# Characterization of Malicious URLs Using Machine Learning and Feature Engineering

Sidwendluian Romaric Nana(✉), Didier Bassolé, Jean Serge Dimitri Ouattara, and Oumarou Sié

Laboratoire de Mathématiques et d'Informatique, Université Joseph KI-ZERBO, Ouagadougou, Burkina Faso
sidnanaroma@gmail.com
https://www.ujkz.bf

**Abstract.** In this paper, we use Machine Learning models for malicious URL detection and classification by Feature Engineering techniques. These models were implemented with scikit-learn using Random Forest, Support Vector Machine and XGBoost classifier algorithms. Our models were trained, tested, and then optimized with a dataset of 641,125 URLs (benign, defacement, malware, and phishing) from several sources including ISCX-URL2016 from the University of New Brunswick. Through iterative learning, we have shown that the combination of certain hyperparameters and features reduces the false positive rate. The results obtained are interesting with scores close to 100% and zero false positive rates for some types of URLs. We then evaluated the performance of the models against other related works models.

**Keywords:** Malicious URL · Characterization · Feature Engineering · Detection · Classification

## 1 Introduction

Development of the web has been accompanied by new practices and new uses, among which we can note the multiplication of websites as well as a diversification of contents. Web sites handle sensitive data (passwords, credit card identifiers, banking transaction information, etc.), making them a prime target for hackers and malicious users. The latter multiply initiatives to steal sensitive information, usurp the identity of internet users or cause an interruption of services, thus discrediting the company that owns the website and damaging its brand image.

There are a wide variety of threats on the web and the techniques used to implement these threats are also diverse. Among the techniques, we can mention website defacement, SQL injections, DDoS attacks, ransomware, formjacking, spamming, use of phishing Uniform Resource Locator (URL), other malicious links, etc. According to the Internet Security Threat Report in 2019, attacks


A. Seeam et al. (Eds.): InterSol 2023, LNICST 541, pp. 15–32, 2024.
https://doi.org/10.1007/978-3-031-51849-2_2

using URL propagation techniques are increasing in number of attacks as well as in level of dangerousness [1].

An URL is an address that identifies a resource on the World Wide Web. An URL has two main components [2]:

- protocol identifier (indicates the protocol to be used);
- resource name (specifies the IP address with port or domain name where the resource is located)

The protocol identifier and resource name are separated by a *colon* (:) and *two slashes* (//), as in the Fig. 1 (inspired by [2]).

Malicious URLs lead users to unsolicited resources, phishing sites or web pages from which attackers can execute malicious code on their victims' computers or install malware. It is therefore imperative to detect these malicious URLs before internet users use them, which can compromise resources. Several approaches have been encountered in the literature for the detection of malicious URLs, notably methods based on signatures [2,3]. These methods have proven ineffective in recognising new malicious URLs in view of the increasing number of attacks and web pages consulted daily in the world[1].

Face of this new reality of massive data and widespread web attacks, methods for detecting malicious URLs have improved through analysis of URL behaviour and attributes. These new methods leverage the performance of Machine Learning (ML) and Deep Learning (DL) algorithms to detect and classify URLs [2,4,5]; but the challenges of false positives and false negatives remain.

The main purpose of this work is to improve the performance of ML models in the detection of malicious URLs and their classification (benign, defacement, phishing, malware). This includes reducing the False Positive Rate (FPR) and detecting new malicious URLs whose dangerousness has not yet been proven.

The rest of this paper is organised as follows: Sect. 2 deals with related works. Section 3 presents our methodological approach. Section 4 shows our implementation environment and the results obtained. We conclude this work in Sect. 5.

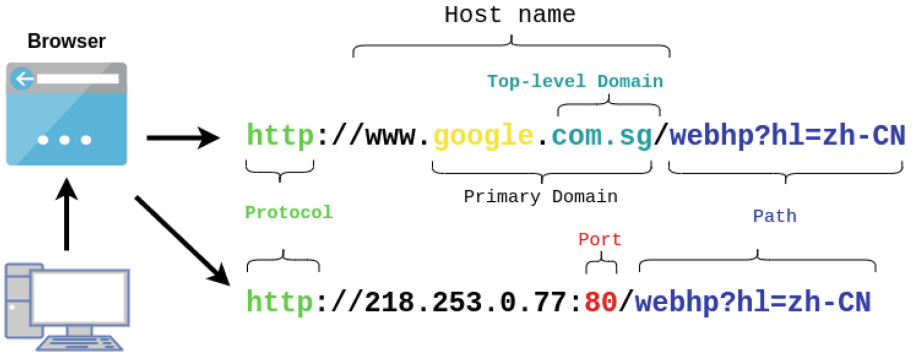


**Fig. 1.** Example of a URL

[1] https://www.akamai.com/fr.

