ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

Malicious URL Detection Using Decision Tree-based Lexical Features Selection and Multilayer Perceptron Model

Warmn Faiq Ahmed, Noor Ghazi M. Jameel

Technical College of Informatics, Sulaimani Polytechnic University, Sulaimani 46001, Kurdistan Region, Iraq

ABSTRACT

Network information security risks multiply and become more dangerous. Hackers today generally target end-to-end technology and take advantage of human weaknesses. Furthermore, hackers take advantage of technology weaknesses by applying various methods to attack. Nowadays, one of the greatest dangers to the modern digital world is malicious URLs, and stopping them is one of the biggest challenges in the field of cyber security. Detecting harmful URLs using machine learning and deep learning algorithms have been the subject of various academic papers. However, time and accuracy are the two biggest challenges of these tools. This paper proposes a multilayer perceptron (MLP) model that utilizes two significant aspects to make it more practical, lightweight, and fast: Using only lexical features and a decision tree (DT) algorithm to select the best relevant subset of features. The effectiveness of the experimental outcomes is evaluated in terms of time, accuracy, and error reduction. The results show that a MLP model using 35 features could achieve an accuracy of 94.51% utilizing only URL lexical features. Furthermore, the model is improved in time after applying the DT as feature selection with a slight improvement in accuracy and loss.

Index Terms: Multilayer Perceptron, Lexical Feature, Feature Selection, Malicious URL, Synthetic Minority Oversampling Technique

1. INTRODUCTION

The internet expands at an unprecedented rate. Most of the time, malicious software is spread via the internet. Malicious websites can be referred to as any website that has been designed to cause harm. It is similar to a legitimate URL for regular users but hosts unsolicited content. The attacker usually builds a website identical to the target or embeds the exploit code of browser vulnerabilities on the webpage. Then, it tricks the victim into clicking on these links to obtain the victim's information or control the victim's computer [1]. In many circumstances, people do not check the complete website URL, and the attacker can obtain essential and personal information once they visit a malicious website [2].

Malicious URL detection always comes at the top in the research area. However, having protection against these attacks is not an option anymore. According to Google's Transparency Report, 2.195 million websites made their list of "Sites Deemed Dangerous by Safe Browsing" category as of January 17, 2021. The vast majority of those (over 2.1 million) were phishing sites. Only 27,000 of Google's removed websites were delisted because of malware [3]. Several forms of a malicious URL proceed with the attack and deliver unsolicited content, mainly named spam, phishing, and drive-by download. Spam is a web page with many links to unwanted websites for other purposes; the

Access this article online

DOI:10.21928/uhdjst.v6n2y2022.pp105-116

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: warmn.faiq.a@spu.edu.iq

Received: 20-08-2022 Accepted: 01-10-2022 Published: 13-11-2022

pages may pretend to provide assistance or facts about a subject. Phishing is a type of social engineering attack used to steal sensitive data. Finally, drive-by downloads refer to the unintentional download of malicious code to the device, leaving it open to a cyber-attack [4].

There are currently several approaches to detect dangerous websites on the internet. Nowadays, a malicious URL is mainly detected by black and white list-based and machine learning-based URL detection methods. According to the first technique, a website cannot be viewed until the URL is checked against the blacklist database to ensure it is not on the list. Blacklist is essentially a listing of URLs that were previously identified as malicious. Its advantage is that it is fast, easy, and has a meager false-positive (FP) rate. However, the main problem with this method is that it has a high false-negative (FN) rate and fails to detect newly generated URLs [1], [5], [6]. Nevertheless, it has been widely utilized in several major browsers, including Mozilla Firefox, Safari, and Chrome, among others, due to its simplicity and efficiency [5]. In addition, the blacklisting approach is also utilized by many antivirus systems and internet businesses. However, due to some limitations, the blacklisting strategy is insufficient to identify non-blacklisted threats [7]. Whitelist is another aspect that provides security when accessing a website. It is similar to the blacklist method technique. The difference is that in the whitelist, only those websites are allowed to access that is in the list. The limitation of this method is denying access to many newly generated websites that are legal and safe to visit [5]. On the other hand, machine learning techniques use a collection of URLs specified as a set of attributes and train a prediction model based on them to categorize a URL as good or bad, enabling them to recognize new, possibly harmful URLs [1].

In this paper, the multilayer perceptron (MLP) model is used to detect malicious URLs based on the features of the URLs. Since a lightweight method is challenging for time efficiency, lexical features are utilized and extracted from the dataset to train the model. The model is tested first without and then with feature selection (FS) to see the result and the differences. The main contribution of this paper is the development of a malicious URL detection system that utilizes only lexical features to construct a light model and selects only high-ranked features to reduce feature extraction (FE) time. Moreover, using decision tree (DT) as a FS algorithm is an advantage to select the best relevant features based on features importance score to improve the model performance and decrease the FE time during the detection process.

The paper is organized as follows. Section 2 is related works. The proposed malicious URL detection system with its phases including dataset collection, features extraction, features selection using DT algorithm, model development, and evaluation is presented in Section 3. All the experimental results and discussions are provided in Section 4. Finally, Section 5 illustrates the conclusion of the paper.

2. RELATED WORKS

Many kinds of research in the area of detecting malicious websites with various techniques, algorithms, and methods exist. The machine learning technique is one of the approaches used to solve the problem of malicious URL detection. Multiple studies have been done in the era. Xuan *et al.* proposed support vector machine (SVM) and random forest (RF) as machine learning algorithms to classify benign and malicious URLs by extracting features and behaviors of the URLs. The researchers created an extensive set of features to improve the model's ability and use it as a free tool to detect malicious URLs [8]. Subha *et al.* tested various machine learning algorithms to detect malicious URLs. According to the results, RF scored better than all SVM, Naïve Base, and artificial neural network (ANN) with an accuracy of 97.98 and the F1 score of 92.88 [9]. Furthermore, Islam *et al.* used three machine learning algorithms to detect malicious URLs: NN, K-nearest neighbor (KNN), DT, and RF. The results showed that the neural network (NN) scored the worst, whereas DT and RF achieved the best scores. The study mentioned that the lack of ability to detect malicious URLs by NN is due to the small size of the dataset, while NN is suitable for large datasets [10].

