Journal of Information Security and Applications 62 (2021) 102967

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

SeizeMaliciousURL: A novel learning approach to detect malicious URLs

Dipankar Kumar Mondal [a], Bikash Chandra Singh [a,b,*], Haibo Hu [b], Shivazi Biswas [a], Zulfikar Alom [c], Mohammad Abdul Azim [c]

[a] Department of Information and Communication Technology, Islamic University, Kushtia, Bangladesh
[b] Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong Special Administrative Region
[c] Department of Computer Science, Asian University for Women, Chattogram, Bangladesh

ARTICLE INFO

Keywords:
Malicious URLs detection
Machine learning
Classification

ABSTRACT

Malicious websites are increasing in abundance, which brings serious web security threats to users. As a result, individuals lose their assets, values, information, etc. to unauthorized parties and become victims while visiting such websites. The research community put efforts into developing effective and efficient models for detecting malicious URLs to make available notifications about websites that users access. Perceiving this, numerous methods make use of various types of machine learning (ML) approaches. However, until now, no technique has perfectly detected malicious URLs, as they are susceptible to false positive and false negative decisions in classifying URLs in malicious and non-malicious groups. To improve this issue, this article proposes a new approach based on a machine learning technique. The proposed approach uses multiple classifiers (i.e., ensemble learning) to predict the class probabilities of URLs and then applies a threshold to filter the decisions of multiple classifiers. Next, the decisions are combined concerning its corresponding class probabilities and find the class label with the highest class probability as the final decision on unlabeled URLs. The results show that the proposed method offers better performance in terms of malicious URL detection than other methods.

1. Introduction

Nowadays, people use the Internet to simplify their daily activities. Hence, they are likely to access various websites using their URLs (Uniform Resource Locator) in obtaining and sharing information. A report shows that there are more than 1.5 billion websites today, and this number is increasing every second.[1] Importantly, all websites are not benign. This means that some of them are designed to harm users. More specifically, hackers have chosen this platform to conduct criminal activities (e.g., theft of users' private information, monitoring traffic transmitted between the client and the server, etc.) and making the users possible victims.

On the other hand, it is remarkable that the number of active Internet users is increasing day by day, reaching nearly 4.6 billion by January 2020.[2] There is no doubt that most users do not have the basic knowledge of accessing the Internet safely. More precisely, they have no idea to distinguish between malicious or benign websites. Therefore, there is a huge possibility of users are likely to become victims by visiting malicious websites. For example, suppose a fraudster *Eve* makes a copy of what appeared to be an original website, that is, a malicious website. A credulous user, *Alice* uses its URL to visit the website and provides her information on the website. In this case, *Alice* becomes a victim since she has no prior knowledge of recognizing malicious URLs.

Generally, malicious websites commit cybercrime by attracting indisputable users to visit the website and obtain private information. Therefore, it is a massive challenge to detect malicious URLs to notify users to avoid visiting such websites that may be harmful to them. Traditionally, the typical method to solve this problem is the blacklisting method [1] and Heuristic method [2,3]. However, the blacklisting method cannot be exhaustive and can detect URLs as malicious or benign only if the URLs exist in the database. In other words, it cannot detect newly generated malicious URLs. The Heuristic method is an extension of the blacklisting method, which can detect threats in new URLs. However, the limitation of this method is that it can only work against common threats rather than working with novel threats in the case of new URLs. To address these issues, researchers have used machine learning approaches to detect malicious URLs properly [4–6]. For example, in [7], the authors proposed linear and non-linear space transformation methods for feature selection and machine learning classifiers for detecting malicious URLs. The paper [4] reported the use

* Corresponding author at: Department of Information and Communication Technology, Islamic University, Kushtia, Bangladesh.
E-mail address: bikash.singh@ice.iu.ac.bd (B.C. Singh).

[1] https://www.internetlivestats.com/total-number-of-websites/
[2] https://www.statista.com/topics/1145/internet-usage-worldwide/

https://doi.org/10.1016/j.jisa.2021.102967

Available online 17 September 2021

of URL lexical analysis with machine learning techniques to classify malicious web pages. In [8], the authors use machine learning methods to categorize and detect malicious URLs based on the types of attacks.

The previous research does not use machine learning techniques extensively. In other words, they only focus on a limited number of existing machine learning methods and consider the less diversity in the dataset to prove the rationality of their proposed model. Moreover, these types of works mainly focus on extracting high-quality learning features rather than exploring appropriate machine learning algorithms to detect malicious URLs. However, feature engineering plays a vital role in improving malicious URL detection. Nevertheless, improving the performance of the learning classifiers is also an important task. Consequently, this article employs the vectorization technology for feature extraction to improve the performance of machine learning methods in detecting malicious URLs [9]. The effort simultaneously focuses on the improvement of the accuracy of the classifier.

Moreover, overfitting is a big issue when machine learning models are used in order to learn facts from new data. Usually, overfitting occurs when the model learns the details and noise in the training dataset, so it negatively affects the performance of the model in new data [10]. However, to address this issue, the article uses the concept of *hold back a validation dataset*. The idea is that a set of validation data is being retained from the machine learning algorithms, which is a subset of the training data. After selecting and adjusting a machine learning algorithm on the training dataset, the model can be evaluated and trained on the validation dataset to get a final objective idea of how the model could perform invisible data. It is found that the threshold value to assign the boundary of the decision of learning models whether the decision is malicious or non-malicious.

With this goal, $SeizeMaliciousURL$ is proposed that exploits the concept of ensemble learning to detect malicious URLs more efficiently and effectively. Generally, ensemble learning is the process of separately constructing multiple classifiers on the dataset and then combining the results of these classifiers to solve a particular problem. The significant fact is that these classifiers may produce probabilities for the labels (i.e., malicious or non-malicious) of URLs very close to each other. Therefore, it resembles that both labels are likely to be in the same class, i.e., these classifiers indicate a specific URL equally malicious and non-malicious, consequently drives ambiguity into the decision process. Intuitively in this specific confusing state, the ensemble learning classifiers result in prediction errors and performance degradation. To address this issue, this article proposes a condition-based elimination procedure in the $SeizeMaliciousURL$, where the confusing decisions are excluded. This condition is based on a threshold value of δ that filters the aforementioned ambiguous decisions. This threshold-based condition only allows the decisions unambiguous. Based on these ideas, the proposed model $SeizeMaliciousURL$ reduces the prediction errors of the classifiers to improve the performance of the soft voting-based ensemble learning approach predict the malicious and non-malicious URLs.

Therefore, this paper provides the following contributions:

- A novel architecture $SeizeMaliciousURL$ for detecting malicious URLs vs. non-malicious URLs is proposed. The design considers assigning a threshold value between the decisions produced for malicious URLs and non-malicious URLs by the ensemble classifiers.
- The model $SeizeMaliciousURL$ can minimize the prediction errors in detecting malicious URLs or non-malicious URLs generated by ensemble learning classifiers.
- This article compares the experimental results produced by the proposed model with the other traditional machine learning models and finds which model is more fitted for detecting malicious URL datasets.

The rest of the paper is structured as follows. Section 2 presents related work. Section 3 proposes an overall system architecture for malicious URLs detection. In contrast, Section 4 describes the learning approaches considered in this paper. Section 5 explains the dataset and performance metrics, Section 6 illustrates the experimental results, and Section 7 figures out the conclusion of this paper and shows the future work for this project.

2. Related work

Many approaches have been proposed and developed to detect malicious URLs. The majority of these approaches can be categorized into four types: (i) blacklists, (ii) content-based classification, (iii) URLs-based classification, and (iv) deep learning-based approach.