## 2 Related Works

The web is one of the largest sources of information and knowledge sharing in the world. Unfortunately, attacks on the web are diverse. We can cite among others: formjacking, phishing, SQL Injection, ransomwares, File inclusion, DDoS attacks. In view of the enormous political, economic and social stakes of the web, content security has always been at the heart of the preoccupations of governments and the research world. Several approaches have been experimented in the literature to protect internet users against malicious URLs, some more effective than others, using different techniques. In this section, we will review what is known about detecting malicious URLs.

### 2.1 Malicious URL Detection by Signature Based Approach

Malicious URLs are a common and serious threat to cyber security. Malicious URLs lead to websites that host unsolicited content (spam, phishing, drive-by downloads, etc.) and lure unsuspecting users into becoming victims of scams (financial losses, theft of private information and installation of malware), which account for billions of dollars in losses each year [2].

For the detection of malicious URLs, the most intuitive method is blacklisting. This method has been explored in the literature for many years [6–8]. URL blacklists are databases of all known malicious URLs. However, the blacklisting method has significant shortcomings. Indeed, it is very difficult to keep the list of malicious URLs up to date. This method is therefore ineffective in detecting new URLs. Furthermore, cybercriminals use URL reduction services to hide their malicious URLs [9], thus bypassing blacklisting checks.

Signature based approach is a variant of the blacklisting method. A signature is assigned to the most common attacks. Some intrusion detection systems use this type of approach. They have the ability to scan web pages for suspicious signatures. Search for malicious signatures could also be done by analysing the dynamic execution of web pages [3]. For example the creation of an unusual process, a multiple redirection. This type of manipulation must be done in a controlled environment in order to limit the risks in case of an attack during the execution of web pages. On the other hand, if an attack is not launched automatically during the execution of the page, we could be faced with a case of false negative.

### 2.2 Malicious URL Detection by Behaviour Based Approach

Method of detecting malicious URLs based on behavioural analysis techniques adopts machine or deep learning algorithms to classify URLs according to their behaviour [5].

The behaviour and characteristics of URLs can be divided into two main groups: static and dynamic. In their studies [10,11], authors presented methods for analysing and extracting the static behaviour of URLs, including lexical features [2,12]; content based features [13–15]; host based features [16]. The

machine learning algorithms used in these studies are online learning algorithms and SVMs. Malicious URLs detection using dynamic actions of URLs is presented in [17,18].

Gerardo Canfora et al. [18] proposed a method to count system calls (write, send, mkdir, open, etc.) when a web page is executed. The study of these system calls showed that malicious web pages execute system calls more frequently than others; then SVM was used to classify malicious and benign web pages. Maheshwari, Shantanu et al. [19] conducted a comparative study of Machine Learning algorithms in detecting malicious URLs. This study which consisted of a binary classification (malicious and benign URLs) revealed good performance for the Random Forest.

Nguyet Quang Do et al. [4] analysed the performance of various DL algorithms in detecting phishing activities. Four DL algorithms were involved in this empirical study: Deep Neural Network (DNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).

Clayton Johnson et al. [20] conducted a performance comparison between ML aglorithms (Random Forest, Classification and Regression Tree, k-Nearest Neighbor, Support Vector Machine, Logistic Regression, Linear Discriminant Analysis, AdaBoost, and Naive Bayes) and DL models (Fast.ai, Keras-TensorFlow across CPU, GPU, and TPU architectures).

Experimentation with RF, Fast.ai, Keras-TensorFlow proved more successful for binary classification (benign and malicious URLs) and mutiple classification (benign URLs, spam, defacement, phishing, malware) [20].

### 2.3 URL Feature Representation Methods

Doyen Sahoo et al. [2] have listed some key attribute groups for malicious URL detection. Their paper is a review of the literature on malicious URL detection techniques using ML. However, particular emphasis is placed on URL feature representation approaches. There are three types of URL feature found in previous works: lexical features, host-based features and content-based features.

**Lexical features** are obtained from the properties of string URL. The main idea is that the constitution of string URL could tell us more about its malicious nature. This idea is further supported as cyber-criminals attempt to circumvent blacklisting through URL obfuscation techniques [2]. The most common lexical features include the statistical properties of the string URL such as the length of the URL, the length of each URL component (Hostname, Top Level Domain, Primary Domain, etc.), the number of special characters, etc.

**Host-based features** describe the properties of the website host as identified by the hostname part of the URL. They enable us to determine approximately "where" malicious sites are hosted, "who" owns them and "how" they are managed. The use of host-based features is based on the assumption that malicious web sites may be hosted in less reputable hosting centers, on machines that are

not those of conventional hosting providers, or via disreputable registrars [16]. The following properties can be considered when building host-based features: WHOIS information, Host location, connection speed, presence of URL on a blacklist.

**Content-based features** determine whether a URL is malicious by analyzing the content of the corresponding web page and the events (or actions generated) after a URL has been visited. The workload for these features is quite heavy, as a lot of information needs to be extracted, and access to this URL may pose security problems. Features based on the content of a website can be extracted mainly from its HTML content and the use of JavaScript [13–15]. Hyunsang Choi et al. [14] identify some native JavaScript functions that are often used in XSS (cross-site scripting) attacks and others web malware. These functions are escape(), eval(), link(), unescape(), exec() and search(). More recently, Yao Wang et al. [15] have applied deep learning techniques to build feature representations from JavaScript code.

