Malicious URL Detection Using Machine Learning

Chapter · November 2020

DOI: 10.4018/978-1-7998-5101-1.ch008

3 authors, including:

Ferhat Ozgur Catak
University of Stavanger (UiS)
72 PUBLICATIONS **426** CITATIONS
SEE PROFILE

Kevser Şahinbaş
Istanbul Medipol University
13 PUBLICATIONS **75** CITATIONS
SEE PROFILE

Artificial Intelligence Paradigms for Smart Cyber-Physical Systems

Ashish Kumar Luhach
The PNG University of Technology, Papua New Guinea

Atilla Elçi
Hasan Kalyoncu University, Turkey

A volume in the Advances in Systems Analysis, Software Engineering, and High Performance Computing (ASASEHPC) Book Series

Published in the United States of America by
IGI Global
Engineering Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA, USA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Names: Luhach, Ashish Kumar, 1985- editor. | Elçi, Atilla, editor.
Title: Artificial intelligence paradigms for smart cyber-physical systems / Ashish Kumar Luhach and Atilla Elçi, editors.
Description: Hershey, PA : Engineering Science Reference, [2021] | Includes bibliographical references and index. | Summary: "This book focuses upon the recent advances in the realization of Artificial Intelligence-based approaches towards affecting secure Cyber-Physical Systems. It features contributions pertaining to this multidisciplinary paradigm, in particular, in its application to building sustainable space by investigating state-of-art research issues, applications and achievements in the field of Computational Intelligence Paradigms for Cyber-Physical Systems"-- Provided by publisher.
Identifiers: LCCN 2020019421 (print) | LCCN 2020019422 (ebook) | ISBN 9781799851011 (h/c) | ISBN 9781799858461 (s/c) | ISBN 9781799851028 (ebook)
Subjects: LCSH: Computer networks--Security measures--Data processing. | Internet of things--Security measures--Data processing. | Cooperating objects (Computer systems) | Artificial intelligence--Industrial applications.
Classification: LCC TK5105.59 .A79 2021 (print) | LCC TK5105.59 (ebook) | DDC 005.8--dc23
LC record available at https://lccn.loc.gov/2020019421
LC ebook record available at https://lccn.loc.gov/2020019422

This book is published in the IGI Global book series Advances in Systems Analysis, Software Engineering, and High Performance Computing (ASASEHPC) (ISSN: 2327-3453; eISSN: 2327-3461)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 8
Malicious URL Detection Using Machine Learning

Ferhat Ozgur Catak
https://orcid.org/0000-0002-2434-9966
Simula Research Laboratory, Oslo, Norway

Kevser Sahinbas
Istanbul Medipol University, Turkey

Volkan Dörtkardeş
Şahıs Adına, Turkey

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.

INTRODUCTION

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 mali-

DOI: 10.4018/978-1-7998-5101-1.ch008

cious 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 bedeviled 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.

Background

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.

Shabtai et al. present a classification model for malware detection mainly by focusing on machine learning algorithms such as Bayesian Networks, K-Nearest Neighbor, Artificial Neural Networks, Naive Bayes, and feature selection techniques. Gupta and Singhal examine that the RF tree achieves an excellent result to detect Phishing URLs in minimum execution time. Firdausi et al. analyzed malware and benign files by collecting 250 unique benign and 220 individual malware software samples to train Support Vector Machine (SVM), Multilayer Perceptron (MLP) neural network, k-Nearest Neighbor, Naive Bayes and J48 decision tree on their dataset. They gained the highest accuracy of 96.8% by the J48 decision tree. Bazrafshan et al. categorize detecting malicious software into behavior-based methods, signature-based methods, and heuristic-based methods. Santos et al. collected 1, 000 for benign files and 1, 000 malicious files to apply SVMs, Decision Trees, and KNNs to detect malicious URLs. Souri et al. provide two methods for malware detection approaches as signature-based methods and behavior-based methods. Parekh et al. propose a model that applied Random Forest and gained a 95% accuracy rate as the highest to detect phishing websites. Rieck et al. gathered 10, 072 unique samples and divided them into 14 malware families to train Support Vector Machine and achieved 88% accuracy in testing correct malware. Ucci et al. present a model that applied machine learning algorithms to feature types extracted from Portable Executable files. Ye et al. analyze feature selection, feature extraction, and classification by applying machine learning techniques for malware detection. Alshboul et al. applied Support Vector Machine (SVM) classifiers and Decision Trees for Malicious URL/web detection and claimed that SVM classifier is one of the most widely used for Malicious URL Detection. In the literature, Logistic Regression has attracted increasing attention for Malicious URL Detection . Canali et al. proposed a model that applied Naive Bayes for Malicious URL Detection. The extreme Learning Machines (ELM) approach is used for classifying the phishing web sites . Singh et al. propose a method for malware detection by applying the Support Vector Machine. Kazemian et al. have used machine learning techniques such as K-Nearest Neighbor, Support Vector Machines, Naive Bayes Classifier, and K Means rather than traditional methods of detecting whether they exist in a predetermined blacklist for detecting harmful web pages. Hou et al. identify dynamic HTML codes that made it difficult to detect harmful web pages by easily hiding and changing with the method they proposed in a study they conducted.

For the tests, a data set consisting of 176 harmful samples and 965 harmless samples collected from StopBadware site were used. In tests performed using the Decision Tree, Naive Bayes, Support Vector Machines, and AdaBoost Decision Tree classifiers, it was determined that the best result belonged to the AdaBoost Decision Tree classifier with a rate of 96.14%. Komiya et al. aim to detect SQL Injection and XSS attacks in a study they conducted. In tests performed using the Nearest Neighbor and Support Vector Machine, a success rate of 99.16% for SQL Injection and 98.95% for XSS was obtained in the classification process using SVM and Gauss cores. In a study by Liu et al., they aimed to classify suspicious URLs using machine learning techniques. For this purpose, regardless of the content of a web page, using URL features directly, harmless (secure web pages), harmful (web pages that disrupt computer operations, collect sensitive information or private access systems) and phishing (username, passwords, credit card information, etc.) labeling has been carried out in three classes: web pages that appear to be reliable, designed to capture data. Using only URL features has eliminated runtime latency and the possibility of users exposed to browser-based vulnerabilities. Manek et al. aim to detect harmful web pages by using the features of the web page based on URL-based properties, server information, and the content of the web page. Support vector machines and Naive Bayes classifiers and web pages created for phishing and malware distribution were determined with high accuracy.