Whittaker et al. [11] developed a scalable machine learning classifier to detect phishing blacklist automatically. The trained model examines the URLs and contents of a page to determine whether or not a page is phishing. The authors claim that the proposed model is a perfect classifier and a robust system. However, the claim is arguable as the model only identifies a phishing page after being published and visible to Internet users. Likewise, Sahingoz et al. [12] presented a real-time anti-phishing detection system. The authors used seven different classification algorithms and natural language processing (NLP) methods to detect phishing URLs. Results demonstrate that the proposed model provides greater accuracy. The accuracy rate of 97.98% is achieved for the detection of phishing URLs. The study [13] demonstrates how a framework can be developed to detect blacklisted URLs (i.e., phishing websites). The authors intend to design a phishing detection system so that users can be alerted while browsing or accessing a particular website. Since the proposed system can be used for identification and authentication, it can be used as a legitimate tool to prevent naive users from getting tricked.

Zhang et al. [14] presented a content-based approach, called $CANTINA$, for detecting phishing websites. The proposed model takes Hyperlinks and used the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm to detect phishing websites. The authors present that the TF-IDF model can catch phishing sites with 6% false positives. In contrast, combining the TF-IDF and some simple heuristics features, the proposed model can catch phishing sites with only 1% false positives. Likewise, Jain et al. [15] proposed a content-based model named $PhishSKaPe$, to escape phishing attacks. The authors used two approaches, namely (i) conventional TF-IDF and (ii) weighted TF-IDF, to detect phishing attacks. The experimental results demonstrate that the weighted TF-IDF method performs better than conventional TF-IDF. Similarly, [16] demonstrates a model, named $CatchPhish$, can detect phishing websites. To detect phishing websites, $CatchPhish$ uses the hostname, URLs, TF-IDF features, and phish hinted words from suspicious URLs. Results show that the proposed model with only TF-IDF features achieved an accuracy of 93.25%. Contrarily, combining the TF-IDF and hand-crafted features achieved a significant accuracy of 94.26%, which is much better than the existing models.

Sameen et al. [17] proposed a detection system called $PhishHaven$ to identify machine-generated and human-crafted phishing URLs. The proposed model exploits lexical analysis for feature extraction. It uses an unbiased voting mechanism to avoid misclassification. Moreover, it employs a multi-threading approach to execute the classification in parallel. Therefore, it speeds up the model's performance, which leads to real-time detection. Similarly, J Ma et al. [18] proposed a light-weighted detecting approach for detecting malicious URLs using lexical and host-based features of the associated URLs. The authors compared the performance and claimed that their methods are better suited to URL classification. The results demonstrate that the proposed model gives 99% accuracy over a balanced data set.

In recent days, deep learning-based approaches have been used to detect malicious URLs (e.g., [19–22]). For instance, Vinayakumar et al. [19] proposed deep learning methods for detecting malicious

and phishing URLs. The proposed models use various deep learning architectures, specifically recurrent neural network (RNN), identity-recurrent neural network (I-RNN), long short-term memory (LSTM), convolution neural network (CNN), and convolutional neural network short-term memory (CNN-LSTM) architectures to phishing URL detection. The authors claim that the proposed deep learning architectures performed well in comparison to the conventional algorithms. Similarly, Bahnsen et al. [20] proposed a deep learning model that applied RNN and LSTM applied to detect phishing URLs. The authors show the performance comparison between the deep learning model with character-level embedding and conventional machine learning with feature engineering methods. The results show that both models perform well. The performance of LSTM was good in comparison to the conventional machine learning methods. Ref. [21] proposed a neural network, which uses a convolution neural network (CNN) with character level Keras embedding for detecting malicious URLs, file paths, and registry keys. Likewise, [22] proposed a model called *DeepURLDetect* (DUD) in which raw URLs are encoded using character level embedding. The authors evaluated the performance of various deep learning architectures for malicious URL detection. Likewise, numerous works exist that use different techniques, e.g., content-based, deep learning architectures, etc., for malicious URL detection.

In addition, researchers have developed many machine learning (ML) based methods for detecting malicious activities (i.e., attacks) in social media. For instance, Alom et al. [23] investigated the nature of malicious users on Twitter to improve existing spam detection mechanisms. In this paper, the authors proposed several new features, which are more effective and robust than existing used features to detect spammers. As an alternative to ML-based detection, [24] presented a new approach based on deep learning (DL) techniques to detect malicious activities in social media networks. Muna et al. [25] proposed a deep learning model for detecting malicious activities in the industrial Internet of Things (IoT). Mohammadi et al. [26] proposed a cyber intrusion detection model by combining different feature selection algorithms. Moreover, substantial research efforts (e.g., [27–30]) have also been made to suggest privacy preferences to users through the use of machine learning techniques and/or graph theories. Singh et al. [27] proposed a suite of semi-supervised approaches to learning the privacy aptitudes of Personal Data Storage owners. Likewise, [29] implemented context-based privacy preference to learn the privacy aptitudes of users. In [31], authors proposed a model that can ensure users' privacy and data security based on users' risk–benefit metrics.

Table 1 summarizes the aforesaid references in accordance with their methodologies. However, in this work, a new approach based on machine learning techniques is proposed. The proposed approach differs from all the proposals above as it is learning based on the class probabilities of URLs. Moreover, the proposed approach has used multiple classifiers (i.e., ensemble learning) that can be effectively used to predict the class probabilities of URLs.

Moreover, a threshold to filter the decisions of multiple classifiers is applied. The model combines the corresponding class probabilities and finds the class label with the highest class probability as the final decision on unlabeled URLs. The results demonstrate that the proposed model improves the performance in terms of detecting malicious URLs compared to the other existing methods (see Section 6).

3. Overall architecture

Fig. 1 shows the overall system architecture of the proposed model to detect malicious URLs. The dataset contains both (i) malicious and (ii) non-malicious URLs. Note that URLs are made up of text, and to use the text in a machine learning environment, data preprocessing is required.

In data preprocessing, first, the concept of the TF-IDF technique to extract the features from all URLs where TF stands for *Term Frequency*, and IDF stands for *Inverse Document Frequency* [9] is taken. In this technique, every word is counted, and a numerical value (corresponding to the word) is given according to the importance of the word in the document. The output TF-IDF description is the product of the TF and IDF values for each word. More precisely, TF is defined as the number of times a term or word appears in a given document, while IDF can be defined as follows [9]: $IDF(t) = \log((1+n)/(1+DF(t))) + 1$, where n is the number of documents in the document set, and DF(t) represents the number of documents in the document set containing words or terms t. Finally, the result of TF-IDF is normalized using the Euclidean norm.

Example 3.1. Let us suppose that there are three documents, D1 = "https://google.com", D2 = "http://www.facebook.com", D3 = "www.youtube.com". There are 4 words (i.e., *http*, *www*, *facebook*, *com*) in $D2$ and www appears once in $D2$. Therefore, the term frequency of www is TF(www) = 1/4 = 0.25. Now, the word www appears in 2 documents (i.e., $D2$, $D3$) in the document set (i.e., D1, D2, D3). Therefore, the inverse document frequency of www is IDF(www) = log((1 + 3)/(1 + 2)) + 1 = 1.20. TF-IDF is the product of these two quantities, that is, TF-IDF=TF(www) * IDF(www)= 0.25 * 1.20 = 0.30. Finally, the result for the document is normalized by using Euclidean norm. At this stage it can be computed the numerical value of all features from URLs in a similar fashion.

It is therefore important to note that every word in the URL separated by special characters such as {"/", "//", ",", ".", "@", ":", ...} is counted as a feature for that URL. For instance, if there is a URL "http://bankofamerica.com", it can extract three features [http, bankofamerica, com] from it.

Concisely, those above preprocessing, i.e., TF-IDF, provide us a mechanism to extract features from all URLs and convert these features into numerical values applicable to the machine learning methods.

In the second step, the entire dataset is divided into three parts: (i) the training dataset, (ii) the validation dataset, and (iii) the testing dataset. Utilizing the training dataset, the learning models are built to predict the possible labels (i.e., malicious URL or non-malicious URL) of the testing dataset. Moreover, the evaluation uses the validation dataset to compute the threshold value δ. It uses this value to filter the error-prone decisions of the soft ensemble to predict the class label of a testing dataset (see Section 4.3 for more details).

