Information Sciences Letters
*An International Journal*

http://dx.doi.org/10.18576/isl/130312

# A Malicious Website Detection Approach Using Random Forest and Pearson Correlation-based Feature Selection Method

*Abdu H. Gumaei*[1,*], *Amr F. Shawish*[2] *and Ahmed Emam*[3]

[1]Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj, 11942, Kingdom of Saudi Arabia
[2]Department of Computer Science, Faculty of Computers and Information, Damanhur University, Beheira, Egypt
[3]Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Menoufia, Egypt

Received: 9 Feb. 2024, Revised: 18 Mar. 2024, Accepted: 02 Apr. 2024.
Published online: 1 May 2024

**Abstract:** With the advancement of the Internet of Things (IoT), smart cities have evolved from traditional urbanization to contemporary urbanization of technology. IoT networks enable scattered smart devices to gather and analyze data through an open channel known as the Internet. As a result, security, privacy, centralization, scalability, and transparency in which smart cities may be developed. Detecting malicious Uniform Resource Locators (URLs) in an IoT context is critical for protecting the network and devices from security risks. Malicious URLs identification is an essential aspect of cybersecurity for interconnected devices of smart cities, employing various techniques and technologies to identify potentially harmful links. Recently, a number of machine learning methods has been used in several studies to classify URLs into malicious or benign classes based on their statistical characteristics and features. However, selecting the most significant features in the preprocessing stage plays a key role in improving the detection accuracy of trained machine learning classifiers. In this paper, we propose an effective approach for detecting malicious websites with a particular emphasis on attack payloads and broader feature space. The importance of URLs' features is first obtained and ranked using a random forest-based comprehensive analysis. Then, Pearson's correlation analysis is used to select the most important features that have a strong correlation with the class labels. The proposed approach is evaluated using four machine learning algorithms: *k*-nearest neighbor (*k*-NN), random forest (RF), support vector machine (SVM), and logistic regression (LR). The experimental results show the efficiency of our approach, achieving 96% accuracy using an RF algorithm.

**Keywords:** Smart cities, Internet of Things (IoT), malicious website detection, feature selection, machine learning.

## 1 Introduction

With the advancement of the Internet of Things (IoT), smart cities have evolved from traditional urbanization. Malicious website is one that tries to download a malware onto a device, which is a catch-all word for anything that may obstruct computer operation, collect personal data, or, in the worst case, take complete control of the system [1,2]. Malicious website detection is an important aspect of cybersecurity for protecting interconnected devices of smart cities, employing various techniques and technologies to identify potentially harmful links. In the instance of a drive by download, however, the websites may attempt to install software on a computer without first requesting permission; normally, this necessitates some action on the user side [3]. Malicious attackers infiltrate systems and use them to install malware [4,5], gain access and privileges, compromise private or sensitive information, sabotage systems, or engage in other attacks like DDoS [6].

Preventing the devices and information systems from being compromised is almost difficult and sometimes impossible. Attackers actually employed a variety of methods to be successful with the incursions, such as drive-by downloads from websites that took advantage of browser flaws [7] or network-accessible flaws [8]. Additionally, social engineering attacks similar to phishing and malicious attachments in emails allow for the user-authorized installation of destructive software [9]. The purpose of malicious programmers is to gain access to a machine and interfere with its functionality. There are several different methods that malicious software can get onto a computer. Now more than ever, it is critical

*Corresponding author e-mail: a.gumaei@psau.edu.sa to recognize these URLs since they have the capacity to corrupt systems or store passwords secretly. Machine learning techniques are being utilized to automate the classification of URLs. Overall performance is improved, and human participation is reduced. For both identifying websites and preventing cyberattacks, a variety of strategies have been put forth and implemented [10-12]. One of the most common tactics is to keep a central blacklist of dubious and malicious domains, IP addresses, and websites [13].

Confidentiality, integrity, and availability are the three components of information security [14]. For Internet systems, Internet gadgets, Internet of Things services, and information security is necessary. In attacks involving DDoS [15], buffer overflows, drive by downloads, phishing websites, SQL injection, and Internet system services are most frequently utilized. The assault using drive-by downloads is shown in Figure 1. When a user visits malicious URLs, one of the website's programmers searches for security holes in the user's system before attempting to launch an attack. The terminal device immediately downloads and runs a virus or a malware program if the attack is successful. The user's system or device is now a component of the hacker's botnet. The primary issue addressed in this study is to distinguish between dangerous and benign URLs and to give an analyst 21 types of features based on Alexa-based, domain-based, and complication technique-based information from the Internet.

The creation of malicious URLs detection systems and the provision of 22 features, including applications layer and network layer features, constitute the contribution to this work. We can identify which traits are more significant than others based on a feature analysis. This can attest to the usefulness of the functionality we suggested for this assignment. Then, accuracy and precision performance are improved accordingly.

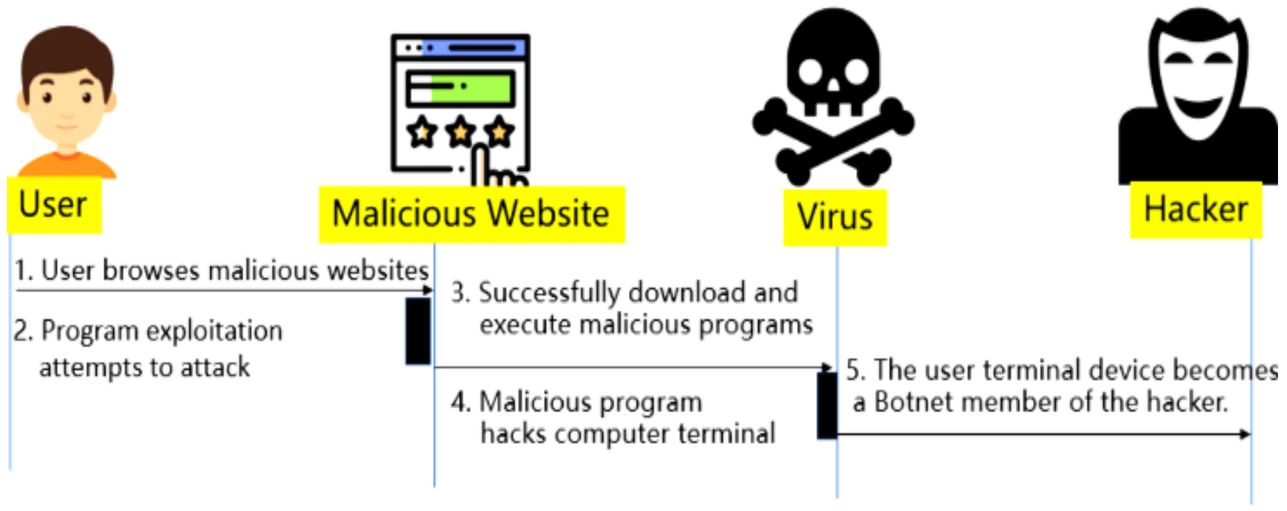

User
Malicious Website
Virus
Hacker
1. User browses malicious websites
2. Program exploitation attempts to attack
3. Successfully download and execute malicious programs
4. Malicious program hacks computer terminal
5. The user terminal device becomes a Botnet member of the hacker.

**Fig. 1:** The type of drive-by downloads flow attack.

The remainder of the paper is organized as follows. Section 2 provides a detailed literature review of malicious website related work. Section 3 explains the proposed framework for classifying website URLs. This includes an explanation on how dataset is collected for training and testing, and how the best features are selected, as well as, how classifiers are built and trained to detect malicious website URLs. Section 4 presents the experimental results. Finally, Section 5 gives the conclusion and future work.

