Received October 15, 2021, accepted October 28, 2021, date of publication November 1, 2021, date of current version November 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3124628

Robust Ensemble Machine Learning Model for Filtering Phishing URLs: Expandable Random Gradient Stacked Voting Classifier (ERG-SVC)

PUBUDU L. INDRASIRI[1], MALKA N. HALGAMUGE[2], (Senior Member, IEEE), AND AZEEM MOHAMMAD[1]

[1]School of Computing and Mathematics, Charles Sturt University, Melbourne, VIC 3000, Australia
[2]Department of Electrical and Electronic Engineering, The University of Melbourne, Melbourne, VIC 3010, Australia

Corresponding author: Malka N. Halgamuge (malka.nisha@unimelb.edu.au)

This work was supported by Charles Sturt University (CSU).

ABSTRACT As cyber-attacks grow fast and complicated, the cybersecurity industry faces challenges to utilize state-of-the-art technology and strategies to battle the consistently present malicious threats. Phishing is a sort of social engineering attack produced technically and classified as identity theft and complicated attack vectors to steal information of internet users. In this perspective, our main objective of this study is to propose a unique, robust ensemble machine learning model architecture that provides the highest prediction accuracy with a low error rate while proposing few other robust machine learning models. Both *supervised* and *unsupervised* techniques were used for the detection process. For our experiments, seven classification algorithms, one clustering algorithm, two ensemble techniques, and two large standard legitimate datasets with 73,575 URLs and 100,000 URLs were used. Two test modes (percentage split, K-Fold cross-validation) were utilized for conducting experiments and final predictions. Mechanisms were developed to (I) identify the best N, which is the optimal heuristic-based threshold value for splitting words into subwords for each classifier, (II) tune hyperparameters for each classifier to specify the best parameter combination, (III) select prominent features using various feature selection techniques, (IV) propose a robust ensemble model (classifier) called the *Expandable Random Gradient Stacked Voting Classifier* (*ERG-SVC*) utilizing a voting classifier along with a model architecture, (V) analyze possible clusters of the dataset using k-means clustering, (VI) thoroughly analyze the *gradient boost* classifier (*GB*) with respect to utilizing the "criterion" parameter with the Mean Absolute Error (*MAE*), Mean Squared Error (*MSE*), and *Friendman_MSE*, and(VII) propose a lightweight preprocessor to reduce computational cost and preprocessing time. Initial experiments were carried out with 46 features; the number of features was reduced to 22 after the experiments. The results show that the *GB* classifier outperformed with the least number of *NLP* based features by achieving a 98.118% prediction accuracy. Furthermore, our stacking ensemble model and proposed voting ensemble model (*ERG-SVC*) outperformed other tested approaches and yielded reliable prediction accuracy results in detecting malicious URLs at rates of 98.23% and 98.27%, respectively.

INDEX TERMS Phishing URLs, cybersecurity, machine learning, NLP, supervised, unsupervised.

I. INTRODUCTION

Since about a decade, internet usage has increased exponentially, and internet users have used it to find and accomplish their various demands like communication, shopping, payment transactions, and more by utilizing the web instead of using time-squandering conventional techniques [1]. The internet is empowering for many activities and makes life easy. Even so, it has its own shortcomings and weaknesses. Cybercriminals abuse the weaknesses of the internet and exploit them to defraud innocent users [2]. An adaptive time-based algorithm was proposed in a recent study [3] identifying the likelihood of malicious attacks with high accuracy. Phishing is the most popular tool amongst hackers for executing attacks in an endeavor to obtain sensitive information such as our account credentials, bank account information and sometimes social media information by deceiving and misleading the user to pay into the hacker's account, and

The associate editor coordinating the review of this manuscript and approving it for publication was Matti Hämäläinen.

more. This is achieved by; (I) posing as a legitimate institution, (II) using human emotions like fear, generosity and greed to lure the user into clicking on a link on a web page that appears genuine, (III) making user download and install malware. The attacker's advanced phishing procedures and semantics-based attack structure make it difficult for users to distinguish genuine web content and phishing attacks [4]. Hence, it is challenging for the network administrators and cyber security experts to impede these attacks, efficiently using human and computer weaknesses. Therefore, advanced algorithms are needed to shield users from such attacks.

Phishing attacks have become a global threat due their expanded extremely fast expansion in the most recent couple of years [5]–[8]. It is absurd to expect a 100% phishing attack detection approach, as attackers routinely change their attacking methods. As such, various solutions have been suggested by experts over the previous years to detect and mitigate phishing attacks. However, the burden of phishing attacks still exists, and developing an efficient anti-phishing approach has become challenging. Moreover, most anti-phishing solutions produce high false positives and are not capable of dealing with zero-hour attack. Email is the mainstream which attackers use to deploy phishing attacks. In addition, messaging has now bought into the mainstream in delivering phishing attacks. Phishing approaches are usually separated into two groups: user awareness and a systematic approach.

For various reasons, the user awareness approach is not adequate to prevent phishing attacks [9]. Some of these reasons include (I) user having lack of knowledge about URLs, (II) user's uncertain of websites to trust, (III) the existence of malicious URLs that are usually hidden from the users, and (IV) malicious websites that look identical to original websites. Hence, most previous works have focused on systematic approaches to detect and extenuate. The traditional systematic approach is to use a list-based (*black list*, *white list*) method to detect phishing attacks. A very-high-security environment is generated by detecting systems based on whitelists, where filtering the incoming URL in the list, allows only genuine emails to reach the end-user. However, the problem with this type of detection system is that it considers benign, newly created, and unlisted legitimate URLs to be malicious.

Hence, companies are currently using various software-based solutions such as image processing, natural language processing, ML or AI to detect malicious URLs [10]. Phishing attacks can be detected effectively by AI and machine language (ML) methodologies instead of static techniques. To help alleviate the phishing detection problem, this paper introduces a solution with a robust ML model that provides high phishing-attack detection accuracy by evaluating and verifying results using various datasets, thus leading to a globally acceptable solution.