Besides, some of the researches used NNs as a solution for classifying malicious URLs from benign ones. Liu and Lee proposed a detection method using a convolutional neural network (CNN). The research adopted the end user's perspective and used CNN to learn and recognize screenshot images of the websites. The results showed that although the training period is lengthy, it is tolerable, especially with powerful graphics processing units. The testing is efficient once the training is completed; therefore, time is often not an issue with this procedure [11]. Balamurugan *et al.* proposed a NN to classify the websites as good and bad URLs with optimizing network parameters using genetic algorithms. The article showed a good improvement when optimizers were applied to the NN model in both classification and convergence [12]. Furthermore, Chen *et al.* used CNN for malicious URL detection. The study showed that the

proposed method achieved satisfying detection accuracy with an accuracy of 81.18% [13].

Moreover, hybrid systems are also proposed by some recent studies as a solution to the problem. Naresh *et al.* proposed a machine learning-based system that combines a SVM with logistic regression using a combination of URL lexical options, payload size, and python supply options as features to recognize the malicious URLs. As a result, an accuracy of 98% was achieved, which is an improvement compared to a conventional method. According to some recent articles, using NNs as a hybrid system can achieve satisfying performance [14]. Yang *et al.* proposed a system to detect malicious websites based on integrated CNNs and RF system. The results showed that the proposed integrated system achieved better results than traditional machine learning algorithms due to their shallow design, which cannot examine the complicated link between safe and malicious URLs [2]. Another research is by Das *et al.* who tested three NN algorithms, RNN, LSTM, and CNN-LSTM, to see the effectiveness of these algorithms in classifying benign and malicious URLs. The results showed that with an accuracy of 93.59%, the CNN-LSTM architecture exceeds the other two [15]. Furthermore, Peng *et al.* proposed attention-based CNN-LSTM for malicious URL detection. The results showed that the proposed method achieved better than shallow NNs and single deep NNs such as CNN and LSTM individuals with an accuracy of 96.74 [16].

3. THE PROPOSED MALICIOUS URL DETECTION SYSTEM

The proposed system is constructed using a lightweight method. Only lexical features are utilized to build the model. Python is used for programming the phases of the proposed system with famously fast and reliable libraries such as Pandas, Numpy, Scikit-learn, Imblearn, Pyplot, TensorFlow, and Keras.

The architecture of the proposed system starts with loading the dataset and then preprocessing stages to prepare the data for training. The training stage starts after the data are prepared. Then the testing stage; the trained model classifies whether the URL is malicious or benign. Finally, evaluation metrics are applied to compute the performance of the model. The system architecture is shown in Fig. 1.

3.1. Dataset Collection

In this work, a proposed model was trained and tested on a dataset conducted from malicious and benign websites that were utilized to create the suggested model and evaluate its predictions [17]. The dataset initially consisted of 420,464 URLs, 344,821 benign (good), and the rest of 75,643 websites are malicious (bad), as shown in Table 1. Therefore, the number of URLs in each class is imbalance, as shown in Fig. 2. A sample of the instances is shown in Fig. 3.

3.2. Data Preprocessing

3.2.1. Data cleaning

One of the most critical preprocessing stages in machine learning is data cleaning. Having clean, accurate noiseless data give precise models and results. Starting with cleaning the data, 9216 duplicated URLs were found and removed. The dataset was then checked for missing values, and there were no missing values in the dataset.

3.2.2. URL Lexical Feature Extraction

Several characteristics separate a safe URL and its webpage from a malicious URL. In certain instances, attackers employ direct IP linkages rather than domain names. Another tactic use by attackers is short names or abbreviations for websites unrelated to legitimate brand names. Algorithms for the detection method involve a wide variety of characteristics. To detect malicious websites using machine learning techniques, several distinct characteristics were retrieved from various academic research, such as lexical, host-based, and content-based features.

Since lexical features are fast to extract, they are also more applicable due to facing some casual problems when using content-based and host-based features. Most of the time, content-based features cannot be extracted from malicious URLs since most are blacklisted and cannot be accessed to get the contents such as HTML, JavaScript, and visual features. Besides, the security risks when accessing such websites need precautions such as using special sandbox services to reduce the risk. Host-based FE also faces problems such as a very long time taking due to the vast number of online requests from the database servers such as WHOIS that sometimes lead to another problem: Closing sockets for some of the websites and not getting the required information. In this study, lexical features are utilized to recognize malicious websites and distinguish them from legitimate ones. These characteristics are derived from the URL address's elements like a string. It should be able to identify malicious URLs because it bases its decision on how the URL appears. By replicating the names and making minor modifications, many attackers may make dangerous URLs seem normal. However, from the perspective of machine learning, it is not feasible to take the actual name of the URL. Instead, the URL's string must be handled to obtain valuable properties. Sixty lexical