## 2 Related Work

In order to get the greatest outcomes and fit into the situation at hand, numerous academics have researched a variety of ways for malicious website identification [16]. As a background for the study included in the remaining sections of the paper, we give an overview of related work. Given a website's Uniform Resource Locator (URL) as input, Vyawhare et al. [17] created a method for categorizing malicious and benign websites. Malicious URL categorization models are tested for accuracy and stability using the XGBoost (eXtreme Gradient Boosting) algorithm. With great classification performance and efficiency, the XGBoost classified the malicious URLs using just nine features. With a focus on a broad of attack-payloads, feature space, and flexibility of the methods that should be adaptive with changes in characteristics of the malicious web pages, and the usability of all real-world systems for defending the users against the malicious attacks. Eshete and Weldemariam [18] presented a comprehensive approach for detecting and analyzing malicious websites. An innovative method for identifying and thwarting attacker efforts to get around malicious website detection algorithms was put out by Singhal et al. [19].

Results demonstrate that the suggested approach is practical for spotting concept deviations. A machine learning framework for intelligently detecting malicious URLs was put forth by Chen et al. [12]. From domain, obfuscation, and Alexa data procedures, 41 features of malicious URLs are retrieved. The 17 most crucial traits are determined using the XGBoost algorithm and the ANOVA (Analysis of Variance) test. The XGBoost classifier is learned on the collected dataset and has achieved a detection accuracy of more than 99%. The authors in [20] presented a novel approach to analyze holistically the characteristics necessary for identifying fraudulent websites using machine learning. Singh and Goyal compared machine learning attributes to identify fraudulent websites.

A total of 25 criteria that are typically used to identify fraudulent websites were taken into consideration. The computational resources needed for extraction and processing, as well as the classification accuracy in identifying harmful websites, were examined in relation to these features. The top 5 attributes for identifying fraudulent websites were determined based on the analysis.

By examining and comparing honey clients with low and high levels of engagement, Zhang and Qassrawi [21] addressed honey clients. In order to compare the two varieties of honey, they established a comparison feature. Additionally, they discussed ways malicious websites could avoid detection and fingerprint honey clients and offered suggestions for getting around these evasion methods. They developed criteria to specify and gauge the efficiency of honey clients by looking at honey client features.

First, they assessed a combination of features that had not previously been examined in the previous works, principally involving embedding obtained using a type of neural networks-based transformer, according to the two methodologies provided in [22]. The second part of the contributions is a brand-new dataset for this issue called GAWAIN that was created in a way that gives other researchers access to all of our data collection and processing methods as well. The investigation of unexpected finding is the absence of characteristics linked to content (such as HTML and JavaScript) from the top-10 list. When contrasting the results of predictions made by models trained on widely used features.

A model for Feed Forward Neural Network-based Malicious Website URL Detection was presented by Emmah, Victor Thomas, and Ukorma [23]. Object-Oriented Analysis and Design is used to design the methodology employed in their research. The model employs a dataset of 98,012 website URLs that include 48,006 genuine instances and 48,006 malicious instances. They trained a feed-forward neural network and achieved a competitive accuracy result. For the purpose of identifying malicious URLs, Alazab and Soman [24] tested a number of cutting-edge character-level embedding techniques based on deep learning.

A DeepURLDetect (DURLD) is a character-level embedding model used to encode raw URLs in order to capitalize on and transform the performance advantage. Comparing DURLD to other related deep learning-based character level embedding approaches shows that it is computationally less expensive and can detect harmful URL variants. Table 1 provides a summary of the prior work methodologies.

While preprocessing is a vital step in developing a website malicious detection system, there are several issues that need to be considered to ensure the effectiveness of the preprocessing step. Some of the problems of preprocessing for websites' malicious detection are:

- Data imbalance: The distribution of benign and malicious websites in the dataset may not be balanced, with a much larger number of benign websites than malicious ones. This can lead to bias in the model, as the model may better detect benign websites than malicious ones. Some techniques, such as under-sampling, over-sampling, or synthetic data generation, can help address data imbalance.

- Data quality: The quality of the data can affect the accuracy of the detection model. Poor quality data, such as incomplete or inconsistent data, can negatively impact the model's performance. Therefore, data cleaning and normalization can help improve the data quality.

- Feature selection: The selection of features for analysis can also impact the accuracy of the detection model. Feature selection or dimensionality reduction can help to classify the most important features and reduce the data dimensionality.

- Overfitting: It occurs if the model fits the training set and achieves a poor performance on a test set. The preprocessing techniques, such as regularization or cross-validation, can reduce overfitting problem and enhance the model's generalization.

- Time sensitivity: Some malicious websites are designed to operate for a short period of time and may be taken down quickly. Feature extraction must be conducted in real-time or near-real-time to ensure the data is up-to-date and accurate.

**Table 1:** A summary of the previous work to detect malicious websites.

| Author | Datasets | Algorithm | Techniques |
|---|---|---|---|
| Chaitanya [17] | Benign URL's Website [25] | XGBoost | Machine learning Classification |
| Birhanu Eshete [18] | HoneyMonkey [26] | A holistic approach combines page identity, URLs tokens, page contents, and execution trace | Feature extraction |
| Singha [19] | PhishTank [27] | Gradient Boosted Trees, Random Forest, Feed Forward Neural Networks | Machine learning Classification |
| Chen [12] | Alexa [28] urlquery.net [29], urlscan.io Rao et al. [28] and GitHub [30] | ANOVA, XGBoost | Machine learning Classification |
| Singh & Goyal [20] | MalCrawler [30] | C4.5 and Naive Bayes. | Machine learning Classification |
| Chaiban et al. [22] | Singh et al. [31] | XGBoost | Machine learning Classification |
| Emmah et al. [23] | Emmah et al. [23] | Neural Network | Deep learning classification |
| Srinivasan et al. [24] | Two types of datasets are utilized, derived from Singh research [30] and [31] | Malicious URL Detection Engine | Deep learning-based character level embedding techniques |

In summary, preprocessing for website malicious detection can be challenging due to issues such as data imbalance, data quality, feature selection, overfitting, and time sensitivity. It is important to carefully consider these issues when developing a preprocessing pipeline to ensure the effectiveness of the detection model. There are several methodologies that can be used to preprocess websites to detect malicious activity. Here are some possible steps:

1. *Collecting a large data set of website's code, content, and behavior. This can include static analysis of the HTML, CSS, and* JavaScript files, as well as dynamic analysis of the website's behavior when accessed with a browser.
2. Processing and cleaning the collected data of the previous step. This can include the values of features such as the presence of certain keywords or phrases associated with malicious activity, the use of obfuscation techniques in the website's code, or abnormal patterns of user behavior. It also normalizes the values of the collected dataset.
3. Extracting the important features by removing any irrelevant or noisy features and ensuring that the remaining data is formatted in a consistent manner.
4. Selecting the most informative features for detecting malicious activity. This can be done using various techniques, such as correlation analysis or machine learning algorithms.
5. Training a number of machine learning models on the selected features to classify websites as either benign or malicious. This can include techniques such as decision trees, support vector machines, or deep learning algorithms.
6. Evaluating the performance of models using some metrics such as recall, precision, and F1-score. This can be done using a test dat*aset of known benign and malicious websites.*

Overall, the methodology of malicious detection depends on the specific requirements of the task and the available data. However, the steps outlined above can provide a useful framework for building an effective detection system.

## 3 Proposed Framework

To improve the limitations of current related work, a framework is proposed to detect malicious websites effectively. A malicious website is defined as a URL that is used to upload a file, download a file, install an application, collect data, display an advertisement, open a pop window, or any mixture of the above operations without any knowledge or consent of the user. The framework utilizes a dataset of websites' URLs with their labels, which are either malicious or benign, based on trusted website directories for training the machine learning models: SVM, RF, *k*-NN, and LR. The components of the proposed framework are illustrated in Figure 2.

