STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVIII**, Number 1, 2023
DOI: 10.24193/subbi.2023.1.02

# MALICIOUS WEB LINKS DETECTION - A COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS

COSTE CLAUDIA-IOANA

Abstract. One of the most challenging categories of threats circulating into the online world is social engineering, with malicious web links, fake news, clickbait, and other tactics. Malware URLs are extremely dangerous because they represent the main propagating vector for web malware. Malicious web links detection is a challenging task because the detection mechanism should not influence the consumers' online experience. The proposed solutions must be sensitive enough, and fast enough to perform the detection mechanism before the user accesses the link and downloads its content.

Our paper proposes three goals. The main purpose of this paper is to refine a methodology for malicious web links detection that may be used to experiment with machine learning algorithms. Moreover, we propose to use this methodology for training and comparing several machine learning algorithms such as Random Forest, Decision Tree, K-Nearest Neighbor. The results are compared, justified, and placed in the malicious web links literature. In addition, we propose to identify the most relevant features and draw some observations about them.

## 1. Introduction

Starting with the early 2000, most services: media and news, education, public administration, shopping etc. have moved their content, and customers online. Now, almost every household with an Internet connection needs to surf the Internet to satisfy its basic needs. Thus, consumers are more susceptible to becoming victims of malicious links and web-malware in general. Malicious web links are used to trick the users into giving away personal information. Through malicious links, consumers may be compelled to give access

Received by the editors: 1 March 2023.

2010 *Mathematics Subject Classification.* 68T99, 68U99.

1998 *CR Categories and Descriptors.* 68T99 [**Artificial Intelligence**]: Applications and Expert Systems – *Experiments for Malicious Web Links Detection*; 68U99 [**Management of Computing and Information Systems**]: Security and Protection – *Malicious Web Links*.

*Key words and phrases.* malicious web links detection, web-malware, artificial intelligence.

to their computer's resources or to consume low-quality and fake web content, increasing the incomes of content providers with views, clicks, and page visits. According to Cofense [4], in 2021, 38% of all phishing emails analyzed contained a malicious link. In addition to that, shortened URLs have become a real threat, since they are difficult to be identified as malicious [5]. Moreover, as stated by [4], 50% of credential phishing attacks were done using *.com* domains and 84% of all phishing sites use SSL and HTTPS protocol [2].

A web link represents a Uniform Resource Locator (URL), an identifier for a web page resource. A malicious link can be an attack vector for many types of threats such as: phishing attacks, cross-site request forgery (CSRF), cross-site scripting (XSS), drive-by-downloads, redirections without user's consent to cloned web pages etc. Most attacks target credentials theft, which may lead to important data breaches, and it may have financial advantages for the attackers.

Our identified problem in the domain of malicious web links is to detect malware or benign links. The problem is a binary classification with the URL as input and the class (malicious or benign) as output. The problem definition is mathematically defined as: $f : URLs \to \mathbb{R}^d$, f(url) = ($x_1$, $x_2$, $x_3$, ..., $x_d$), where d is the number of features and $x_d$, for each d $\in \mathbb{N}$ is a feature. We address the problem of malicious web links detection with three machine learning (ML) algorithms: K-Nearest Neighbor (KNN), Decision Tree (DT), and Random Forest (RF). The experiments made follow the next steps: parameters calibration and feature importance for DT and RF. We propose a methodology for refining the parameters values through experiments. Moreover, we would like to compare our models and analyze how they relate with other solutions presented in literature, especially with models proposed by Islam et al. [7], since it is using the same dataset. Another aim for the present paper is to investigate the feature importance in the case of DT and RF models.

The present article is structured in the following four sections. Malicious Web Links Detection presents the previous work done in our research niche. Proposed Methodology contains relevant details about the methodology we followed when driving the experiments. Results and Discussions has our metrics, comparisons and critical analysis on the experiments delivered. Conclusions and Future Work draws the final conclusions and discusses future directions of research.

## 2. Malicious Web Links Detection

Most previous work done in the malicious web links detection field can be split into two categories: dynamic and static. The dynamic approaches involve malicious code execution, and the static ones predict maliciousness of a link based on features, without code execution. In the execution-based category, there are included honey pots, sandbox-based systems and ML approaches that use features related to the code execution. Static detection systems contain blacklists solutions, signature-based ones, approaches based on complex networks and ML solutions taking into consideration static features.

The features used for link classification can be split into blacklists features, host-based ones, lexical characteristics, and content-based [14]. Blacklists features are extracted from public blacklists and lists with trusted domains. Host-based characteristics are usually extracted through web crawling and are related to IP address, port, protocol, HTTP/HTTPS headers, WHOIS and DNS information, geo-location etc. Lexical features involve traditional lexical properties (e.g., Bag-of-Words, N-grams) [14]. Moreover, lexical features may also involve the number of different special characters, number of words, URL length, query parameters number, domain name, etc. Content-based features are formed by characteristics extracted from the web page content, mostly code: HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript. In addition, content-based features include visual features, used for catching consumers' attention and metadata characteristics used for search engine optimization.

There are more problem types when discussing malicious web links detection. Some approaches consider a binary classification (most approaches) and others a multi-class problem ([8, 16]). There are articles ([10], [11]) testing the proposed models across multiple datasets to prove their adaptability.