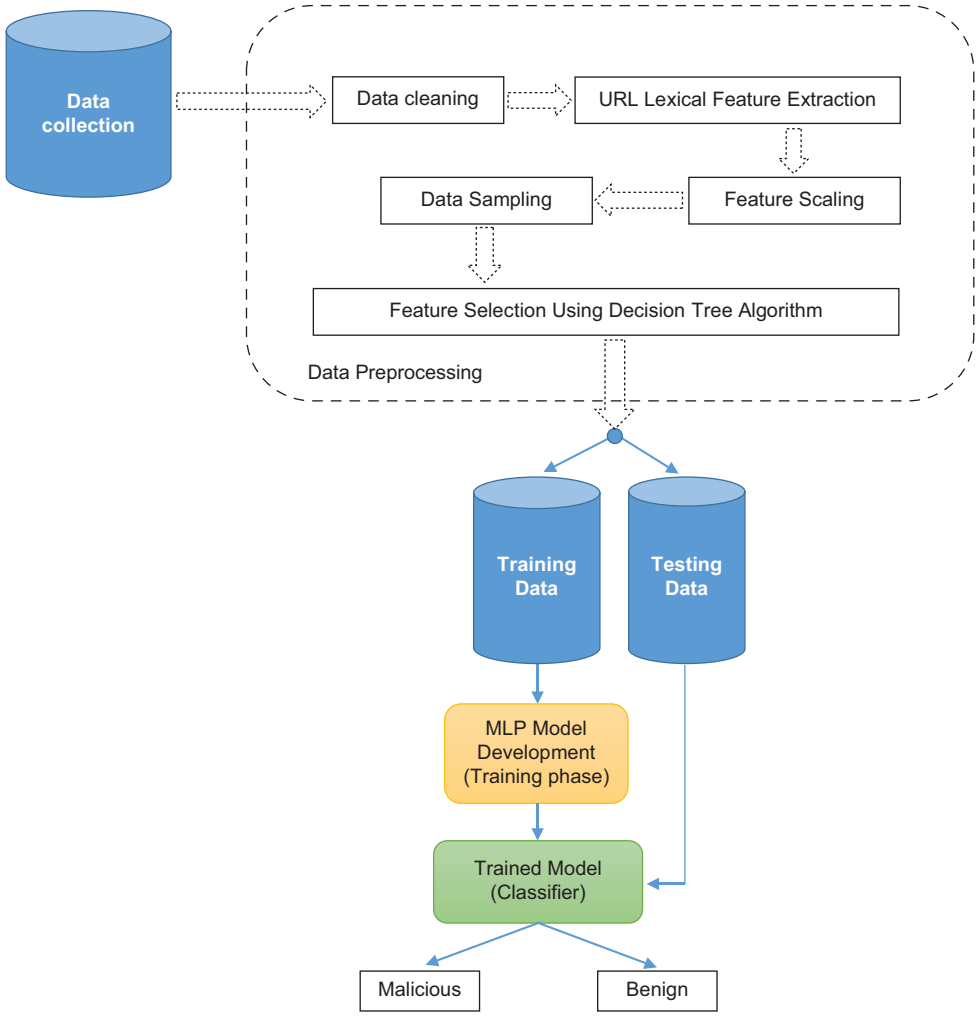

Fig. 1. The proposed system architecture.

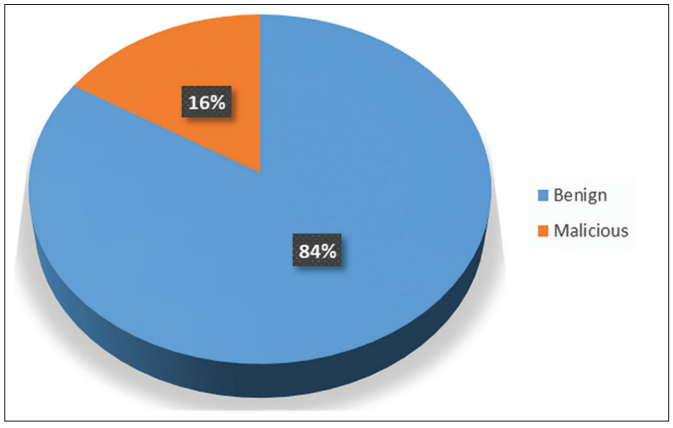

Fig. 2. Dataset class distribution.

features were collected from literature, then extracted from the web links as listed in Table 2.

TABLE 1: Dataset description

Type	No. of URLs
Benign	344,821
Malicious	75,643
Total URLs	420,464

3.2.4. Feature scaling

Feature scaling or normalization is often advised and sometimes crucial. Normalization is vital for NNs since unnormalized inputs to activation functions might cause trapping in a relatively flat domain region. Feature scaling helps optimize NN algorithms by accelerating training and preventing optimization from being trapped in local optima. Models of NNs establish a mapping between input and output variables. As a result, each variable's size and distribution of the data extracted from the domain may change. Input variables can have distinct scales because of

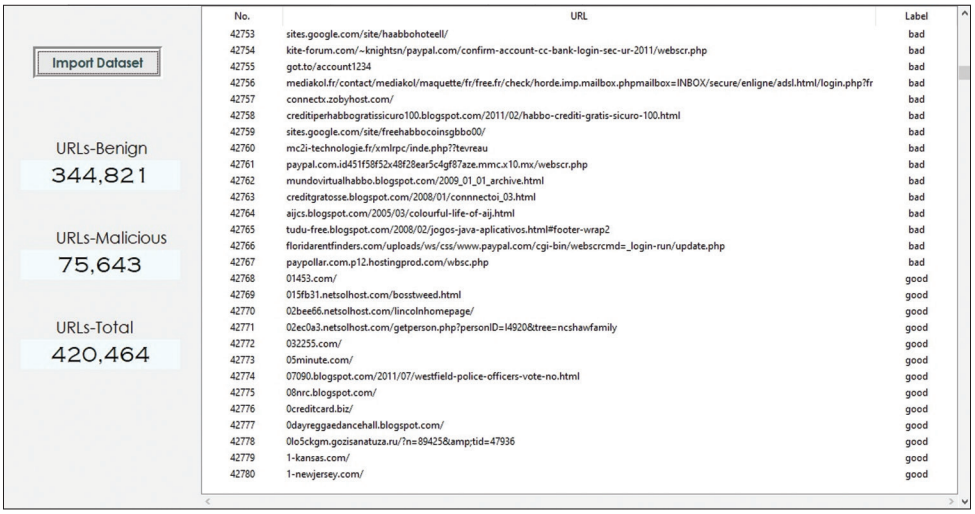

Fig. 3. Sample of the dataset instances.

TABLE 2: List of URL lexical features

Feature No.	Feature names	Data type	Description	References
f0	Count dots	Integer	Number of character “.” in URL	[7], [8], [18]-[21]
f1	url depth	Integer	The depth of the URL	[8]
f2	url length	Integer	The length of the URL	[7], [8], [14], [16], [18]-[20], [22]-[26]
f3	hyphen	Integer	Number of the dash character “-” (hyphen)	[8], [20], [22], [23]
f4	AT symbol	Boolean	There exists a character “@” in URL	[8], [22], [23], [27]
f5	Tide symbol	Boolean	There exists a character “~” in URL	[8]
f6	numUnderscore	Integer	Number of the underscore character	[8], [22]
f7	numPercent	Integer	Number of the character “%”	[8], [20]
f8	numAmpersand	Integer	Number of the character “&”	[8], [20], [22]
f9	numHash	Integer	Number of the character “#”	[8], [22]
f10	countQuestionMark	Integer	count the number of “?” in url	[20]
f11	countSemicolon	Integer	count the number of “;” in URL	[22]
f12	httpsInUrl	Boolean	Check if there exists a HTTPS in website URL	[8], [19], [22], [28]
f13	ipAddress	Boolean	Check if the IP address is used in the hostname of the website URL	[7], [8], [16], [22], [23], [25]
f14	urlRedirection	Boolean	There exists a slash “//” in the link path	[8], [19], [22], [23], [27]
f15	Count alpha	Integer	Number of the alphabetic character	[20], [22]
f16	Alpha ratio	Floating point	The proportion of alphabetic characters in the URL to the total length of the URL	[22]
f17	Count digit	Integer	Number of the numeric character	[8], [20], [22], [29]
f18	Digit ratio	Floating point	The proportion of numeric characters in the URL to the total length of the URL	[22]
f19	Count special chars	Integer	Number of any special characters like”,' %”,”$”,”,’ =”, etc.	[4], [7], [8], [14], [16], [18], [19], [22], [24]-[26]
f20	Special chars ratio	Floating point	The proportion of special characters in the URL to the total length of the URL	[16], [22]
f21	Count lowercase	Integer	The number of lowercase English letters in the URL	[16], [22]
f22	Lowercase ratio	Floating point	The proportion of lowercase English letters in the URL to the total length of the URL	[16], [22]
f23	Count uppercase	Integer	The number of uppercase English letters in the URL	[16], [22]
f24	Uppercase ratio	Floating point	The proportion of uppercase English letters in the URL to the total length of the URL	[16], [22]
f25	Count_subdomain	Integer	Number of subdomains in the URL	[8], [18]
f26	Short URL	Boolean	Using tiny url/short url service	[14], [23], [25]
f27	Length_of_ hostname	Integer	Length of hostname	[8], [19]