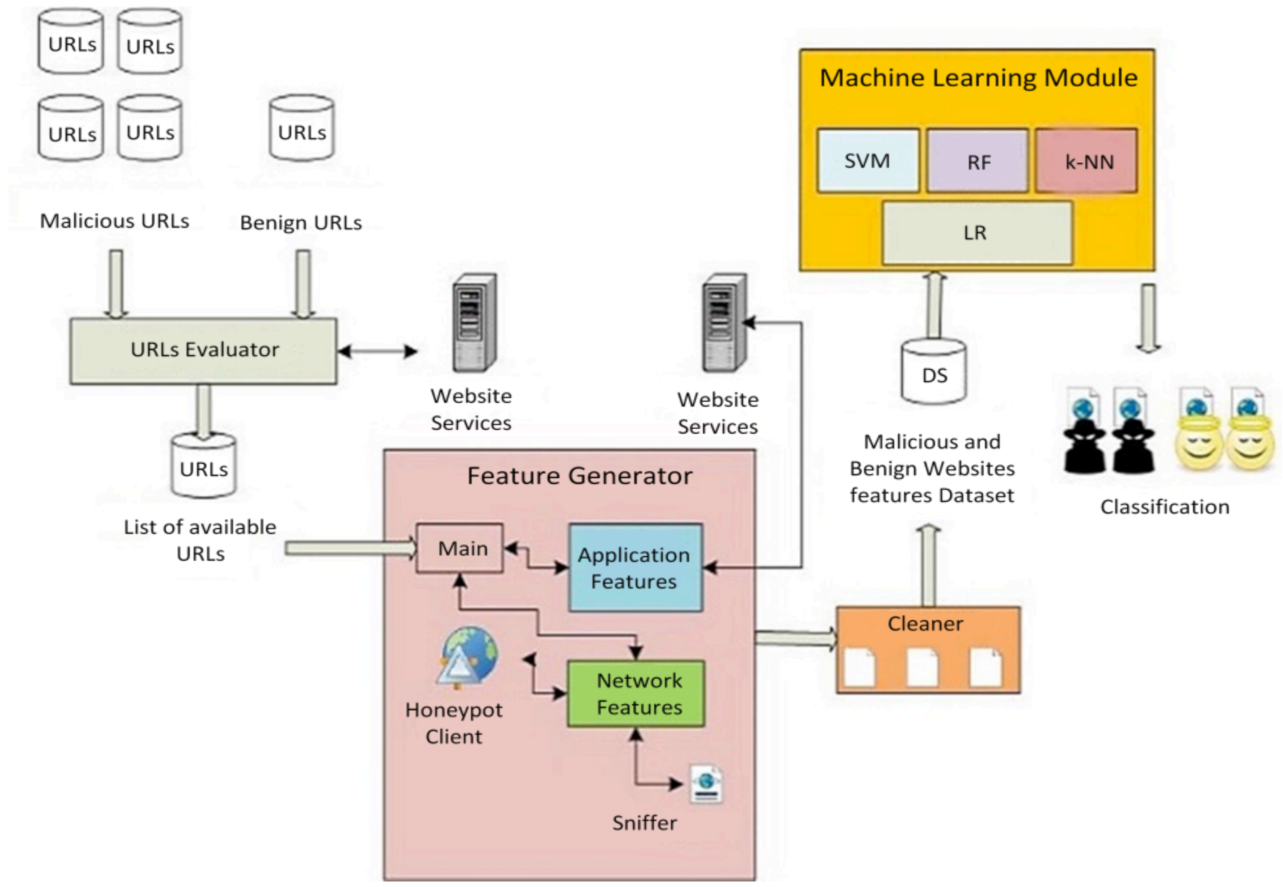


**Fig. 2:** Proposed framework to detect malicious websites.

It contains four components: URLs evaluator, feature generator, cleaner, and machine learning module. The URLs evaluator is designed to assess the legitimacy, safety, or potential risks associated with a given URL, known as a web address. It determines whether a website is malicious, or benign for users. The output of URLs evaluator is a list of available URLs. This output will be an input to the feature generator component that generates a set of features related to URLs for network and application layer protocols. It considers the specific characteristics and contexts in which these URLs are encountered. These features can be extracted from network packets, HTTP traffic logs, web server logs, or through dedicated URL analysis tools. They provide valuable insights into the characteristics and behaviors of URLs at both the network and application layers, facilitating the detection of malicious activities and ensuring network and application security. Cleaner component is designed to preprocess the URLs before they are subjected to be prepared as a dataset for training the models of machine learning module, or even for classification in the deployment phase. It removes noise, irrelevant information, or obfuscation techniques from URLs to enhance the effectiveness and efficiency of subsequent steps.

The next subsections describe the data collection and feature importance process along with the feature selection and classification tasks. Moreover, we explain how the dataset is collected and divided for training and testing. It also explains how the machine learning classifiers work. Finally, a description of the evaluation matrices is provided.

### *3.1 Data Collection*

The dataset used for this study includes both malicious and benign URLs. The benign websites' URLs are the general URLs for organizations in the Internet network service systems. The malicious URLs are gathered from the malware domain list and other open-source datasets [32], getting a total of 185,181 websites' URLs. On the other hand, the benign links are taken from KAGGLE [33], which acquired 345,000 URLs. Check the status of each web page in the URL dataset. For this, a tool was developed in Python, and with the library urllib2, it will be verified whether each URL is active or inactive. A result, a total of 35,279 malicious links were obtained active (19%), and a sample was selected from a random number of benign URLs to which they have applied the process, and as a result, a size list of 27,912.

The URLs mainly consist of four parts: top-level domain, hostname, paths, and port of the web address, as shown in Figure 3.

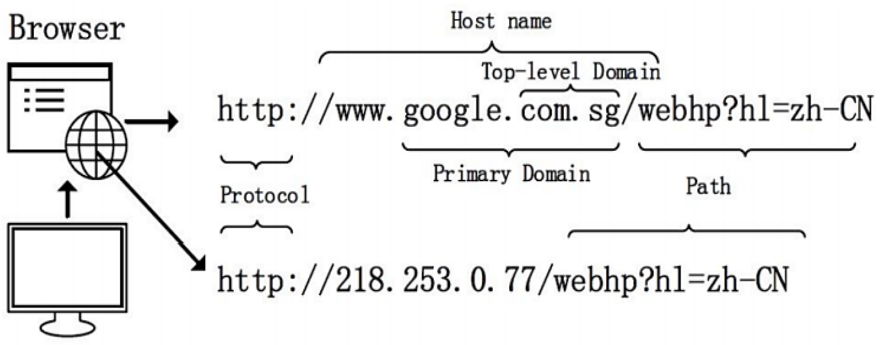

Browser
Host name
Top-level Domain
http://www.google.com.sg/webhp?hl=zh-CN
Primary Domain
Path
Protocol
http://218.253.0.77/webhp?hl=zh-CN

**Fig. 3:** URL structure.

### *3.2 Data Collection*

The feature generation step follows the same characteristics studied by Xu [3]. We analyze the data generated from a set of active URLs to obtain the capacities of a honeypot type low interaction client using a Thug tool. A Thug is a public Python tool released in March 2012 and used to mimic the behaviors of a web browser. Each URL is run on Thug for 4 seconds with its normal setup, concurrent with our process script that captures web traffic using PyShark. Finally, the method produced 1,743 recordings of benign network traffic and 756 records of malicious network data.

*Application Layer Features:* The application layer features extracted from collected data are given as follows:

(A1): Content length represents the size of the total characters in the URL.

(A2): Number of special characters is the total of special characters that appear on the URL (for example: '?', '%', '#', '&', ',').

(A3): HTTP Header content length represents to the size of the header content HTTP.

(A4): HTTP Header server provides information about the web page server, including its name, type, and version.

(A5): HTTP Header charset indicates the code - cation of each web page (for example, ANSI, ISO-8859-1, UTF8).

(A6): Whois registration date indicates the date on which the website server was registered.

(A7): Who is updated date is the last date when the server was updated.

(A8): Whois country specifies the country where the website server.

(A9): WHOIS state Pro represents the location on where the website was registered.

(A10): Whose Domain indicates the domain of the website.

*Network Layer Features:* The features of the network layer, which are extracted from the collected data, are given as follows:

(R1): TCP conversation exchange sums the number of packets between the honeypot and website by TCP protocol.