Regarding multi-class solutions, Johnson et al. [8] experiments with RF, DT, KNN, Support Vector Machine (SVM), Logistic Regression (LR), etc., and two deep learning models developed with Fast.ai and TensorFlow-Keras. Best results for both multi-class and binary-class prediction are obtained by RF and the two deep models. Finally, RF is seen as the most suitable option since it does not require many computational resources. Tung et al. [16] is solving the multi-class problem as well, having four classes: benign, spam, malware, and phishing. The classification step is done using DT and RF models. By comparison, the RF algorithm outperforms the DT model with an accuracy of 97.49% for each class. They concentrate their effort in the feature selection process, where the solutions prove an improvement when adding three host-based features.

The classification done by Oshingbesan et al. [11] is binary and it is an experiment across multiple different datasets with multiple artificial intelligence algorithms (SVM, DT, RF, KNN etc.). According to [11], KNN is the most robust model that achieved a high performance in a cross-dataset environment. Similarly, Naveen et al. [10] is implying usage of tree-based models (DT, RF) and other algorithms (LR, Linear SVM, KNN etc.) for experiments across multiple datasets. The model that distinguishes its performance is KNN.

Taking into consideration binary classification with multiple ML models, Shantanu et al. [15] is using many ML models for experiments: SVM, DT, RF, KNN, LR, Naive Bayes (NB), and Stochastic Gradient Descent. The best metrics are obtained by RF. Similarly, Ibrahim et al. [6] analyzes malicious websites that deliver drive-by downloads attacks, using NB, JRip and J48. In the same direction of research, Catak et al. [3] develops two models, a RF with default parameters and a Gradient Boosting classifier. The best accuracy of 98.6% is obtained by RF, which proves to be faster than its counterpart algorithm. Implying multiple ML algorithms, Pakhare et al. [12] is using: LR, SVM with linear, RBF, and sigmoid kernels, KNN, RT and DT. But besides this, there are proposed ensembles formed out of these algorithms. The best performance is achieved by the ensemble formed with DT, KNN, and SVM. Islam et al. [7] is proposing solutions on the dataset found in [17] with the following ML algorithms: KNN, DT, RF and Multilayer Perceptron. They achieved 90% F1 score for KNN model, 83% for Neural Net and 99% for DT and RF model.

## 3. Methodology

Our research purpose is to propose an experimental methodology for approaching the malicious web links detection problem. This methodology will be exemplified with three ML algorithms: KNN, DT and RF. We aim to calibrate their parameters, accordingly, compare them and analyze the feature importance obtained. The methodology contains the next steps: dataset selection, pre-processing, counteracting the imbalancement problem, the classification step, which includes in total three phases of experiments for parameter calibration and feature importance. The methodology will be detailed in the next paragraphs.

The dataset we selected is a free available dataset [17], having 1781 records (216 malicious, 1565 benign). We propose to first experiment with little data such that we could easily get an insight on how we should approach this malicious links classification problem, without involving many computation resources. Moreover, the dataset has 17 already extracted attributes: lexical (URL length, number of special characters) and host-based ones (WHOIS & DNS information, content length, charset, server, number of ports open on the server, TCP packets count, number of bytes transported over the network, number of IPs connected to the server etc.).

Next step was preprocessing the data. Firstly, we cleared the records from missing or not defined values. Categorical features were transformed into numbers using supervised ratio algorithm (total number of samples with the category present in the positive class divided by the total number of records [7]) and weight of evidence algorithm (equation 1 [7]). The positive class is annotated with 1, indicating the malicious records, while the negative one is labeled with 0.

$$
\begin{aligned}
&X_{new} = \ln \frac{\frac{P_i}{TP}}{\frac{N_i}{TN}}; where\\
&P_i \; - \; number\ of\ records\ with\ positive\ class\ value\ for\ the\\
&categorical\ attribute\ value\ in\ dataset;\\
&N_i \; - \; number\ of\ records\ with\ negative\ class\ value\ for\ the\\
&categorical\ attribute\ value;\\
&TP \; - \; total\ number\ of\ records\ with\ positive\ class\ value;\\
&TN \; - \; total\ number\ of\ records\ with\ negative\ class\ value.
\end{aligned}
\tag{1}
$$

As the final preprocessing step, data normalization was done using two implementations: the Min-Max and Standard Scaler from Sklearn library [13]. The Min-Max Scaler normalizes the data on feature range (0,1). The Standard Scalar is transforming data based on the difference between data and the mean of data divided by the standard deviation [13]. Even though, Islam et al. [7] is not mentioning a normalization step, we chose to add it since we consider it to be important to balance the values in our dataset. For instance, timestamps for datetime features are represented by a long number, while the URL length is represented with a number between 16 and 33.

Since the data of the dataset is unbalanced, the problem is approached by using specific metrics: ROC-AUC score, precision, recall and F1 score. Moreover, when splitting the data into training and testing sets, it was parameterized with the *stratify* argument to keep the initial ratio between the two classes. In addition to this, the *class weight* parameter was used for DT and RF models. This parameter takes into consideration the imbalancement of the data when executing the algorithm and making a node split.