(Contd...)

TABLE 2: *(Continued)*

Feature No.	Feature names	Data type	Description	References
f28	Length_of_path	Integer	Length of the link path	[8], [19], [20]
f29	Length_of_query	Integer	Length of the query	[8], [20]
f30	Length_of_scheme	Integer	Length of the URL scheme	[20]
f31	Presence_sus_file_ext	Boolean	Checking the URL string for the presence of the following file extensions·exe,·scr,·vbs,·js,.xml, .docm,·xps, .iso, .img, doc, .rtf,·xls, pdf, .pub, .arj, .lzh, .r01, .r14, .r18, .r25, .tar, .ace, .zip, .jar, .bat, .cmd, .moz, .vb, .vbs, .js, .wsc, .wsh, .ps1, .ps1×ml, .ps2, .ps2×ml, .psc1 and .psc2.	[25]
f32	Count_ar_num	Integer	The number of Arabic numerals in the URL	[16]
f33	Is_tld_in_top5	Boolean	Whether the top-level domain is the top five domains (com, cn, net, org, cc)	[16]
f34	Paypal_in_path	Boolean	If “paypal” is contained in the PATH section.	[30]
f35	Ali_in_path	Boolean	If “ali” is contained in the PATH section.	[30]
f36	Jd_in_path	Boolean	If “jd” is contained in the PATH section.	[30]
f37	Safety_in_path	Boolean	If “safety” is contained in the PATH section.	[30]
f38	Verify_in_path	Boolean	If “verify” is contained in the PATH section.	[30]
f39	Google_in_path	Boolean	If “Google” is contained in the PATH section.	[30]
f40	Apple_in_path	Boolean	If “apple” is contained in the PATH section. if_facebook_u	[30]
f41	Facebook_in_path	Boolean	If “Facebook” is contained in the PATH section.	[30]
f42	Amazon_in_path	Boolean	If “amazon” is contained in the PATH section.	[30]
f43	Porn_in_path	Boolean	If “porn”-related words are contained in the PATH section.	[30]
f44	Gamble_in_path	Boolean	If “gamble” related words are contained in the PATH section.	[30]
f45	Paypal_in_domain	Boolean	If “paypal” is contained in the DOMAIN section.	[30]
f46	Ali_in_domain	Boolean	If “ali” is contained in the DOMAIN section.	[30]
f47	Jd_in_domain	Boolean	If “jd” is contained in the DOMAIN section.	[30]
f48	Safety_in_domain	Boolean	If “safety” is contained in the DOMAIN section.	[30]
f49	Verify_in_domain	Boolean	If “verify” is contained in the DOMAIN section.	[30]
f50	Google_in_domain	Boolean	If “Google” is contained in the DOMAIN section.	[30]
f51	Apple_in_domain	Boolean	If “apple” is contained in the DOMAIN section.	[30]
f52	Facebook_in_domain	Boolean	If “Facebook” is contained in the DOMAIN section.	[30]
f53	Amazon_in_domain	Boolean	If “amazon” is contained in the DOMAIN section.	[30]
f54	Porn_in_domain	Boolean	If “porn” related words are contained in the DOMAIN section.	[30]
f55	Gamble_in_domain	Boolean	If “gamble” related words are contained in the DOMAIN section.	[30]
f56	Has keyword “client”	Boolean	If the word “client” is contained in the URL	[31]
f57	Has keyword “admin”	Boolean	If the word “admin” is contained in the URL	[31]
f58	Has keyword “server”	Boolean	If the word “server” is contained in the URL	[31]
f59	Has keyword “login”	Boolean	If the word “login” is contained in the URL	[31]

their varied. The difficulty of the problem being modeled could be exacerbated by differences in the scales across the input variables. A model may learn tremendous weight values due to large input values, such as a spread of thousands of units, makes the result to be biased toward the bigger units. When features are of comparable size and nearly normally distributed, several machine learning methods work better or converge more quickly. Min-max algorithm is used to scale all the features between 0 and 1. Equation (1) uses for min-max feature scaling which helps the model to understand and learn better and faster without biasing to the more significant values [20].

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

Where, x_{max} and x_{min} are the maximum and the minimum values of the feature (x), respectively.

3.2.5. Data sampling

Initial examination of the dataset revealed that there were 5.18 times fewer occurrences of harmful websites than benign ones. Therefore, due to the stark disparity in the number of malicious and benign website instances, the model affect to be biased due to this significant class imbalance

as it learns from a far higher percentage of benign website occurrences.

A balanced class dataset is necessary for classification issues. As most machine learning algorithms used for classification were developed based on the presumption that there are an equal number of instances of each class, the imbalance of types in classification presents problems for predictive modeling. Therefore, a balanced classification dataset is also necessary for a classification model to produce accurate judgments.

There are several ways to handle an imbalanced dataset. The synthetic minority oversampling technique (SMOTE) was utilized to address this issue. The SMOTE technique uses KNN machine learning algorithm to produce new instances. Using it, additional instances of the minority class have been created, matching the proportion of instances of each class to the majority class to balance the classes. To balance the dataset, the minority class must thus be oversampled unless both groups have almost an equal number of cases. After balancing, the minority class were oversampled, which caused the data size to grow. Finally, the 344,800 occurrences of each class result in a balanced distribution, as shown in Fig. 4.