DATASET AND ANALYSIS

Overview

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.

Data

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).

Features

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. ***Table 1*** shows the types of lexical and host-based features

Table 1. Feature breakdown on Day 100 of the experiments

Lexical		Host-Based	
Feature type	**Count**	**Feature type**	**Count**
Hostname	835,764	WHOIS info	917,776
Primary domain	738,201	IP prefix	131,930
Path tokens	124,401	AS number	39,843
Last path token	92,367	Geographic	28,263
TLD	522	Conn. Misc.	37
Lexical	1,791,261	Host-Based	1,117,901

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 fulfill 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.

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

METHODS

This section presents the theoretical review of Random Forest and Gradient Boosting algorithms.

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. In the formula below, the mathematical calculation of the Gini index is indicated:

$$Gini(p) = \sum_{k=1}^{K} P_k (1 - P_k) \tag{1}$$

P_k 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 d_i is calculated.

$$Feature\ Importance\ Score = \frac{\bar{d}}{SH_{d_i}} \tag{2}$$

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 (c_i) selects a sample X_i from training dataset X. Then the algorithm builds an independent decision tree algorithm using this sub dataset. Algorithm 1 shows the pseudo-code of the Random Forest algorithm.

Algorithm 1: *Random Forest*

```
Input:Number of classifier c, Training dataset X
for i to c do
Random sampling $\mathbf{X}_i$ with replacement from X
Build full decision tree classifier using $\mathbf{X}_i$
Return all classifiers
```

Gradient Boosting

In general, augmentation algorithms try to find a robust prediction model by combining weak prediction models in a recurring manner according to specific rules. Gradient Boosting, which is a boosting algorithm, was first proposed for use in classification problems, and over time it has become a preferred method both for classification and regression problems (Friedman, 2002). Gradient Boosting creates a f_1 function that generates predictions during the first iteration. Calculates the difference between estimates and target value and creates f_2 function for these differences. In the second refresh, f_1 and f_2 combine functions and the difference between re-estimates and calculated targets are combined. In this way, it tries to add the success of the f_1 function continuously and reduce the difference between predictions and goals to zero (Bengio, Simard, & Frasconi, 1994).

Although the gradient boosting method is generally similar to the random forest method, it contains some structural differences. The trees created in this method are interdependent. When adding a new tree, some correction is provided in error generated by the tree, which was created and trained before. **Figure 1** indicates the general overview of the Gradient Boosting Algorithm.

Experiments

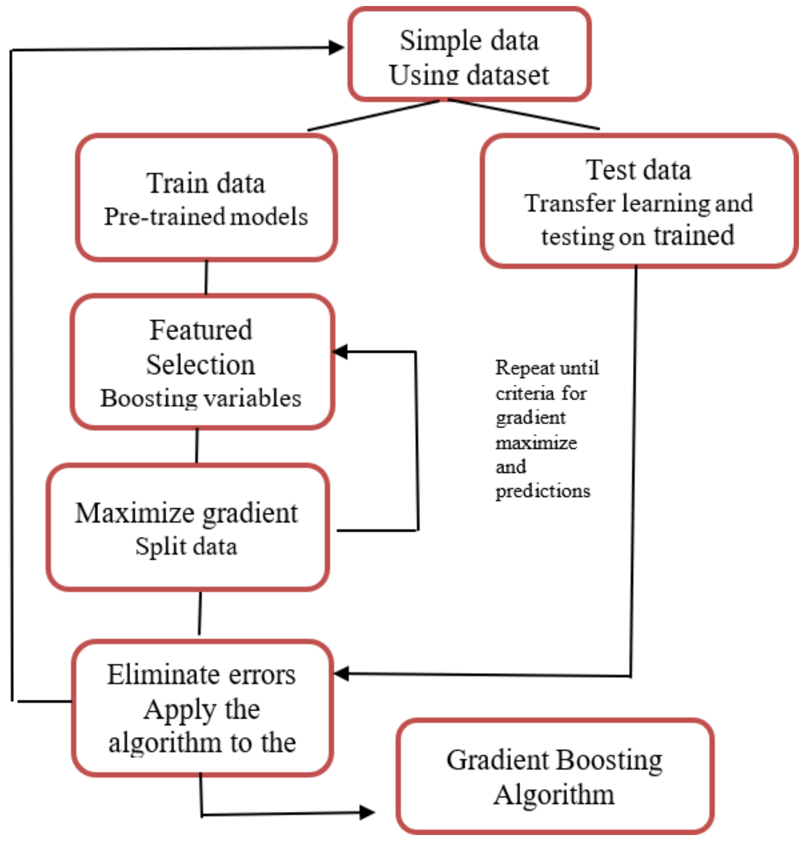
Dataset

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 3231961, 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.