The algorithms we have chosen for this experimental research were KNN, DT and RF. We considered these algorithms because they are efficient, and do not require a lot of configuration and training time. Moreover, these algorithms are often used in other literature articles proposing solutions for malicious web links detection.

## 3.1. **K-Nearest Neighbor.**

KNN is a statistical algorithm, used in [7], where it achieves an F1 score of 90%. Shantanu et al. [15] is using KNN along other ML algorithms, yet all models are outperformed by RF. Johnson et al. [8] is experimenting with multiple algorithms, including KNN, which scores 97.47 accuracy for the binary classification. In [11] and [10], KNN proves to be the most flexible and adaptive model across multiple datasets. Pakhare et al. [12] uses KNN in the best performing ensemble model, including SVM and DT. This ensemble is the one outperforming the others with an accuracy of 94.93%.

3.2. **Decision Tree.** A DT model is a directed connected acyclic graph with a root node, child nodes and leaves. In [7], DT achieves a 99% accuracy, being one of their best models. Also, DT is used in the best ensemble model with SVM and KNN in [12]. Johnson et al. [8] experiments with DT for binary classification and achieves an accuracy of 97.63. For Oshingbesan et al. [11] and Naveen et al. [10], DT performs well for classifying links based on lexical features but is not the best performing model. Tung et al. [16] achieves a multi-class accuracy of 96% for the DT model.

3.3. **Random Forest.** RF is an ensemble learning method characterized by a voting system and formed with multiple Decision Trees. For Islam et al. [7], RF is one of the best models deployed, considering their experiments. Adas et al. [1] employs a Decision Forest reaching 99.8% accuracy. Similarly, as in the DT case, RF model achieves good accuracy for lexical features according to Naveen et al. [10]. In [8], RF reaches a performance of 98.68 accuracy for the binary classification case and outperforms more complex models. RF's multi-class accuracy is 96.26, being close to the accuracy of the fast.ai model. In the experiments done by Pakhare et al. [12], RF has the worst accuracy of 87.34%. Tung et al. [16] is deploying a RF model, with 97.49% accuracy, surpassing its comparing algorithm (DT). Catak et al. [3] get its best performance with a RF (98.6% accuracy).

The implementations of the ML algorithms (KNN, DT, RF) used in our experiments were from Sklearn Python Library [13], version 1.0.2. For running the experiments, we used Jupyter notebooks with Google Colab and the Python version was 3.7.15.

The experiments done followed three stages. The first and second stages of experiments consisted of calibrating the parameters for all algorithms. The values used for parameters were chosen based on a preliminary set of experiments (first stage). These experiments were made using as a starting point the configuration provided by Islam et al. [7], then we varied each parameter at a time. The resulting configuration was sorted based on an average metric computed as the average mean of all metrics (precision, recall, F1 score and ROC AUC score * 100). We chose the best 20 configurations and extracted from them the parameters and values relevant. Next, for the selected parameters and values we tried all the combinations possible (second stage of experiments).

| | **Model by [7]** | **Preliminary stage** | **Second stage** |
|---|---|---|---|
| **scaler** | not mentioned | Min-Max & Standard | Min-Max & Standard |
| **n neighbors** | 5 (default) | {1, 2, ..9} ∪ {10, 20, . . . , 90} | {1, 2, .., 10 } |
| **weights** | uniform (default) | {uniform, distance} | {distance} |
| **algorithm** | auto (default) | {brute, ball tree, kd tree, auto} | {ball tree, kd tree, auto} |
| **leaf size** | 30 | {1, 6, . . . 16} ∪ {20, 39} ∪ {40, 50, . . . , 90} | {1,3, .., 13} ∪ {20, 22, .., 34} ∪ {43, 44, 45, 46} ∪ {50, 51, .., 56} ∪ {65, 66, .., 69} ∪ {85, 86, .., 91} |
| **p** | 2 | {1, 2, . . . , 10} (with metric = minkowski) | {1, 2, . . . , 10} (with metric = minkowski) |
| **metric** | minkowski | {euclidean, chebyshev, manhattan, minkowski } | {manhattan, euclidean, minkowski} |
| | | **Configurations: 376** | **Configurations: 68400** |

Table 1. KNN algorithm with the parameters we calibrated and their values

| | **Model by [7]** | **Preliminary stage** | **Second stage** |
|---|---|---|---|
| **scaler** | not mentioned | Min-Max & Standard | Min-Max & Standard |
| **criterion** | gini | {gini, entropy} | {gini, entropy} |
| **min samples leaf** | 32 | {1, 2, . . . 99} | {1, 2, .., 20} ∪ {30, 31, . . . , 35} |
| **min samples split** | 2 (default) | {2, 22, 42, . . . , 1762} | {2,3, .., 10} |
| **max depth** | 12 | {1, 2, . . . 99} | {5, 6, .., 15} |
| **max leaf nodes** | None (default) | {2, 4, . . . , 98} | {18, 20, 22, 24} |
| **max features** | None (default) | {1, 2, . . . , 19} ∪ {log2, sqrt, auto} | None (default) ∪ sqrt |
| **min weight fraction leaf** | 0.0 (default) | {0, 0.1, 0.2, 0.25, 0.3, 0.4, 0.5} | 0.0 (default) |
| **min impurity decrease** | 0.0 (default) | {0, 20, 40, . . . , 1780} | 0.0 (default) |
| **ccp alpha** | 0.0 (default) | {0.0, 0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 1} | 0.0 (default) |
| | | **Configurations: 1882** | **Configurations: 82368** |

Table 2. DT algorithm with the parameters we calibrated and their values

In the tables 1, 2 and 3, there are aggregated all parameters for each algorithm and the values chosen for both stages of the experiments. Finally, the best configuration was chosen out of the top 20 configurations resulting from the second stage of experiments, which were run multiple times. The scores for the best configuration are computed as an average of all execution scores.