A more complete survey of URL representation methods can be found in reference [2]. overall, we can see from this literature review that detection and classification of malicious URLs have been the subject of research for several years. The techniques used are constantly evolving, leading to improved results. Unfortunately, very few studies use False Positive Rate (FPR) as a metric for evaluating the approaches they propose. However FPR is a very important criterion for assessing the quality of an anomaly or hazard detection system. Furthermore, the absence of a study on the importance score of features makes it impossible to understand the contribution of each feature to the overall performance of models. Our detection approach, based on Feature Engineering with lexical features as representation method, aims to overcome the shortcomings identified in the related works.

## 3 Methodological Approach

In this section, we present our approach for detecting and classifying malicious URLs. We will first present the feature extraction technique (Feature Engineering), the dataset, and finally our learning models.

### 3.1 Feature Engineering

Feature Engineering is the process of transforming raw data into features that better represent the underlying problem for predictive models, thereby improving the accuracy of the model on data that it has not yet been subjected to.

In other words, feature engineering is the process of creating new features from existing data for machine learning models. These features are extracted

from the accumulated raw data and then transformed into formats suitable for the machine learning process. For this purpose, knowledge of the data domain is essential, as well as programming and mathematical skills are also mandatory to apply feature engineering methods.

Performing feature engineering involves the following steps for machine learning algorithms:

- data preparation;
- exploratory analysis;
- Benchmarking.

### 3.2 Presentation of the Dataset and Exploratory Analysis

Our dataset contains 651,191 URLs divided into four (04) categories:

- 428,103 benign URLs;
- 96,457 website defacement URLs;
- 94,111 phishing URLs;
- 32,520 other malicious URLs.

**Benign URLs:** benign URLs are legitimate URLs that do not lead to any infectious websites and do not attempt to inject any harmful malware into the user's computer.

**Defacement URLs:** defacement of a website usually means changing some aspect of the site, such as its visual appearance and some of its content. The hacker takes advantage of a security hole and manages to change the content of a web page without the permission of the original 'owner' of the site [21].

**Malware URLs:** malware URLs lead the user to a malicious website that usually installs malware on the user's device, which can be used for identity theft, file corruption and even keystroke logging, etc.

**Phishing URLs:** phishing URLs usually entice the user to visit a fake website (with a URL that looks like the real one) and try to steal as much information as possible. Sometimes a simple typo in a URL can easily lead a user to a phishing site. Phishing can be defined as the hacker's intention to steal private information such as credit card numbers and other digital identities using social engineering techniques [22].

URLs of our dataset come from various sources[2,3,4] including [23] [URL dataset (ISCX-URL2016) of the University of New Brunswick]. The dataset is presented as a table with 651,191 rows and 2 columns. Table 1 presents the first three (03) rows of the dataset. A summary analysis of the dataset allows us to observe the proportion of URLs according to the type of threat: benign, defacement, phishing, malware (Table 2).

**Table 1.** Dataset overview

| URLs | TYPE |
|---|---|
| br-icloud.com.br | phishing |
| mp3raid.com/music/krizz_kaliko.html | benign |
| bopsecrets.org/rexroth/cr/1.htm | benign |

**Table 2.** Distribution of URLs

| URLs | Number of Urls | Percentage |
|---|---|---|
| benign | 428,103 | 65.74% |
| defacement | 96,457 | 14.81% |
| phishing | 94,111 | 14.45% |
| malware | 32,520 | 5.00% |
| **TOTAL** | **651,191** | **100%** |

In accordance with the Feature Engineering techniques outlined above, preprocessing and exploratory analysis of the data are important steps. The preprocessing on our dataset allowed us to detect duplicates (duplicated lines). The removal of these duplicates reduced our dataset to 641,125 rows distributed as shown in the Table 3.

**Table 3.** Distribution of URLs after preprocessing

| URLs | Number of Urls | Percentage |
|---|---|---|
| benign | 428,080 | 66.77% |
| defacement | 95,308 | 14.86% |
| phishing | 94,092 | 14.68% |
| malware | 23,645 | 3.69% |
| **TOTAL** | **641,125** | **100%** |

For a binary analysis of URLs (benign/malicious), we can group URLs of type (phishing, defacement, malware) under the common denomination "malicious URLs". Table 4 shows us the results of this binary division.

[2] https://www.phishtank.com/developer_info.php.
[3] https://github.com/faizann24/Using-machine-learning-to-detect-malicious-URLs/tree/master/data.
[4] https://research.aalto.fi/en/datasets/phishstorm-phishing-legitimate-url-dataset.

