993788)

1CSE(AI) Student, Department of Computer Science and Engineering, Chitkara University, Punjab, India.

2CSE(AI) Student, Department of Computer Science and Engineering, Chitkara University, Punjab, India.

RESEARCH PAPER

ABSTRACT

Recently, with the increase in Internet usage, cybersecurity has been a significant challenge for computer systems. Different malicious URLs emit different malicious software and try to capture user information. Signature-based approaches have often been used to detect such websites and detected malicious URLs have been attempted to restrict access by using various security components. This chapter proposes using host-based and lexical features of the associated URLs to better improve the performance of classifiers for detecting malicious web sites. Random forest models and gradient boosting classifier are applied to create a URL classifier using URL string attributes as features. The highest accuracy was achieved by random forest as 98.6%. The results show that being able to identify malicious websites based on URL alone and classify them as spam URLs without relying on page content will result in significant resource savings as well as safe browsing experience for the user.

malicious uniform resource locator (URL), i.e., Malicious websites are one of the most common cybersecurity threats. They host gratuitous content (spam, malware, inappropriate ads, spoofing, etc.) and tempt unwary users to become victims of scams (financial loss, private information disclosure, malware installation, extortion, fake shopping site, unexpected prize etc.) and cause loss of billions of rupees each year. The visit to these sites can be driven by email, advertisements, web search or links from other websites. In each case, the user must click on the malicious URL. The rising cases of phishing, spamming and malware has generated an urgent need for a reliable solution which can classify and identify the malicious URLs. Traditional classification techniques like blacklisting, regular expression, and signature matching approach are challenged because of huge data volume, patterns and technology changing over time, along with complicated relationship among features. In this paper, we address the detection of malicious URLs as a binary classification problem and evaluate the performance of several well-known machine learning classifiers. We adopted a public dataset from Kaggle comprising of 450000 URLs to train the model. The best classifier was used to detect malicious URLs from open phish website. It was found to give better results. Index Terms—Malicious URL, Machine learning, Phishing, Spamming, Malware, Spoofing.

LITERATURE SURVEY

This section presents a review of the existing approach and a summary of the literature in the field of URL classification. Previous work on this topic has involved content analysis of the page itself (Ntoulas, Najork, Manasse, & Fetterly, 2006). These typically include creating features from the HTML structure of the page, links, and anchor text, such as the number of words on a page, average word-length, and the number of words in the title. Other methods involve looking at the amount and percentage of hidden content (not visible to a user) on a page. Another approach is first to determine what are important features in terms of ranking in a search engine and then find which features are likely to be used by spammers (Egele, Kolbitsch, & Platzer, 2009). The downside to this approach is that it is infeasible to enumerate every ranking element, and thus important features may be missed. Another work attempt to classify web spam into buckets, such as link spam, redirection, cloaking, and keyword stuffing (Gyongyi & Garcia-Molina, 2005). While splitting spam into more specific buckets will likely lead to improvements in classifier ability, this paper will focus on building a general classifier for all types of spam. While relying on the page content and links increase the amount of data available for spam classification, there are strong motivations for being able to classify spam before crawling a page. This paper explores using the URL string as the primary feature in spam classification.

KEY TERMS

Lexical Features: It is the feature that distinguishes malicious URLs from benign URLs.

Malicious URLs: Compromised URLs that are used for cyber-attacks are termed as malicious URLs.

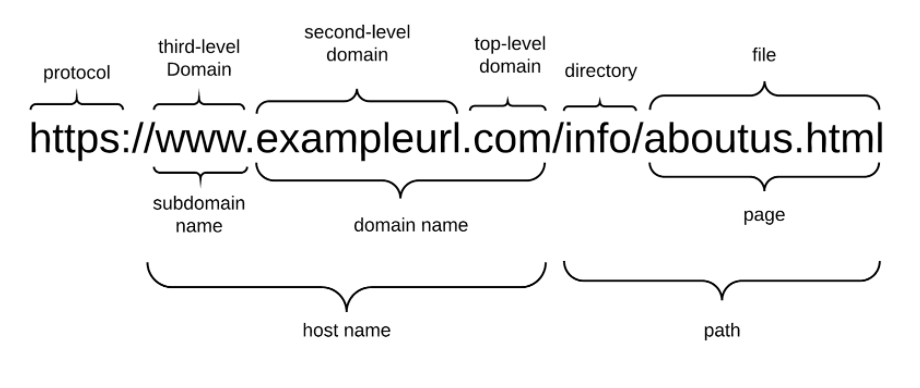
RF: Random Forest creates a forest and somehow does it randomly. The “forest” that is established is a collection of decision trees that are mostly trained by the method of “bagging”.

URLs: The abbreviation of uniform resource locator, which is the global address of documents and other resources on the World Wide Web.