Figure 1. Training and testing the dataset using a gradient boosting algorithm


Simple data
Using dataset
Train data
Pre-trained models
Test data
Transfer learning and testing on trained
Featured Selection
Boosting variables
Repeat until criteria for gradient maximize and predictions
Maximize gradient
Split data
Eliminate errors
Apply the algorithm to the
Gradient Boosting Algorithm

Data Pre-Processing

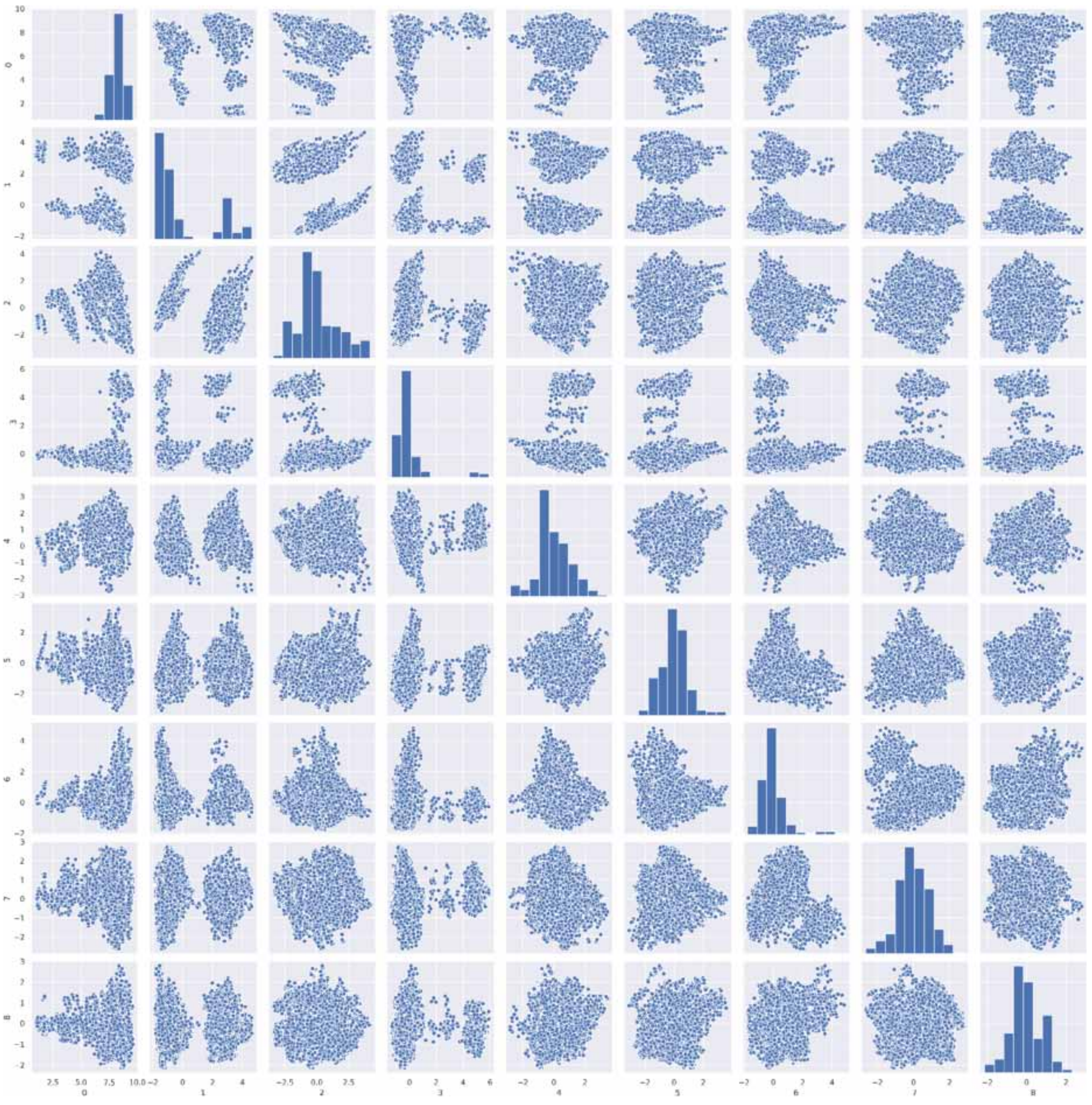
Raw data is available in the SVM light data format. *load_svmlight_file* function is used to read the data from operating system files into a sparse matrix.

The data to be used for learning has over 3 million features, and it is far more than samples of data available for training. To be used for training, feature reduction is to be done using the appropriate dimensionality reduction technique. The two methods that are available for dimensionality reduction are Principal Component Analysis (PCA) and Truncated SVD. PCA uses the covariance matrix for factorization, and it requires operating on the entire matrix, and therefore it cannot be here. TruncatedSVD is used here for dimensionality reduction as it can perform on sparse matrices.

TruncatedSVD's kernel is failing for n_components greater than 32, so we will continue further analysis based on these 32 features. Figure 2 indicates the explanation of variance by all 32 features.

Figure 4 presents that 32 dimensions can capture about 46% of the overall variation. Labels are encoded into 0 and 1 using the LabelEncoder.