Note that previous studies exploit various learning models to perform these tasks [32–34]. Due to the large number of labeled items in the available dataset, the researcher commonly uses supervised learning methods. This article presents RF, DT, k-NN, NB, and SVM as supervised learning methods compared to the proposed classifier. When the classifier individually predicts the class labels on an instance, then it is called the single-view method (for more details, see Section 4.1). However, the single-view method cannot cover all aspects of learning to build a model in some application scenarios. In this context, the texts of URLs and extract various kinds of features are used. Thus, it is worthy to look at features with multiple classifiers and predicting class labels for unlabeled URLs. More specifically, multiple supervised classifiers are used to make multiple decisions on an item and consider a hard/soft (see Sections 4.2.1 and 4.2.2 for details) voting approach to make the final decision by combining these decisions.

However, the hard voting approach considers the predicted class labels of items in computation instead of considering the class probabilities generated by the classifiers. This approach, in general, is susceptible to false positive/negative decisions. Here, soft voting is in use to minimize this problem. The soft voting approach uses multiple classifiers to calculate the probability for each class (i.e., non-malicious/malicious). Then, all probabilities associated with a given class are summed, and the class with the highest probabilities is chosen as the final decision. However, a difficulty exists if some classifiers compute the probability for classes that are very close to each other (i.e., non-malicious/malicious). These cases are naturally prone to error

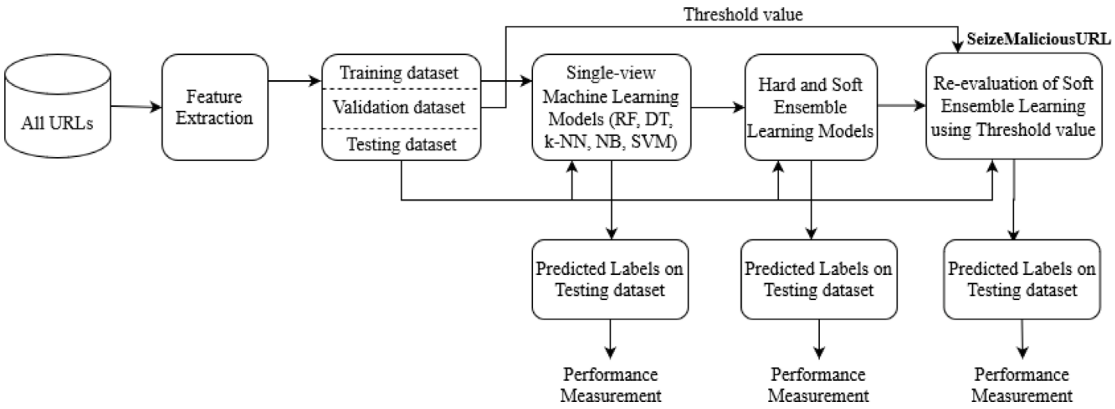

Fig. 1. Overall model architecture.

Table 1
Summary of the techniques used by the surveyed article.

Paper ID	Methodology
Whittaker et al. [11]	*Scalable Machine Learning*
Sahingoz et al. [12]	*Supervised Machine Learning*
Zhang et al. [14]	*Content – based Approach*
Jain et al. [15]	*Content – based Approach*
Sameen et al. [17]	*Ensemble Learning*
J Ma et al. [18]	*Machine Learning*
Vinayakumar et al. [19]	$I-RNN$, $LSTM$, $CNN-LSTM$
Bahnsen et al. [20]	CRF, $LSTM$ *Model*
Saxe et al. [21]	*Character level CNN*
V Rao et al. [16]	*Deep Neural Network*
Muna et al. [25]	*Deep Learning Model*
Singh et al. [27]	*Semi – Supervised Approach*
Alom et al. [29]	*Context – based Privacy Preferences*
The proposed approach	**Machine Learning, Hard and Soft Voting Based Ensemble Learning, Threshold-based Soft Voting Ensemble Learning**

Algorithm 1: Single-view Learning Algorithm

Input: L_{URL}, the set of labeled URLs in training dataset; U_{URL}, the set of unlabeled URLs in testing dataset
Output: 1 if a URL be malicious in U_{URL} ; 0 otherwise
1 Let $(x_1, x_2, x_3, \ldots.)$ be the features set of L_{URL} and U_{URL}
2 $Build\_Classifier = Classifiers(L_{URL}(x_1, x_2, x_3, \ldots))$
3 **for** $X \in U_{URL}$ **do**
4 $Predicted\_Probability\_For\_X = Build\_Classifier(X(x_1, x_2, x_3, \ldots))$
5 Let $P_X(Malicious)$ be the probability of X being predicted as a malicious URL
6 Let $P_X(Non-Malicious)$ be the probability of X being predicted as a non-malicious URL
7 **if** $P_X(Malicious) == P_X(Non-Malicious)$ **then**
8 Return: The label corresponding to $P_X(Malicious)$
9 **else**
10 Return: The label corresponding to $Max(P_X(Malicious), P_X(Non-Malicious))$

results degrading the performance of that particular machine learning method. To overcome the issue, this article proposes an alternative approach that imposes a threshold to filter the decisions of a soft voting-based ensemble learning approach that is likely to belong to the same class. More particularly, the proposed technique calculates the absolute value of the subtraction between the class probabilities and, if that value is greater than or equal to the threshold, the system considers the case as a decision; otherwise, not. By examining all the predicted decisions and adding up all the probabilities associated with the corresponding classes, the final decision is made based on the class associated with the highest probabilities. This approach is termed $SeizeMaliciousURL$ (see Section 4.3 for more details).

4. Learning approaches

4.1. Single-view learning approach

According to this approach, all of the features are used together to build a learning classifier. Notably, the method treats each URL as a single instance. It uses a set of labeled URLs (e.g., training dataset) to build a single classifier to predict the class of unlabeled URLs (e.g., testing dataset). However, it takes several supervised learning approaches (e.g., RF, DT, k-NN, NB, SVM) separately to build a single classifier.

Algorithm 1 illustrates the overall idea of the single-view learning process. Here, once the classifier is built (see line 2), it can predict the label of unlabeled URLs, denoted as U_{URL}. The single-view method takes into account the class label associated with the highest class probability value (see lines 3–10). Specifically, the probabilities of the concerned U_{URL} are evaluated for both malicious and non-malicious perspectives. The maximum of the probabilities dictates the particular U_{URL}'s class (malicious/non-malicious); otherwise, it returns the class label associated with $P_X(Malicious)$.

4.2. Ensemble approach

In addition to the single-view learning approach, this article uses the ensemble learning approach to detect malicious URLs. The ensemble learning approach solves a specific problem by combining the decisions computed by multiple classifiers [35]. For example, let patient XX possess multiple physical symptoms. XX decides to go to three different doctors denoted as doctor x, doctor y, and doctor z for the treatment. Suppose doctor x diagnoses the disease as heart disease, doctor y diagnoses it as just a blood pressure problem, and z makes a diagnosis of heart disease. Given the doctor's decision, XX can combine all this information to make the final decision. Here, since most three doctors predict that it is heart disease, it can be simply considered heart disease as the final decision. This is how ensemble learning works.

In machine learning, the ensemble learning approach utilizes multiple classifiers (e.g., RF, DT, k-NN, NB, SVM, etc.), combines the predicted decisions of these classifiers, and finds the final decision as the majority give this decision. However, there are two main ways to implement the ensemble learning approach, (i) have a single dataset with features and build multiple learning classifiers using these features. (ii) leverage a single learning classifier across multiple sub-datasets in terms of considering splitting a single dataset to build multiple classifiers.