(R2): Destination remote TCP port is the total number of ports rather than those exposed in TCP.

(R3): Remote IPS represents the number of addresses IPs connected to the honeypot.

(R4): PKT without DNS is an array of all non-DNS packets.

(R5): TCP URG packets represent the number of TCP packets with the URG flag.

(R6): Source app packets are the number of packets sent by the honeypot to the server remote.

(R7): Remote app packets are the variable of the volume in bytes of communication between the web servers and the honeypot.

(R8): Duration is the duration time of the website.

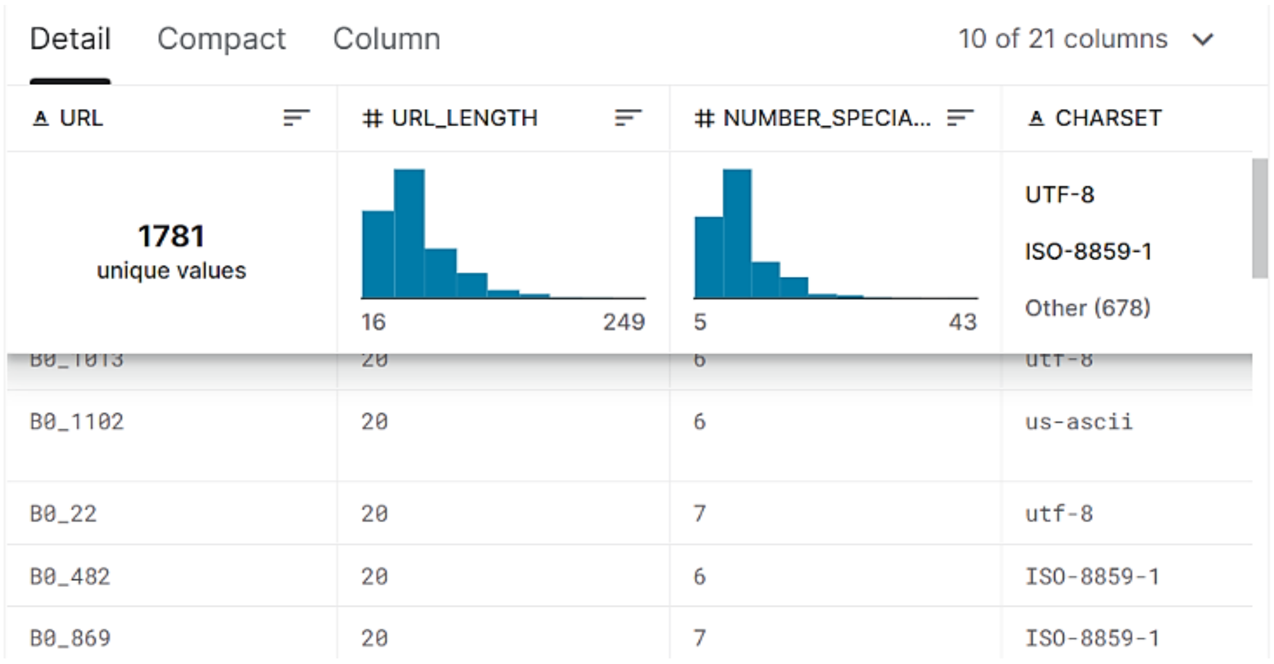
(R9): AVG. local PKT rate is the average of local IP packets per second (packets sent over duration) sent from the crawler to the web server.

(R10): AVG. The remote PKT rate is the average number of remote IP packets sent per second from the remote server to the crawler.

(R11): Application packets are the total number of IP packets generated in the query of the URL, in which the DNS ones are included.

(R12): DNS query times list of DNS layers.

Once the mentioned characteristics have been obtained, a normalization rule is applied to the numerical data in order to represent them in a range between 0 and 1. Additionally, the data categories are symbolized as binary, apart from raw data, which are consolidated, as shown in Figure 4.



| URL | URL_LENGTH | NUMBER_SPECIA... | CHARSET |
|---|---|---|---|
| B0_1013 | 20 | 6 | utf-8 |
| B0_1102 | 20 | 6 | us-ascii |
| B0_22 | 20 | 7 | utf-8 |
| B0_482 | 20 | 6 | ISO-8859-1 |
| B0_869 | 20 | 7 | ISO-8859-1 |

**Fig. 4:** Row data features.

### *3.3 Machine Learning Classifiers*

Various machine learning (ML) classification methods, including *k*-nearest neighbor (*k*-NN), support vector machine (SVM), logistic regression (LR), and random forest (RF) are built using a training part of a collected dataset to classify the malicious website in the test set. The following subsections briefly explain these classification algorithms.

*K-Nearest Neighbor (k*-NN*)*: It is a common machine learning method that may be used for both classification and regression. The *k*-NN approach is non-parametric, which means it makes no assumptions about the data's underlying distribution.

The basic idea of the *k*-NN algorithm is to predict the class or value of a new data point by looking at the *k* nearest data points in the training set and taking a majority vote or averaging their values. The choice of k is a hyper-parameter that can be tuned to achieve better data performance.

For classification, the *k*-NN algorithm assigns the most common class label among the *k*-NN of a test point. For regression, the *k*-NN algorithm computes the mean or median of the *k*-nearest neighbors' target values and uses this as the predicted value for the test point.

The *k*-NN algorithm has some advantages, such as being easy to understand and implement and having the ability to handle multi-class problems. However, it can also have some disadvantages, such as being sensitive to irrelevant features and being computationally expensive for large datasets. The *k*-NN is a machine learning algorithm used for classification and regression problems. The algorithm is based on the idea that similar data points are likely to have similar class labels or output values.

The mathematical representation of the *k*-NN algorithm can be described as follows:

Given a dataset $x = \{x_1, x_2, x_3, \dots, x_n\}$ and the corresponding class labels or output values $y = \{y_1, y_2, y_3, \dots, y_n\}$.

1- The *k*-NN algorithm finds the *k* nearest neighbors' values of a new data point $x$ based on some distance metric (e.g., Euclidean distance, Manhattan distance, etc.).

2- For classification problems, the class label of the new data point is predicted based on the class labels of the *k*-nearest neighbors. The majority class label among the *k* nearest neighbors is assigned as the predicted class label.

3- For regression problems, the output value of the new data point is predicted based on the output values of the *k*-nearest neighbors. The average or median of the output values among the *k*-nearest neighbors is assigned as the predicted output value.

Overall, *k*-NN is a simple and effective algorithm that works well in low-dimensional spaces and is widely used in various applications, including recommender systems, image recognition, and anomaly detection. Estimating the number of neighbors, *k*, is discovered experimentally. It starts with the classifier's error rate at *k*=1 and then repeats the process with each additional *k* to check a different neighbor. The number of training trails typically rises as a *k* value increases, and the best *k* value is chosen based on the lowest error rate. Because distance-based comparisons are used, which give each attribute equal weight, the main drawback of the *k*-NN classifier is its low accuracy, particularly with noise and irrelevant attributes [34].

*Support Vector Machine (SVM)*: The SVM is a learning method used for classification and regression problems. In classification, SVM tries to find the hyperplane that best separates the two classes, while regression tries to fit a curve that best fits the data. The mathematical representation of SVM can be described as follows:

Given a dataset $x = \{x_1, x_2, x_3, \dots, x_n\}$ and corresponding binary labels $y = \{y_1, y_2, y_3, \dots, y_n\}$, where $y_i$ is either -1 or 1.

The SVM can find the hyperplane $w * x + b = 0$ that maximizes the margin between the two classes. Here, $w$ is the weight vector, and $b$ is the bias term.