A. RELATED WORKS

This section addresses the various ways that have been proposed to deal with phishing attacks by using ML techniques and different ensemble techniques, which were enhanced in order to obtain better results. Most studies have trained the classification algorithms by extracting features from phishing websites. Those characteristics can be divided into several classes, a few of which are shown in Figure 1. Therefore, it is easy to enable predicting the credibility of unseen URLs and the detection of phishing attacks by training ML algorithms with a rich collection of extracted features. Table 1 demonstrates the literary analysis of the research carried out in the same context.

On the other hand, an enhanced bagging technique was developed in the study [11], utilizing the misclassified predictions by the previous ML algorithm of the proposed ensemble architecture. The recent study [12] did a comparative analysis on gradient boosting algorithms, XGBoost, LightGBM, and CatBoost, regarding both accuracy and speed. A successful novel ensemble approach was proposed by Rojarath and Songpan [13] using the voting classifier. It uses probability-weight, which leverages the training data to generate its own probability calculations for each model. An effective prediction strategy based on stacking ensemble learning was proposed in study [14] to achieve reliable prediction results.

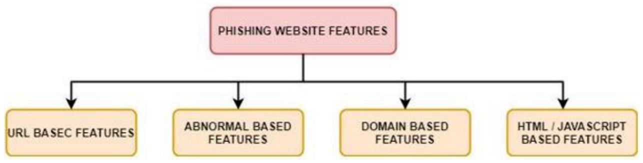


FIGURE 1. Few key categories of URL based features.

B. MOTIVATIONS AND CONTRIBUTIONS

An antiphishing solution with high accuracy, low false positives, and low false negatives could protect users from online threats. A limited dataset could be used to develop a solution using ML; however, a better product could be developed at an actual production level by using multiple datasets with large sample sizes for better outcome accuracy. For certain ML models, a small sample size typically provides a higher accuracy rate than that of a very large dataset and provides a biased performance while executing k-fold cross-validation and parameter-tuning-related experiments [25]. A very large feature set would dramatically increase the complexity of ML models and increase model computation time. Therefore, a well-organized feature-engineering process would lead to a simpler, more accurate ML model. To improve accuracy scores, building ensemble models is essential. Even so, computation time must be the biggest concern in such approaches. Additionally, a preprocessor is one of the vital components focused on ML-based examinations because it provides valuable information to train and test a model from raw data. Hence, having a robust preprocessor leads to highly accurate results. Preprocessor designers must determine specific heuristic-based threshold values for decision-making; however, those values may not provide optimal results for each classifier. A dynamic preprocessor is likely to provide more accurate results for those types of problem statements.

TABLE 1. Summary of background literature.

No	Study	Proposed System	Dataset	Classifier	Accuracy
1	Sindhu et al. (2020) [15]	Three different ML models using limited dataset	13,000	*RF* / SVM / Neural Networks	97.45%
2	Zamir et al. (2020) [16]	A method for using the stacking model to identify phishing websites and use feature elimination techniques while using limited dataset.	13,000	*Stacking* Classifier / *AdaBoost* / RF / *Bagging* / *KNN* / *J48*	97.4%
3	Rao et al. (2019) [17]	A model analyzing image features and certain heuristic features and use PCA analysis for feature engineering and used *Random forest* classifier for classification with limited dataset while using third part services based features	3,500	*RF* / *BN* / *SVM* / *J48* / *MLP* / *SMO*	99.31%
4	Babagoli et al. (2019) [18]	A phishing detection model using meta-heuristic based non-linear regression algorithm and used 20 features. *Decision tree* classifier is used to obtain results and it includes third party services based features.	11,000	*Harmony search* / *SVM*	96.32%
5	Feng et al. (2018) [19]	A model by classifying using neural network and Monte Carlo algorithm. No third party services based features included, however, it need to download the whole web page.	11,000	*NB* / *LR* /*KNN* / *DT* / *LSVM* / *RSVM* / *LDA*	97.71%
6	Smadi et al. (2018) [20]	A model with classified using neural network-based approach together with reinforcement with real time results. Even so, it utilizes about 50 features for training	9,118	*Neural network*	98.63%
7	Peng et al. (2018) [21]	A machine learning model using *NLP* based features and Naïve bayes classifier for classification process.	10,000	*Naive Bayes*	95%
8	Jain et al. (2018) [22]	A machine learning model to detect phishing attacks and it used in client side. *Random forest* classifier was used to obtain results using different types of extracted features. (URL based, CSS based and etc.)	4,000	*RF* / *SVM* / *Neural networks* / *LR* / *NB*	99.39%
9	Shirazi et al. (2018) [23]	An improved framework for detect phishing attacks using deep learning with 28 different types of features and obtain and analysis results using different classifiers including third party services based features.	5,000	Support vector machine (*SVM*)	96%
10	Buber et al. (2017) [24]	A machine learning model using extracted *NLP* based features. Three machine learning models were created by classifying the extracted emails	7,300	*NB* / *DT* /*RF* /*AdaBoost* / *SVC*	97.2%

C. MAIN CONTRIBUTIONS

The main contributions of this paper include the following.

1) Identifies the best heuristic threshold values (N) to split words into subwords and obtain the best results from ML models.
2) Introduces a novel lightweight preprocessor that used the minimum number of features to obtain the highest accuracy scores.
3) Identifies the optimal number of features using a well-defined feature selection process with six different techniques such as constant and quasi constant removal.
4) Design of a rich ensemble model architecture using the voting classifier (ERG-SVC).
5) Comparison of the results of our method to those of others described in the literature.

D. STRUCTURE OF THE PAPER

The rest of this paper is structured in the following manner. Section (II) illustrates the methods and methodologies used for building ML models. The results are highlighted in section (III). Section (IV) discusses the findings of the study and sections (V) and (VI) elaborate on future directions and the conclusion.

II. MATERIALS AND METHODS

Figure 2 shows the step-by-step methodology that was followed throughout entire study. It shows how ML was used to distinguish between legitimate URLs and those devised for phishing by using classification, clustering, and ensemble ML techniques. The entire process included some major subprocesses: a feature extraction module, a best N selection module, and a feature selection module. All subprocesses and ML model building methodologies are discussed in detail in later sections.

A. DATA COLLECTION AND PREPARATION

A reliable and acceptable dataset is required as a key input for an ML-based detection approach for URL validity predictions. Therefore, three separate datasets were collected as shown in Table 2. The best ML model was determined by analyzing and comparing the proposed methodologies' outcomes using various performance evaluation metrics on all datasets.