Algorithm 2: Hard Voting based Ensemble Learning Algorithm

Input: L_{URL}, the set of labeled URLs; U_{URL}, the set of unlabeled URLs

Output: 1 if a URL be malicious in U_{URL} ; 0 otherwise

1 Let $(x_1, x_2, x_3, \ldots.)$ be the features set of L_{URL} and U_{URL}
2 Let N, be the set of supervised learning classifiers used to build an ensemble approach
3 Let $Label_X(Malicious)$, $Label_X(Non-Malicious)$ be initialized to zero
4 **for** $i \in \{1, \ldots, |N|\}$ **do**
5 $\quad Build\_Classifier_i = Classifiers_i(L_{URL}(x_1, x_2, x_3, \ldots))$
6 **for** $X \in U_{URL}$ **do**
7 $\quad$ **for** $i \in \{1, \ldots, |N|\}$ **do**
8 $\quad\quad Predicted\_Label\_For\_X_i = Build\_Classifier_i(X(x_1, x_2, x_3, \ldots))$
9 $\quad\quad$ Let $Label^i_X(Malicious)$ be the labeled of X being predicted as a malicious URL
10 $\quad\quad$ Let $Label^i_X(Non-Malicious)$ be labeled as X being predicted as a non-malicious URL
11 $\quad\quad Label_X(Malicious) = Label_X(Malicious) + Label^i_X(Malicious)$
12 $\quad\quad Label_X(Non-Malicious) = Label_X(Non-Malicious) + Label^i_X(Non-Malicious)$
13 $\quad$ Return: The label corresponding to $Max(Label_X(Malicious), Label_X(Non-Malicious))$

Algorithm 3: Soft Voting based Ensemble Learning Algorithm

Input: L_{URL}, the set of labeled URLs; U_{URL}, the set of unlabeled URLs

Output: 1 if a URL be malicious in U_{URL} ; 0 otherwise

1 Let $(x_1, x_2, x_3, \ldots.)$ be the features set of L_{URL} and U_{URL}
2 Let N, be the set of supervised learning classifiers used to build an ensemble approach
3 Let $P_X(Malicious)$, $P_X(Non-Malicious)$ be initialized to zero
4 **for** $i \in \{1, \ldots, |N|\}$ **do**
5 $\quad Build\_Classifier_i = Classifiers_i(L_{URL}(x_1, x_2, x_3, \ldots))$
6 **for** $X \in U_{URL}$ **do**
7 $\quad$ **for** $i \in \{1, \ldots, |N|\}$ **do**
8 $\quad\quad Predicted\_Probability\_For\_X_i = Build\_Classifier_i(X(x_1, x_2, x_3, \ldots))$
9 $\quad\quad$ Let $P^i_X(Malicious)$ be the probability of X being predicted as a malicious URL
10 $\quad\quad$ Let $P^i_X(Non-Malicious)$ be the probability of X being predicted as a non-malicious URL
11 $\quad\quad P_X(Malicious) = P_X(Malicious) + P^i_X(Malicious)$
12 $\quad\quad P_X(Non-Malicious) = P_X(Non-Malicious) + P^i_X(Non-Malicious)$
13 $\quad P_X(Malicious) = P_X(Malicious)/|N|$
14 $\quad P_X(Non-Malicious) = P_X(Non-Malicious)/|N|$
15 $\quad$ **if** $P_X(Malicious) == P_X(Non-Malicious)$ **then**
16 $\quad\quad$ Return: The label corresponding to $P_X(Malicious)$
17 $\quad$ **else**
18 $\quad\quad$ Return: The label corresponding to $Max(P_X(Malicious), P_X(Non-Malicious))$

This paper employs the former to implement the ensemble learning approach. Using the training dataset, separate learning classifiers, such as RF, DT, k-NN, NB, SVM, and each classifier, are built to predict the testing dataset labels. To combine each classifier's predicted labels and make a final decision on an instance, the ensemble learning method mainly considers two methods, (i) hard voting and (ii) soft voting. It is considered both concepts to combine the decisions of multiple classifiers to make the final decision. The rationale behind this is to compare the approaches. Below, these concepts are explained in detail.

4.2.1. Hard voting based ensemble learning approach

In this approach, multiple classifiers with the training dataset (i.e., labeled URLs) are constructed, and each of the classifiers produces the expected label on the testing dataset (i.e., unlabeled URLs). The next step is to find the URL label produced by most classifiers, which is the final decision as to whether a URL is malicious.

As described in Algorithm 2, multiple classifiers ($N = 5$) are built using the labeled URLs L_{URL} (line 5). Then, the classifier calculates the predicted labels of the unlabeled URLs (for example, U_{URL}) (lines 6–12). The used five classifiers will have five predicted labels for the unlabeled URL. Finally, a label is a voting base designed to an instance (i.e., unlabeled URL) by aggregating each class's label (i.e., malicious, non-malicious). As presented in Algorithm 2, the label returned for U_{URL} is the one associated with the majority classifiers voted for (line 13).

4.2.2. Soft voting based ensemble learning approach

The soft voting approach works differently from hard voting. The approach considers class probabilities instead of class labels. In the soft ensemble approach, the classifiers compute the class probabilities and add up the corresponding class probabilities predicted by the classifiers to find the final class label for unlabeled URLs that has comparatively larger than the probabilities of other classes. As shown in Algorithm 3, multiple classifiers with the labeled URLs L_{URL} is built (line 5). Once the classifiers have been built, given a new URL, this design identifies to which class (label) the URL most likely belongs to (see lines 6–12). Here, the bagging method [36] is used. It is worth noting that bagging is an effective method for ensemble learning, where the final label is associated with the average of the largest membership probabilities returned by the obtained classifiers (see lines 13–18 in Algorithm 3), otherwise Algorithm 3 returns the class label associated with $P_X(Malicious)$.

4.3. $SeizeMaliciousURL$ approach

$SeizeMaliciousURL$ is an extended version of the soft voting-based ensemble learning approach. The method essentially introduced a threshold δ that consecutively helps in selecting more appropriate decisions. The threshold is used to filter out the bad decisions made by an individual classifier (or a set of classifiers). The detailed description of the proposed method is systematically demonstrated as follows.

Threshold-based Soft Voting Ensemble Learning Approach. As a variant of the ensemble learning approach, $SeizeMaliciousURL$ comprises several inbuilt classifiers. Among these classifiers, some of the classifiers may calculate the probability value of a class (e.g., malicious/ non-malicious) very close to each other. This phenomenon is common, intuitive, and well-known. Let, a classifier $classifier1$ computes the probability of an unlabeled URL (i.e., U_{URL1}) for the malicious class of 0.52 and a probability of 0.48 for non-malicious class, respectively. In this instance, the difference between these calculated values is minimal. Therefore, it can be concluded that this situation may produce prediction errors. More specifically, such a situation is error-prone, reflecting the degradation of the classifier's performance.

Conversely, let classifier $classifier2$ calculates the probability of being malicious and non-malicious of the same URL, U_{URL1} is 0.70 and 0.30, respectively. Such calculation can provide a strong decision level for $URL1$ as malicious.

The philosophy behind the design is to handle this phenomenon above by incorporating the threshold δ value that filters the predicted

decision of individual classifier(s) results in improving the overall prediction accuracy of the ensemble learning approach. Essentially, the design transforms the soft voting-based ensemble learning approach. Let consider ignoring the decisions of those classifiers that do not satisfy the threshold δ assigned for checking the difference between the probability values of the predicted classes (malicious and non-malicious). Let term this approach as a threshold-based ensemble learning approach. This method introduces a malicious URL detection system named $SeizeMaliciousURL$.

Algorithm 4 depicts the entire threshold-based soft voting ensemble learning approach. The algorithm is very similar to the soft voting based ensemble learning approach, except for the part shown in lines 5–20 in Algorithm 4. More particularly, it is needed to find the appropriate value of δ that can filter the error-prone decisions. In the initial state, zero is assigned to the δ. Then, the optimization of δ is achieved voting-based on by advancing 0.01 from the range of {0, 1}, and offers the highest accuracy in the validation dataset. Multiple classifiers with a training dataset is first built (see lines 2–4 in Algorithm 4). Next, the class probabilities of the URLs are predicted that appeared in the validation dataset by the built classifiers (see line 9). Here, $P_X^i(Malicious)$ and $P_X^i(Non-Malicious)$ represent the class probabilities for malicious URL and non-malicious URL, respectively, produced by the multiple classifiers (see lines 10–11). Then the absolute difference is found, i.e., $|P_X^i(Malicious)$- $P_X^i(Non-Malicious)|$, and check whether this value meets the condition, that is, $|P_X^i(Malicious)$- $P_X^i(Non-Malicious) \geq \delta$ (see line 12).