The SVM can also be represented mathematically as an objective function, written in the following equation:

$$Minimize\ \left(\frac{1}{2} * \left|\left|w\right|\right|^2 subject\ to\ yi(w * x + b) >= 1\right) for\ all\ i \quad (1)$$

Here, the objective is to minimize the magnitude of the weight vector while ensuring that all data points are correctly classified by the hyperplane. The inequality constraint ensures that the hyper-plane separates the two classes with a margin of at least one. The margin is well-defined as the perpendicular distance between the hyperplane and the closest data point. The optimization problem can be solved using quadratic programming (QP), which involves solving a set of linear equations to find the optimal values of w and b. The QP formulation can be written as follows:

$$Minimize\ \left(\frac{1}{2} * x^T * P * x + q^T * x\right) \quad (2)$$

Subject to

$$Gx <= h,\ Ax = b \quad (3)$$

Here $x = (w, b)$, $P$ is a matrix, $q$ is a vector, $G$ is a matrix, $h$ is a vector, $A$ is a matrix, and $b$ is a vector. The optimization problem is subject to constraints that ensure that the solution is feasible.

Once the optimal values of $w$ and $b$ are obtained, the decision function for classifying a new data point $x$ can be defined as follows:

$$f(x) = sign(w * x + b) \quad (4)$$

Here, $f(x)$ returns -1 or 1 depending on which side of the hyper-plane the point $x$ falls on. If the value is less than 0, the data point is classified as the negative class, and if the value is greater than 0, it is classified as the positive class. SVM is a powerful algorithm that works well in high dimensional spaces and is widely applied in various applications, including bioinformatics, image classification, and natural language processing.

The SVM algorithm employs a nonlinear mapping technique to raise the dimension of the original training data before attempting to locate the linearly ideal separation hyper-plane using margins and support vectors. SVM can accurately simulate complex nonlinear decision boundaries despite being incredibly sluggish. SVM labels classes with one of two values, either +1 or -1, when there are two linearly separable classes.

The optimal line with the least amount of classification error must be found, but if there are an infinite number of separating lines that might be drawn. In other words, the optimum hyper-plane must be identified if we generalize to n dimensions. This is done by looking for the hyperplane with the highest margin, known as the maximum marginal hyperplane [34].

*Logistic Regression (LR)*: It is a machine learning algorithm used for classification problems where the goal is to predict binary class labels (e.g., 0 or 1). It models the probability of the positive class as a function of the input variables using a logistic function, which maps any real-valued input to a probability between 0 and 1.

The mathematical representation of the Logistic Regression algorithm can be described as follows:

Given a dataset $x = \{x_1, x_2, x_3, \dots, x_n\}$; and corresponding binary class labels $y = \{y_1, y_2, y_3, \dots, y_n\}$.

LR can model the probability of the positive class as follows:

$$p(y = 1|x) = \frac{1}{(1 + exp(-z))} \tag{5}$$

Here, z is a linear function of the input variables $x$, i.e.

$$z = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_d x_d \tag{6}$$

Where $w_0, w_1, w_2, w_d$ are the model parameters or weights that are learned from the training data, and $d$ is the number of input variables.

The LR uses Maximum Likelihood Estimation (MLE) to learn the model parameters that maximize the likelihood of the observed data. The likelihood function $L(w)$ is defined as follows:

$$L(w) = prod\{p(y_i = 1|x_i)\hat{}y_i * (1 - p(y_i = 1|x_i))\hat{}(1 - y_i)\} \tag{7}$$

Here, $y_i$ is the binary class label of the i$th$ data point, and $x_i$ is its corresponding input variables. The product is taken over all $n$ data points in the training set.

The log-likelihood function is easier to optimize than the likelihood function, and it is defined as follows:

$$L(w) = sum\{y_i * log(p(y_i = 1|x_i)) + (1 - y_i) * log(1 - p(y_i = 1|x_i))\} \tag{8}$$

Here, the sum is taken over all $n$ data points in the training set. The LR uses Gradient Descent or other optimization methods to find the values of $w$ that maximize the log-likelihood function $l(w)$.

LR is a simple and effective algorithm that works well in many applications, including medical diagnosis, fraud detection, and credit scoring. It can be extended to handle multiclass classification problems using one-vs-all or softmax methods.

Statistical and data mining researchers regularly use the LR algorithm, one of the most important methodologies, to examine and classify binary and proportional response datasets. One of its important benefits is that LR can be applied to multi-class classification problems and calculate probability automatically. Another benefit is that most techniques employed in LR model analysis adhere to the same fundamentals as those in linear regression. Furthermore, LR is amenable to most unconstrained optimization techniques [35]

The logistic function can be given by the following formulas:

$$Logit\ (p_i) = 1/\ (1 + exp\ (-p_i)) \tag{9}$$

$$ln(p_i/(1 - p_i)) = Beta_0 + Beta_1 * x_1 + \dots + B_k * K_k \tag{10}$$

In Eq. (9), $logit(p_i)$ is a response or a dependent variable, and the variable $x$ is an independent variable. The model's coefficient parameter, or beta parameter, is usually projected via a maximum likelihood estimation (MLE).

*Random Forest Algorithm (RF)*: Random Forest is a machine learning technique that operates on an ensemble basis. It combines numerous decision trees to increase accuracy while minimizing overfitting. The algorithm consists of two primary steps: training and prediction. During the training process, a portion of the training data is chosen at random, followed by a set of features. The selected characteristics and data subset are then used to create a decision tree. The previous stages are then repeated for a certain number of times (or until a stopping requirement is reached). Finally, the decision trees are kept or stored. In the prediction stage, the new data points are routed through each decision tree in the forest, and the class of each decision tree is projected using the tree's decision rule. The forecasts from all of the trees are then added together. Following that, categorization may be done with a majority vote, and regression can be done with an average of the projected values. Mathematically, the random forest algorithm can be represented in the following algorithms:

**Algorithm 1.** Training algorithm of RF.

```
1. Given a training dataset $x = \{x_1, x_2, x_3, \dots, x_n\}$ with corresponding labels $y = \{y_1, y_2, y_3, \dots, y_n\}$.
2. for t = 1 to T (where T is the number of decision trees):
    2.1. Randomly select a subset of the training data $D_t$ of size $n_t$, where $n_t <= n$.
    2.2. Randomly select a subset of the features $F_t$ of size $m_t$, where $m_t <= m$ (where $m$ is the total
         number of features).
    2.3. Construct a decision tree by $D_t$ and $F_t$.
3.  Store the decision trees as a forest, $F = \{T_1, T_2, \dots, T_t\}$
```

**Algorithm 2.** Prediction algorithm of RF.

```
1.  Given a new data point $x$.
2.  for t = 1 to T:
    2.1. Predict the class label of $x$ using the decision tree $T_t$.
    2.2. Aggregate the predictions from all the trees to get the final prediction. For classification, this can
         be done by majority vote. For regression, this can be done by taking the average of the predicted
         values.
3.  Return the class label of $x$.
```

### *3.4 Evaluation Metrics*

To evaluate the performance of a model that classifies websites as either benign or malicious, we can use the following measures:

- Accuracy (Acc.): The accuracy measures the percentage of correct predictions made by the model out of all predictions in terms of true positive (TP), true negative (TN), false negative (FN), and false positive (FP). It is calculated as:

$$\text{Accuracy (Acc.)} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{11}$$

- Precision (Pre.): It measures the proportion of true positives out of all positive predictions. Precision is calculated as:

$$\text{Precision (Pre.)} = \frac{\text{TP}}{(\text{TP+FP})} \tag{12}$$

- Recall (Rec.): It measures the proportion of true positives out of all actual positives. Recall is calculated as:

$$\text{Recall (Rec.)} = \frac{\text{TP}}{(\text{TP+FN})} \tag{13}$$

- F1 Score: It is the harmonic mean of precision and recall and is a way to balance these two metrics. F1 Score is calculated as:

$$\text{F1-score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision+Recall})} \tag{14}$$

We would use a test dataset of known benign and malicious websites to evaluate the model using these measures. We first use the model to predict each website's class (benign or malicious) in the test dataset and then compare these predictions to the actual class labels. We then compute the accuracy, recall, precision, and F1-score of the model based on these predictions and actual labels.