TABLE 2. Summary of collected datasets.

Dataset	Phishing	Legitimate	Source
$DataSet_1$	37,375	36,400	Buber et al. (2019) [10]
$DataSet_2$	40,000	60,000	NetSecExplained website [26]
$DataSet_3$	6,000	6,000	UCI database [27]

B. URL FEATURE EXTRACTION MODULE (PRE-PROCESSING)

Pre-processing data is vital before feeding it into the model. A study by Sahingoz *et al.* [10] proposed 40 different URL-based features to extract using natural language processing and third-party services. Our experiment,

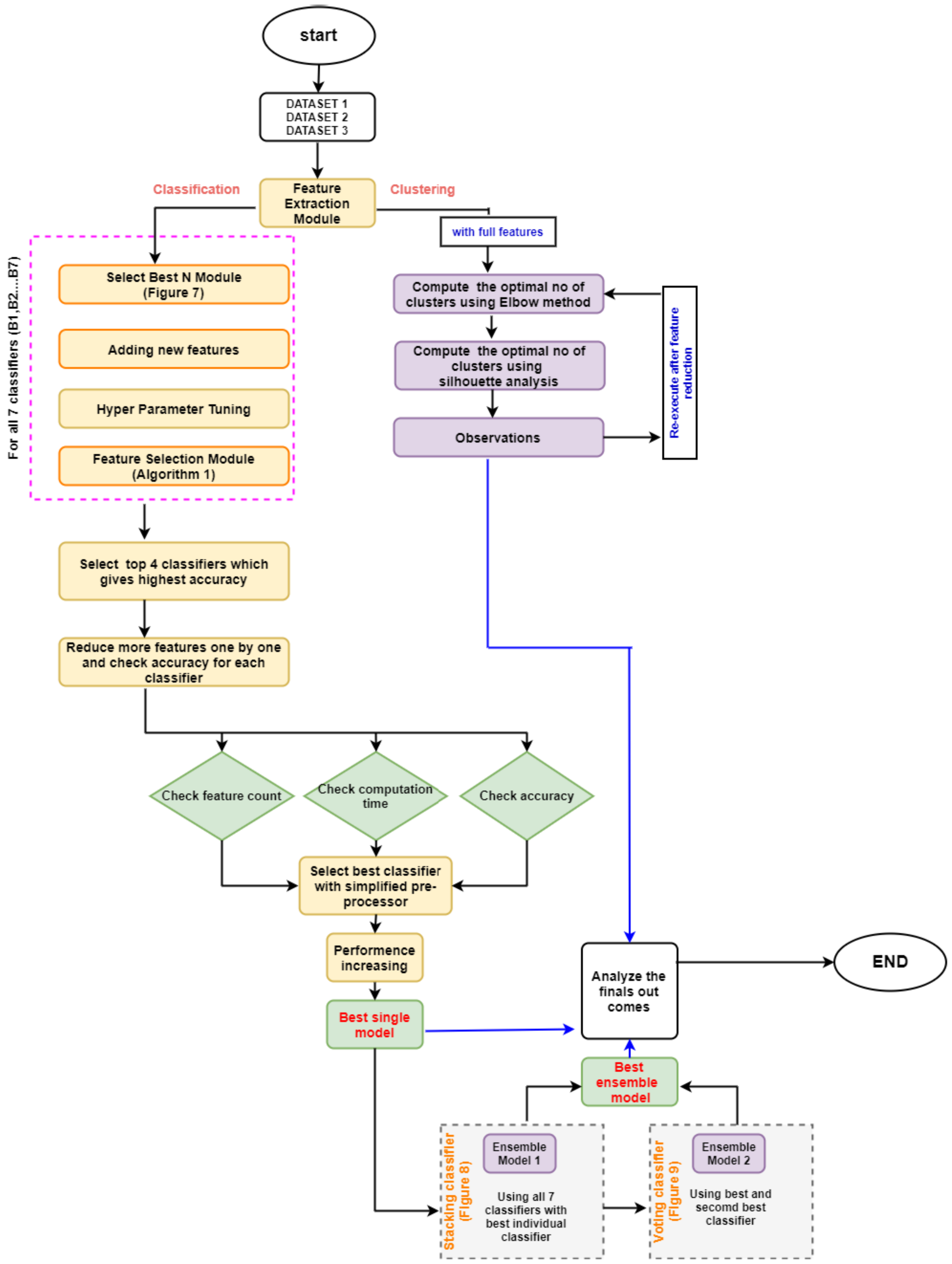

start
DATASET 1
DATASET 2
DATASET 3
Feature Extraction Module
Classification
Clustering
with full features
For all 7 classifiers (B1,B2....B7)
Select Best N Module (Figure 7)
Adding new features
Hyper Parameter Tuning
Feature Selection Module (Algorithm 1)
Compute the optimal no of clusters using Elbow method
Compute the optimal no of clusters using silhouette analysis
Observations
Re-execute after feature reduction
Select top 4 classifiers which gives highest accuracy
Reduce more features one by one and check accuracy for each classifier
Check feature count
Check computation time
Check accuracy
Select best classifier with simplified pre-processor
Performence increasing
Best single model
Analyze the finals out comes
END
Best ensemble model
Stacking classifier (Figure 8)
Ensemble Model 1
Using all 7 classifiers with best individual classifier
Voting classifier (Figure 9)
Ensemble Model 2
Using best and secomd best classifier

FIGURE 2. Methodology proposed for building various machine learning models.

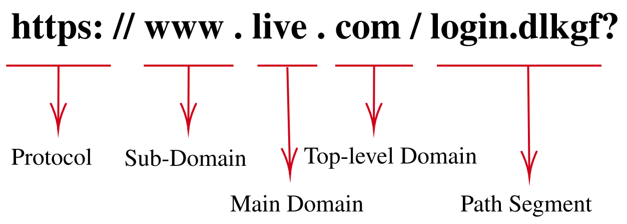


FIGURE 3. URL Structure which is used for extracting features.