If this condition is met, the corresponding class probabilities are summed. The same process for all classifiers (see lines 13–14) is performed. After this step, the $Max()$ function is used to calculate the predicted label of the validation dataset V_{URL} associated with the maximum class probability (see line 15). Then the accuracy of the validation dataset is measured with the $accuracy()$ function. This may lead to a list of accuracy values since it depends on the δ and the value of δ is changed (see lines 7–20). The value of δ is chosen and then assigned to δ', which produces the highest accuracy on the validation dataset (see line 19).

After fixing the value for δ, system proceeds to predict the class label of each URL in the testing dataset, U_{URL}. It first, predicts the class probabilities computed by the multiple classifiers (see lines 24–31), and then check the condition $|P_X^i(Malicious)$- $P_X^i(Non-Malicious)| \geq \delta'$ (see line 28). If the condition is true (i.e., the count is greater than zero), the algorithm sums up the corresponding class probabilities (see lines 29–30) and the algorithm returns the class label of the unlabeled URLs associated with the highest probability value (see lines 34–40), otherwise it returns the class label as $Malicious$ (i.e., count equals to zero, see lines 32–33).

Algorithm 4: $SeizeMaliciousURL$ Algorithm

Input: L_{URL}, the set of labeled URLs in training dataset; U_{URL}, the set of unlabeled URLs in the testing dataset; V_{URL}, the set of labeled URLs in the validation dataset

Output: 1 if a URL be malicious in U_{URL} ; 0 otherwise

1 Let $(x_1, x_2, x_3, \ldots)$ be the features set of L_{URL} and U_{URL}
2 Let N, be the set of supervised learning classifiers used to build an ensemble approach
3 **for** $i \in \{1, \ldots, |N|\}$ **do**
4 $\quad Build\_Classifier_i = Classifiers_i(L_{URL}(x_1, x_2, x_3, \ldots))$
5 Let K be initialized to 1; δ be initialized to Zero
6 Let $Check_{ACC}$, $P_VURL(Malicious)$, $P_VURL(Non-Malicious)$ be initialized to zero
7 **while** $\delta \leq K$ **do**
8 $\quad$ **for** $i \in \{1, \ldots, |N|\}$ **do**
9 $\quad\quad Predicted\_Probability\_For\_V_i = Build\_Classifier_i(V_{URL}(x_1, x_2, x_3, \ldots))$
10 $\quad\quad$ Let $P_V^iURL(Malicious)$ be the probability of V_{URL} being predicted as a malicious URL
11 $\quad\quad$ Let $P_V^iURL(Non-Malicious)$ be the probability of V_{URL} being predicted as a non-malicious URL
12 $\quad\quad$ **if** $(abs(P_V^iURL(Malicious)$- $P_V^iURL(Non-Malicious)) \geq \delta)$ **then**
13 $\quad\quad\quad P_VURL(Malicious) = P_V^iURL(Malicious) + P_VURL(Malicious)$
14 $\quad\quad\quad P_VURL(Non-malicious) = P_V^iURL(Non-malicious) + P_VURL(Non-malicious)$
15 $\quad Predicted_{label} = Max(P_VURL(Malicious), P_VURL(Non-Malicious))$
16 $\quad ACC = accuracy(Predicted_{label}, Actual_{label})$
17 $\quad$ **if** $(ACC \geq check_{ACC})$ **then**
18 $\quad\quad check_{ACC} = ACC$
19 $\quad\quad \delta' = \delta$
20 $\quad \delta = \delta + 0.01$
21 Let $P_X(Malicious)$, $P_X(Non-Malicious)$ be initialized to zero
22 **for** $X \in U_{URL}$ **do**
23 $\quad$ Let $count$, be initialized to zero
24 $\quad$ **for** $i \in \{1, \ldots, |N|\}$ **do**
25 $\quad\quad Predicted\_Probability\_For\_X_i = Build\_Classifier_i(X(x_1, x_2, x_3, \ldots))$
26 $\quad\quad$ Let $P_X^i(Malicious)$ be the probability of X being predicted as a malicious URL
27 $\quad\quad$ Let $P_X^i(Non-Malicious)$ be the probability of X being predicted as a non-malicious URL
28 $\quad\quad$ **if** $(abs(P_X^i(Malicious)$- $P_X^i(Non-Malicious)) \geq \delta')$ **then**
29 $\quad\quad\quad P_X(Malicious) = P_X(Malicious) + P_X^i(Malicious)$
30 $\quad\quad\quad P_X(Non-Malicious) = P_X(Non-Malicious) + P_X^i(Non-Malicious)$
31 $\quad\quad\quad count++$
32 $\quad$ **if** $count == 0$ **then**
33 $\quad\quad$ Return: The label as $Malicious$
34 $\quad$ **else**
35 $\quad\quad P_X(Malicious) = P_X(Malicious)/count$
36 $\quad\quad P_X(Non-Malicious) = P_X(Non-Malicious)/count$
37 $\quad\quad$ **if** $P_X(Malicious) == P_X(Non-Malicious)$ **then**
38 $\quad\quad\quad$ Return: The label corresponding to $P_X(Malicious)$
39 $\quad\quad$ **else**
40 $\quad\quad\quad$ Return: The label corresponding to $Max(P_X(Malicious), P_X(Non-Malicious))$

5. System evaluation elements

5.1. Dataset

This article considers two separate datasets to evaluate the performance of the proposed malicious URL detection model. These datasets are explained as follows.

Dataset-I. Dataset-I is essentially the data set [8]. Five different types of URLs, specifically: benign URLs, spam URLs, phishing URLs, malware URLs, and defaced URLs, are considered in this dataset. There are a total of 165362 URL examples in this dataset. This article terms benign URLs as non-malicious URLs and treats all other URLs as malicious URLs. Among the 165362 URLs reported in Table 2, 35378 URLs are non-malicious URLs and other URLs are malicious URLs. 80% of the total number of URLs is reserved for training the machine learning model, which is 132289. The remaining 20% of the URLs are split into two parts. (i) Validation set — about 6.66% of the URLs, i.e. 11024 URLs are chosen for the validation dataset. The validation dataset is used to find the appropriate threshold value δ that produces the highest accuracy

Table 2
Dataset.

Dataset	# Non-malicious URLs	# Malicious URLs	# Total URLs
Datatset-I	35378	129984	165362
Datatset-II	344821	75643	420464

Table 3
Dataset splitting Strategy.

Dataset	# Training	# Validation	# Testing	# Total URLs
Datatset-I	132289	11024	22049	165362
Datatset-II	336371	28031	56062	420464

Table 4
Confusion matrix.

		Predicted class	
		$MaliciousURL$	$Non-maliciousURL$
Actual class	$MaliciousURL$	TP	FN
	$Non-maliciousURL$	FP	TN

Table 5
Metrics definition.

Accuracy	=	(TP+TN)/(TP+FN+FP+TN)
Precision	=	TP/(TP+ FP)
Recall	=	TP/(TP+ FN)
F1	=	2* (Precision*Recall)/(Precision+Recall)

in the validation dataset. This practice is a standard mechanism in the machine learning field where the validation dataset is used for the hyperparameter tuning. (ii) Test set — The test data is reserved just to test the designed system's performance. The experiment considers 13.34% URLs, i.e., 132289, for performance testing of the proposed model.

Furthermore, this article continues focusing on the other large datasets. The following describes the second dataset utilized to evaluate the proposed malicious URL detection technique.

Dataset-II. The dataset is collected from GitHub.[3] Dataset-II consists of an adequate amount of entries to assess a malicious URL detection technique. There are 420464 URL samples in this dataset. Of these, 344821 URLs are non-malicious, and others are reported as malicious URLs in Table 2. 80% of the complete URLs is used for training purposes, i.e., 336371 entries. The remaining 20% URLs are divided into two parts, (i) validation — experiments consider about 6.66% URLs for the validation dataset, that is, 28031. (ii) testing: 13.34% of URLs is considered for testing the performance of the proposed models, i.e., 56062. (See Table 3.)