A good model would generally have high accuracy, high precision, high recall, and high F1- score, indicating that it correctly identifies most malicious websites while minimizing false positives. However, the importance of each measure would depend on the specific use case and the consequences of false positives or negatives

## 4 Experimental Results of Proposed Approach

This section gives the experimental results of each step in the proposed approach. First, an explanation of the results of the data preprocessing step on the collected dataset will be introduced, including the results of feature selection and extraction. Then, the classification results of adopted ML methods will be described with a performance comparison between them in terms of accuracy, precision, recall, and F1-score metrics.

### *4.1 Results of Data Preprocessing and Feature Importance*

The data preprocessing involves preparing the dataset to be suitable for machine learning models and also selecting the significant features for improving their accuracy. Feature analysis is the main step in our proposed framework. The data features with string values are transformed to integer values by using the one-hot encoding of the data, in which the URLs are split into different parts, and patterns of each part are encoded, creating columns for each unique data point of the string data. After preparing the data features, the random forest method is used to get the rank of important features. Table 2 shows the importance of the features and their ranking, as obtained by the comprehensive analysis of the random forest.

### *4.2 Results of Feature Selection*

Feature extraction is the second step, which is important for effectively building the machine learning models, optimizing their accuracy, and reducing the computational time. In this step, we have employed Pearson's correlation analysis to select the features strongly correlated with the URLs dataset class label feature. Figure 5 presents the correlation between features, as shown as a heat map. The heat map provides a visual tool for understanding the relationships between the features in the dataset. It can interpret the patterns and identify potential insights, such as identifying groups of features that move together, indicating the dependencies or relationships of features, and detecting multicollinearity issues, where multiple features are highly correlated with each other, which can affect the stability and interpretability of models. There are some features, such as, 'URL_LENGTH', 'APP_BYTES', 'TCP_CONVERSATION_EXCHANGE', 'REMOTE_APP_BYTES', 'SOURCE_APP_PACKETS', 'REMOTE_APP_PACKETS', 'SOURCE_APP_BYTES', are highly correlated. Figure 6 shows the scatter and density plot selected features. The scatter and density plot are a commonly used visualization in data analysis and is particularly useful for understanding the distribution and relationship between features in the dataset.

**Table 2:** Features importance

| No. | Feature Name | Score |
|---|---|---|
| 1 | WHOIS_STATEPRO | 0.1274 |
| 2 | WHOIS_COUNTRY | 0.1010 |
| 3 | SERVER | 0.0876 |
| 4 | WHOIS_REGDATE | 0.0732 |
| 5 | DIST_REMOTE_TCP_PORT | 0.0673 |
| 6 | SOURCE_APP_BYTES | 0.0578 |
| 7 | REMOTE_APP_PACKETS | 0.0569 |
| 8 | WHOIS_UPDATED_DATE | 0.0482 |
| 9 | NUMBER_SPECIAL_CHARACTERS | 0.0413 |
| 10 | URL_LENGTH | 0.0400 |
| 11 | REMOTE_IPS | 0.0378 |
| 12 | CONTENT_LENGTH | 0.0368 |
| 13 | CHARSET | 0.0362 |
| 14 | APP_BYTES | 0.0357 |
| 15 | REMOTE_APP_BYTES | 0.0356 |
| 16 | SOURCE_APP_PACKETS | 0.0322 |
| 17 | APP_PACKETS | 0.0320 |
| 18 | DNS_QUERY_TIMES | 0.0265 |
| 19 | TCP_CONVERSATION_EXCHANGE | 0.0264 |

In Figure 6 we visualize the distribution and relationship between each pair of six selected features regarding the class labels, colored with orange and blue colors. For example, we can see an overlap between the distribution of REMOTE_IPS and URL_LENGTH, which makes the classification between malicious and benign URLs difficult.

### *4.3 Results of Machine Learning Methods*

Through the use of the original collected data sample as both training and test sets for building the ML models, a 10-fold cross-validation is used to evaluate the results of adopted models and select the best model as the recommended model for websites malicious detection. In the evaluation process of the 10-fold cross-validation technique, the dataset set is divided into 10 parts, using one distinct part as the test set and the other nine parts as the training set, as depicted in Figure 7.

This step individually applies four typical machine learning algorithms (*k*-NN, LR, RF, and SVM), and their performance is compared to select the best model in the domain of malicious detection. The comparison results of implemented ML models based on a 10-fold cross-validation technique is given in Table 3. Using accuracy as the main standard, it can be found that the RF algorithm performs better than the others.