**Table 4.** Benign/malicious URLs

| URLs | Number of Urls | Percentage |
|---|---|---|
| benign | 428,080 | 66.77% |
| malicious | 213,045 | 33.23% |
| **TOTAL** | **641,125** | **100%** |

The knowledge of the field of study (URLs) led us to take an interest in certain elements of the URLs. Thus, we were interested in the domain extensions (Top-Level Domain) most represented in the URLs of our dataset according to the type of threat (phishing, malware, defacement). The extensions encountered are among others the most popular domain extensions such as: .com, .net, .org, .info. However, there are country domain extensions such as: .jp, .uk, .ru, .br, .de, .it, etc.

The rest of the exploratory analysis of our dataset was based on the practices of malicious users. Thus we were interested in the presence in our dataset of:

- the use of IP addresses in URLs;
- the length of URLs;
- the use of URL shortening services;
- the use of the "@" character in URLs;
- hidden redirects using "//" symbols;
- the use of dashes (-) in the domain name;
- use of the https protocol;
- use of the term "https" in the domain name;
- the use of suspicious words such as: *banking, secure, login, account, bank, confirm, paypal, update, hacking, hacker*, etc.

By applying the different steps of Feature Engineering as described above, we have extracted features from our dataset that we believe are relevant for our models. These features are summarised in the Table 5.

**Table 5.** List of features

| FEATURES | TYPE | DESCRIPTION |
|---|---|---|
| 1. length_url | Integer | Number of characters in the URL |
| 2. number_letter | Integer | Number of alphabetical characters |
| 3. numeric_character | Integer | Number of numeric characters |
| 4. number_dot_com | Integer | Number of ".com" |
| 5. number_dot_co | Integer | Number of ".co" |
| 6. number_dot_net | Integer | Number of ".net" |
| 7. number_of_slash | Integer | Number of "/" in the URL |
| 8. number_of_uppercase | Integer | Number of uppercase characters in the URL |
| 9. number_of_lowercase | Integer | Number of lowercase characters in the URL |
| 10. number_dot_info | Integer | Number of ".info" |
| 11. number_of_https | Integer | Number of "https" in the URL |
| 12. number_of_http | Integer | Number of "http" in the URL |
| 13. num_of_www_point | Integer | Number of "www." |
| 14. special_characters | Integer | Number of special characters as &, _, #, ;... |
| 15. num_of_percentage | Integer | Number of character "%" in the URL |
| 16. question_mark | Integer | Number of "?" in the URL |
| 17. number_of_dash | Integer | Number of "-" in the URL |
| 18. dash_in_domain | Integer | Use of "-" dashes in the domain name |
| 19. num_equal_symbol | Integer | Number of "=" in the URL |
| 20. percentage_character | Real | Ratio of "%" in the URL |
| 21. space_in_url | Boolean | Use of "%20" in the URL path |
| 22. suspicious_words | Boolean | Presence of sensitive words in the URL |
| 23. ip_adress | Boolean | Presence of the IP address in the URL instead of the domain name |
| 24. protocol_https | Boolean | Use of the https protocol |
| 25. https_domaine | Boolean | Use of the term "https" in the domain name |
| 26. number_sub_domain | Integer | Number of subdomain in the URL |
| 27. short_link | Boolean | Use of short links in the URL |
| 28. number_at | Integer | Number of symbol "@" in the URL |
| 29. num_double_slash | Integer | Hidden redirection by using the // symbol in the URL |
| 30. hostname_length | Integer | Hostname length |
| 31. path_length | Integer | Path length in the URL |
| 32. first_directory_length | Integer | First directory length in the URL |
| 33. tld_length | Integer | Length of the Top Level Domain in the URL |

### 3.3 Implementation Scenario

Authors of the ISCX-URL2016 dataset propose 79 features that can be extracted from the URLs [24]. Contrary to other works in the literature [5,20], our hypothesis is that the plethora of features does not necessarily contribute to the performance of models. Thus the reduction of features could improve the performances. Using Feature Engineering techniques, we have defined 33 features (see Table 5) that are essential for the detection and classification tasks which is a contribution of this research paper.

For this study, three (03) scenarios were implemented:

- detection of malicious URLs in general;
- detection of malicious URLs specifically (defacement, phishing, malware);
- classification of malicious URLs.

The detection of malicious URLs consisted of a binary classification. Using the dataset presented in Table 4, a model was trained to predict whether a URL is benign or not. For the second scenario, the aim was to create datasets by contrasting benign URLs with other types of URLs. Thus, three (03) datasets were created with the following pairs: (benign - malware), (benign - defacement), (benign - phishing). These datasets were trained to recognise one type of URL (phishing, defacement, malware). For the third scenario, we used the dataset presented in Table 3 and trained the model to recognize the four (04) classes of URLs (benign, phishing, defacement, malware). Figure 2 illustrates the description of scenarios.