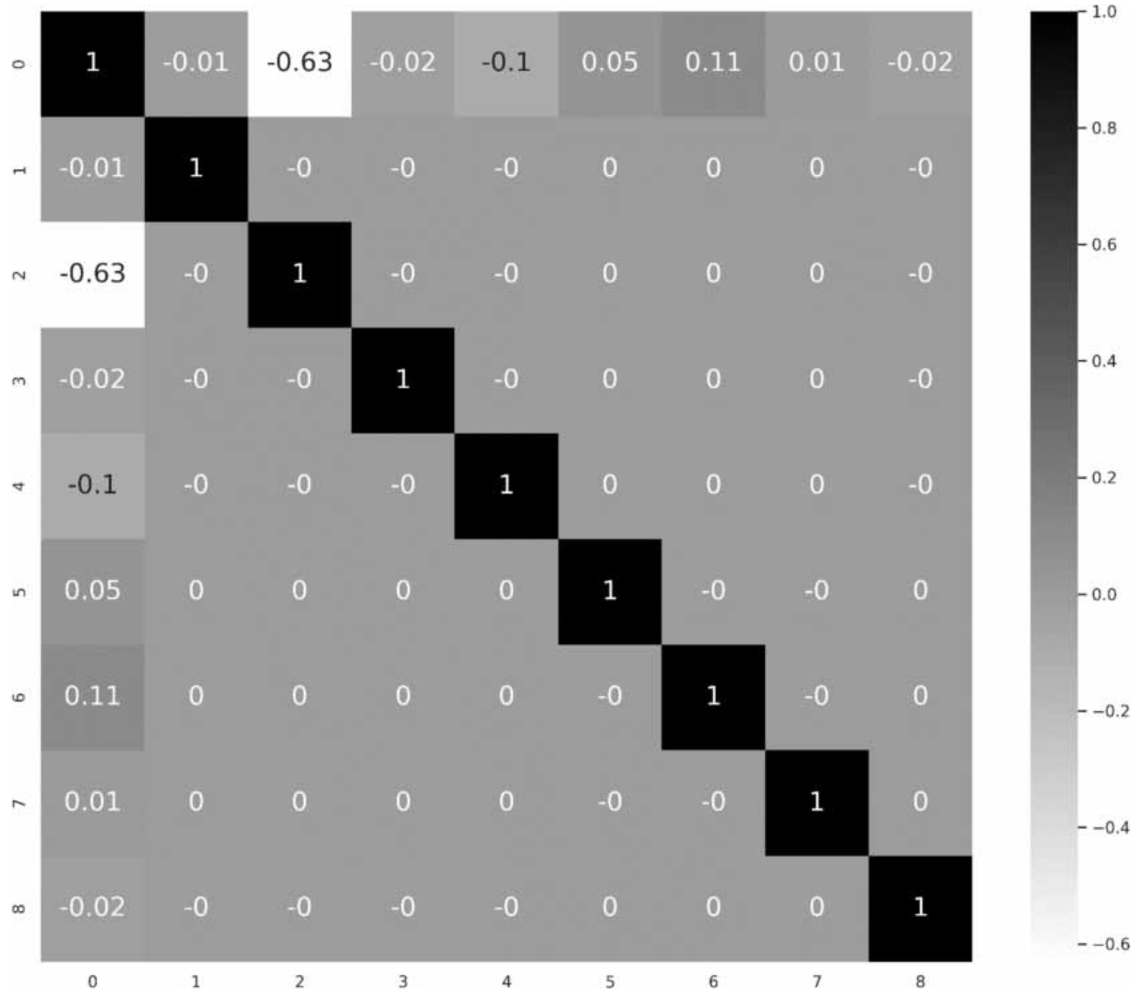
Figure 2. Pairplots of the 9 PCA features



Dataset Split Strategy

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.

Figure 3. Correlation matrix of the URL reputation dataset



	0	1	2	3	4	5	6	7	8
0	1	-0.01	-0.63	-0.02	-0.1	0.05	0.11	0.01	-0.02
1	-0.01	1	-0	-0	-0	0	0	0	-0
2	-0.63	-0	1	-0	-0	0	0	0	-0
3	-0.02	-0	-0	1	-0	0	0	0	-0
4	-0.1	-0	-0	-0	1	0	0	0	-0
5	0.05	0	0	0	0	1	-0	-0	0
6	0.11	0	0	0	0	-0	1	-0	0
7	0.01	0	0	0	0	-0	-0	1	0
8	-0.02	-0	-0	-0	-0	0	0	0	1

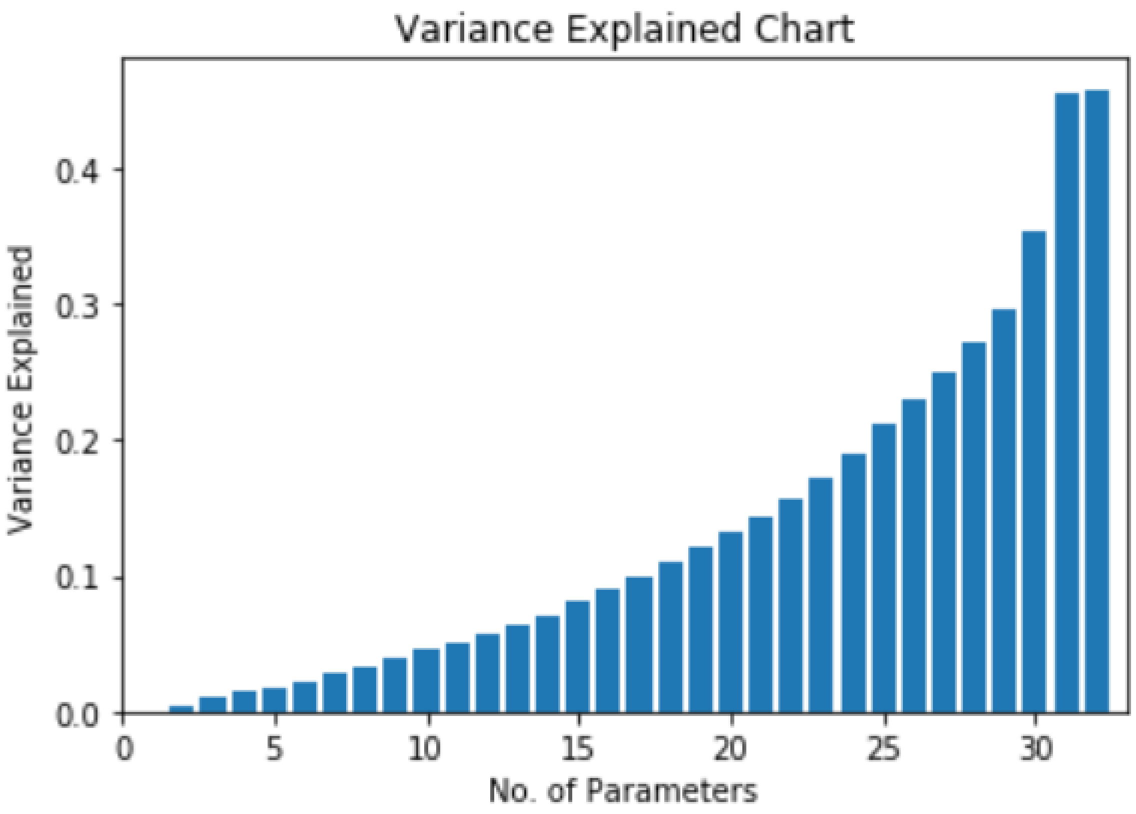
Modified data is split into training and test set using the train_test_split function. 30% of the data is set aside for testing and evaluation. We have used the confusion matrix and time taken for execution here for evaluating the ML models.