WHOIS Information Feature: It includes domain name registration dates, registrars, and registrants. So, if the same individual registers a set of malicious domains, we used such control as a malicious indicator

INTRODUCTION

The Covid 19 has a great impact on the growth of online businesses such as e-banking, e-commerce, and social networking. Unfortunately, the technological advancements accompany state of the art techniques to exploit users. Such attacks generally include malicious websites that steal all kinds of private information that a hacker can exploit. In Malicious URL detection, traditional classification techniques like blacklisting [1], regular expression [2], and signature matching [3] approaches are challenged because of huge data volume, patterns changing over time, and complicated relationship among features. Inevitably, several malicious sites do not seem to be blacklisted. As any file on a computer is to be found by giving its filename, similarly, URL can be used to trace any website. It is the address of a resource on the WWW. Each URL has two main components. The first is Protocol. For URL https://www.google.com, the protocol identifier is HTTPS. Hypertext Transfer Protocol Secure (HTTPS) which is used to fetch hypertext documents. Other protocols include File Transfer Protocol (FTP), Domain Name System (DNS) etc. The second is Resource identifier. For URL https://www.google.com, the resource name is www.google.com. The resource identifier is the address of a webpage on the internet. The proposed work in this paper considers the identification of bad URLs and examines the evaluation metrics of various Machine Learning classifiers [4]. The source of data is a public dataset from Kaggle [5] comprising of 450000 URLs. The best classifier is used to detect malicious URLs from the open phish [6] website. The remaining paper is divided into the following sections. The experimental results achieved.



The significance of the World Wide Web (WWW) has attracted increasing attention because of the growth and promotion of social networking, online banking, and e-commerce. While new development in communication technologies promote new e-commerce opportunities, it causes new opportunities for attackers as well. Nowadays, on the Internet, millions of such websites are commonly referred to as malicious web sites. It was noted that the technological advancements caused some techniques to attack and scam users such as spam SMS in social networks, online gambling, phishing, financial fraud, fraudulent prize-winning, and fake TV shopping (Jeong, Lee, Park, & Kim, 2017). In recent years, most attacking methods are applied by spreading compromised URLs and fishing, and malicious Uniform Resource Locators (URLs) addresses are the leading methods used by hackers to perform malicious activities. Common types of attacks using malicious URLs can be categorized into Spam, Drive-by Download, Social Engineering, and Phishing (Kim, Jeong, Kim, & So, 2011). Spam is called to be sent to unsolicited messages by force for advertising or phishing, which we do not request and do not want to receive. These attacks have caused a tremendous amount of damage (Verma, Crane, & Gnawali, 2018). The download of malware while visiting a URL is called as Drive-by download (Cova, Kruegel, & Vigna, 2010). Lastly, Social Engineering and Phishing attacks guide users to reveal sensitive and private information by acting as genuine web pages (Heartfield & Loukas, 2015). The attackers create copies of the popular web pages used by users such as Facebook and Google and compromise victim computers by placing various pieces of malicious code in the manipulated web site’s HTML code. Besides, the ubiquitous use of smartphones encourages the increase of mobile and Quick Response (QR) code phishing activities, especially to deceive the elderly that encode fake URLs in QR codes. The dark side of the Internet has attracted increasing attention and bedevilled the world (Patil & Patil, 2015). Internet security software cannot always detect malware from malicious websites and drive-by downloads. It can, however, prevent you from getting them in the first place (Symantec, 2020). Malicious URLs detection is not adequately addressed yet and causes enormous losses each year. In the fourth quarter of 2019, more than 162,000 unique phishing URLs were detected globally (Statista, 2020). Even though the security components used today are trying to detect such malicious sites and web addresses, these components are evading by using different methods implemented by the attackers. Researchers have studied to gather effective solutions for Malicious URL Detection. One of the most popular ways is the blacklist method that uses records of known malicious URLs to filter the incoming URLs. However, blacklists have some limitations, and this approach useless for new malicious sites that are created continuously. Security components have started to use innovative applications of machine learning and artificial intelligence-based prediction models to cope with this problem, during the last decades (Garera, Provos, Chew, & Rubin, 2007) (Kuyama & Kakizaki, 2016) (Ma, Saul, Savage, & Voelker, beyond blacklists: learning to detect malicious web sites from suspicious URLs, 2009) (Ma, Saul, Savage, & Voelker., Learning to Detect Malicious URLs, 2011). They have started to prefer machine learning and artificial intelligence prediction instead of being signature-based for Malicious URL Detection. Machine Learning approaches apply a set of URLs as training data and learn a prediction function to classify whether a URL is malicious or benign. This approach allows them to generalize to new URLs, unlike blacklisting methods. Soon, these solutions will need to be used in Cyber-Physical Systems (CPS), and the other area will be to identify harmful sites and URL addresses. As a result, it can be noted that Artificial Intelligence-based antimalware tools will aid to detect recent malware attacks and develop scanning engines. This chapter aims to present the basics of machine learning-based malicious URL detection. The rest of the chapter is organized as follows. In the Background section, a review of the existing approach and a summary of the literature in the field of URL classification is presented, In Dataset and Analysis section, the publicly available dataset is discussed. In Method Section, the fundamentals of machine-learning models are explained. In the Experiment section, the detection performance comparisons of different algorithms are evaluated. Lastly, conclusion and future directions remarks are given.