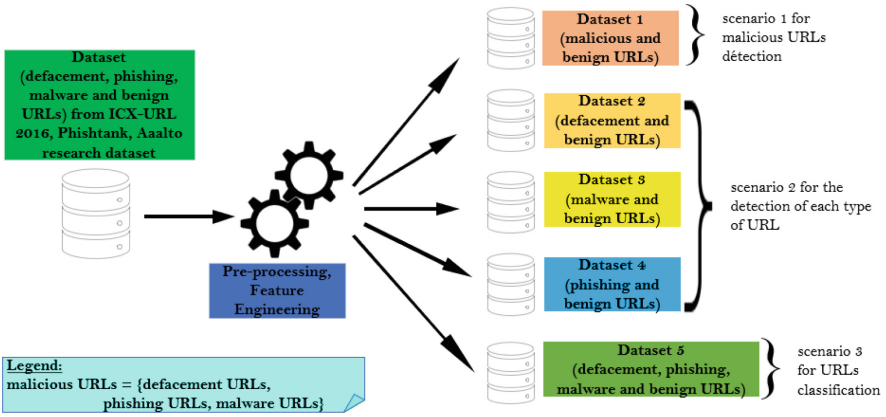


**Fig. 2.** Scenarios description

## 4 Implementation and Results

### 4.1 Execution Environment and Tools Used

We carried out this work using *open source* tools. The programming language used to write the scripts is Python with the Jupyter Notebook editor. Our work environment is a laptop computer with the Ubuntu 20.04 LTS operating system and the following characteristics:

- Processor: Intel® Core™ i7-4810MQ CPU @ 2.80 GHz × 8 threads
- RAM: 16 Gb
- Graphics: NVE6/Intel® HD Graphics 4600 (HSW GT2)

From our dataset, we performed data processing, feature extraction and model implementation tasks. To do so, we wrote python scripts with a set of libraries such as Pandas[5], Numpy[6], Matplotlib[7], Scikit-learn[8], xgboost[9], Seaborn[10], Pickle[11].

### 4.2 Models Implementation

Before submitting our datasets to the different learning models: Random Forest Classifier (RF), Support Vector Classification (SVC), XGBoost Classifier (XGB) (see table 6), we subdivided them into two parts: training data (train set) and test data (test set) with *train_test_split* function in scikit-learn. This data subdivision principle, which we describe below, has been respected for all types of experimentation (binary and multi-label classification). These are:

- spliting: Train set 80%, Test set 20%, Shuffle = True, random_state = 5. **shuffle** is a boolean that indicates whether or not the data will be shuffled before it is subdivided. **random_state** is an integer that controls the level of mixing of the data.
- determination of features (X_train, X_test) and labels (y_train, y_test)
- saving variables in different files in csv format. This ensures the reusability of these variables for all training models

By doing this we ensure that the different models have been trained and tested on the same variables. The comparison of the results of these models was done on a fairness basis. By tuning the hyperparameters, we obtained the best performance of each model before comparing them on the basis of scores and metrics. Figure 3 presents model implementation process. Table 6 presents a summary of the comparative analysis of some ML models [26–32].

[5] https://pandas.pydata.org/.
[6] https://numpy.org/.
[7] https://matplotlib.org/.
[8] https://scikit-learn.org/stable/.
[9] https://xgboost.readthedocs.io/en/stable/.
[10] https://seaborn.pydata.org/.
[11] https://docs.python.org/3/library/pickle.html.

**Table 6.** Summary of the comparative analysis of some ML models

| Model | Advantages | Disadvantages |
|---|---|---|
| RF | no overfitting | complex in the implementation |
| | requires less data pre-processing | requires more computing resources for optimal implementation |
| | supports large data volumes | longer processing time |
| | suitable for multi-label classification | less intuitive as the number of decision trees increases |
| XGB | regularization | a model with multiple hyperparameters |
| | parallel processing | a bad extrapolation tool |
| | supports large data volumes | |
| | handling of missing values | |
| | integrated crossvalidation | |
| SVM | SVM works best when the data are linear | choosing the right kernel is not easy |
| | SVM is more efficient in high dimensions | SVM does not perform well on a large dataset |
| | Complex problems can be solved by SVM | Hyperparameters are difficult to adjust |
| | SVM is not sensitive to outliers | It is difficult to visualise the impact of hyperparameters |