3.2.6. Feature Selection using DT Algorithm

The quality of FS and importance is one of the crucial differentiators in every machine learning task. Due to computational limitations and the need to remove noisy variables for more accurate prediction, FS becomes necessary when there is a large amount of data that the model may process.

In this study, a DT algorithm is used to select the best and most relevant lexical features based on the feature importance score. DTs apply various techniques to decide whether to divide a node into two or more sub-nodes. The homogeneity of newly formed sub-nodes is increased by sub-node formation. The threshold value of an attribute is used to divide the nodes in the DT into sub-nodes. The classification and regression tree algorithm uses the Gini index criteria to find the sub-nodes with the best homogeneity. The DT divides the nodes based on all factors that are accessible before choosing the split that produces the most homogenous sub-nodes. At the same time, the target variables are considered while selecting an algorithm. It is a visual depiction of every option for making a choice based on specific criteria according to the algorithm. Conditions on any characteristics are used to make judgments in both situations. The leaf nodes reflect the selection based on the conditions, whereas the inside nodes represent the conditions. Finding the attribute that provides the most information is necessary for DT construction. By building the tree in this way, feature importance scores can be accessed and used to help interpret the data, ranking, and select features that are most useful to a predictive model. It aids in determining which variable is chosen to be used in producing the decisive internal node at a specific point. The steps of FS using a DT are described in an (Algorithm 1). At this phase, the list of features with their importance values is calculated and selected by the DT algorithm.

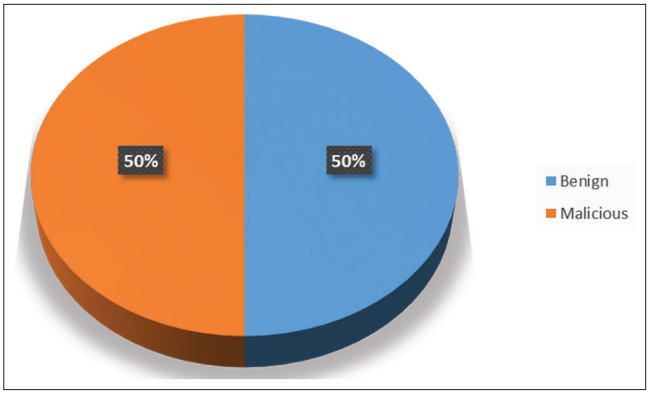

Fig. 4. Dataset after data sampling using SMOTE.

Begin
repeat
Step-1: Start at the Root node with all instances *S*
Step-2: Select an attribute based on splitting criteria
minGini ← 0
splitTree ← ∅
for all attributes a in S **do**
$giniIndex \leftarrow 1 - \sum_a P_a^2$
if giniIndex < minGini **then**
minGini ← giniIndex
splitA ← a
end if
end for
Step-3: Partition instances according to selected attribute recursively
CART(attributes-a, instances, target attribute)
until all instances processed
Step-4: There are no remaining attributes for further partitioning
End

Algorithm 1. Classification and regression tree [32].

3.3. MLP Model

The most practical variety of NNs is MLP which is frequently used to refer to the area of ANNs. A perceptron is a single-neuron model that serves as the basis for more extensive NNs. Artificial neurons are the basic units of NNs. The feed-

TABLE 3: The parameters of the proposed MLP model

Layer no.	No. of neurons/dim	Optimizer	Activation function	Learning rate	Batch size	No. of epochs
Layer 1	400	Adam	Sigmoid	0.005	200	1500
Layer 2	300		Sigmoid			
Layer 3	200		Sigmoid			

forward NN is supplemented by the MLP. There are three layers: The input layer, the output layer, and the hidden layer.

The proposed MLP model consists of three hidden layers besides the input and output layers to describe the model. The first hidden layer has 400 neurons, the second hidden layer has 300 neurons, and the last hidden layer has 200 neurons. The output layer has one neuron as it is a binary classification with two outputs, 1 and 0, whereas 1 represents a malicious URL and 0 represents a benign one. The other parameters are set as a batch size of 200, a learning rate of 0.005, a sigmoid function as an activation function, and Adam as an optimizer, as shown in Table 3.

3.4. Model Evaluation

The goal is not just to create a predictive model. It involves building and choosing a model that performs well on out-of-sample data. Therefore, verifying the model's correctness is essential before computing estimated values. To assess the models, many indicators are considered. A crucial phase in the machine learning pipeline is evaluating the learned model's effectiveness. Machine learning models are either adaptable or non-adaptive based on how effectively they generalize to new input. When an ML model is applied to new data without being adequately evaluated using a variety of metrics and without relying on accuracy, it may produce inaccurate predictions. Besides, the accuracy, precision, recall, and F1 score have been taken into account for the model reliability and considering the aspect of the errors when the model classifies between malicious and benign URLs. The definition of classification accuracy, which may be the most straightforward criterion to use and apply, is the ratio of correct predictions to all other predictions and calculated using Equation (2) [33].

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \quad (2)$$

Confusion matrix produces a matrix that summarizes the overall effectiveness of the model. For example, the confusion matrix for binary classification, which is the case in this work, is a two-by-two matrix. The confusion matrix shows the number of correct and incorrect classification for both actual and predicted values, including true positive indicates the

TABLE 4: Confusion matrix

Actual values	Predicted values	
	Negative	Positive
Negative	TN	FP
Positive	FN	TP

TP: True positive, TN: True negative, FP: False positive, FN: False negative

TABLE 5: List of used hardware and software specifications

Hardware and software specification	Description
PC	Core i3 gen6
RAM	20 GB
Storage	SSD SATA 256 GB
Operation system	Windows 10 pro

number of samples that are correctly classified as positive and true negative shows the number of instances that are correctly identified as negative, besides, there is FP that indicates the number of samples that are incorrectly identified as positive, and finally, FN that indicates the number of instances that are incorrectly identified as negative. The confusion matrix for binary classification is shown in Table 4.