The last stage of our experiments was about analyzing feature importance in the case of the DT or RF model. The feature importance was computed on the best model with Mean Decrease in Impurity (MDI) and Mean Decrease Accuracy (MDA) formulas, both being integrated in Sklearn library [13]. We proposed to analyze the feature importance such that we could give relevant

| | **Model by [7]** | **Preliminary stage** | **Second stage** |
|---|---|---|---|
| **scaler** | not mentioned | Min-Max & Standard | Min-Max & Standard |
| **n estimators** | 100 | {1, 11, . . . , 71} ∪ {80, 85, . . . , 115} ∪ {120, 130, . . . , 240} ∪ {250, 275, . . . , 475} ∪ {500, 550, . . . , 950} | {50, 51, 90, 100, 110} |
| **criterion** | gini | {gini, entropy} | {gini, entropy} |
| **min samples leaf** | 32 | {1, 6, . . . 96} | {1, 3, 5} ∪ {11, 13, . . . , 21} ∪ {32} |
| **min samples split** | 2 (default) | {2, 3, 23, 43, . . . , 1763} | {2, 4, .., 10} |
| **max depth** | 12 | {None} ∪ {1, 2, . . . , 15} ∪ {16, 18, . . . , 98} | {10,11, . . . , 15} ∪ {50, 51, . . . , 54} |
| **max leaf nodes** | None (default) | {2, 22, 42, . . . , 1762} | {None, 680, 681, 682, 683, 684, 1560, 1561, 1562, 1563, 1564} |
| **max features** | sqrt (default) | {1, 3, . . . , 19} ∪ {log2, sqrt} | sqrt (default) |
| **min weight fraction leaf** | 0.0 (default) | {0, 0.1, 0.2, 0.25, 0.3, 0.4, 0.5} | 0.0 (default) |
| **min impurity decrease** | 0.0 (default) | {0, 20, 40, . . . , 1780} | 0.0 (default) |
| **bootstrap** | True (default) | {true, false} | True (default) |
| **oob score** | False (default) | {true, false} | False (default) |
| **ccp alpha** | 0.0 (default) | {0.0, 0.1, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.7, 0.75, 0.8, 0.9, 1} | 0.0 (default) |
| **max samples** | None (default) | {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1} | None (default) |
| | | **Configurations: 1816** | **Configurations: 99000** |

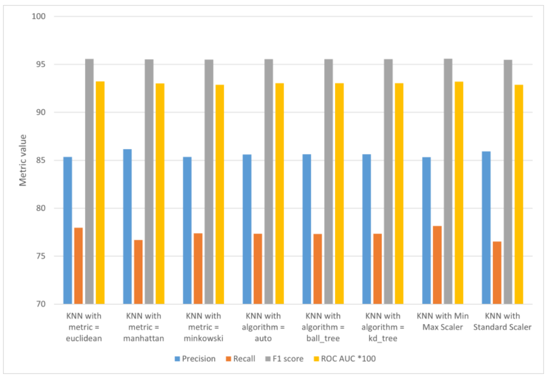
Table 3. RF algorithm with the parameters we calibrated and their values

and interesting tips based on how to spot the malicious link and explain how the tree models work in links classification.

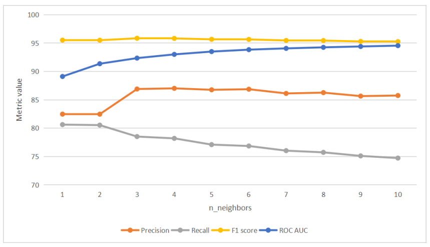
## 4. Results and discussions

In the current section, we will present the results for all our experiments described in Section 3. The first stage of experiments was the preliminary phase. The second stage of experiments was represented by the parameter's calibration. Both phases were carried out for all three algorithms. In the next paragraphs, we will present and analyze the results obtained for each algorithm. For the preliminary stage we ran 376 configurations for KNN, 1882 for DT and 1816 for RF. Based on the results obtained from these experiments we chose the interval values for relevant parameters in our model. The values and parameters were presented in detail in Tables 1, 2 and 3.