DATASET COLLECTION AND PREPROCESSING

While there are a variety of features that one can use to classify if a web page is spam, this project aims to use only the URL and limited metadata information to classify if web pages are spam/not spam. This choice was made for performance reasons, as scraping HTML from web pages is resource-intensive and not useful since the page must have already been crawled. In the context of a search engine, it is often advantageous to be able to detect if a given URL is malicious before a page being crawled. This way, URL’s that are likely to be malicious can be deprioritized during crawling, and those resources can be used to crawl more useful pages that are less likely to be malicious.

In this work, we used an open public original dataset of UCI Machine Learning Repository and seen in this address: https://archive.ics.uci.edu/ml/datasets/URL+Reputation. It is an Anonymized 120-day subset of the ICML-09 URL data containing 2.4 million examples and 3.2 million features. Each day’s data is stored in separate files in SVM-light format. A label of +1 corresponds to a malicious URL, and -1 corresponds to a benign URL (Ma, Lawrence, Saul, Stefan, & Geoffrey, 2009)

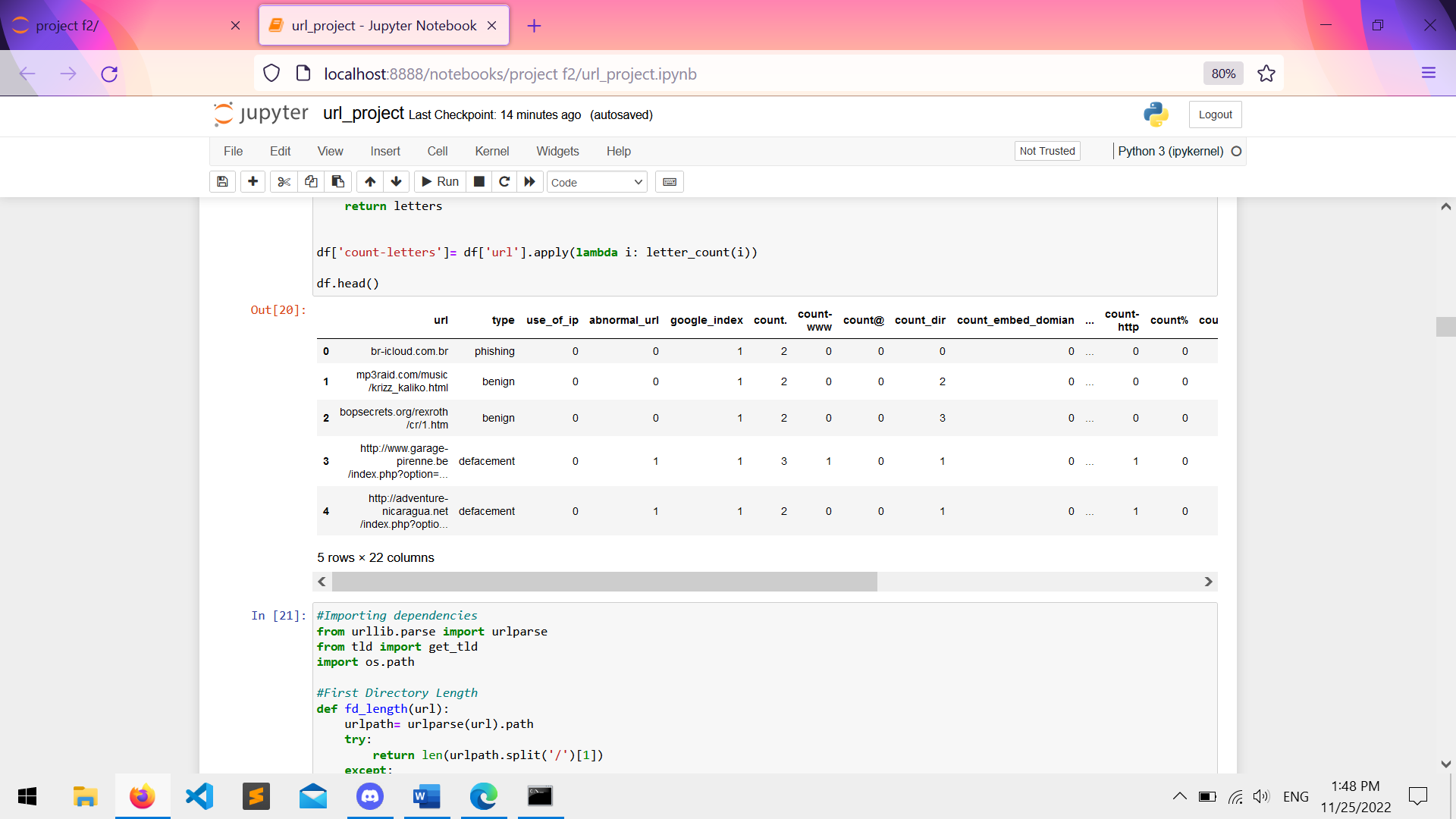
The features used in this research are anonymized. On the other hand, the features match the lexical and host-based features obtained for each URL.