From the confusion matrix, some important metrics are calculated and taken into consideration along with the accuracy to ensure that the model performs well and is not biased because of issues such as dataset imbalance. Therefore, precision, recall, and F1 score are used as model evaluation metrics. Precision indicates how accurate the positive predictions are, recall is the coverage of actual positive samples, and the F1 score is the harmonic mean of precision and recall, and they are calculated using Equations (3), (4), and (5), respectively [22], [29], [34].

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (4)$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5)$$

TABLE 6: List of features with their importance score

Feature No.	Feature importance	Feature No.	Feature importance	Feature No.	Feature importance	Feature No.	Feature importance
f0	0.11828	f16	0.05732	f32	0	f48	0
f1	0.07532	f17	0.04211	f33	0.07169	f49	0
f2	0.03691	f18	0.04414	f34	0.00132	f50	0
f3	0.01727	f19	0.01206	f35	0.00158	f51	0
f4	0.00161	f20	0.13231	f36	0.00022	f52	0
f5	0.00185	f21	0.02187	f37	0.00009	f53	0
f6	0.01472	f22	0.02058	f38	0.00041	f54	0
f7	0.00227	f23	0.00755	f39	0.00241	f55	0
f8	0.0018	f24	0.01264	f40	0.00031	f56	0.00053
f9	0.00009	f25	0.02609	f41	0.00228	f57	0.01168
f10	0.00874	f26	0.01038	f42	0.00009	f58	0.00089
f11	0.00997	f27	0.1204	f43	0.00017	f59	0.02466
f12	0.00017	f28	0.05412	f44	0		
f13	0.00007	f29	0.00539	f45	0		
f14	0.00058	f30	0.00068	f46	0		
f15	0.01788	f31	0.0065	f47	0		

4. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, the details of the experimental results are presented. The experiments are implemented on a malicious URL dataset [19] aiming to find the set of relevant URL lexical features based on their importance score using DT algorithm and evaluating the MLP model performance using the selected features. The final prepared dataset after the main steps of data preprocessing which includes data cleaning, data sampling, and FE, consists of a total of 689,600 URLs with 60 lexical features and a class label that has a 0 for benign and 1 for malicious. The software and hardware specifications used for the experiments are explained in Table 5.

After running the DT algorithm for FS, the importance score or weight for each variable was calculated. Features with lowest importance scores were deleted and features with highest scores were kept. This type of FS can simplify the problem that is being modeled, speed up the modeling process, and improve the performance of the model. The list of all lexical features' importance scores is illustrated in Table 6. After this phase, 35 features were selected and 25 features were eliminated. The selected features are the top 35 features with highest importance values which are f0, f1, f2, f3, f4, f5, f6, f7, f8, f10, f11, f15, f16, f17, f18, f19, f20, f21, f22, f23, f24, f25, f26, f27, f28, f29, f31, f33, f34, f35, f39, f41, f57, f58, and f59.

As a result of eliminating 25 features, a significant decrease in FE time achieved, which is an essential factor in this problem situation, as shown in Table 7 and Fig. 5.

TABLE 7: Feature extraction time before and after feature selection

No. of features	Feature extraction time in seconds
60 features, the whole dataset (Before FS)	134 s
35 features, whole dataset (After FS)	92 s

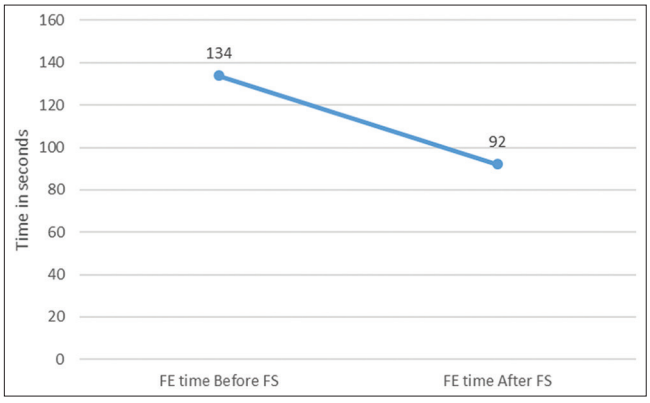

Fig. 5. FE time differences before and after FS.

For MLP model evaluation, the 35 selected features were fed to the model as input. The stratified technique was used for splitting the dataset into train and test sets to preserve the same proportions of instances in each class as in the original dataset. It is obvious that most of the data in the dataset are advised to be used for training to let the model learn well. Different ratios for training and testing have been used by the researchers such as 80% for training and the other 20% for testing or 70% for training by 30% for testing. Many factors are taken into consideration when train test split is done, such as the number of instances in the dataset, hyperparameters

TABLE 8: List of tested scenarios

Scenario	No. of epochs	No. of features	Batch size	Learning rate	No. of neurons in hidden layers
s1	100	35	200	0.005	200, 120, 80
s2	100	35	200	0.005	400, 200, 100
s3	100	35	200	0.005	400, 300, 200
s4	100	35	200	0.005	600, 400, 200
s5	100	35	200	0.005	800, 600, 400
s6	500	35	200	0.005	400, 300, 200
s7	500	35	200	0.005	600, 400, 200
s8	500	35	200	0.005	800, 600, 400
s9	1000	35	200	0.005	400, 300, 200
s10	1500	35	200	0.005	400, 300, 200

TABLE 9: Results of all the 10 scenarios

Scenarios	Train time in seconds	Test time in seconds	Train loss	Train accuracy (%)	Test accuracy (%)	Precision	Recall	F- score	Confusion matrix
s1	933.4	15.0	0.145	93.90	92.82	0.923	0.935	0.929	([95321 8119] [6735 96705])
s2	2258.5	28.7	0.142	94.00	92.95	0.919	0.943	0.930	([94797 8643] [5938 97502])
s3	2553.3	17.8	0.123	94.79	93.45	0.927	0.944	0.935	([95733 7707] [5840 97600])
s4	2847.4	23.3	0.122	94.86	93.51	0.927	0.944	0.936	([95807 7633] [5798 97642])
s5	6984.2	31.8	0.125	94.74	93.51	0.935	0.936	0.935	([96659 6781] [6636 96804])
s6	10487.2	18.3	0.091	96.21	94.18	0.937	0.948	0.942	([96822 6618] [5415 98025])
s7	17460.9	25.3	0.098	96.00	94.08	0.939	0.943	0.941	([97118 6322] [5918 97522])
s8	27800.3	37.7	0.095	96.09	94.15	0.937	0.946	0.942	([96877 6563] [5546 97894])
s9	22684.7	19.7	0.086	96.49	94.25	0.938	0.947	0.943	([97010 6430] [5460 97980])
s10	62791.6	30.3	0.075	96.93	94.51	0.941	0.950	0.945	([97233 6207] [5146 98294])