(I) omitted two third-party-based features (*Alexa check* and *Alexa trie*), out of 40 features from [10] study, (II) added 8 extra features from previous studies [28], [29], and (III) modified and retuned extra features for best results.

Figure 3 shows the key parts of the URLs that were used as inputs for feature extraction. Natural language processing was used to extract most of the features using URLs, and two features (**F2** and **F8**) were extracted from third-party services. Newly added features (**F1**, **F2**, ..., **F8**) are explained in detail after Figure 3.

1) F1 - HavingIP

Using an IP address instead of a URL domain is an alarming indication of malicious intent. Example: "http://125.98.3.123/fake.html"

$$f(\text{domain}) = \begin{cases} \text{Phishing} & \text{If the domain part has an IP} \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{1}$$

2) F2 - PageRank

The PageRank (PR) is a probabilistic algorithm utilized by Google in its search engine to evaluate the quality of the website and rank those web pages accordingly with their search results. PageRank works by checking the number and nature of associations with a page to roughly estimate how critical the site is. The basic assumption is that more significant sites will more likely get more links from other sites. It plays a role in detecting a phishing site by ranking a website from 0 to 1. A website is important or considered the best quality when it has a greater PageRank value

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If PageRank} < 0 \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{2}$$

3) F3 - DoubleSlashRedirecting

Having "//" inside the URL implicates that the user will be redirected to another page or web site. For instance, a URL is: "http:/ www.legitimate.com//http://www.phishing.com." We inspect the position of the "//" within the URL. It is legit to have a "//" in the 6th position of the URL

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If position(//)} > 7 \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{3}$$

4) F4 - PORT

Ports help us to detect if a particular service (e.g. HTTP) is running or down on a server. A non-standard port in a URL could also be an indicator of a bad URL. The Network Address Translation (NAT), Firewall, and proxy will, by default, block the majority of the open ports which are non-standard or that are not required to be open

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If port no is non-standard} \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{4}$$

5) F5 - SHORTENING SERVICE

URL shortening services shorten a long URL into a much shorter length and redirect to the original target website link. For example,: the URL "http://portal.hud.ac.uk/" is shortened to "bit.ly/19DXSk4". The shortened bit.ly link will redirect to the original URL "http://portal.hud.ac.uk/". These shortening links are legitimately used for analytics. Even if every shortening URL is not a phishing URL, most of them has a potential to be phishing [30]. Attackers use these services to disguise the mala fide URL's

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If URL is short} \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{5}$$

6) F6 - HTTPs TOKEN

The threat actor would add the *HTTPS* token as a part of the domain to trick the target user into thinking that the site is secure and legitimate. Example: http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If using ``HTTPS'' token in domain} \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{6}$$

7) F7 -URL LENGTH

A lengthy URL is used by phishers to hide the insure part in the address bar. Following threshold values are determined after analyzing the average URL length of our Dataset 1

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If URL length} > 75 \\ \text{Suspicious} & \text{If 54> URLength >75} \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \tag{7}$$

8) F8 - DOMAIN AGE

Most phishing URLs are either short-lived or recently created. The URL's age can be verified from the WHOIS domain database, which publishes information about the domain and its age. The best threshold value for distinguishing phishing from legitimate URLs was determined based on the domain age by analyzing the distribution plot (*distplot*), shown in Figure 4, and the *box plot*, shown in Figure 5, derived from the Python *Seaborn* library. If any domain age was shown as 0 by WHOIS lookup, that domain was ignored for this experiment.

According to the distplot in Figure 4, phishing URLs are likely to have a shorter life than of legitimate ones. Even

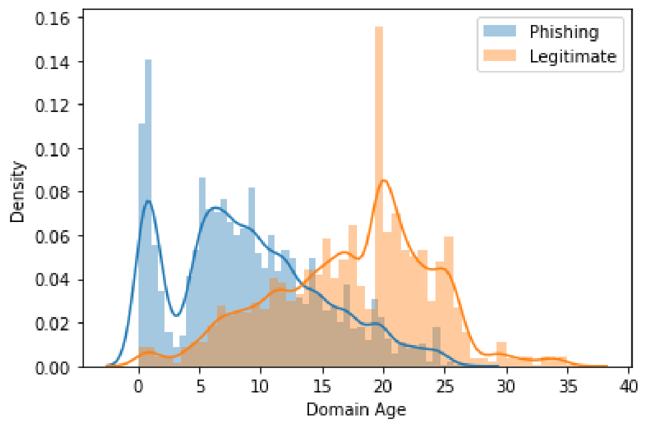


FIGURE 4. Analysis of normal distribution of URL domain age using *distplot* visualization.

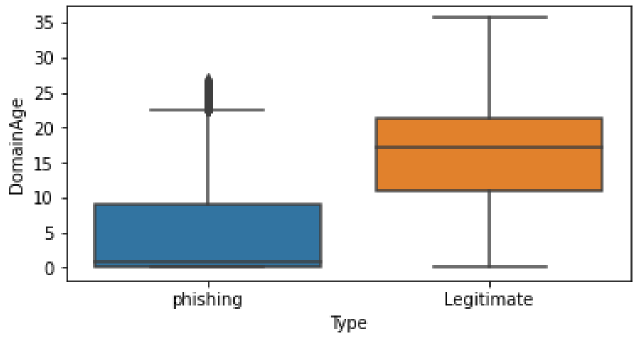


FIGURE 5. Analysis of normal distribution of URL domain age using *boxplot* visualization.

so, it was difficult to determine the best threshold value for distinguishing phishing and legitimate URLs based on the domain ages in the distplot. After the box plot was evaluated, the most suitable threshold value for the separation process was determined to be 10. It was evident that the median value of the domain age for phishing URLs was close to 0, and that of valid URLs was close to 20. It was also possible to identify certain anomalies; however, they were ignored at that stage

$$f(\text{url}) = \begin{cases} \text{Phishing} & \text{If domain age} <= 10 \text{ (months)} \\ \text{Legitimate} & \text{Otherwise.} \end{cases} \quad (8)$$

C. PAIR PLOT VISUALIZATION

Pair plot visualization included in the Python *Seaborn* library is used to gain an interpretation of the nature of a dataset. A *pair plot* calculates by a variant combination between every feature combination and plots the results in a 2D diagram. Significant overlapping was observed after the output shown in Figure 6 was analyzed. The theoretical background of *logistic regression (LR)* shows that it creates a straight line to divide data points. Since the analysis showed significant overlapping, it was challenging to create a proper straight line through the data points using algorithms like *LR*. Thus, linear algorithms are not recommended to use for this kind of problem statement.