the types of lexical and host-based features and the numbers of each class contribution. Word types make up 62% of the features, and host-based types make up 38%. Feature types and reasons for including them for classification will be explained. Lexical features allow us to understand the difference between malicious URLs that lead to “look different” than benign URLs. For example, the appearance of the ‘.com’ token at the URL ‘www.google. com’ is usual. On the other hand, the presence of ‘.com’ in ‘www.google.com.phishy.biz’ or ‘phish.biz/ www.google.com/index.aspx’ may show an attempt to emulate the domain name of a valid business web site. Furthermore, there are explicitly indicating keywords that tend to appear in malicious URLs — e.g., ‘googleisapi’ would frequently appear in the context of URLs trying to spoof a Google page. To fulfil these features, we use a bag-of-words description of tokens in the URL, where ‘/’, ‘?’, ‘.’, ‘=’, ‘-’, and” are delimiters. We discover tokens that appear in the hostname, path, the top-level domain (TLD), the primary domain name, and the last token of the path. Consequently, ‘com’ in the TLD position of a URL would be a different token from ‘com’ in other parts of the URL. We also use additional features; the lengths of the hostname and the URL as features. Host-based features explain characteristics of the Web site host as recognized by the hostname portion of the URL. This feature allows us to approximate “where” malicious sites are hosted, “who” own them, and “how” they are managed. We examine the following sets of properties to construct host-based features: The location feature refers to the host’s geography, IP address prefix, and the autonomous system (AS) number. If a specific IP prefix of an Internet service provider (ISP) hosts malicious URLs, then this ISP is considered as malicious. Connection speed feature: If a malicious URL tends to live on compromised residential machines, then host connection speed is recorded. Membership in blacklists features: If the URLs were present in blacklists. Other DNS-related properties feature: These include time-to-live (TTL), spam-related domain name heuristics, and whether the DNS records share the same ISP.

While measuring the performance of a model to be used, it should not only be looked at the learning algorithm used but also the size of the training and test sets, incorrect classification or class distribution play an important role. The data set is divided into two as education and test data sets. Here, the training data set is used for the training of the selected classifier model. The most appropriate parameter values and performance measures for the model creation phase are determined at this stage. The test data set is also used to measure the overall performance of the model.

Modified data is split into training and test set using the train\_test\_split function. 30% of the data is set aside for testing and evaluation. We have used the confusion matrix and time taken for execution here for evaluating the ML models

Feature Engineering

In real-world problems, the data volume is huge; the correlations among features and patterns are changing over time are complicated in malicious URLs detection. To cope with these problems, feature engineering has a significant role in addressing these problems. The main idea behind feature engineering is to provide features to machine learning algorithms to apply better. Feature engineering studies are used at the stage of obtaining very critical and processable data for data science. In many cases, there are many different advantages, such as the solution of missing data, the ability to solve many various problems, such as text processing, image processing, which are typically difficult to process, and that the data can be used as a time-dependent series. In most cases, it is also possible to say that the feature extraction consists of steps connected in the form of a pipeline. It is possible to see attribute engineering as a process that increases the success of the system in general. However, this approach has the possibility of misleading. 

In general, the results achieved are a result of the selected models and attributes, and excellent results do not always indicate a competent data mining process. For example, correct attributes obtained as a result of good attribute engineering enable simpler models to work more successfully. Simple models, on the other hand, are significant for building systems that operate faster, are understood, and can be maintained simply. In this respect, it can be said that attribute engineering contributes to the flexibility of the system. Feature engineering is used in many different fields. For example, extracting attributes on time dependent values and time series, mining data on text, and even using some metrics obtained on social networks as attributes pose some common problems in the literature. In some studies, it is seen that these features are extracted from different areas and used in cross areas. Feature extraction consists of five necessary steps, and these steps can be listed as follows (Zhang, Ren, & Jiang, Application of feature engineering for phishing detection, 2016):

• Indicator Variables

• Interaction Features

• Feature Representation

• External Data

•Error Analysis

PROPOSED ALGORITHM

Random Forest

Random Forest (RF) is the ensemble classifier, which collects the results of many decision trees by majority vote. In ensemble learning, the results of multiple classifiers are brought together, and a single decision is made on behalf of the community. Each decision tree in the forest is created by selecting different samples from the original data set using the bootstrap technique. Then, the decisions made by many different individual trees are subject to voting and present the class with the highest number of votes as the class estimate of the committee. In the RF method, trees are created by CART (Classification and Regression Trees) algorithms and boot bagging combination method. The data set is divided into training and test data. From the training data set, samples are selected as Bootstrap (resampled and sampled) technique, which will form trees (in a bag) and data that will not build trees (out of the bag). 1/3 of the training set is divided into data that will not form trees, and 2/3 of them will be data that will build trees. m variables are selected in each node among all variables, and the best separations are provided by using the Gini index.