5.2. Hardware and schedule

The models are trained on one machine with Intel(R) Core(TM) i5-6200U CPU @2.30 GHz 2.40 GHz RAM – 8 GB, Windows pro, 64 bit. However, the evaluation time for the testing dataset I and II is approximately 480 ms and 1500 ms, respectively. Moreover, elapsed time for single testing data of dataset I is 6.5×10^{-2} ms.

5.3. Performance metrics

Precision, recall, F1 score, and accuracy metrics are used to assess the performance of machine learning techniques to detect malicious URLs. To do this, a confusion matrix is built as shown in Table 4. A confusion matrix is a well-known approach that represents a table, shows the visualization of the performance of a classification algorithm [37]. Each row represents the instances in an actual class; on the other hand, each column expresses the instances in a predicted class. This paper considers two classes like $Non-maliciousURL$ and $MaliciousURL$. Therefore, the confusion matrix has two rows and two columns that represent a 2×2 dimensional matrix. The diagonal units represent TP as a true positive and TN as a true negative. These are all the correct predictions made by a classifier. The remaining cells represent false positive (FP) and false negative (FN), which are prediction errors, represented by values as the other diagonal elements, as shown in Table 4. Table 5 shows the definition of the metric that is used to evaluate the performance of the classifiers. Accuracy refers to the ratio of the total number of instances correctly predicted by the total number of samples. Precision is the ratio of the number of correctly predicted positive instances by the total number of predicted positive instances. While the recall is defined as the ratio of the number of correctly predicted positive instances divide by the total number of positive instances. The F1 score considers both precision and recall.

[3] https://github.com/Jcharis/Machine-Learning-In-Julia-JCharisTech

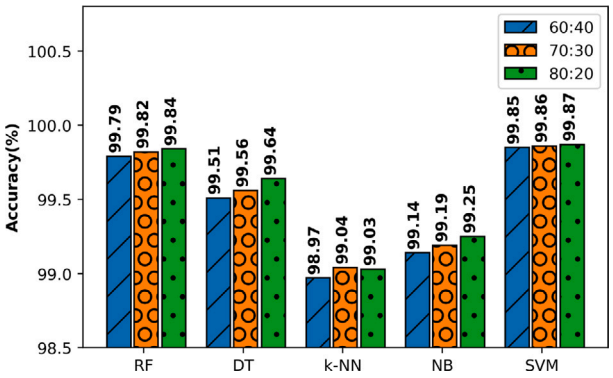

Fig. 2. Accuracy obtained with classifiers for various training and testing ratios using dataset-I.

6. Experimental results and discussion

To evaluate the effectiveness of the proposed approaches, several experiments were conducted. The first experiment measures accuracy by changing the size of the training and testing datasets. The main goal of this experiment was to find the appropriate size of the training dataset so that the learning method can produce relatively high accuracy in the testing dataset.

Next, experiments were conducted to see the performance between the considered machine learning models and the proposed model to detect non-malicious and malicious URLs. This experiment compared the accuracy and F1 scores produced by these methods and concludes which model had the highest performance compared to other models.

The final experiment computed the threshold δ between the class probabilities generated by the individual classifier of the $SeizeMaliciousURL$ model. It is important to note that this experiment has (i) tested all possible values from 0 to 1, (ii) found the optimized value, and (iii) set it to the threshold δ that can produce the highest accuracy in the validation dataset. This value was utilized to filter the error-prone decisions on the testing dataset produced by the soft voting-based broadly ensemble learning approach.

6.1. Various ratio of training and testing datasets

This experiment focused on showing the predicted accuracy with different learning approaches, given different proportions of training and testing data sets. Specifically, three ratios of training and testing datasets were considered, such as 60:40, 70:30, and 80:20. Several supervised learning models with different sizes of training datasets were to run (for example, 60%, 70% and 80%, respectively) and calculated the accuracy of each of the learning models in different sizes of testing datasets (for example, 40%, 30%, and 20% respectively). The results been shown in Fig. 2 for dataset-I and Fig. 3 for dataset-II.

Table 6 has shown the accuracy of all the classifiers for both of the datasets in different test-train splits. The highlighted items have represented the best results obtained by the setting is at 80%:20% ratio of the training and testing datasets. Consequently, this ratio in all of the remaining experiments was used.

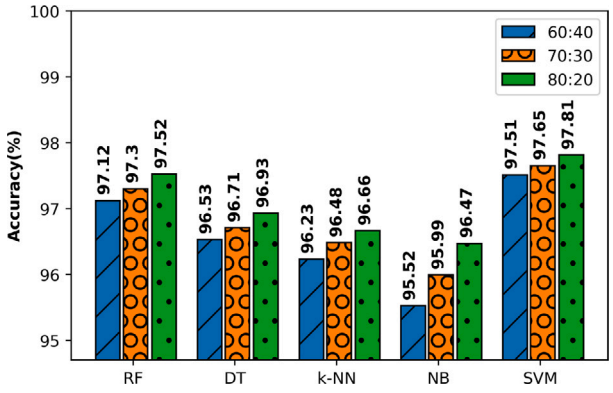

Fig. 3. Accuracy obtained with classifiers for various training and testing ratios using dataset-II.

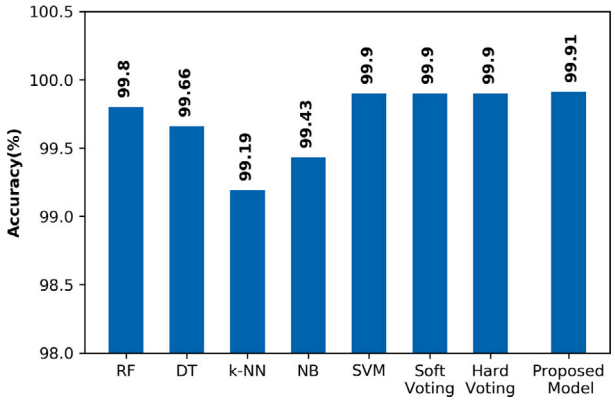

Fig. 4. Accuracy comparison among classifiers using dataset-I.

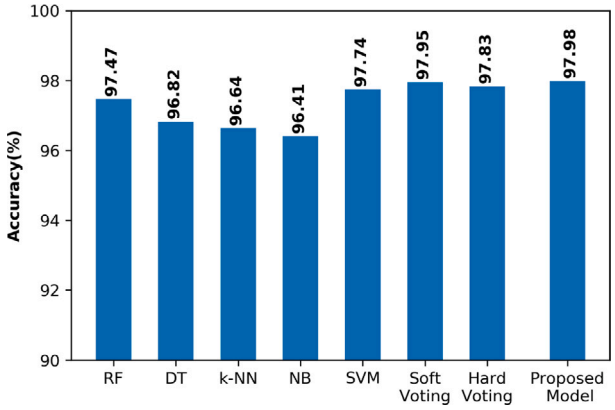

Fig. 5. Accuracy comparison among classifiers using dataset-II.

Table 6
Prediction accuracy of classifiers for various proportions of training and testing datasets using dataset-I and dataset-II.

	Classifiers	60:40 Accuracy (%)	70:30 Accuracy (%)	80:20 Accuracy (%)
Dataset-I	RF	99.79%	99.82%	**99.84%**
	DT	99.51%	99.56%	**99.64%**
	k-NN	98.97%	99.04%	**99.03%**
	NB	99.14%	99.19%	**99.25%**
	SVM	99.85%	99.86%	**99.87%**
Dataset-II	RF	97.12%	97.30%	**97.52%**
	DT	96.53%	96.71%	**96.93%**
	k-NN	96.23%	96.48%	**96.99%**
	NB	95.52%	95.99%	**96.47%**
	SVM	97.51%	97.65%	**97.81%**

6.2. Performance evaluation of the learning models

This experiment largely shown the performance comparison of (i) the considered existing learning models vs. (2) the proposed model. The most common performance metrics: accuracy and F1-score, were calculated. They have been detailed as follows.