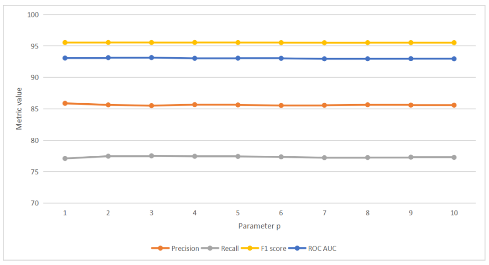
4.1. **Results obtained for KNN.** KNN achieved the best on average performance (see Table 4). The best configuration of KNN was *weights = distance, metric = Euclidean, algorithm = ball tree, n neighbors = 3, leaf size= 86,* *scaler* = *Min-Max*. Parameter calibration for KNN is presented in Figures 1. As observed for DT and RF as well, there was little to no difference between the normalization methods (Figure 1a). From some of the graphics associated with the calibration of parameters it cannot be determined the best value for our model (Figures 1d, 1a, and 1c). Best KNN had the distance value for *weights*, representing that closer neighbors have a higher influence on the classification task. The Euclidean metric got the best recall and F1 score, while the Manhattan metric had the highest precision and Minkowski reached the best ROC AUC score as can be observed in Figure 1a. In the case of the Euclidean metric, the value of the $p$ parameter is not taken into consideration. Ball tree algorithm got the best precision, but the difference to the other algorithm was not significant. On the other hand, the *n neighbors* value could be deducted from the Figure 1b, value 3 achieving the highest precision. *Leaf size* is relevant for the algorithm ball tree chosen, but its optimal value is hard to be observed in Figure 1d.



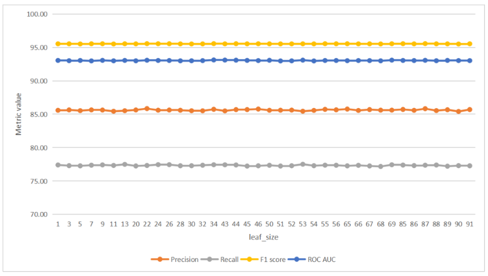
(A) Calibrating algorithm & metric & scaler



(B) Calibrating n neighbors



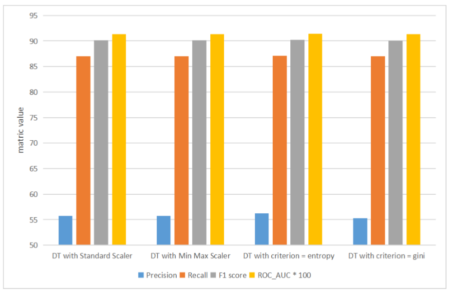
(C) Calibrating p



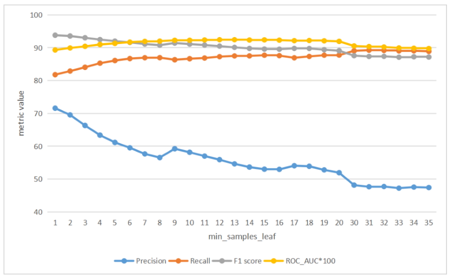
(D) Calibrating leaf size

Figure 1. KNN - calibrating parameters

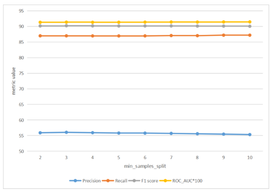
4.2. **Results obtained for DT.** The results for DT confirm that there was little to no difference in using the Min-Max or the Standard Scaler for normalization, as seen in Figure 2a. Considering the *criterion*, in Figure 2a it can be observed that entropy was the better option. In figures 2d, 2c and 2e the problem of choosing the best parameter is quite difficult, not much of a difference between each value. On the contrary, Figure 2b is presenting the best option to be chosen as parameter, the precision score being very sensitive to the variations of *min samples leaf* parameter, where the optimum value was 1. The graphics obtained are strengthened by the selection of the best DT model configuration. This configuration involved: *criterion = entropy, min samples leaf = 1, max depth = 8, max leaf nodes = 20, min samples split = 7, scaler = Standard, class weight = balanced, max features = None.*