Pk is the frequency of instances of class k in the node, and k is the total number of classes. Estimations are made, and estimation errors are calculated in the model established thanks to the data that does not create trees. By combining the out of bag estimates made by each tree, the error of the decision tree is estimated. Each tree is given a weight based on the out of bag error rate, and the tree with the lowest error rate receives the highest weight, while the tree with the highest error rate receives the lowest weight (Han, Kamber, & Pei, 2012). Each decision tree that classifies gets individual votes, and at the end of the transaction, the classification made by the decision tree with the highest vote is used. Since each decision tree cannot show the same performance when it encounters a different data group than the data group it is trained in, the method combines a large number of decision trees, thereby increasing the classification performance and correct classification rate. In the RF method, the tree starts with a single node. If all the samples belong to the same class, the node ends as a leaf, and a class label is given. If the examples are not included in the same class, the feature that will best divide the samples into classes is selected. One of the advantages of the RF method is that it determines the degree of importance among the features. While deciding the feature importance, the following steps are carried out:

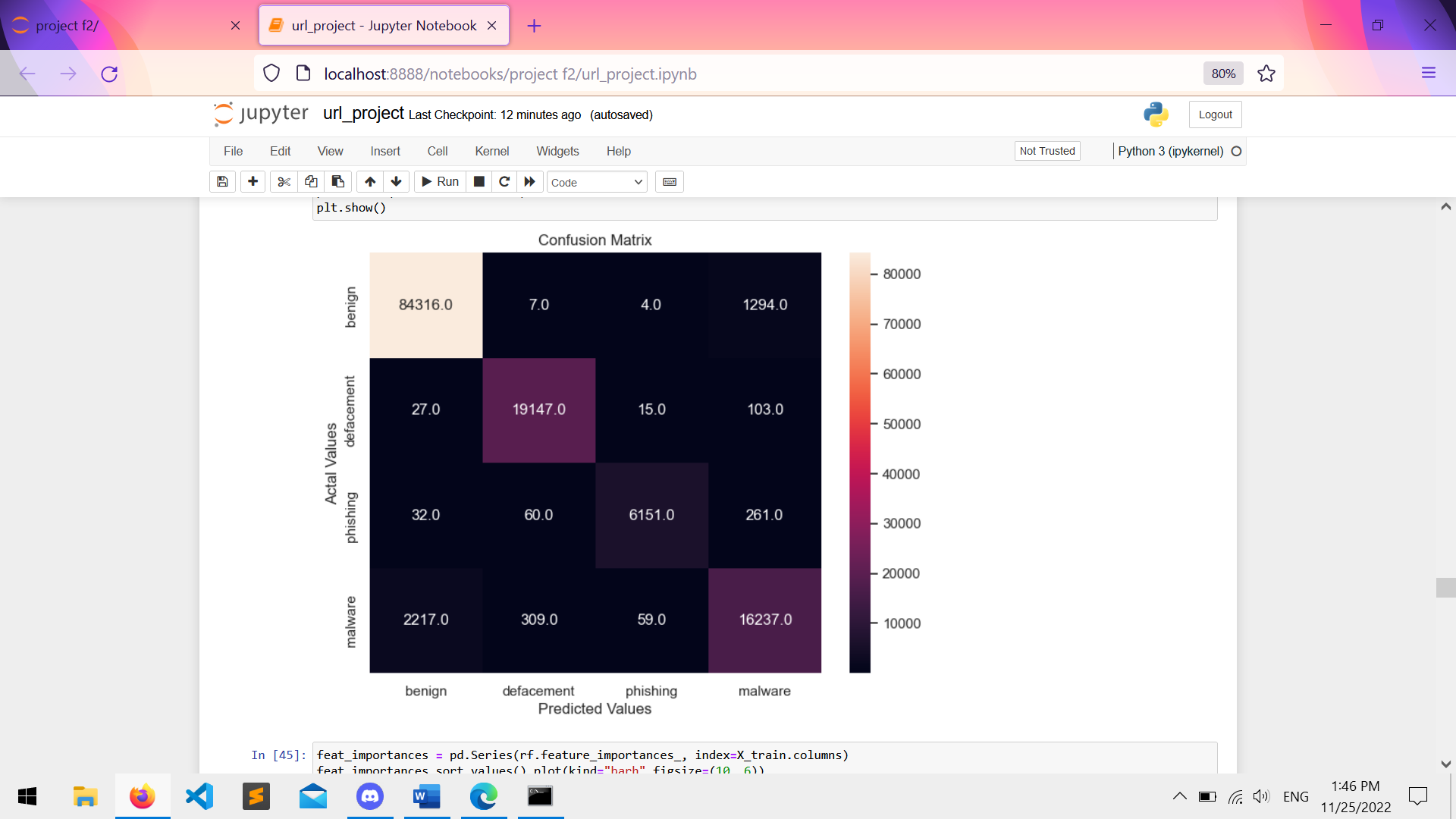
• After the decision tree is created, the classes predicted according to out of the bag are placed from top to bottom, and the correct classification number is recorded.