Evaluation

Based on the TP, TN, FP, and FN metrics, we compute the accuracy, sensitivity, and specificity of our proposed model. The equations introducing the metrics are equations 3, 4, 5, and 6 (Makhoul, Kubala, Schwartz, & Weischedel, 1999).

Accuracy is one of the most widely used criteria for classification performance, showing the overall correct classification success of the model created. With the correctly classified samples, the ratio to the total number of samples is calculated as in Equation 2.

Figure 4. Explanation of variance by these 32 features



$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Precision is the probability that a negative sample is also negative as a result of estimation.

$$Precision = \frac{TP}{TP + FN} \tag{4}$$

The recall is the possibility of a positive sample in the real case to be positive as a result of the estimate.

$$Recall = \frac{TN}{TN + FP} \tag{5}$$

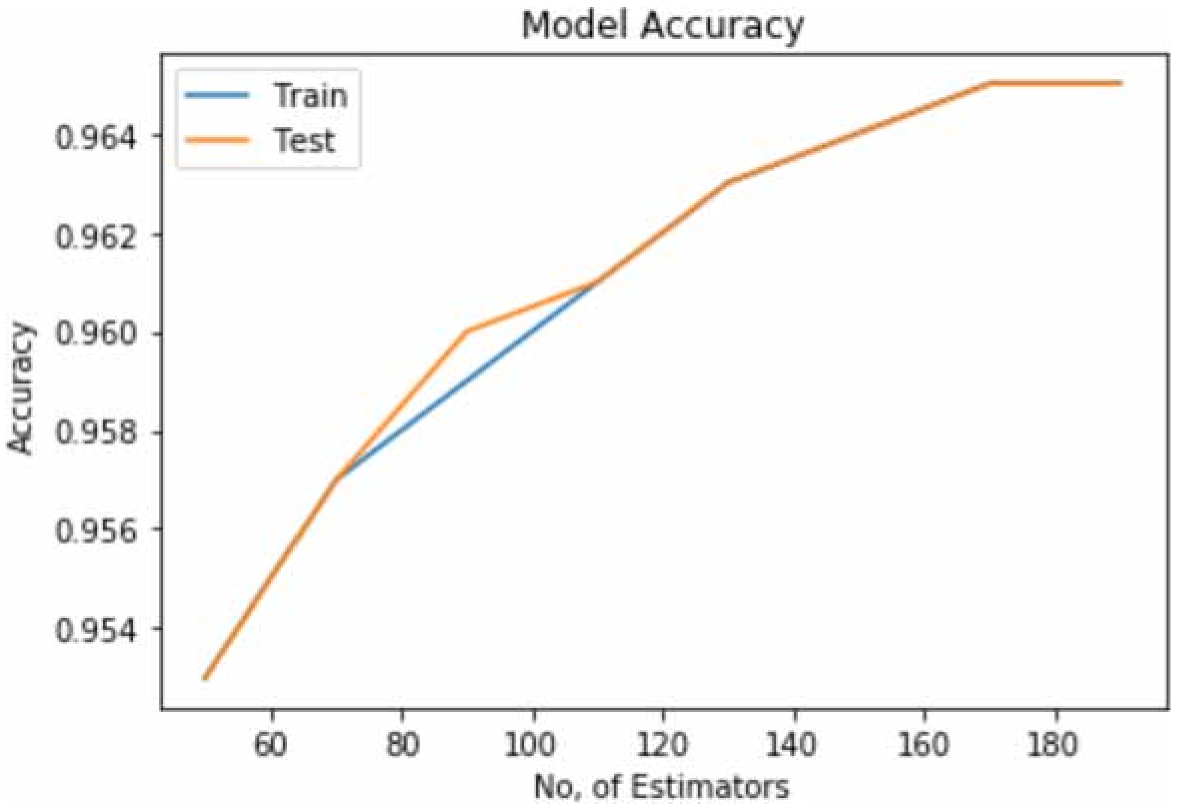
F1-score represents the harmonic average of precision and sensitivity values.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

Model Training and Tuning

In this study, Random Forest Classifier is used as it is with the default parameters. Gradient Boosting Classifier is a sophisticated algorithm with many more parameters. The first parameter to be decided is the learning rate. The algorithm is trained using a different number of estimators and the performance of the algorithm. **Figure 5** shows the model accuracy of the gradient boosting classifier algorithm.

Figure 5. Model accuracy of the gradient boosting classifier algorithm



Findings from **Figure 5** indicates that Model Accuracy plateaus are 96.5% at 170 estimators with test accuracy. We applied 170 estimators for this study. **Figure 6** shows the model Processing graph of the Gradient Boosting Classifier Algorithm.

Model processing time increases linearly with the number of estimators.

Table 2 summarizes the execution time in seconds with different depths. Model accuracy increases marginally with the increase in the Maximum Depth, but the processing time increases 2n times as shown.

Table 3 provides the execution time in seconds with a different number of features. Whereas increasing the number of features, does not cause any improvement in the performance of the model as shown.