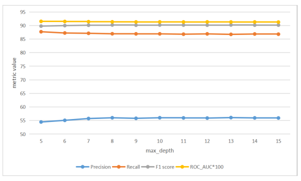
(A) Calibrating scaler & criterion



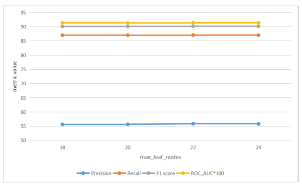
(B) Calibrating min samples leaf



(C) Calibrating min samples split



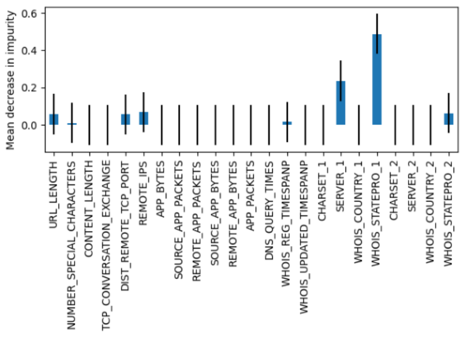
(D) Calibrating max depth



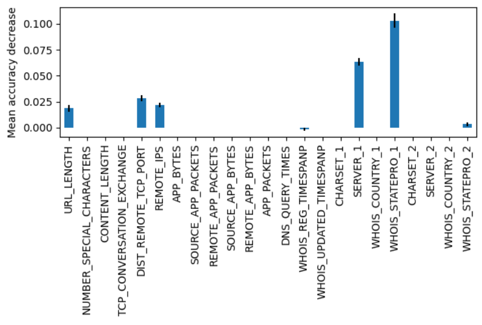
(E) Calibrating max leaf nodes

FIGURE 2. DT - calibrating parameters

Considering feature importance analysis, the results for the DT model are presented in Figure 3a for MDI and in Figure 3b for MDA. Features suffixed with "_1" are categorical features preprocessed with the supervised ratio algorithm. The ones suffixed with "_2" are categorical features transformed with weight of evidence algorithm (Equation 1). Taking into consideration the results, we observed a high relevancy for host-based features such as: *SERVER* and *WHOIS STATEPRO*. *SERVER* refers to the operation system running on the Web Server. *WHOIS STATEPRO* is represented by the state (approximate geo-location) from where the server responded to the request. Thus, malicious link attacks might be a geographically segregated attack. *SERVER* feature indicates importance since an outdated operating system can be more vulnerable to threats. It can be observed that categorical features processed with the supervised ratio algorithm reached a higher degree of importance. From the lexical features, *URL LENGTH* was the most important one, its score being comparable with other host-based characteristics: *DIST REMOTE TCP PORT* and *REMOTE IPS*. These last two attributes refer to the number of ports detected to be opened on the server (excluding the current connection port when data was collected) and respectively, the number of IPs addresses connected to the web server. These two attributes were relevant since they show a high activity on the server.



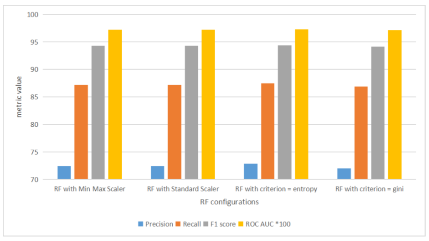
(A) Mean Decrease in Impurity (MDI)



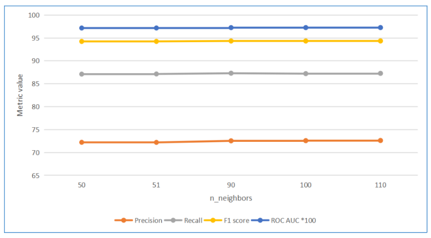
(B) Mean Decrease Accuracy (MDA)

FIGURE 3. DT - feature importance

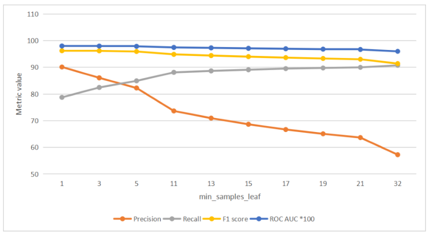
4.3. **Results obtained for RF.** RF on average managed to improve performance compared to DT (Table 4). The best calibration of RF model was: *criterion = entropy, n estimators = 110, max depth=53, min samples leaf=3, max leaf nodes=1564, min samples split=4, class weight=balanced, scaler = Min-Max*; which outperformed the others. The best model configuration was confirmed by the graphics presenting parameters calibration, even though in the cases of *max depth*, *max leaf nodes*, *min samples split* and *n estimators* as can be observed from Figures 4e, 4f, 4d and respectively Figure 4b, there was little to no difference between the parameters values. Considering the *criterion* used, entropy was the value scoring the highest value along all metrics as seen in Figure 4a. Entropy was the value used in the best model calibration as well. Considering the *scaler* used for the normalization step, just the same conclusion here as for KNN model or DT model, there was almost no difference between them. Still, the best performance was achieved using Min Max scaler, which scored a higher precision and F1 score than its counterpart. Regarding the calibration of *min samples leaf* parameter, as can be seen quite easily in Figure 4c the value, which significantly increased the recall is 3, this one being used in the best model configuration.