Accuracy. This experiment compared the prediction accuracy among (i) the traditional supervised learning approaches (i.e., RF, DT, k-NN, NB, SVM), (ii) the ensemble learning approaches (i.e., soft voting, hard voting) and, (iii) the proposed model for both of the datasets. These experiments computed the accuracy of the considered machine learning approaches, and the proposed method for predicting malicious and non-malicious URLs is performed.

Figs. 4 and 5 have shown the accuracy of the considered machine learning approaches and the proposed method using dataset-I and dataset-II, respectively. It can be observed that SVM has given the highest accuracy of 99.90% for dataset-I and 97.74% for dataset-II, respectively, among the traditional supervised learning approaches (i.e., RF, DT, k-NN, NB, SVM). However, the significant fact being that the ensemble approaches (i.e., soft, complex) have provided better performances than the SVM for both datasets. In addition, the soft voting-based ensemble approach produced better accuracy than the challenging voting-based ensemble approach.

Further notably, Figs. 4 and 5 have shown that the proposed approach outperforms all of the other available schemes, i.e., 99.91% for dataset-I and 97.98% for dataset-II, respectively. In other words, the proposed model and soft voting ensemble models incorrectly have detected 19 and 20 malicious URLs for dataset-I and 1134 and 1149 for dataset-II, respectively. This implies that the proposed model have performed almost the same performance as soft voting for dataset-I but improves more in dataset-II. That is, 15 more URLs detected correctly than a soft voting ensemble. This has been due to the fact that all learning models can correctly detect approximately 100% of malicious URLs in dataset-I. However, there has still been some room for improvement in DataSet-II. Therefore, evidently, the proposed model shows better performance in dataset-II than dataset-I. Also, it is worth noting that the proposed model has shown a slight improvement in detecting malicious URLs in both datasets compared to the other models. However, the important thing is that all malicious URLs are dangerous to users. So, improvements to the model used to detect malicious URLs were worthwhile.

F1-score. The F1-score uses the precision and recall (of a classifier) and takes its harmonic average to make a single metric. It is mainly used to compare the performance of classifiers. For example, suppose that classifier X has a higher recall, and classifier Y has higher precision. In this case, the F1-score for both classifiers can be used to see which one yields better results. This experiment finds the F1-score for non-malicious and malicious classification of URLs. Figs. 6 and 7 have shown that the F1-scores of the considered machine learning approaches and the proposed approach for dataset-I and dataset-II,

Table 7
Accuracy (%) comparison among classifiers.

Datasets	RF	DT	k-NN	NB	SVM	Soft voting	Hard voting	Proposed model
Dataset-I	99.80%	99.66%	99.19%	99.43%	99.90%	99.90%	99.90%	**99.91%**
Dataset-II	97.47%	96.82%	96.64%	96.41%	97.74%	97.95%	97.83%	**97.98%**

Table 8
F1-score for malicious vs non-malicious classification.

	Metrics	URL label	RF	DT	k-NN	NB	SVM	Soft voting	Hard voting	Proposed model
Dataset-I	Precision	Non-malicious	99.43%	99.20%	99.61%	98.81%	99.68%	99.85%	99.81%	99.83%
		Malicious	99.90%	99.80%	99.08%	99.61%	99.97%	99.93%	99.92%	99.94%
	Recall	Non-malicious	99.64%	99.26%	96.61%	98.56%	99.87%	99.72%	99.72%	99.77%
		Malicious	99.84%	99.78%	99.90%	99.68%	99.91%	99.96%	99.95%	99.95%
	F1-score	Non-malicious	**99.53%**	**99.23%**	**98.09%**	**98.68%**	**99.78%**	**99.79%**	**99.77%**	**99.80%**
		Malicious	**99.87%**	**99.79%**	**99.49%**	**99.64%**	**99.94%**	**99.94%**	**99.94%**	**99.95%**
Dataset-II	Precision	Non-malicious	97.37%	97.53%	98.13%	95.91%	97.65%	98.01%	97.63%	98.06%
		Malicious	98.00%	93.34%	89.97%	99.39%	98.24%	97.65%	98.82%	97.58%
	Recall	Non-malicious	99.61%	98.61%	97.76%	99.89%	99.65%	99.52%	99.77%	99.50%
		Malicious	87.71%	88.63%	91.51%	80.57%	89.05%	90.79%	88.97%	91.02%
	F1-score	Non-malicious	**98.47%**	**98.07%**	**97.95%**	**97.86%**	**98.64%**	**98.76%**	**98.69%**	**98.78%**
		Malicious	**92.57%**	**90.92%**	**90.73%**	**89.00%**	**93.42%**	**94.10%**	**93.64%**	**94.18%**

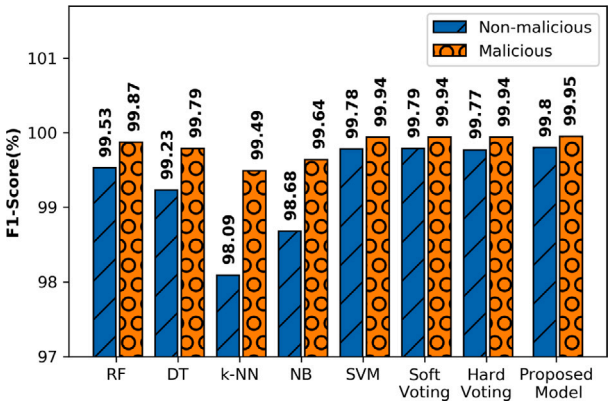

Fig. 6. F1-score obtained with classifiers using dataset-I.

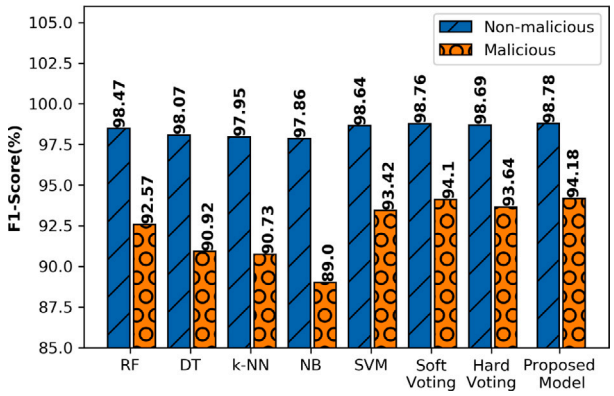

Fig. 7. F1-score obtained with classifiers using dataset-II.

respectively. It can be seen that the F1-score of the approach has provided the highest score than other approaches for both datasets. Moreover, the precision and recall value of the proposed model was not always greater than other models, as shown in Table 8. However, the significant fact was that since these two parameters are mainly used to compute the value of F1-score, and the proposed model gives the maximum F1-score, it implies that the approach was better than the others.

Based on the discussion above, it can be concluded that of the proposed $SeizeMaliciousURL$ model is more effective than other considered machine learning models for detecting malicious URLs.

6.3. Threshold value (δ) optimization for $SeizeMaliciousURL$ model

This experiment was run to find the optimization value of the threshold, δ. Here, the δ was considered in between 0 and 1 with an increment of 0.01. The idea was to assign a δ value that produces the highest accuracy in the validation dataset. After that, this value was used to predict the test dataset's class label and check the accuracy of the proposed model.

In Fig. 8, the experiment has found that the threshold value δ is of 0.34 and 0.30 for the dataset-I and dataset-II, respectively. Here, the points have been presented by the black dots in Fig. 8. Note that these points have yielded the highest accuracy on the testing dataset as shown in Table 7, which have been highlighted in bold.

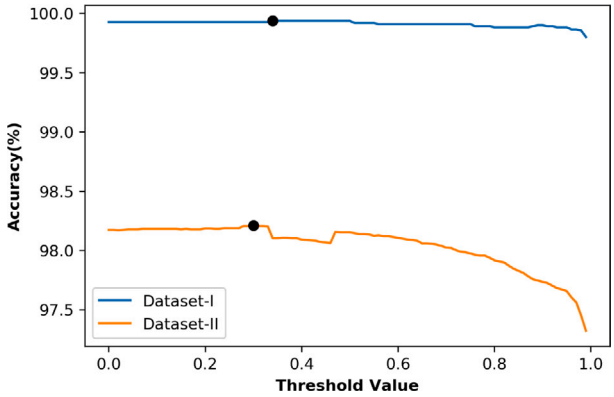

Fig. 8. Threshold value δ obtained from the validation dataset.