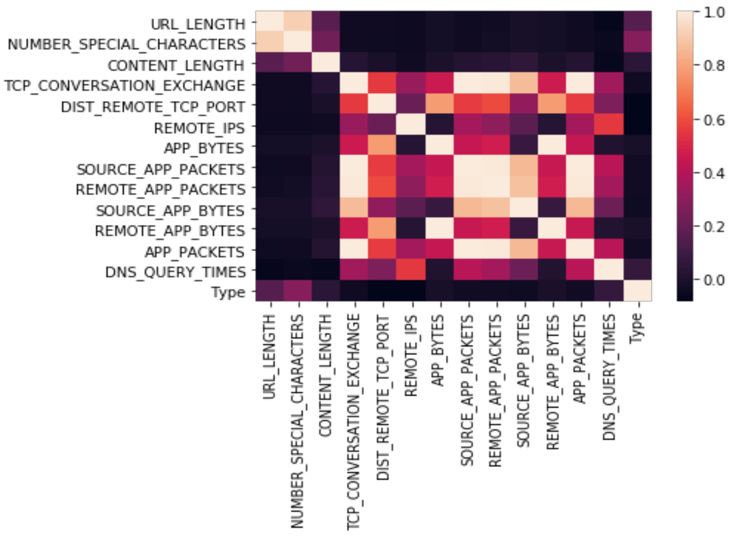


**Fig. 5:** Heat map correlation feature

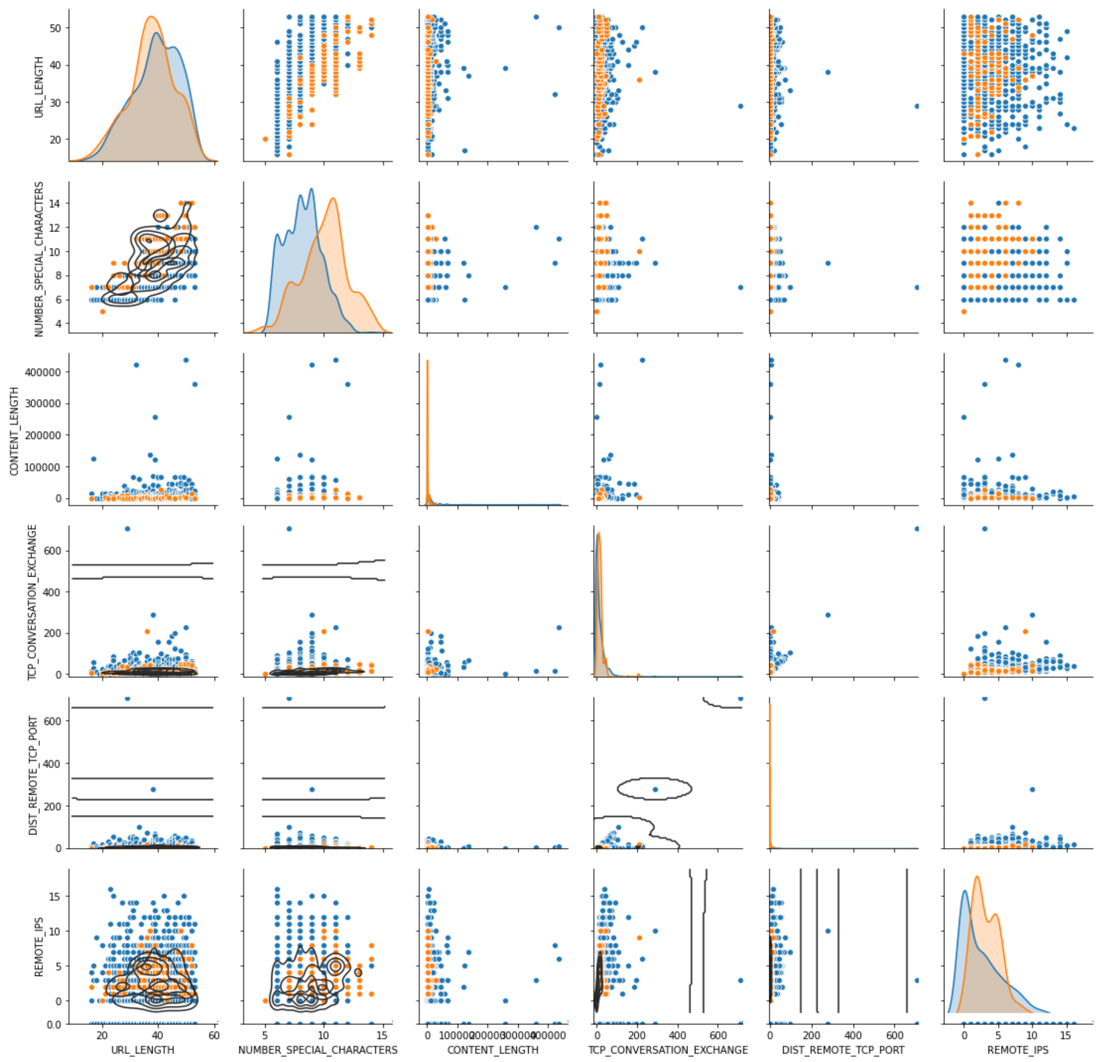


**Fig. 6:** Scatter and density plot.

Figures 8-11 show the learning curve of the learning classification algorithms. In Figure 8, we present the *k*-NN classifier's learning curve at different neighbors. It achieves 89% of testing accuracy. Figure 9 displays the learning curve of the LR classifier. It achieves 94% of testing accuracy. Figure 10 gives the learning curve of the RF classifier. It achieves 96% of testing accuracy. Finally, Figure 11 visualizes the learning curve of the SVM classifier. It achieves 86% of testing accuracy. Among these classifiers, the RF model attains the best accuracy rate for classifying malicious and benign websites' URLs.

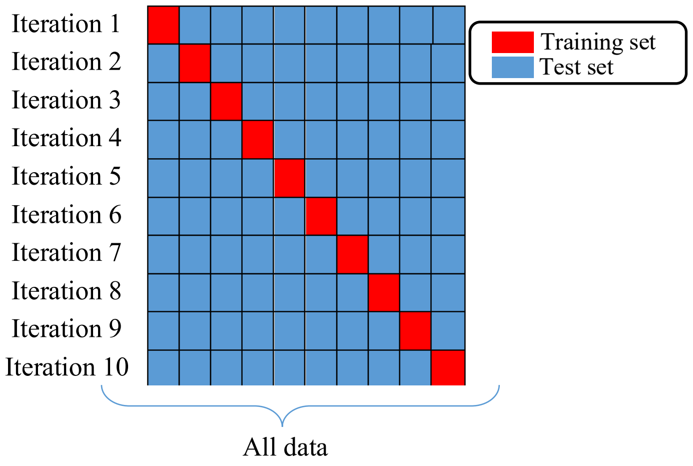


**Fig. 7:** a 10-fold cross-validation technique used for evaluating the proposed approach.

**Table 3:** Comparisons of adopted machine learning algorithms.

| Algorithms | Acc. | Pre. | Rec. | F1-score |
|---|---|---|---|---|
| KNN | 0.89 | 0.88 | 0.89 | 0.88 |
| LR | 0.94 | 0.94 | 0.94 | 0.93 |
| RF | 0.96 | 0.96 | 0.96 | 0.95 |
| SVM | 0.86 | 0.88 | 0.87 | 0.80 |

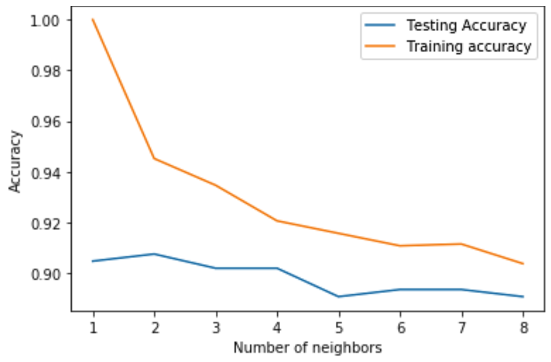


**Fig. 8:** Learning curve of *k*-NN classifier.

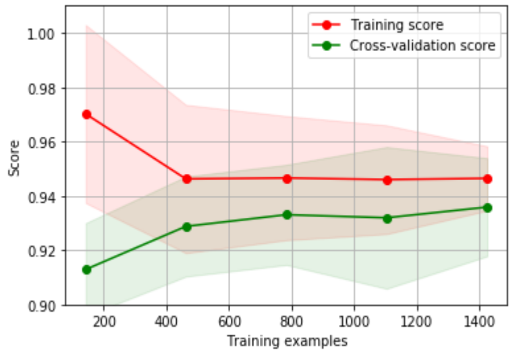


**Fig. 9:** Learning curve of LR classifier.

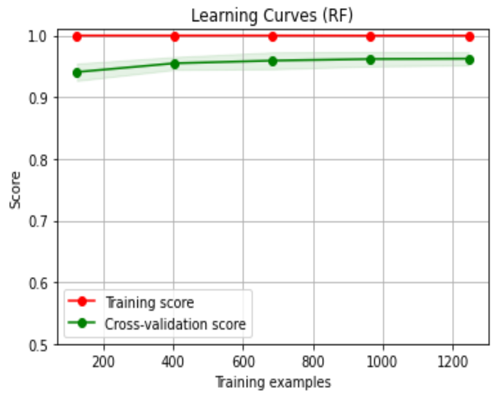


**Fig. 10:** Learning curve of RF classifier.

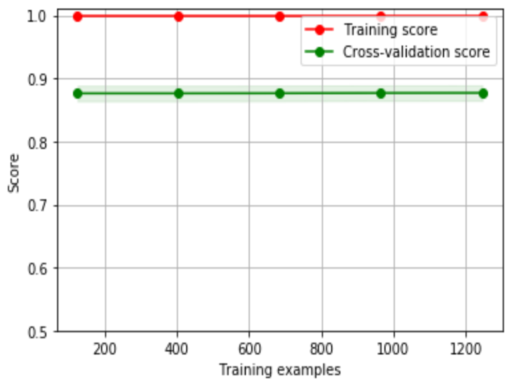


**Fig. 11:** Learning curve of SVM classifier.

## 5 Conclusion and Future Work

In malicious website detection, a preprocessing technique extracts features from the website data, such as URLs, HTML tags, and content, which can be used as input to machine learning models for classification. The preprocessing technique is important in building accurate and reliable machine-learning models for website malicious detection. However, it is important to continually evaluate and improve the effectiveness of the preprocessing technique as new threats and attack techniques emerge. Current methods for automated detection and analysis of malicious websites have some drawbacks in differentiating the ideal webpage and being resistant to potential evasion.

Additionally, selecting effective features while maintaining feature composition still has significant issues that need to be resolved. In fact, in order to deal with changing features and attackers' attempts, existing machine learning-based algorithms also need to be improved. This paper proposes an effective and efficient approach to detect and analyze malicious websites. Specifically, we enhanced the selection and extraction of the feature set for characterizing malicious payloads by merging URL tokens, page identity, execution trace, and page content of malicious websites. Extending the current related work, the effectiveness of preprocessing the data features using RF and Pearson's correlation analysis is investigated to improve the detection task of malicious websites. The proposed approach is evaluated using four classification algorithms (*k*-NN, LR, RF, and SVM).