• m in out of the bag. The values of the variable are relocated and now become the changed out of the bag.

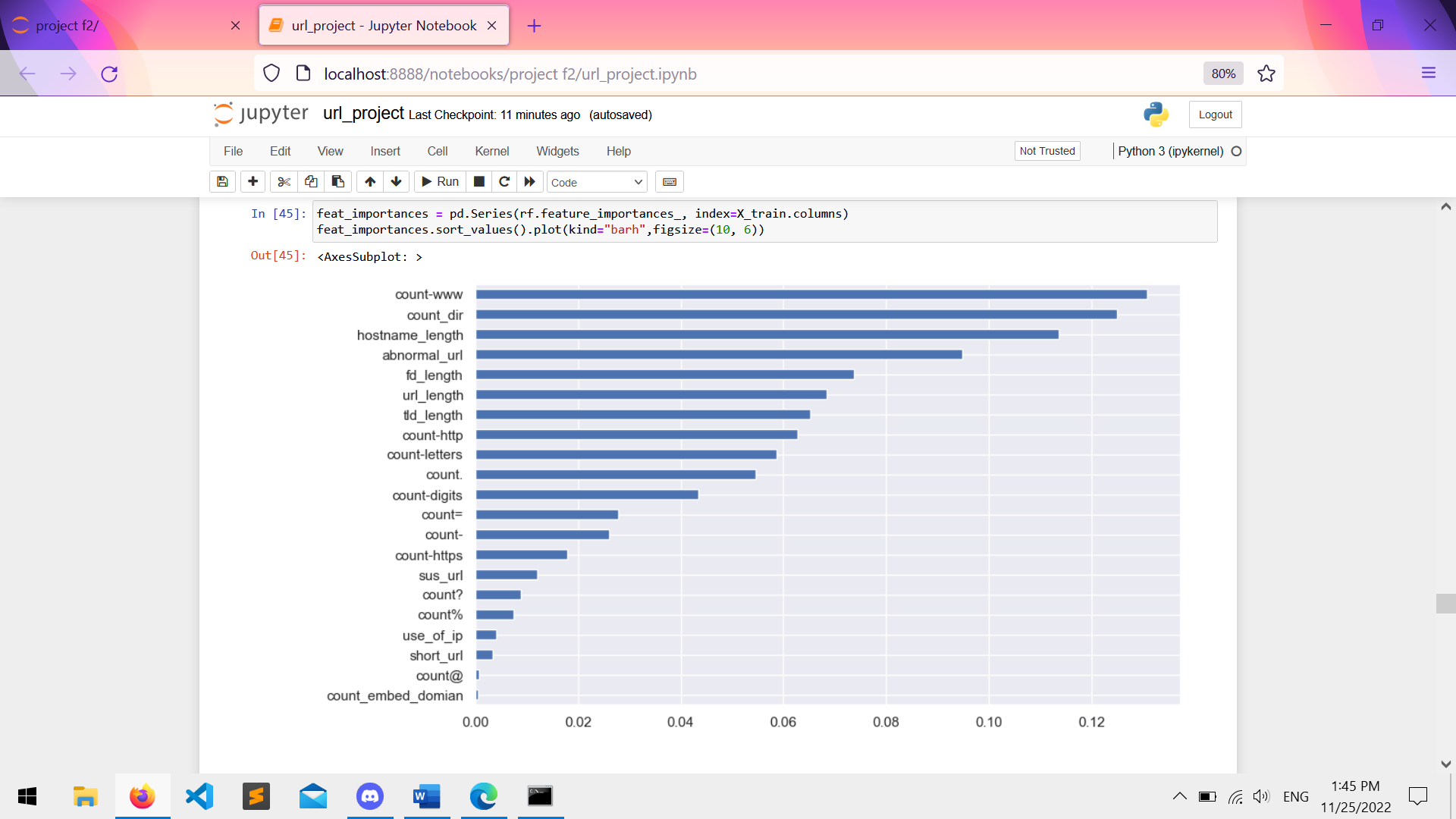
• The modified out of bag values are placed from top to bottom on the previously created decision tree, and the correct classification number is calculated.

• The out of bag correct classification number is changed from the out of bag exact classification number and the di is calculated.