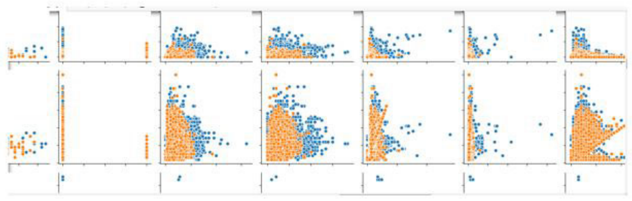


FIGURE 6. Part of the plot of the comparison of variant combinations between every feature using *pair plot* visualization.

D. CLASSIFIER SELECTION

As our dataset contained more overlaps, a decision was made to go through a list of nonlinear classification algorithms beneficial for building different ML models. *Decision trees (DTs)*, *random forests (RFs)*, *k-nearest neighbours (KNN)*, and other nonlinear classification algorithms could quickly solve this problem. (Note: In Section III, it is justified why *LR*, a linear classification algorithm, was not suitable for this problem statement). Thus, in this study *DT*, RF, XgBoost (*XGB*), AdaBoost (*ADB*), *KNN*, *GB*, and *LR* classifiers were considered for the experiments.

E. BEST N MODULE

A study by Sahingoz *et al.* [10] tried to split meaningless lengthy words into meaningful subwords and extract features by splitting words. Those split words might have produced potential connectivity with phishing URLs. Their study used a fixed heuristic-based threshold value (N), $N = 7$ for all classifiers, which meant that only the words longer than seven characters were split into subwords. Our study determined the best heuristic-based threshold value for each classifier used and compared the accuracy score for each classifier using the newly obtained values with the initial accuracy using the $N = 7$ threshold value. For this experiment, two different URL datasets ($DataSet_1$, $DataSet_2$) and two different experiment test modes (*percentage split* and *k-fold cross-validation*) were used.

As a first step, for executing the best N module, sub-datasets were constructed using $DataSet_1$ and $DataSet_2$ by changing the threshold value $7 > N > 3$ for each dataset using a preprocessor. At the end of this, five different sub-datasets were created for each *DataSet*.

1) IMPORTANCE OF DIFFERENT HEURISTIC BASED THRESHOLD VALUES

Heuristic values can have a great effect on the final outcome of ML modules. In our analysis, the values of the split word-related features (split word count, average split word length, Etc.) were totally dependent on the N value. Table 3 demonstrates a practical example of the importance of using different N values.

A brief explanation of two methods are defined in Table 4 Finally, we select the best N value by analyzing both outputs from the mentioned two methods in Table 4 for each classifier and select the sub dataset extracted using each classifier's best N for further experiments. The complete process is shown in Figure 7.

TABLE 3. Example of splitting words into subwords with different *N* values.

i | n | t | o	This word will split into two words only if the threshold value set as 3. "In" and "to" are stop words which can be useful in extracting feature values and stop words count might be an important feature.

$$\text{IF}\begin{cases} N = 3 \rightarrow \text{algorithm is splitting all the remain words which has length greater than 3 and others are left} \\ N = 4 \rightarrow \text{algorithm is splitting all the remain words which has length greater than 4 and others are left} \\ \vdots \\ N = 10 \rightarrow \text{algorithm is splitting all the remain words which has length greater than 10 and others are left} \end{cases}$$

TABLE 4. Two test modes used for best *N* module.

No	Method	Explanation
1	Percentage split	Each sub dataset in both main datasets is executed and one sub dataset is executed 3 times and get the average accuracy with reshuffling the dataset in each time.
2	K-Fold cross validation	Each sub dataset in both main datasets is executed and calculated the accuracy rate using K-Fold cross validation when K = 10.

F. HYPERPARAMETER TUNING FOR CLASSIFIERS

While creating a ML model, generally, we do not promptly have an idea of what the ideal model architecture should be for a given model, and, thus, we might need to investigate a range of possibilities. In real ML style, we preferably request that the machine carry out this investigation and select the ideal model design consequently. Parameters which characterize the model design are referred to as *hyperparameters* and the searching the optimal model architecture is called as *hyperparameter tuning*.

After obtaining model accuracies for each classifier using each best N value, hyperparameter tuning was performed prior to running the feature selection module. After the feature selection procedure was completed, two library functions *GridSearchCV* and *RandomizedCV* were used to re-execute hyperparameter tuning for the optimal features.

G. FEATURE SELECTION MODULE

Machine learning algorithms build models by learning from data with various features. Dataset features affect the training time and effectiveness of ML algorithms because they are entirely dependent on those features. Preferably, in the dataset, only the features that help the ML framework to learn information are retained. Excessive and repetitive characteristics increase an algorithm's training time and reduce its efficiency. Therefore, to minimize features and dimensionality, six different feature selection strategies were used to choose optimal features from the initial feature set. The step-by-step process for the complete selection of features is briefly outlined in Table 5, and the techniques for selecting features are listed in Steps 3 to 8. For the optimal feature selection process, *Algorithm 1* demonstrates the proposed algorithm. It is observed that the proposed algorithm has a complexity of $O(n)$. This is crucial as many devices have limited processing capabilities.