Figure 6. Model processing graph of the gradient boosting classifier algorithm

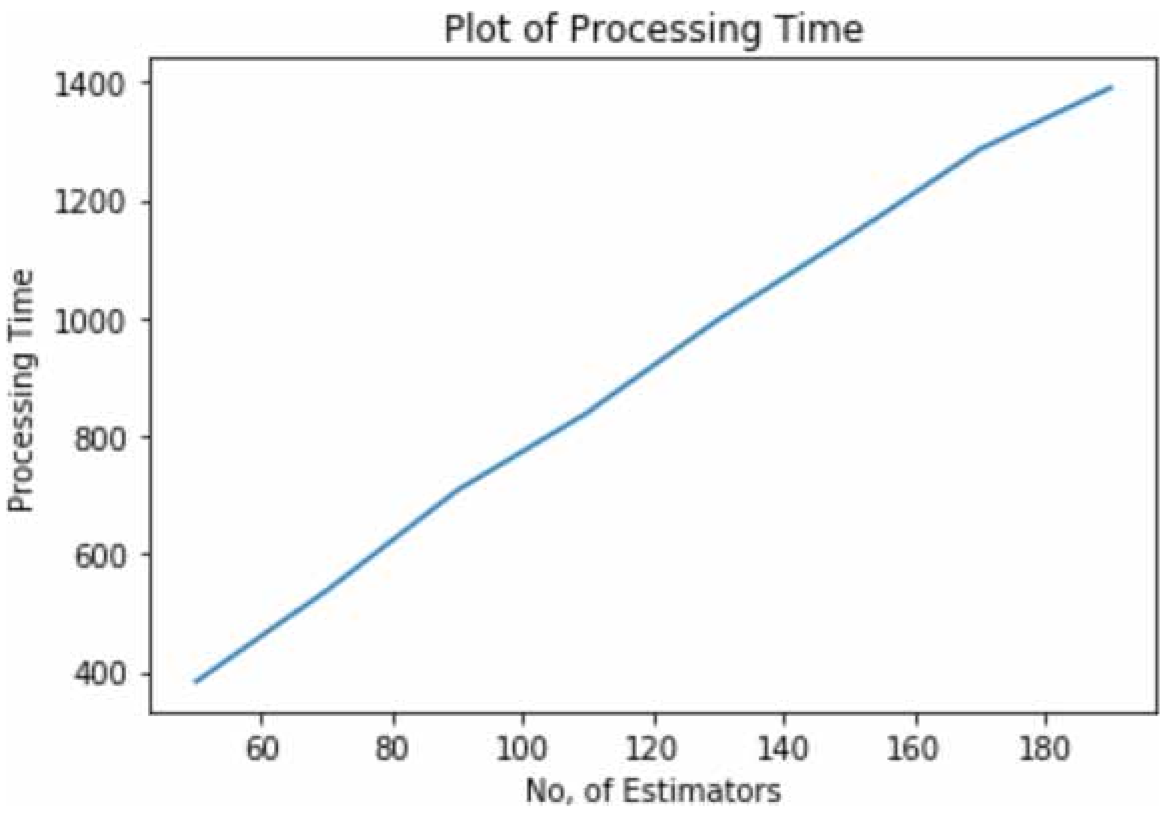


Table 2. The execution time in seconds with different depth

Features	Depth	Test Accuracy	Training Accuracy	Execution Time in Secs
4	4	0,945	0,945	147,0
4	5	0,954	0,954	263,0
4	6	0,958	0,959	488,0
4	7	0,963	0,963	754,0
4	8	0,967	0,967	1169,0

SOLUTIONS AND RECOMMENDATIONS

Figure 7 and **Figure 8** indicate the feature importance of the dataset for gradient boosting classifier and random forest, respectively.

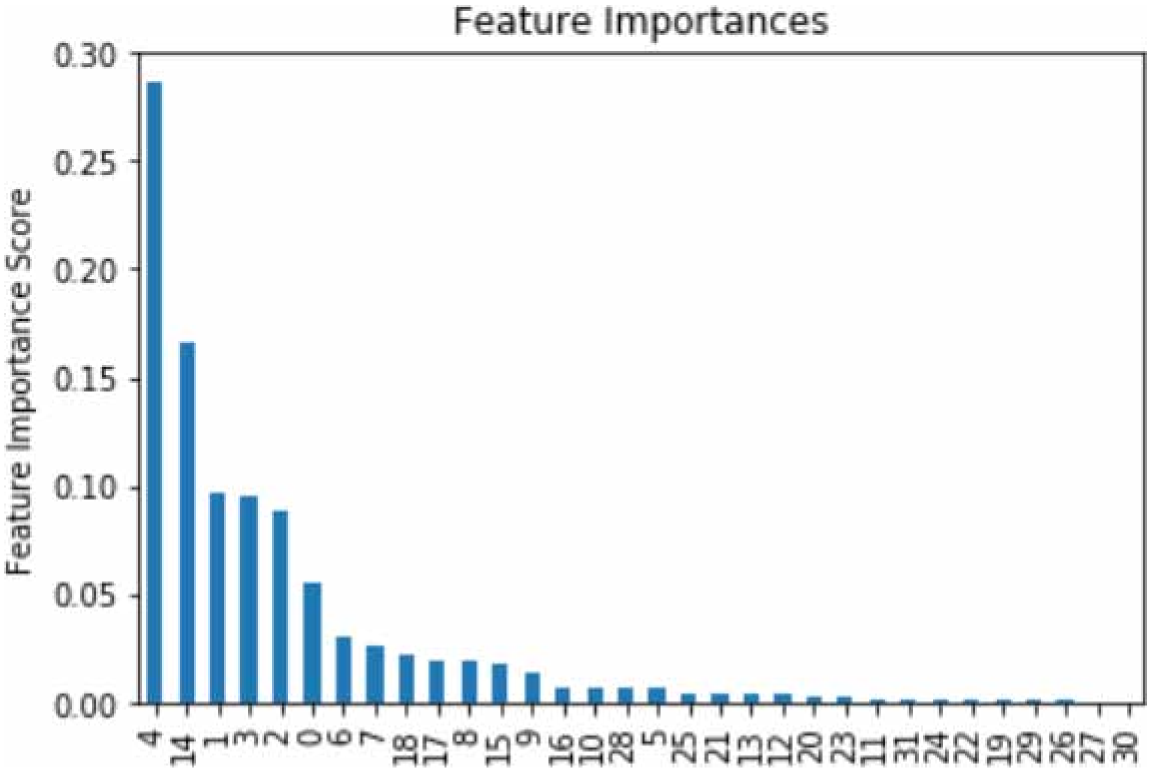
Both models are mainly in agreement concerning feature importance as shown in the table below and thereby validates the correctness of the models. Here we have demonstrated 12 features out of 32. ***Table 4*** shows the feature rank of the dataset.