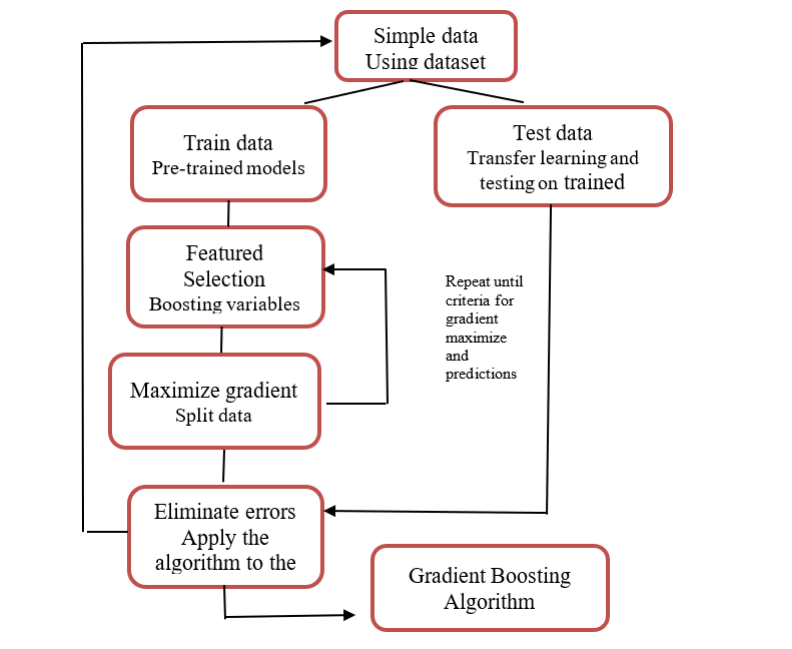
The steps described are applied individually for each variable. Thus, the severity scores of each variable are calculated. Another method for varying severity levels was calculated with the help of Gini values. The difference between the Gini index values before the branching from the variable takes place, and the Gini values after the branching are taken. This value is calculated for all trees, and the values obtained are summed. This value is calculated for all variables, and their significance is calculated from here. Random forest is a well-known ensemble method using different decision trees using independent and identically distributed random samples from the input dataset. Each decision tree classifier (ci) selects a sample Xi from training dataset X. Then the algorithm builds an independent decision tree algorithm using this sub dataset.