In accordance with the algorithm, the model accuracy was calculated at the end of each feature reduction technique and

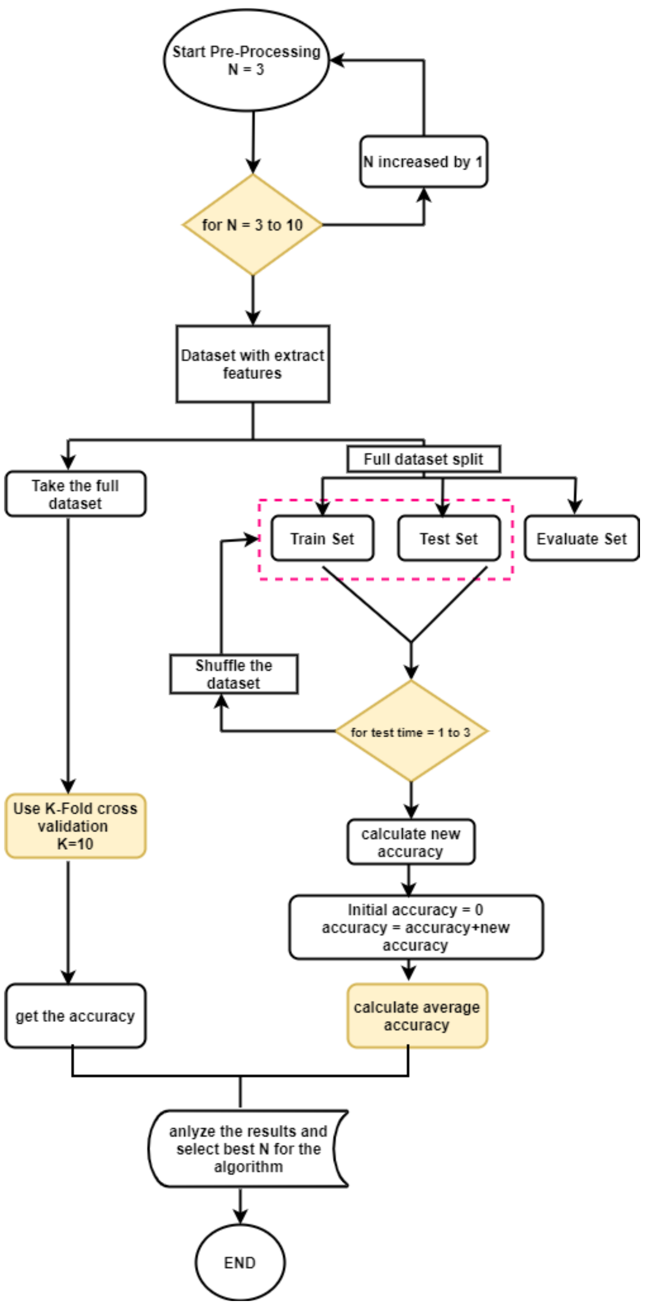


FIGURE 7. Best *N* module process.

compared with the initial accuracy. If the current accuracy score was more than 0.5% lower than the original accuracy, the last feature reduction method was ignored, and the techniques up to the ignored one were considered. That procedure was executed separately for all classifiers because they had different subsets from different N values.

H. SELECTING THE BEST MACHINE LEARNING MODEL

Features were reduced sequentially by using six feature engineering techniques. Each ML model's prediction accuracy and computation time were calculated at the end of each technique. After the feature selection process, the top four

TABLE 5. Feature selection process with required python libraries and functions in brief.

Steps	Libraries	Functions
Step 1: Collect the URL dataset	Manual	-
Step 2: Divided the collected dataset into 3 parts (train, test, evaluate)	Automated/sklearn.model_selection	train_test_split
Step 3: Remove constant features which are features having same values as outputs.	Automated/sklearn.feature_selection	VarienceThreshold,
Step 4: Remove quasi constants which are similar to constants with large sub set of same output.	Automated/sklearn.feature_selection	constant_filter
Step 5: Remove duplicate features which are having similar values for each feature.	Automated/sklearn.feature_selection	VarienceThreshold,
Step 6: Remove correlated features.	Automated	quasi_constant_filter
Step 7: Remove using *ANOVA* test.	Automated/sklearn.feature_selection	transform(T)
Step 8: Remove using *CHI squared* test. The feature selection is performed according to the score under fisher criterion.	Automated/sklearn.feature_selection	corr ()
Step 9: Check the feature importance of the remain features for each algorithm.	Automated	f_classif ()
Step 10: Remove least important features one by one for all machine learning models.	Manual/automated	chi2 (chi-square test)
Step 11: After every removal of least important feature, compare the current accuracy with the accuracy after step 8.	Manual/automated	chi2 (), pd.series
Step 12: Remove features until the final accuracy percentage goes down not less than 0.5% for each model.	Manual/automated	drop ()

Algorithm 1 Adaption of Best Features

```
Data: $\theta$
Notations:
    $\theta$ = dataset
    $\theta_{train}$ = training dataset
    model($\theta_{train}$) = model trained with training dataset
    Accuracy(model($\theta_{train}$)) = Accuracy of the model trained
with the training dataset
Result: Model with the highest accuracy model($\theta_{train}$)
1  K $\longleftarrow$ Accuracy(model($\theta_{train}$))
2  preprocessingSteps $\longleftarrow$ [removeConstants,
                                  removeQuasiConstants,
                                  removeDuplicates,
                                  removeCorrelatedFeatures,
                                  removeFeaturesUsingAnovaTest
                                  removeFeaturesUsingChi2Test]
3  previousModel $\longleftarrow$ currentModel $\longleftarrow$ model($\theta_{train}$)
4  Load raw URL dataset
5      Split Data into $\longrightarrow$ Training: Test
6  Function GenFnRecursive ($\theta_{train}$)
7      for $i \leftarrow 1..6$ do
8          $\theta_{train}$ $\longleftarrow$ preprocessingSteps[i]
9          currentModel $\longleftarrow$ model($\theta_{train}$)
10         if (Accuracy(model($\theta_{train}$)) - K) < 0.5 then
11             return previousModel
12         else
13             previousModel $\longleftarrow$ currentModel
14         end
15     end
16     return currentModel
17 end
```

classifiers that provided the highest accuracy scores were chosen. Then, more features were eliminated one by one by considering the feature importance of each classifier until the accuracy of each model was reduced by a maximum of 0.2% from the current accuracy. This experiment was also done using *DataSet*1 and *DataSet*2. The classifier that provided the highest accuracy from both datasets with the least number of features and minimum computing time was chosen as the best initial individual model. The optimal feature count for both datasets was determined. After the best model was selected, other significant performance evaluation metrics were measured, such as *precision*, *recall*, *ROC-AUC* (Area Under the Receiver Operating Characteristic Curve), and other required metrics for all classification models. Hyper-parameter tuning was done again with the minimized feature set. The final best individual ML model was chosen, and a lightweight URL preprocessor was proposed after the outcomes of all performed experiments were considered.