We have presented two models to identify malicious URLs before crawling the page. Table 5 illustrates the classification results. Findings in ***Table 5*** indicate that the Random Forest model provides the

Table 3. The execution time in seconds with the different number of features

Features	Depth	Test Accuracy	Training Accuracy	Execution Time in Secs
4	4	0,964	0,964	604,0
5	4	0,964	0,964	711,0
6	4	0,965	0,965	1286,0
7	4	0,964	0,965	940,0

Figure 7. Importance of features as computed by gradient boosting classifier



highest classification performance and performs the best out of the models tested, with 98.6% accuracy on the test data.

We used 64. bit python 3.5 environments in our development. ***Table 6*** indicates the processing environment.

FUTURE RESEARCH DIRECTIONS

With the development of new machine learning and AI-based models, new security services are expected which may change many aspects of our security issue through CPS.

Figure 8. Feature importance as per random forest

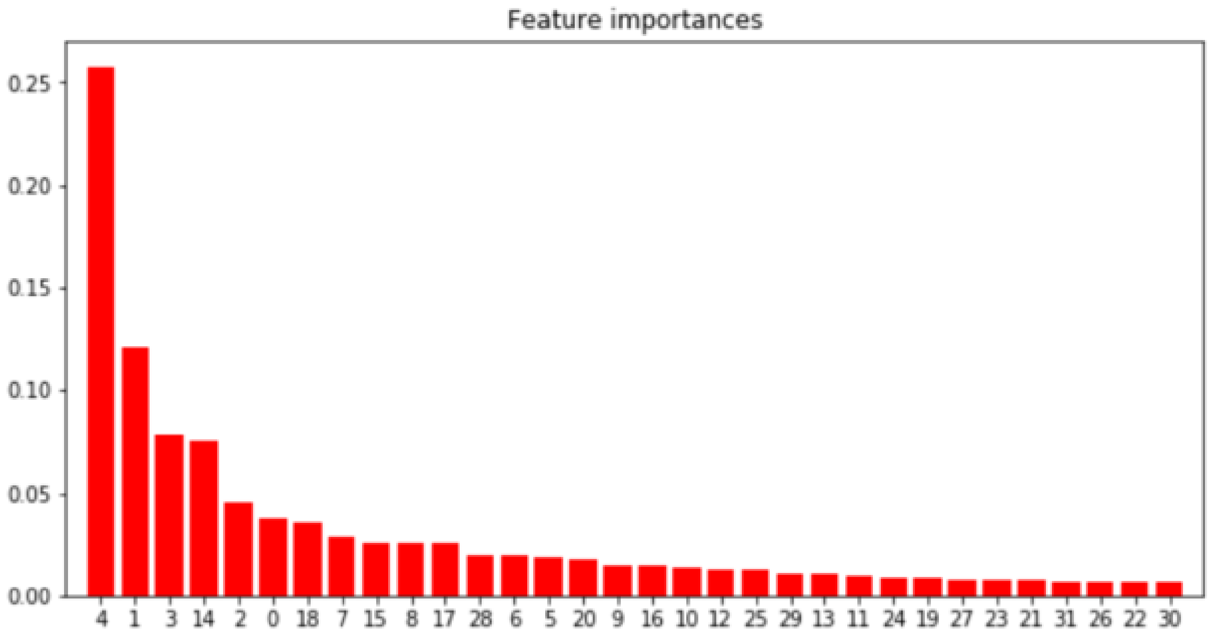


Table 4. Agreement of both models

Feature Rank	Random Forest	Gradient Boosting Classifier
1	4	4
2	1	14
3	3	1
4	14	3
5	2	2
6	0	0
7	18	6
8	7	7
9	15	18
10	8	17
11	17	8
12	28	15

Table 5. Test results of the models

Model	Precision	Recall	F1 Score	Accuracy	Execution Time in Secs
Random Forest	0,965	0,986	0,979	0,986	325,0
Gradient Boosting	0,965	0,965	0,947	0,965	825,0

Table 6. Processing environment

Machine Configuration	
System	HP Pavilion 14-ce1003tx
Operating System	Windows 10
Processor	Intel® Core™ i7-8565U (1.8 GHz base frequency, up to 4.6 GHz with Intel® Turbo Boost Technology, 8 MB cache, 4 Cores)
Memory	16 GB DDR4-2400 SDRAM (1 x 16 GB)
Graphics	NVIDIA® GeForce® MX150 (2 GB GDDR5 dedicated)
Hard Disk	512 GB PCIe® NVMe™ M.2 SSD
Machine Learning Software and Libraries	
Dataset	https://archive.ics.uci.edu/ml/datasets/URL+Reputation
Language	Python
Libraries	Pandas, Pandas_ml, SciPy, Numpy, Scikit Learn, os, Time, Matplotlib and Prettytable

The results of the proposed model are highly encouraging and are worth further investigation in terms of processing time, improvement in performance in terms of prediction accuracy, and deploying the solution in the production environment to test the efficacy of the proposed solution.

As future work, by use of J48, SVM, KNN, and NB machine learning algorithms can incorporate to compare the algorithms concerning processing time and performance of accuracy for malicious URL detection.

CONCLUSION

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 behavior 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 Center, 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 behavior 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 analyzed 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

Cova, M., Kruegel, C., & Vigna, G. (2010). Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th international conference on World wide web (WWW '10)* (pp. 281-290). Raleigh, NC: Association for Computing Machinery. 10.1145/1772690.1772720

Egele, M., Kolbitsch, C., & Platzer, C. (2009). Removing web spam links from search engine results. *Journal in Computer Virology*, *7*(1), 51–62. doi:10.100711416-009-0132-6

Firdausi, I., Lim, C., & Nugroho, A. (2010). Analysis of machine learning techniques used in behavior-based malware detection. In *2nd International Conference on Advances in Computing, Control and Telecommunication Technologies*. ACT. 10.1109/ACT.2010.33

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378. doi:10.1016/S0167-9473(01)00065-2

Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode* (S. 1-8). Alexandria, VA: Association for Computing Machinery. 10.1145/1314389.1314391

Gupta, S., & Singhal, A. (2018). *Dynamic Classification Mining Techniques for Predicting Phishing URL. In Soft Computing: Theories and Applications* (pp. 537–546). Singapore: Springer.