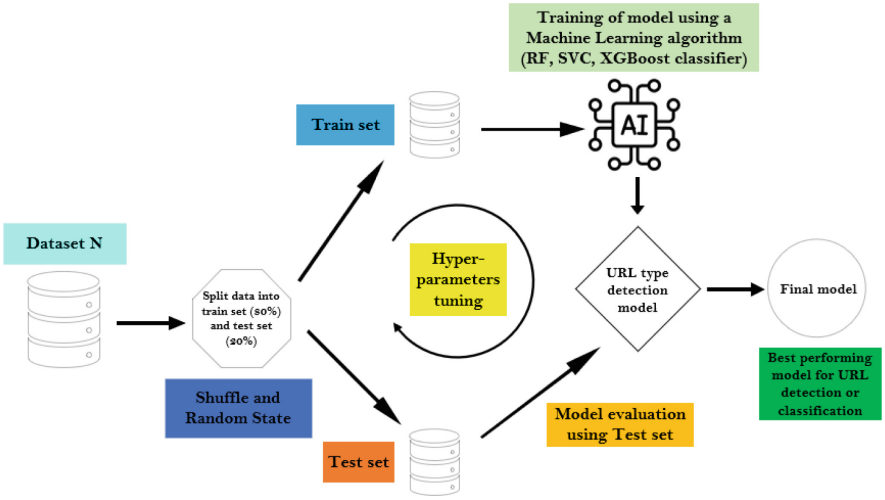


**Fig. 3.** Model implementation process

### 4.3 Scores and Metrics

To evaluate the models, we used the performance indicators of the algorithms. These indicators are based on the confusion matrix presented in Table 7. The confusion matrix is used to summarise and visualise the results of a classification problem.

**Table 7.** Confusion matrix

| | Negative prediction | Positive prediction |
|---|---|---|
| Benign | TN (True Negative) | FP (False Positive) |
| Malicious | FN (False Negative) | TP (True Positive) |

- TN: benign URLs are identified as benign
- FP: benign URLs are identified as malicious
- FN: malicious URLs are identified as benign
- TP: malicious URLs are identified as malicious

Among the performance indicators based on the confusion matrix, we have *Precision* (Pr), *Recall* (Re), *F1-score* (F1), *Accuracy* (Acc), *False Positive Rate* (FPR), *False Negative Rate* (FNR):

$$Pr = \frac{TP}{TP + FP} \tag{1}$$

$$Re = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

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

$$FNR = \frac{FN}{FN + TP} \tag{6}$$

*FEATURE IMPORTANCE* refers to a set of techniques for assigning scores to the features (input characteristics) of a predictive model that indicate the relative importance of each feature in the prediction. Feature importance scores can be calculated for regression problems as well as for classification problems. Thus in our study, the features with the highest importance scores for the detection of each type of URL are:

- **defacement URL**
  - number of times the term “http” appears in the URL;
  - number of times the term “www.” appears in the URL;

- number of times the equal symbol "=" appears in the URL;
- length of the first directory of the URL;
- number of special characters in the URL.

– **malware URL**
- number of times the term "http" appears in the URL;
- path length of the URL;
- URL length;
- presence of the IP address in the URL instead of the domain name;
- whether or not the https protocol is used;
- number of subdomains in the URL.

– **phishing URL**
- number of times the term "www." appears in the URL;
- path length of the URL;
- number of directories in the URL (Number of "/" in the URL);
- presence of sensitive words in the URL (e.g. "secure", "account", "webscr", "login", "ebayisapi", "sign inv, "banking", "confirm", "hacker", "pirate", "paypal", ...);
- length of the Top Level Domain in the URL.

Since we worked with unbalanced data (not all classes had the same number of examples), we will use the F1-score metric to compare the performance of different models. We will also compare the models on the basis of the FPR values. In Tables 8, 9, 10, 11, 12, we present results of our implementation.

**Table 8.** Malicious URLs detection

(a) Results of malicious URLs detection

| | RF | SVC | XGB |
|---|---|---|---|
| Accuracy | 97.8% | 91.6% | 97.1% |
| FPR | **1.13%** | **2.15%** | **1.24%** |

(b) Comparaison of F1-score values

| | RF | SVC | XGB |
|---|---|---|---|
| Benign | **0.98** | 0.94 | **0.98** |
| Malicious | **0.97** | 0.86 | **0.95** |

**Table 9.** Defacement URLs detection

(a) Results of defacement URLs detection

| | RF | SVC | XGB |
|---|---|---|---|
| Accuracy | **100%** | **99.2%** | **99.9%** |
| FPR | **0.02%** | **0.24%** | **0.03%** |

(b) Comparaison of F1-score values

| | RF | SVC | XGB |
|---|---|---|---|
| Benign | **1.00** | **1.00** | **1.00** |
| Defacement | **1.00** | **0.98** | **1.00** |

**Table 10.** Malware URLs detection

(a) Results of malware URLs detection

| | RF | SVC | XGB |
|---|---|---|---|
| Accuracy | 99.9% | 99.4% | 99.9% |
| FPR | **0.02%** | **0.01%** | **0.03%** |

(b) Comparaison of F1-score values

| | RF | SVC | XGB |
|---|---|---|---|
| Benign | **1.00** | 1.00 | **1.00** |
| Malware | **0.99** | 0.94 | **0.99** |

**Table 11.** Phishing URLs detection

(a) Results of phishing URLs detection

| | RF | SVC | XGB |
|---|---|---|---|
| Accuracy | **97.2%** | 90.4% | **96.6%** |
| FPR | **1.15%** | **1.77%** | **1.13%** |

(b) Comparaison of F1-score values

| | RF | SVC | XGB |
|---|---|---|---|
| Benign | **0.98** | 0.94 | **0.98** |
| Phishing | **0.92** | 0.67 | **0.90** |