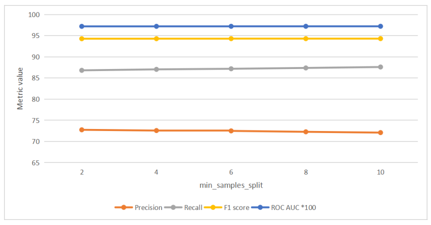
(A) Calibrating scaler & criterion



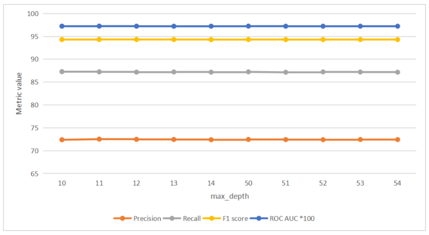
(B) Calibrating n estimators



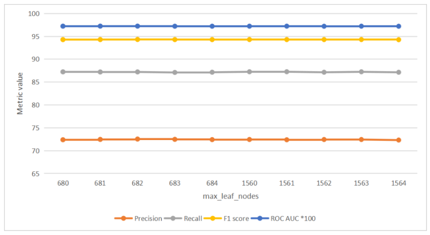
(C) Calibrating min samples leaf



(D) Calibrating min samples split



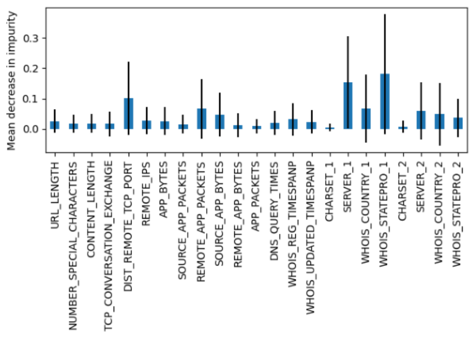
(E) Calibrating max depth



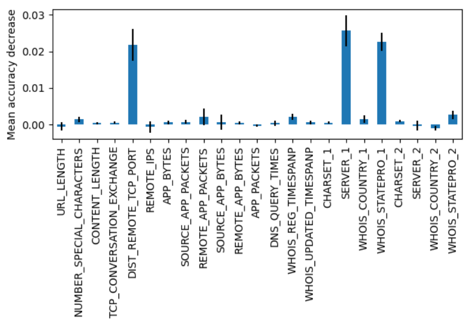
(F) Calibrating max leaf nodes

Figure 4. RF - calibrating parameters

The scores of the features were computed on the best RF model configuration. Results for the RF model showed us that the importance score is more distributed along more features, unlike the DT model case. Feature scores were computed based on the MDI (Figure 5a) and MDA (Figure 5b). Similarly, as in the case of the DT model, categorical features were transformed in numbers using the supervised ratio algorithm and the weight of evidence algorithm. Best features were *SERVER*, *WHOIS STATEPRO*, *DIST REMOTE TCP PORT*, all of them being host-based features. *SERVER* represents the operating system of the web server serving the HTTP request, which is relevant to predict the vulnerability of an outdated system. *WHOIS STATEPRO* refers to the approximate geo-location of the server and it was expected to be relevant since most cyber-attack are geographically segregated. *DIST REMOTE TCP PORT* counts the open ports waiting for a connection on the web server and it shows a high activity on the server, maybe even serving multiple web pages. From the lexical features, *URL LENGTH* was the attribute achieving the highest score.



(A) Mean Decrease in Impurity (MDI)



(B) Mean Decrease Accuracy(MDA)

FIGURE 5. RF - computing feature importance

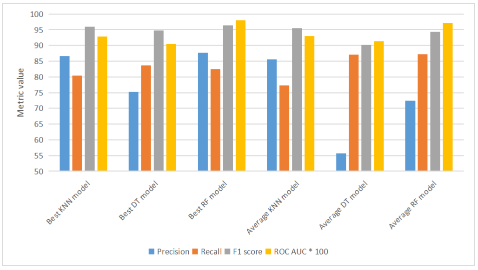
| | Precision | Recall | F1 score | ROC AUC * 100 |
|---|---|---|---|---|
| KNN on average | 85.62 | 77.33 | 95.53 | 93.03 |
| Best KNN | 86.6 | 80.43 | 96.02 | 92.78 |
| DT on average | 55.73 | 87.01 | 90.11 | 91.35 |
| Best DT | 75.23 | 83.62 | 94.7 | 90.51 |
| RF on average | 72.43 | **87.19** | 94.29 | 97.21 |
| Best RF | **87.6** | 82.55 | **96.39** | **98.08** |

TABLE 4. Results for KNN, DT and RF models

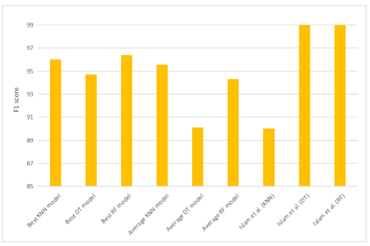
4.4. **Discussions.** From the experiments done and the statistics made, we observed that precision is a highly sensitive metric to parameters' calibration. We considered it to be highly relevant for our binary classification because it was computed on the malicious class. By comparing our models, their average and best performance as presented in Figure 6a and Table 4 it can be observed that the highest precision score was obtained for the best RF model. Moreover, this model managed to get the highest F1 score and ROC AUC score. The second most precise model was the best KNN model, having comparable results with the best RF model. On average, KNN performed best having the highest precision and F1 score across the average models. Unfortunately, the DT model performed the worst, on average achieving just five percent above a random baseline model.

When comparing our results with the performance denoted by Islam et al. [7] models, we managed to significantly improve the KNN model. In the experiments from [7], KNN gets 90% F1 score, while our best KNN model had 96.02% F1 score and on average 95.53% F1 score. For the case of DT and RF models, their approach has a better performance of 99% F1 score. Our DT model had a 94.7% F1 score at best and RF reached 96.39% at best. Still, we consider that our results were comparable with their experiments.

To our perspective, we manage to improve the methodology needed for running experiments in this benign/ malicious links classification field. Thus, the purpose of our paper was not to improve state-of-the-art detection algorithms for malicious links detection, but to try and perfect the strategy followed when finding solutions for this problem.