The experimental results show the effectiveness of our approach, which has a detection accuracy of 96% with an RF algorithm due to its robustness and ability to handle noisy data and achieve a high accuracy result in classification tasks, especially when dealing with complex datasets. A potential direction for future study is fully training neural networks on the raw URLs, content, and images rather than depending on simple models and embedding to strengthen the feature type coverage.

Classifying website URLs into malicious or benign classes in IoT environments presents unique challenges due to the constrained resources and heterogeneity of IoT devices. By developing the proposed framework, it can address these challenges and effectively classify website URLs into malicious or benign classes, enhancing cybersecurity in IoT deployments. The selected best model of the framework can be deployed in the IoT environment to classify website URLs in real-time. Continuous monitoring of model performance and retraining with updated data would be essential to maintain effectiveness over time.

## Conflicts of Interest Statement

*The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.*

***Acknowledgment:***

The authors extend their appreciation to Prince Sattam bin Abdulaziz University for funding this research work through the project number (PSAU/2023/01/27118).

## References

[1] Yan, X., Xu, Y., Cui, B., Zhang, S., Guo, T. et al. (2020). Learning URL embedding for malicious website detection. *IEEE Transactions on Industrial Informatics, 16*(**10**), 6673-6681.

[2] Stevanovic, D., Vlajic, N., An, A. (2013). Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Applied Soft Computing, 13*(**1**), 698-708.

[3] Xu, L., Zhan, Z., Xu, S., Ye, K. (2013). Cross-layer detection of malicious websites. *Proceedings of the third ACM conference on Data and application security and privacy*, pp. 141-152. San Antonio, Texas, USA.

[4] McGahagan IV, J., Bhansali, D., Pinto-Coelho, C., Cukier, M. (2021). Discovering features for detecting malicious websites: An empirical study. *Computers Security, 109*, 102374.

[5] O'Kane, P., Sezer, S., Carlin, D. (2018). Evolution of ransomware. *LET Networks, 7*(**5**), 321-327.

[6] Papadie, R., Apostol, I. (2017). Analyzing websites protection mechanisms against DDoS attacks. *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pp. 1-6.

[7] Jang-Jaccard, J., Nepal, S. (2014). A survey of emerging threats in cybersecurity. *Journal of Computer System Sciences, 80*(**5**), 973-993.

[8] Jaiswal, M. (2017). Computer Viruses: Principles of Exertion, Occurrence and Awareness. *International Journal of Creative Research Thoughts*, 648-651.

[9] Peng, Y., Tian, S., Yu, L., Lv, Y., Wang, R. (2019). A joint approach to detect malicious URL based on attention mechanism. *International Journal of Computational Intelligence Applications, 18*(**03**), 1950021.

[10] Do Xuan, C., Nguyen, H. D., Tisenko, V. N. (2020). Malicious URL detection based on machine learning. *International Journal of Advanced Computer Science Applications, 11*(**1**).

[11] Alsaedi, M., Ghaleb, F. A., Saeed, F., Ahmad, J., Alasli, M. (2022). Cyber threat intelligence-based malicious URL detection model using ensemble learning. *Sensors, 22*(**9**), 3373.

[12] Chen, Y.-C., Ma, Y.-W., Chen, J.-L. (2020). Intelligent malicious URL detection with feature analysis. *Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1-5. Rennes, France.

[13] Li, K., Yu, X., Wang, J. (2021). A Review: How to Detect Malicious Domains. *Proceedings, Part III: Advances in Artificial Intelligence and Security 7th International Conference, ICAIS 2021*, pp. 152-162. Dublin, Ireland.

[14] Rai, A., Singh, A. S., Kumar, A. S. (2020). A Review of Information Security: Issues and Techniques. *International Journal for Research in Applied Science Engineering Technology, 8*(**5**), 953-963.

[15] Cova, M., Kruegel, C., Vigna, G. (2010). Detection and analysis of drive-by-download attacks and malicious JavaScript code. *Proceedings of the 19th international conference on World wide web*, pp. 281-290.

[16] Alanazi, A., Gumaei, A. (2023). A Decision-Fusion-Based Ensemble Approach for Malicious Websites Detection. *Applied Sciences, 13*(**18**), 10260.

[17] Vyawhare, C. R., Totare, R. Y., Sonawane, P. S., Deshmukh, P. B. Machine Learning System for Malicious Website Detection using Concept Drift Detection.

[18] Eshete, B., Villafiorita, A., Weldemariam, K. (2011). Malicious website detection: Effectiveness and efficiency issues. *2011 First SysSec Workshop*, pp. 123-126.

[19] Singhal, S., Chawla, U., Shorey, R. (2020). Machine learning & concept drift based approach for malicious website detection. *Proceedings of the International Conference on COMmunication Systems & NETworkS (COMSNETS) 2020*, pp. 582-585. Bengaluru, India.

[20] Singh, A., Goyal, N. (2019). A comparison of machine learning attributes for detecting malicious websites. *Proceedings of the 11th International Conference on Communication Systems & Networks (COMSNETS) 2019*, pp. 352-358. Bengaluru, India.

[21] Qassrawi, M. T., Zhang, H. (2011). Detecting malicious web servers with honeyclients. *Journal of Networks, 6*(1), 145.

[22] Chaiban, A., Sovilj, D., Soliman, H., Salmon, G., Lin, X. (2022). Investigating the Influence of Feature Sources for Malicious Website Detection. *Applied Sciences, 12*(**6**), 2806.

[23] Emmah, V. T., Ukorma, G., Taylor, O. E. (2022). A Model for Malicious Website Detection Using Feed Forward Neural Network. *European Journal of Computer Science Information Technology Control, 10*(**2**), 18-26.

[24] Srinivasan, S., Vinayakumar, R., Arunachalam, A., Alazab, M., Soman, K. (2021). DURLD: Malicious URL detection using deep learning-based character level representations. *Malware analysis using artificial intelligence deep learning*, 535-554.

[25] Castruccio, S., Genton, M. G. (2018). Principles for statistical inference on big spatio-temporal data from climate models. *Statistics Probability Letters, 136*, 92-96.

[26] Wang, Y.-M., Beck, D., Jiang, X., Roussev, R., Verbowski, C. et al. (2006). Automated web patrol with strider honeymonkeys. *Proceedings of the 2006 Network and Distributed System Security Symposium*, pp. 35-49.

[27] Li, X., Yeh, A. G.-O. (2002). Neural-network-based cellular automata for simulating multiple land use changes using GIS. *International Journal of Geographical Information Science, 16*(**4**), 323-343.

[28] Rao, K. V., Govardhan, A., Rao, K. C. (2012). Spatiotemporal data mining: Issues, tasks and applications. *International Journal of Computer Science Engineering Survey, 3*(**1**), 39.

[29] Wang, N., Cheng, W., Zhao, M., Liu, Q., Wang, J. (2019). Identification of the debris flow process types within catchments of Beijing Mountainous Area. *Water, 11*(**4**), 638.

[30] Singh, A., Goyal, N. (2017). Malcrawler: A crawler for seeking and crawling malicious websites. *Proceedings of the 13 Distributed Computing and Internet Technology: 13th International Conference, ICDCIT 2017*, pp. 210-223. Bhubaneswar, India.

[31] Singh, A. (2020). Malicious and benign webpages dataset. *Data in brief, 32*, 106304.

[32] Hidden fraudulent URLs dataset. (10 December 2023). *[online] Available: https://machinelearning.inginf.units.it/data-and-tools/hidden-fraudulent-urls-dataset*.

[33] Malicious_n_Non-Malicious URL. (10 December 2023). *[online] Available: https://www.kaggle.com/datasets/antonyj453/urldataset*.

[34] Han, J., Pei, J., Tong, H. (2022). *Data mining: concepts and techniques*: Morgan kaufmann.

[35] Das, A. (2021). Logistic regression. In *Encyclopedia of Quality of Life and Well-Being Research* (pp. 1-2): Springer.