**Table 12.** Multi-label classification of URLs

(a) Results of multi-label classification URLs

| | RF | SVC | XGB |
|---|---|---|---|
| Accuracy | **97.2%** | 89.1% | **96.2%** |

(b) Comparaison of F1-score values

| | RF | SVC | XGB |
|---|---|---|---|
| Benign | **0.98** | **0.94** | **0.98** |
| Defacement | **0.99** | **0.92** | **0.98** |
| Malware | **0.95** | 0.83 | **0.92** |
| Phishing | **0.91** | 0.58 | 0.87 |

### 4.4 Comparison with Related Works

As we can only compare elements with a common base, we present in the Tables 13, 14, 15 some of the results of our work in comparison with those of the state of the art on similar fields of study (detection of malicious URLs, phishing and multi-label classification of malicious URLs). Our results with Random Forest and XGBoost models are similar to those of [19] and [20]; they outperform some Deep Learning models [4] and [25] with very low false positive rates.

**Table 13.** Comparison of results with the state of the art - benign vs. malicious

| Ref. | Dataset | Models | FPR |
|---|---|---|---|
| [5] | 10 000 URLs | RF[tr. Acc: 99.77%] | 12.037% |
| [5] | 470 000 URLs | RF[tr. Acc: 96.28%] | – |
| [12] | – | RF [Acc: 95%] | 8.15% |
| [19] | 450 000 URLs | RF [F1: 99%] | – |
| [20] | 19 072 URLs | RF [Acc: 98.68%] | – |
| **Our model** | **641 125 URLs** | **RF [Acc: 97.8%; F1: 98%]** | **1.13%** |

**Table 14.** Comparison of results with the state of the art - benign vs. phishing

| Ref. | Dataset | Models | FPR |
|---|---|---|---|
| [4] | – | DNN [Acc: 97.29%] | 3.01% |
| [4] | – | CNN [Acc: 96.56%] | 3.50% |
| [4] | – | LSTM [Acc: 97.2%] | 1.80% |
| [4] | – | GRU [Acc: 96.7%] | 2.48% |
| [25] | Phishtank | RNN-LSTM [Acc: 97.4%; F1: 96.4%] | – |
| [25] | web crawler | RNN-LSTM [Acc: 96.8%; F1: 96.2%] | – |
| **Our model** | **ISCX-URL2016** | **RF [Acc: 97.2%; F1: 97%]** | **1.15%** |

**Table 15.** Comparison of results with the state of the art - multi-label classification

| Ref. | Dataset | Models |
|---|---|---|
| [14] | – | KNN [Acc: 93.11%] |
| [20] | 19 072 URLs | fast.ai [Acc: 96.77%] |
| **Our model** | **641 125 URLs** | **RF [Acc: 97.2%; F1: 97%]** |

## 5 Conclusion

As a result of this work, we have shown that reducing the number of features (from 79 to 33) improves the performance of models (RF, SVC, XGBoost) in detecting and classifying malicious URLs. This confirms our hypothesis formulated in Sect. 3.3. Then, thanks to the feature importance scores, we observed the contribution of the features to the performance of the models. Three (03) features appear on all the experiments as having major contributions:

- number_of_http: number of times the term "http" appears in the URL;
- num_of_www_point: number of times the term "www." appears in the URL;
- path_length: path length of the URL.

Furthermore, the use of ML models to determine these criteria also contributes to the scientific debate on the explainability of artificial intelligence models. All of which reinforces our view that decision tree models such as RF and XGBoost, if well optimised with the different hyperparameters, perform well in detecting malicious URLs. Another lesson that we draw from this work is that despite the modest hardware resources (see Sect. 4.1), the performance of the models is satisfactory.

In terms of perspectives, we plan to train our models and to study their performance with URLs from the Burkinabe cyberspace. To do so, we will set up a strategy to collect URLs from websites hosted in Burkina Faso or dealing with Burkinabe content in order to constitute a dataset of URLs by type (benign, defacement, malware, phishing). Then, we will deepen the reflection on the detection of phishing attacks by proposing other models or by further refining the extraction of features on this type of URL. Finally, we will extend the study to other types of attacks, in particular SQL injections and cross-site scripting (XSS).

## References

1. Internet Security Threat Report. https://docs.broadcom.com/doc/istr-24-2019-en. Accessed 13 June 2022
2. Sahoo, D., Liu, C., Hoi, S.C.H.: Malicious URL detection using machine learning: a survey (2017). https://doi.org/10.48550/arXiv.1701.07179