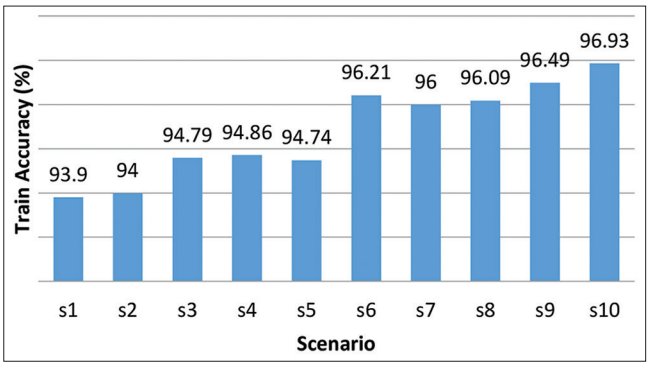

Fig. 6. Train accuracy for the 10 different scenarios.

to tune, the used classifier, and the model use case. Due to the good amount of instances in the dataset, 70% of the final dataset considered for training, while the remaining 30% is used for testing. The model with several scenarios has been tested using a learning rate of 0.005, batch size of 200, and different number of epochs and neurons. The list of scenarios is described in Table 8.

After executing all the 10 scenarios described in Table 8, from the results shown in Table 9, it is obvious that with increasing the number of epochs, the accuracy will increase along with training time, and the training loss will decrease eventually. In this system, the more important parameters for detecting malicious URLs are higher values for test accuracy, precision, and recall with lower training loss. The least important parameter is the training time. Training phase is a one-time process, sometimes it requires a long time to develop a well-trained model with high accuracy and less training loss. Since the last scenario, 1500 epochs outperformed the best scores for the mentioned parameters, it has been chosen to train the model and used for malicious URL detection. As a result,

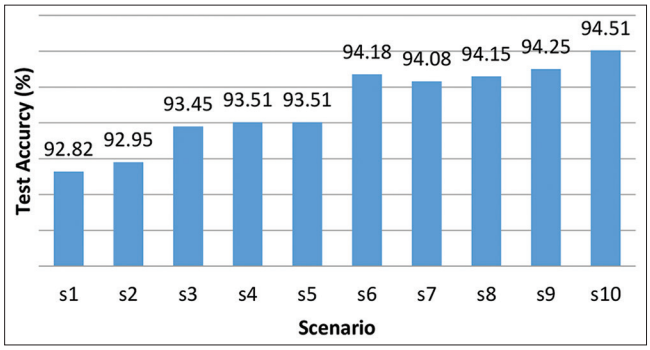

Fig. 7. Test accuracy for the 10 different scenarios.

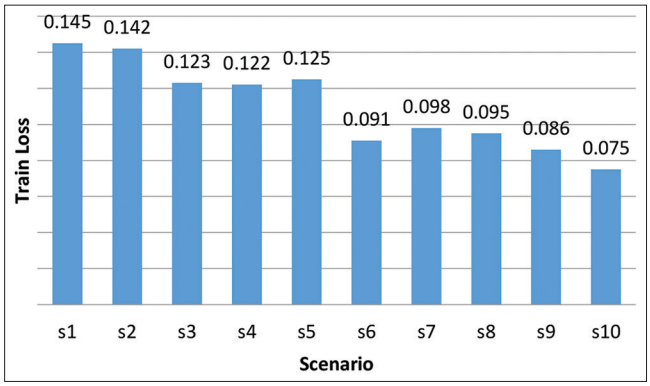

Fig. 8. Train loss for the 10 different scenarios.

the model achieved an accuracy of 94.51, recall of 94.1, the precision of 95.0, and training loss of 0.075. The results are shown in Table 9 and illustrated in Figs. 6-8.

5. CONCLUSION

One of the serious threats on the internet is malicious URL. Hackers have several techniques and algorithms to obfuscate URLs to bypass the defenses. The problem of detecting malicious URLs has been studied in this research with explaining types of possible attacks, features, and detection techniques. The study developed a lightweight malicious URL detection model using URL lexical features only instead of content or host-based features. Content and host-based features take a long time during the extraction. To extract content-based features, the websites should be available for accessing their source code. Host-based features extraction process needs connection with special servers such as WHOIS to get the required information. DT has been used to get the importance scores of all lexical features to select the best features to build a malicious URL detection system with better performance and efficiency. The study shows that using only relevant lexical features, which is more practical to apply, is enough to create a robust lightweight detection model using MLP algorithm. Experiment results have been shown and discussed to explain the differences before and after applying each technique.

REFERENCES

[1] J. Yuan, G. Chen, S. Tian and X. Pei. "Malicious URL detection based on a parallel neural joint model," *IEEE Access*, vol. 9, pp. 9464-9472, 2021.

[2] R. Yang, K. Zheng, B. Wu, C. Wu and X. Wang. "Phishing website detection based on deep convolutional neural network and random forest ensemble learning," *Sensors*, vol. 21, no. 24, pp, 8281, 2021.

[3] S. Cook. "Malware Statistics in 2022: Frequency, Impact, Cost and More," 2022. Available from: https://www.comparitech.com/antivirus/malware-statistics-facts [Last accessed on 2022 Aug 18].

[4] S. Kumi, C. Lim and S. G. Lee. "Malicious url detection based on associative classification." *Entropy*, vol. 23, no. 2, pp. 1-12, 2021.

[5] W. Bo, Z. B. Fang, L. X. Wei, Z. F. Cheng and Z. X. Hua. "Malicious URLs detection based on a novel optimization algorithm." *IEICE Transactions on Information and Systems*, vol. E104.D, no. 4, pp. 513-516, 2021.

[6] Z. Chen, Y. Liu, C. Chen, M. Lu and X. Zhang. "Malicious URL detection based on improved multilayer recurrent convolutional neural network model." *Security and Communication Networks*, vol. 2021, pp. 9994127, 2021.

[7] S. M. Nair. "Detecting malicious URL using machine learning: A Survey." *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, no. 5, pp. 2670-2677, 2020.