I. BUILDING ENSEMBLE MODEL

Multiple classifiers are combined to form ensemble methods to obtain better efficiency. Ensemble strategies benefit from the advantage of achieving improved results with two or more classifiers. It may also be argued that Ensemble models include multiple single models and form a high-capacity system with higher versatility relative to single models. Ensemble approaches are becoming more widely known because of their high capacity potential, flexibility, reliability, and competence. In this paper, two ensemble models using two ensemble techniques (*stacking* and *voting*) are proposed.

1) STACKING

"Stacking" is a machine learning algorithm which includes multiple predictions from several ML models by integrating each other.

2) VOTING

This is generally uses of classification problem statements. Multiple predictions from several ML models are deemed as a "vote" and the final prediction is based on the highest probability.

Model 1: Model 1 used the stacking classifier with two layers using the seven classifiers used for this study. It also

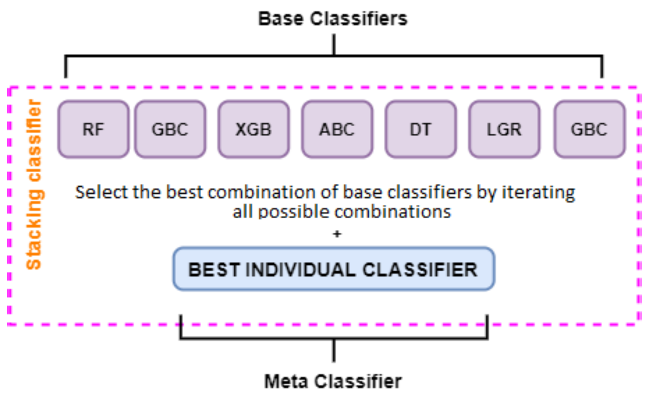

Base Classifiers
Stacking classifier
RF
GBC
XGB
ABC
DT
LGR
GBC
Select the best combination of base classifiers by iterating all possible combinations
+
BEST INDIVIDUAL CLASSIFIER
Meta Classifier

FIGURE 8. **Proposed methodology using stacking classifier.**

used the best individual classifier that was found by previous experiments as the meta classifier. Finally, the best combination of base classifiers was determined by using all base classifier combinations to obtain the best prediction accuracy. Figure 8 demonstrate the *Model 1* architecture.

Model 2 (Expandable Random Gradient Stacked Voting Classifier- ERG-SVC): Model 2 used the *voting classifier* that was used to build the model architecture like a stack to make an accurate prediction. The Model 2 architecture is shown in Figure 9. It was completely dynamic, expandable, and based on the number of base classifiers. This meant that the number of base classifiers (BC) could be increased, however, must be 2^k where $k = \{1, 2, 3, 4, \ldots, n\}$ where k defines the depth of the model. In order to achieve the accuracy of the prediction, the number of classifiers in an ensemble model has a significant impact [31]. In our proposed ERG-SVC model, two key classifiers have been utilized for building the architecture. However, four pairs of the two standard classifiers (BC_1, BC_2) were employed, and seven voting classifier objects with standard classifier, pairs were amalgamated. Although the same classification was used numerous times, a total of 15 classifiers (ensemble size) were utilized in the proposed model. The, base classifier is given by

$$BC_s = BC_1, BC_2, BC_3, BC_4, \ldots, BC_{2^k-1}, VC_{(2^n/2)-1}. \quad (9)$$

3) LEVEL 0 (BASE LAYER)

Level 0 is the layer used to combine base classifier pairs. The combination of classifiers at the base layer is given by

$$\sum_{k=1}^{k}(BC_{2^k-1}, BC_{2^k}) * k \quad (10)$$

where k is the number of pairs. It is possible to utilize the pairs using patterns where: (I) the classifiers used in the first pair of BCs can be used for all the other BC pairs in the architecture, such as ((BC_1, BC_2), (BC_1, BC_2)), (II) all BC pairs could have different classifiers, such as ((BC_1, BC_2), (BC_3, BC_4)), (III) a classifier of a particular BC pair could be repeated for other BC pairs in the architecture, such as ((BC_1, BC_2), (BC_1, BC_3)).

In this study, the pattern (I) architecture was used as discussed above with four classifier pairs, as shown in Figure 9. Six different models using this architecture were examined with BC_1 as GB and BC_2 as the other six classifiers, one at a time. Finally, the best combination of classifiers was selected and regarded as the best BCs for the proposed architecture.

4) (LEVEL $1, 2, \ldots, n$) (MIDDLE LAYERS AND FINAL LAYER)

The middle and final layers were responsible for selecting the best prediction class using base layer classification pairs. Each pair of the BCs was combined using *soft voting* criteria where the class label relied on the argmax of the sum of the predicted probabilities. A particular problem statement can have any number of target classes. In the case of our study, it was either phishing or legitimate. Soft voting always selects the target class that provides the highest probability as the final prediction. For the purpose of explanation, assume that a particular problem statement has a q number of $(1, 2, \ldots, q)$ target classes as shown in Figure 10.

P_1 = Average probability of *Class 1*
P_2 = Average probability of *Class 2*
P_q = Average probability of *Class q*.

Voting Classifier (VC) selects its prediction using the highest average probability of target classes

$$VC = \text{argmax}\{P_1, P_2, \ldots, P_q\} \quad (11)$$

where P_1 was the average probability by BC_1 and BC_2 for *Class 1*, $P2$ was the average probability by BC_1 and BC_2 for *Class 2*, P_q was the average probability by BC_1 and BC_2 for Class q, and VC was the combined prediction by BC_1 and BC_2 using the soft voting criteria. This process was repeated in every BC pair, and if more than one middle layer was used, the same process was repeated in each middle layer pair as well. We computed the proposed model architecture