3. Moshchuk, A., Bragin, T., Deville, D., Gribble, S.D., Levy, H.M.: SpyProxy: execution-based detection of malicious web content. In: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium. USENIX Association, Boston (2007)
4. Do, Q.N., Selamat, A., Krejcar, O., Yokoi, T., Fujita, H.: Phishing webpage classification via deep learning-based algorithms: an empirical study. Appl. Sci. **11**(9210) (2021). https://doi.org/10.3390/app11199210
5. Xuan, C.D., Nguyen, H.D., Nikolaevich, T.V.: Malicious URL detection based on machine learning. Int. J. Adv. Comput. Sci. Appl. (IJACSA) **11**(1) (2020)
6. Seifert, C., Welch, I., Komisarczuk, P.: Identification of malicious web pages with static heuristics. In: 2008 Telecommunication Networks and Applications Conference, ATNAC 2008, Australasian, pp. 91–96. IEEE (2008)
7. Sinha, S., Bailey, M., Jahanian, F.: Shades of grey: on the effectiveness of reputation-based blacklists. In: 2008 3rd International Conference on Malicious and Unwanted Software, MALWARE 2008, pp. 57–64. IEEE (2008)
8. Sheng, S., Wardman, B., Warner, G., Cranor, L.F., Hong, J., Zhang, C.: An empirical analysis of phishing blacklists. In: Proceedings of Sixth Conference on Email and Anti-Spam (CEAS) (2009)
9. Chhabra, S., Aggarwal, A., Benevenuto, F.: The phishing landscape through short URLs. In: Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference. ACM (2011)
10. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 681–688. ACM (2009)
11. Eshete, B., Villafiorita, A., Weldemariam, K.: BINSPECT: holistic analysis and detection of malicious web pages. In: Keromytis, A.D., Di Pietro, R. (eds.) SecureComm 2012. LNICST, vol. 106, pp. 149–166. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36883-7_10
12. Joshi, A., Lloyd, L., Westin, P., Seethapathy, S.: Emphusing lexical Features for malicious URL detection - a machine learning approach. arXiv (2019). https://arxiv.org/abs/1910.06277
13. Hou, Y.-T., Chang, Y., Chen, T., Laih, C.-S., Chen, C.-M.: Malicious web content detection by machine learning. Expert Syst. Appl. **37**, 55–60 (2010)
14. Choi, H., Zhu, B.B., Lee, H.: Detecting malicious web links and identifying their attack types. In: Proceedings of of the 2nd USENIX Conference on Web Application Development (USENIX Association) (2011)
15. Wang, Y., Cai, W., Wei, P.: A deep learning approach for detecting malicious Javascript code. Secur. Commun. Netw. **9**(11), 1520–1534 (2016). https://doi.org/10.1002/sec.1441
16. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs, In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2009)
17. Tao, Y.: Suspicious URL and device detection by log mining. Master of science thesis, Applied Sciences, School of Computing Science, Simon Fraser University (2014)
18. Canfora, G., Medvet, E., Mercaldo, F., Visaggio, C.A.: Detection of malicious web pages using system calls sequences. In: Teufel, S., Min, T.A., You, I., Weippl, E. (eds.) CD-ARES 2014. LNCS, vol. 8708, pp. 226–238. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10975-6_17

19. Shantanu, Janet, B., Joshua Arul Kumar, R.: Malicious URL detection: a comparative study. In: International Conference on Artificial Intelligence and Smart Systems (ICAIS) (2021)
20. Johnson, C., Khadka, B., Basnet, R.B., Doleck, T.: Towards detecting and classifying malicious URLs using deep learning. J. Wirel. Mob. Netw. Ubiquit. Comput. Dependable Appl. JoWUA **11**, 31–48 (2020)
21. Romagna, M., van den Hout, N.: Hacktivism and website defacement: motivation, capabilities and potential threats. In: 27th Virus Bulletin International Conference, Madrid, Spain (2017)
22. Verma, R., Das, A.: What's in a URL: fast feature extraction and malicious URL detection. In: 3rd ACM on International Workshop on Security and Privacy Analytics (IWSPA 2017), Scottsdale, Arizona USA, pp. 55–63 (2017)
23. University of New Brunswick. https://www.unb.ca/cic/datasets/url-2016.html. Accessed 20 June 2022
24. Mamun, M.S.I., Rathore, M.A., Lashkari, A.H., Stakhanova, N., Ghorbani, A.A.: Detecting malicious URLs using lexical analysis. In: Chen, J., Piuri, V., Su, C., Yung, M. (eds.) NSS 2016. LNCS, vol. 9955, pp. 467–482. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46298-1_30
25. Dutta, A.K.: Detecting phishing websites using machine learning technique. PLOS One 1–17 (2021). https://doi.org/10.1371/journal.pone.0258361
26. https://www.upgrad.com/blog/random-forest-classifier/ . Accessed 18 Mar 2023
27. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/ . Accessed 18 Mar 2023
28. https://www.rebellionresearch.com/what-are-the-advantages-and-disadvantages-of-random-forest . Accessed 18 Mar 2023
29. https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article . Accessed 18 Mar 2023
30. http://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of-svm.html . Accessed 18 Mar 2023
31. https://towardsdatascience.com/everything-about-svm-classification-above-and-beyond-cc665bfd993e . Accessed 18 Mar 2023
32. https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6 . Accessed 18 Mar 2023