[8] C. Do Xuan, H. Dinh Nguyen and T. Victor Nikolaevich. "Malicious URL Detection Based on Machine Learning." *International Journal of Advanced Computer Science and Applications*, vol. 11, pp. 148-153, 2020.

[9] V. Subha, M. S. Pretha and R. Manimegalai. " Malicious Url Classification Using Data Mining Techniques." *Journal of Analysis and Computation (JAC)*, pp. 148-153, 2018.

[10] M. Maminur Islam, S. Poudyal and K. Datta Gupta. "Map reduce implementation for malicious websites classification." *International Journal of Network Security and its Applications*, vol. 11, no. 5, pp. 27-35, 2019.

[11] D. Liu and J. H. Lee. "Cnn based malicious website detection by invalidating multiple web spams." *IEEE Access*, vol. 8, pp. 97258-97266, 2020.

[12] P. Balamurugan, T. Amudha, J. Satheeshkumar and M. Somam. "Optimizing neural network parameters for effective classification of benign and malicious websites." *Journal of Physics Conference Series*, vol. 1998, no. 1, 2021.

[13] Y. Chen, Y. Zhou, Q. Dong and Q. Li. "A Malicious URL detection method based on CNN." In: *2020 IEEE Conference on Telecommunications, Optics and Computer Science, TOCS 2020*. IEEE, Piscataway, 2020, pp. 23-28.

[14] N. Khan, R. Naresh, A. Gupta and S. Giri. "Ayon gupta and sanghamitra Giri, malicious URL detection system using combined SVM and logistic regression model." *International Journal of Advanced Research in Science, Engineering and Technology*, vol. 11, no. 4, pp. 63-73, 2020.

[15] A. Das, A. Das, A. Datta, S. Si and S. Barman. "Deep approaches on malicious URL classification." In: *2020 11th International*

Conference on Computer Networks and Communication Technologies. ICCCNT 2020, IEEE, Piscataway, 2020.

[16] Y. Peng, S. Tian, L. Yu, Y. Lv and R. Wang. "Malicious URL recognition and detection using attention-based CNN-LSTM." *KSII Transactions on Internet and Information Systems*, vol. 13, no. 11, pp. 5580-5593, 2019.

[17] Adamyong. "GitHub-Adamyong-zbf/URL_Detection: Data Set." 2020. Available from: https://github.com/adamyong-zbf/URL_detection [Last accessed on 2022 Aug 18].

[18] L. M. Camarinha-Matos, N. Farhadi, F. Lopes and H. Pereira, Editors., *Technological Innovation for Life Improvement*, Vol. 577. Springer International Publishing, Cham, 2020.

[19] S. Singhal, U. Chawla and R. Shorey. "Machine learning concept drift based approach for malicious website detection." In: *2020 International Conference on Communication Systems Networks*, *COMSNETS 2020*, IEEE, Piscataway, pp. 582-585, 2020.

[20] Maheshwari S, B. Janet and R. J. A. Kumar. "Malicious URL Detection: A Comparative Study." In: *Proceedings International Conference on Artificial Intelligence and Smart Systems*, *ICAIS 2021*. IEEE, Piscataway, pp. 1147-1151, 2021.

[21] Y. Peng, S. Tian, L. Yu, Y. Lv and R. Wang. "A Joint Approach to Detect Malicious URL Based on Attention Mechanism." *International Journal of Computational Intelligence and Applications*, vol. 18, no. 3, 2019.

[22] A. S. Raja, R. Vinodini and A. Kavitha. "Lexical features based malicious URL detection using machine learning techniques." *Materials Today Proceedings*, vol. 47, pp. 163-166, 2021.

[23] S. D. Vara Prasad and K. R. Rao. "A Novel Framework For Malicious URL Detection Using Hybrid Model." *Turkish Journal of Computer and Mathematics Education*, vol. 12, pp. 2542, 2021.

[24] S. Ahmad and A. Tamimi, "Detecting Malicious Websites Using Machine Learning," M.S. thesis, Department of Graduate Programs & Research, Rochester Institute of Technology, RIT Dubai, April. 2020. [Online]. Available from: https://scholarworks.rit.edu/theses

[25] T. Manyumwa, P. F. Chapita, H. Wu and S. Ji. "Towards Fighting Cybercrime: Malicious URL Attack Type Detection using Multiclass Classification." In: *Proceedings 2020 IEEE International Conference on Big Data, Big Data 2020*, IEEE, Piscataway, pp. 1813-1822, 2020.

[26] F. Alkhudair, M. Alassaf, R. Ullah Khan and S. Alfarraj. "*Detecting Malicious URL*." IEEE, Piscataway, 2020.

[27] R. R. Rout, G. Lingam and D. V. L. Somayajulu. "Detection of malicious social bots using learning automata with url features in twitter network." *IEEE Transactions on Computational Social Systems*, vol. 7, no. 4, pp. 1004-1018, 2020.

[28] Y. C. Chen, Y. W. Ma and J. L. Chen. "Intelligent malicious url detection with feature analysis." In: *Proceedings Second IEEE Symposium on Computer and Communications*. Vol. 2020. IEEE, Piscataway, 2020.

[29] S. He, J. Xin, H. Peng and E. Zhang. "Research on malicious URL detection based on feature contribution tendency." In: *2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics*, *ICCCBDA 2021*, pp. 576-581, 2021.

[30] T. Li, G. Kou and Y. Peng. "Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods." *Information Systems*, vol. 91, pp. 101494, 2020

[31] R. Ikwu. In: R. E. Ikwu, editor. "*Extracting Feature Vectors From URL Strings For Malicious URL Detection*." Towards Data Science," Canada, 2021. Available from: https://towardsdatascience.com/extracting-feature-vectors-from-url-strings-for-malicious-url-detection-cbafc24737a [Last accessed on 2022 Aug 16].

[32] G. S. Kori and D. M. S. Kakkasageri. "*Classification and Regression Tree (Cart) Based Resource Allocation Scheme for Wireless Sensor Networks*." Social Science Research Network, Rochester, NY, 2022.

[33] N. Hosseini, F. Fakhar, B. Kiani and S. Eslami. "Enhancing the security of patients' portals and websites by detecting malicious web crawlers using machine learning techniques." *International Journal of Medical Informatics*, vol. 132, pp. 103976, 2019.

[34] M. Chatterjee and A. S. Namin. "Deep Reinforcement Learning for Detecting Malicious Websites." *Computer Science*, vol. 15. pp. 55, 2019.