Gyongyi, Z., & Garcia-Molina, H. (2005). Web Spam Taxonomy. *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb 2005).*

Han, J., Kamber, M., & Pei, J. (2012). Data Mining Concepts and Techniques (3rd ed.). Morgan Kaufmann Publishers.

Heartfield, R., & Loukas, G. (2015). A Taxonomy of Attacks and a Survey of Defence Mechanisms for Semantic Social Engineering Attacks. *ACM Computing Surveys*, *48*(3), 39.

Hou, Y. T., Chang, Y., Laih, C. S., & Chen, C. M. (2010). Malicious web content detection by machine learning. *Expert Systems with Applications*, *37*(1), 55–60. doi:10.1016/j.eswa.2009.05.023

Jeong, S., Lee, J., Park, J., & Kim, C. (2017). The Social Relation Key: A new paradigm for security. *Information Systems*, *71*, 68–77. doi:10.1016/j.is.2017.07.003

Kazemian, H. B., & Ahmed, S. (2015). Comparisons of machine learning techniques for detecting malicious webpages. Expert Systems with Applications. *Expert Systems with Applications*, *42*(3), 1166–1177. doi:10.1016/j.eswa.2014.08.046

Kim, W., Jeong, O.-R., Kim, C., & So, J. (2011). The dark side of the Internet: Attacks, costs and responses. *Information Systems*, *36*(3), 675–705. doi:10.1016/j.is.2010.11.003

Komiya, R., Paik, I., & Hisada, M. (2011). Classification of malicious web code by machine learning. *Awareness Science and Technology*, 406-411.

Kulkarni, V. Y., Sinha, P. K., & Petare, M. C. (2016). Weighted hybrid decision tree model for random forest classifier. *J. Inst. Eng*, 209-2017.

Kuyama, M., & Kakizaki, R. S. (2016). Method for Detecting a Malicious Domain by Using WHOIS and DNS Features. *The Third International Conference on Digital Security and Forensics (DigitalSec2016)*, 74-80.

Liu, H., Pan, X., & Qu, Z. (2009). Learning based Malicious Web Sites Detection using Suspicious URLs. *Software Engineering*, 1–3.

Ma, J., Lawrence, K., Saul, K., Stefan, S., & Geoffrey, M. (2009). Identifying Suspicious URLs: An Application of Large-Scale Online Learning. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 681-688). Motreal: ICML. 10.1145/1553374.1553462

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2009). Beyond blacklists: learning to detect malicious web sites from suspicious URLs. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1245–1254. 10.1145/1557019.1557153

Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2011). Learning to Detect Malicious URLs. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 24. doi:10.1145/1961189.1961202

Makhoul, J., Kubala, F., Schwartz, R., & Weischedel, R. (1999). Performance measures for information extraction. *Proceedings of DARPA Broadcast News Workshop*, 249-252.

Manek, A. S., Shenoy, P. D., Mohan, M. C., & Patnaik, L. (2014). DeMalFier:Detection of Malicious Web Pages using an Effective ClassiFier. Data Science & Engineering, 83-88.

Ntoulas, A., Najork, M., Manasse, M., & Fetterly, D. (2006). Detecting spam web pages through content analysis. *Proceedings of the 15th International Conference on World Wide Web*, 83-92. 10.1145/1135777.1135794

Parekh, S., Parikh, D., & Sankhe, S. (2018). A New Method for Detection of Phishing Websites: URL Detection. *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 949-952. 10.1109/ICICCT.2018.8473085

Patil, D. R., & Patil, J. B. (2015). Survey on Malicious Web Pages Detection Techniques. *International Journal of u- and e- Service*. *Science and Technology*, *8*, 195–206.

Rieck, K., Holz, T., Wiiems, C., Dussel, P., & Laskov, P. (2008). Learning and classification of malware behavior. In *International Conference ¨ on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 108-125). Springer. 10.1007/978-3-540-70542-0_6

Santos, I., Devesa, J., Brezo, F., Nieves, J., & Bringas, P. (2013). OPEM: A static-dynamic approach for machine-learning-based malware detection. *Advances in Intelligent Systems and Computing 189 AISC*, 271-280.

Shabtai, A., Moskovitch, R., Elovici, Y., & Glezer, C. (2009). Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Information Security Technical Report*, *14*(1), 16–29. doi:10.1016/j.istr.2009.03.003

Singh, T., Troia, T., Carrado, V. A., Austin, T. H., & Stamp, M. (2016). Support vector machines and malware detection. *Journal of Computer Virology and Hacking Techniques*, 203-212.

Souri, A., & Hosseini, R. (2018). *A state-of-the-art survey of malware detection approaches using data mining techniques*. Human-centric Computing and Information Sciences. doi:10.118613673-018-0125-x

Statista. (2020). *Number of unique phishing sites detected worldwide from 3rd quarter 2013 to 1st quarter 2020*. https://www.statista.com/statistics/266155/number-of-phishing-domain-names-worldwide/

Symantec. (2020). *What is Malicious Website?* https:// us.norton.com/internetsecurity-malware-what-are-malicious-websites.html

Ucci, D., Aniello, L., & Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, *81*, 123–147. doi:10.1016/j.cose.2018.11.001

Verma, R., Crane, D., & Gnawali, O. (2018). *Phishing During and After Disaster: Hurricane Harvey. In Resilience Week (RWS)* (pp. 88–94). Denver, CO: IEEE.

Ye, Y., Chen, L., Wang, D., Li, T., & Jiang, Q. (2008). Sbmds: an interpretable string based malware detection system using svm ensemble with bagging. *J. Comput. Virol.*, 283.

Zhang, W., Jiang, Q., Chen, L., & Li, C. (2016). Two-stage ELM for phishing Web pages detection using hybrid features. *World Wide Web (Bussum).*

Zhang, W., Ren, H., & Jiang, Q. (2016). Application of feature engineering for phishing detection. *IEICE Transactions on Information and Systems*, *E99-D*(D), 1062–1070. doi:10.1587/transinf.2015CYP0005

KEY TERMS AND DEFINITIONS

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.