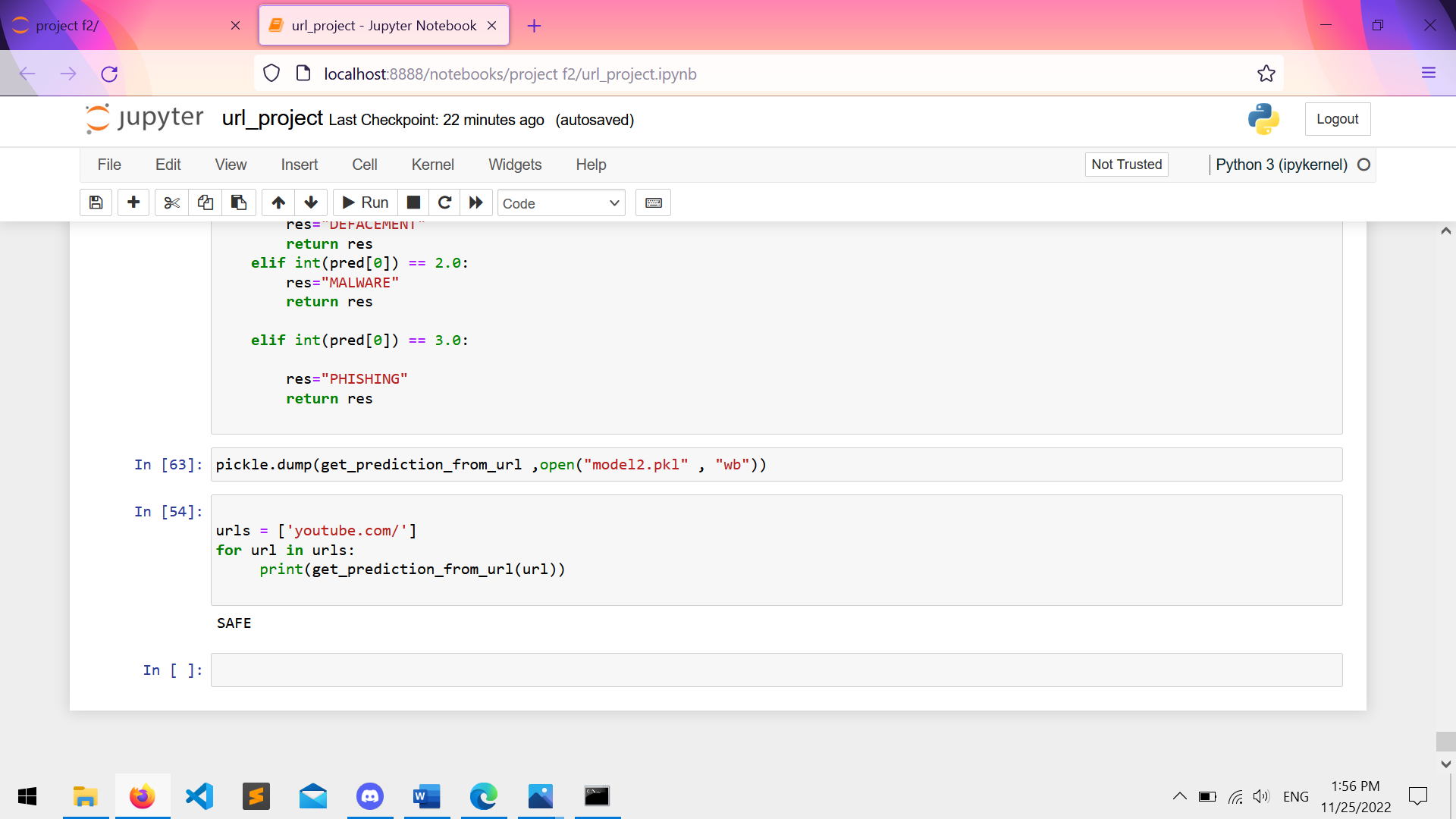
EXPERIMENTS

In this section, we will present detailed information about the dataset. In this study, we used the publicly available URL reputation dataset. We will show the details of the dataset using the data visualization techniques applying the exploratory data analysis methods. Since the number of features of the data set is 651191, we have transformed the data set into nine features using the PCA method. Figure 2 shows the scatter plots of 9 features. As shown in the figure, the scatter plot graphics of the features are separable from each other. Thus, the data set resulting from the applied PCA model, the decision boundaries are clear and linearly separable. The highly correlated features in a data set have adverse effects on the classification performance of the model. 

Besides, the high number of features causes an increase in the execution time of the classification algorithms. For this reason, the correlation matrix of the dataset should be used and examined, and if there are highly correlated features, they should be converted. Figure 3 shows the correlation matrix of the URL reputation dataset. As shown in the figure, the correlation values between the features are very close to 0. For this reason, the new dataset will have a positive effect on both execution time and classification performance. Unfortunately, there is not enough data in the cybersecurity field. Currently, there are more publicly available datasets such as malware and Apache Http logs [44,45]. The URL reputation data set used in this study was created in 2009.



RESULT



CONCLUSION AND FUTURE SCOPE

With the developing computer and system technologies, people exchange information over the Internet that attracts people due to the convenience of services they offer day by day and beyond that, they do many other things related to daily life. During these processes, users have intelligence and critical information such as descriptive usernames and passwords. Most network applications detect their users with them. The rapid increase of the web pages and applications caused them to become the primary target for the attackers. Today, the number of malicious websites has increased considerably. Malicious behaviour of trusted or malicious users threatens network applications. Users who are unaware of anything become a victim only by visiting these harmful pages. Attackers can exploit the web environment more easily by uploading or embedding malicious code on the web page instead of spreading the malware. According to the Google Research Centre, over 10% of web pages contain malicious code. Therefore, the detection of harmful web pages has become very important to protect the users of the web environment from these threats. In this respect, determining whether web pages directed to users are used for malicious behaviour is of great importance for the institution and individual users to overcome the situation with minimum damage. Recent years have witnessed detecting Malicious URL has a significant role in cybersecurity applications. Malicious URL has been a severe threat to cybersecurity. Without any questions, CPS can be considered as a crucial step in the development of data-accessing and data-processing services available on the Internet. Researchers have studied to gather effective solutions for Malicious URL Detection. Machine learning approaches have been widely applied in Malicious URL detection. In this study, the data of the web pages used for phishing shared in the UCI data warehouse are used. This research analysed the performance of machine learning algorithms for malware detection. We applied Random Forest and Gradient Boosting machine learning algorithms for malicious URL detection. The experimental results of the proposed method indicate that the performance of the Machine Learning Model (Random Forest) in processing the large dataset and predicting the website as benign or malicious is significantly pretty impressive (98.6%). This indicates we can very quickly build deployable and reliable machine learning models for malicious URL detection.

REFERENCES

Alshboul, Y., Nepali, R. K., & Wang, Y. (2015). Detecting malicious short URLs on Twitter. Twenty-first Americas Conference on Information Systems, 1-7. Bannur, S. N., Saul, L. K., & Savage, S. (2011).

Judging a site by its content: learning the textual, structural, and visual features of malicious web pages. Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence. 10.1145/2046684.2046686 Bazrafshan, Z., Hashemi, H., & Fard, S. (2013).

A survey on heuristic malware detection techniques. The 5th Conference on Information and Knowledge Technology, 113-120. Bengio, Y., Simard, P., & Frasconi, P. (1994).

Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166. doi:10.1109/72.279181 PMID:18267787 Breiman, L. (2001).

Random forests. Machine Learning, 45(1), 5–32. doi:10.1023/A:1010933404324 Canali, D., Cova, M., Vigna, G., & Kruegel, C. (2011).

Prophiler: a fast filter for the large-scale detection of malicious web pages. In Proceedings of the 20th international conference on World wide web. ACM. 10.1145/1963405.1963436