(A) All models comparison



(B) All models comparison with Islam et al. models [7]

FIGURE 6. Comparing models

If we compare our approach to other literature solutions that experiment with the same algorithms, our results are outperformed by theirs. This is the case in [15], where the authors deliver experiments on multiple ML algorithms, such as KNN (96.2% precision), DT (99% precision), RF (99.8% precision) etc. Reaching a higher score than ours may be because of using a 450,000 records dataset, with the URL and the label. Having a larger dataset can be helpful in properly fitting the models. Shantanu et al. [15] manually engineers lexical features from the URL, while we mostly used host-based features and two lexical ones, which are as well included in their model. Another disparity comes from our dealing with data imbalancement, while in [15] there are no mentions about the actions taken to counteract it. Moreover, we computed the metrics as an average of multiple runs. Comparing our approach on the RF model with the Decision Forest proposed by Adas et al. [1] (99.8% accuracy), the dissimilarity comes from using another methodology and dataset. In [1], the dataset has over 2.4 million URLs and the ratio between malicious and benign samples is not mentioned. Besides that, the features used in classification are different and their training step includes a cross validation step, which may be very effective since their dataset is numerous.

Comparing the last step of the experiments, about feature importance, we managed to confirm the results within other articles. The experiments done in [11] concludes that no particular features dominated the detection algorithm. On the contrary, Johnson et al. [8] manage to prove the relevance of URL related attributes through chi-squared test, as our lexical URL features achieved a medium accuracy score. As well, in [9] the relevant attributes are URL-based, while the host-based ones (WHOIS information) are not as relevant. This is in contrast with our results that proved a higher score for host-based characteristics in general. The relevance computed with information gain in [6] denotes that the length of the URL is a top feature together with the count of dots, which is part of our special characters count attribute.

## 5. Conclusions and future work

In conclusion, malicious web links detection is a complex domain from an experimental point of view. There was almost no difference between the two normalization techniques we applied. Our main contribution is that we managed to propose an experimental methodology for malicious web links detection. Moreover, we got to improve the score metric of KNN model compared to other literature solutions and the RF model achieved the best precision (87.6%). Regarding feature importance analysis, we observed a high score for host-based features. Considering future work, we propose experimenting with more simple and complex ML algorithms on the same dataset such that we could draw a more relevant and complete overview from our experiments. Moreover, we plan to develop a real-time framework that will help users report malicious links. The collected samples of links can be further analyzed, other features can be extracted, especially the host-based ones.

## References

[1] Adas, H., Shetty, S., and Tayib, W. Scalable detection of web malware on smartphones. In *2015 international conference on information and communication technology research (ICTRC)* (Abu Dhabi, UAE, 2015), IEEE, IEEE, pp. 198–201.

[2] APWG. Phishing activity trends report - q4, 2020. Tech. rep., APWG, USA, 2021.

[3] Catak, F. O., Sahinbas, K., and Dörtkardeş, V. Malicious url detection using machine learning. In *Artificial intelligence paradigms for smart cyber-physical systems*. IGI Global, Papua New Guinea, Turkey, 2021, pp. 160–180.

[4] Cofense. Annual state of phishing report. Tech. rep., Cofense, Leesburg, VA, USA, 2021.

[5] Cook, S. Phishing statistics and facts for 2019–2022, Oct 2022.

[6] Ibrahim, S., Herami, N. A., Naqbi, E. A., and Aldwairi, M. Detection and analysis of drive-by downloads and malicious websites. In *International Symposium on Security in Computing and Communication* (Trivandrum, India, 2019), Springer, Springer, pp. 72–86.

[7] Islam, M., Poudyal, S., Gupta, K. D., et al. Mapreduce implementation for malicious websites classification. *International Journal of Network Security & Its Applications (IJNSA) Vol 11* (2019).

[8] Johnson, C., Khadka, B., Basnet, R. B., and Doleck, T. Towards detecting and classifying malicious urls using deep learning. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl. 11*, 4 (2020), 31–48.

[9] Kumi, S., Lim, C., and Lee, S.-G. Malicious url detection based on associative classification. *Entropy 23*, 2 (2021), 182.

[10] Naveen, I. N. V. D., Manamohana, K., and Verma, R. Detection of malicious urls using machine learning techniques. *International Journal of Innovative Technology and Exploring Engineering 8*, 4S2 (2019), 389–393.

[11] Oshingbesan, A., Okobi, C., Ekoh, C., Richard, K., and Munezero, A. Detection of malicious websites using machine learning techniques. *preprint none*, none (06 2021), 1–5.

[12] Pakhare, P. S., Krishnan, S., and Charniya, N. N. Malicious url detection using machine learning and ensemble modeling. In *Computer Networks, Big Data and IoT*. Springer, Singapore, 2021, pp. 839–850.

[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[14] Sahoo, D., Liu, C., and Hoi, S. C. Malicious url detection using machine learning: A survey. *arXiv preprint arXiv:1701.07179* (2017).

[15] Shantanu, Janet, B., and Kumar, R. J. A. Malicious url detection: A comparative study. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (Tamil Nadu, India, 2021), IEEE, IEEE, pp. 1147–1151.

[16] Tung, S. P., Wong, K. Y., Kuzminykh, I., Bakhshi, T., and Ghita, B. Using a machine learning model for malicious url type detection. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems* (Cham, 2022), Y. Koucheryavy, S. Balandin, and S. Andreev, Eds., Springer International Publishing, pp. 493–505.

[17] Urcuqui, C., Navarro, A., Osorio, J., and García, M. Machine learning classifiers to detect malicious websites. *SSN 1950* (2017), 14–17.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeș-Bolyai University, Cluj-Napoca, Romania
*Email address*: `claudia.coste@ubbcluj.ro`