7. Conclusion and future work

This paper describes $SeizeMaliciousURL$- a novel and practical approach to detect and classify malicious URLs. This design exploits the soft voting concept-based ensemble learning approach. A threshold value, δ is imposed between classifiers produced by classifiers to filter out the error-prone decisions in detecting and classifying URLs. Furthermore, this research examined the optimal value of the threshold to facilitate the highest prediction accuracy on the testing dataset (e.g., unlabeled URLs). The paper has shown that the proposed approach is more accurate compared to the other machine learning approaches. More specifically, the system can detect non-malicious vs. malicious URLs with the highest accuracy compared to other models, that is, 99:91% for dataset-I and 97:98% for dataset-II.

Research in this direction has not achieved 100% accuracy in any of the cases; additional research is needed in this direction. In the future, it is planned to execute more effective feature engineering to increase the accuracy of the classification approach.

CRediT authorship contribution statement

Dipankar Kumar Mondal: Conception and design of study, Acquisition of data. **Bikash Chandra Singh:** Conception and design of study, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Haibo Hu:** Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Shivazi Biswas:** Conception and design of study, Acquisition of data. **Zulfikar Alom:** Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Mohammad Abdul Azim:** Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant No: 62072390), and the Research Grants Council, Hong Kong SAR, China (Grant No: 15222118, 15218919, 15203120).

References

[1] 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. IEEE; 2008, p. 57–64.
[2] Sahoo D, Liu C, Hoi SC. Malicious URL detection using machine learning: A survey. 2017, arXiv preprint arXiv:1701.07179.
[3] Kolbitsch C, Livshits B, Zorn B, Seifert C. Rozzle: De-cloaking internet malware. In: 2012 IEEE symposium on security and privacy. IEEE; 2012, p. 443–57.
[4] Darling M, Heileman G, Gressel G, Ashok A, Poornachandran P. A lexical approach for classifying malicious URLs. In: 2015 international conference on high performance computing & simulation. IEEE; 2015, p. 195–202.
[5] Khonji M, Iraqi Y, Jones A. Phishing detection: A literature survey. IEEE Commun Surv Tutor 2013;15(4):2091–121.
[6] Kuyama M, Kakizaki Y, Sasaki R. Method for detecting a malicious domain by using whois and dns features. In: The third international conference on digital security and forensics, 74, 2016.
[7] Li T, Kou G, Peng Y. Improving malicious URLs detection via feature engineering: Linear and nonlinear space transformation methods. Inf Syst 2020;101494.
[8] Mamun MSI, Rathore MA, Lashkari AH, Stakhanova N, Ghorbani AA. Detecting malicious urls using lexical analysis. In: International Conference on Network and System Security. Springer; 2016, p. 467–82.
[9] Hakim AA, Erwin A, Eng KI, Galinium M, Muliady W. Automated document classification for news article in bahasa Indonesia based on term frequency inverse document frequency (TF-IDF) approach. In: 2014 6th International Conference on Information Technology and Electrical Engineering. IEEE; 2014, p. 1–4.
[10] Dietterich T. Overfitting and undercomputing in machine learning. ACM Comput. Surv. 1995;27(3):326–7.
[11] Whittaker C, Ryner B, Nazif M. Large-scale automatic classification of phishing pages. 2010.
[12] Sahingoz OK, Buber E, Demir O, Diri B. Machine learning based phishing detection from URLs. Expert Syst Appl 2019;117:345–57.
[13] Alkawaz MH, Steven SJ, Hajamydeen AI. Detecting phishing website using machine learning. In: 2020 16th IEEE international colloquium on signal processing & its applications. IEEE; 2020, p. 111–4.
[14] Zhang Y, Hong JI, Cranor LF. Cantina: A content-based approach to detecting phishing web sites. In: Proceedings of the 16th international conference on world wide web. 2007. p. 639–48.
[15] Jain AK, Parashar S, Katare P, Sharma I. PhishSKaPe: A content based approach to escape phishing attacks. Procedia Comput Sci 2020;171:1102–9.
[16] Rao RS, Vaishnavi T, Pais AR. CatchPhish: Detection of phishing websites by inspecting URLs. J Ambient Intell Humaniz Comput 2020;11(2):813–25.
[17] Sameen M, Han K, Hwang SO. PhishHaven—an efficient real-time AI phishing URLs detection system. IEEE Access 2020;8:83425–43.
[18] Ma J, Saul LK, Savage S, Voelker GM. Identifying suspicious URLs: An application of large-scale online learning. In: Proceedings of the 26th annual international conference on machine learning. 2009. p. 681–8.
[19] Vinayakumar R, Soman K, Poornachandran P. Evaluating deep learning approaches to characterize and classify malicious URL's. J Intell Fuzzy Systems 2018;34(3):1333–43.
[20] Bahnsen AC, Bohorquez EC, Villegas S, Vargas J, González FA. Classifying phishing URLs using recurrent neural networks. In: 2017 APWG symposium on electronic crime research. IEEE; 2017, p. 1–8.
[21] Saxe J, Berlin K. eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys. 2017, arXiv preprint arXiv:1702.08568.
[22] KP S, Alazab M, et al. Malicious URL detection using deep learning. 2020.
[23] Alom Z, Carminati B, Ferrari E. Detecting spam accounts on Twitter. In: 2018 IEEE/ACM international conference on advances in social networks analysis and mining. IEEE; 2018, p. 1191–8.
[24] Alom Z, Carminati B, Ferrari E. A deep learning model for Twitter spam detection. Online Soc. Netw. Media 2020;18:100079.
[25] Muna A-H, Moustafa N, Sitnikova E. Identification of malicious activities in industrial internet of things based on deep learning models. J. Inform. Secur. Appl. 2018;41:1–11.
[26] Mohammadi S, Mirvaziri H, Ghazizadeh-Ahsaee M, Karimipour H. Cyber intrusion detection by combined feature selection algorithm. J. Inform. Secur. Appl. 2019;44:80–8.
[27] Singh BC, Carminati B, Ferrari E. Learning privacy habits of pds owners. In: 2017 IEEE 37th International conference on distributed computing systems. IEEE; 2017, p. 151–61.
[28] Singh BC, Carminati B, Ferrari E. Privacy-aware personal data storage (P-PDS): Learning how to protect user privacy from external applications. IEEE Trans Dependable Secure Comput 2019.
[29] Alom MZ, Carminati B, Ferrari E. Helping users managing context-based privacy preferences. In: 2019 IEEE international conference on services computing. IEEE; 2019, p. 100–7.
[30] Alom Z, Singh BC, Aung Z, Azim MA. Knapsack graph-based privacy checking for smart environments. Comput. Secur. 2021;105:102240.
[31] Singh BC, Carminati B, Ferrari E. A risk-benefit driven architecture for personal data release. In: 2016 IEEE 17th international conference on information reuse and integration. IEEE; 2016, p. 40–9.
[32] Chen C-M, Huang J-J, Ou Y-H. Efficient suspicious URL filtering based on reputation. J. Inform. Secur. Appl. 2015;20:26–36.
[33] Chen S, Xue M, Fan L, Hao S, Xu L, Zhu H, et al. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. Comput. Secur. 2018;73:326–44.
[34] Haider CMR, Iqbal A, Rahman AH, Rahman MS. An ensemble learning based approach for impression fraud detection in mobile advertising. J Netw Comput Appl 2018;112:126–41.
[35] Dietterich TG, et al. Ensemble learning. In: The handbook of brain theory and neural networks, 2, Massachusetts: MIT press Cambridge; 2002, p. 110–25.
[36] Breiman L. Bagging predictors. Mach Learn 1996;24(2):123–40.
[37] Powers DM. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.