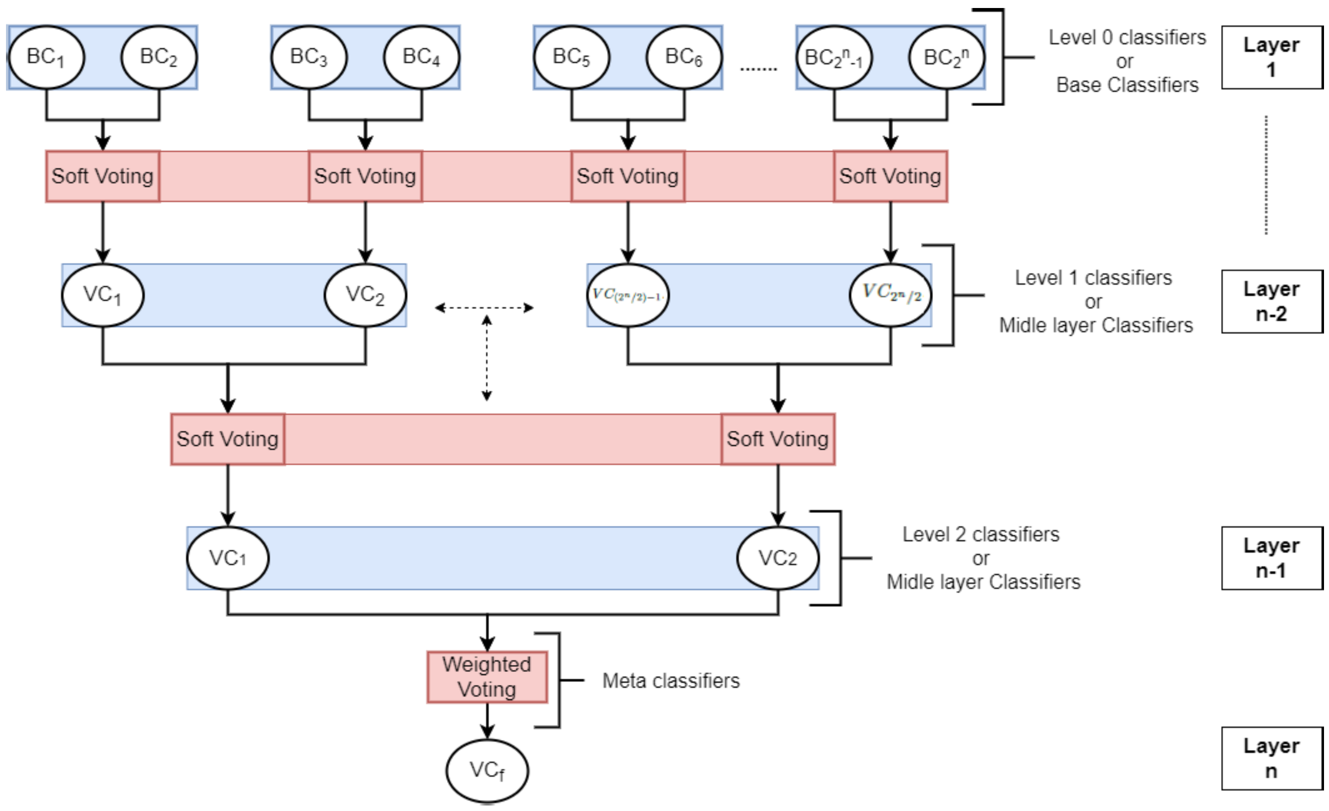
$$\text{ERG-SVC} = \sum_{i=1}^{l}\left\{\text{argmax}\left(\sum_{j=1}^{2^k/2}(BC_1, BC_2)_j\right)\right\}_i \quad (12)$$

where l was the number of layers and *ERG-SVC* was the final prediction by the proposed model. A detailed explanation of the proposed model is elaborated using an example as shown in Figure 11.

In this study, *DataSet*1 predicted that its class belonged to either the phishing class or the legitimate class. Figure 11 shows that there were two pairs of BCs BC_1, BC_2, BC_3 and BC_4. BC_1 predicted the probability for the legitimate class (α_1) and the phishing class (β_1) for data in DataSet1. Similarly, BC_2 predicted the probability for the legitimate class (α_2) and the phishing class (β_2)

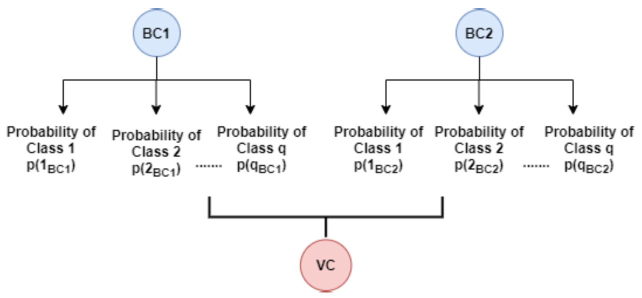
α_1 = probability of BC_1 for predicted legitimate class
β_1 = probability of BC_1 for predicted phishing class
α_2 = probability of BC_2 for predicted legitimate class
β_2 = probability of BC_2 for predicted phishing class.

The soft voting criterion used the average probabilities of each class. Accordingly, it computed *Pair 1* average



BC1
BC2
BC3
BC4
BC5
BC6
BC2^n-1
BC2^n
Level 0 classifiers or Base Classifiers
Layer 1
Soft Voting
VC1
VC2
VC(2^n/2)-1
VC2^n/2
Level 1 classifiers or Midle layer Classifiers
Layer n-2
Level 2 classifiers or Midle layer Classifiers
Layer n-1
Weighted Voting
Meta classifiers
VCf
Layer n

FIGURE 9. Proposed architecture using voting classifier.



BC1
BC2
Probability of Class 1 p(1BC1)
Probability of Class 2 p(2BC1)
Probability of Class q p(qBC1)
Probability of Class 1 p(1BC2)
Probability of Class 2 p(2BC2)
Probability of Class q p(qBC2)
VC

FIGURE 10. Process of a middle layer of proposed architecture using voting classifier.

probabilities of each target class

$$\text{Probability of legitimate class } (\textit{Pair 1}) = p_{\alpha_{p1}} = \frac{\alpha_1 + \alpha_2}{2} \tag{13}$$

$$\text{Probability of phishing class } (\textit{Pair 1}) = p_{\beta_{p1}} = \frac{\beta_1 + \beta_2}{2}. \tag{14}$$

Hence, as highlighted in Figure 11, the combination of BC_1 and BC_2 provided its voting classifier prediction for Pair 1 (VC_1) as

$$VC_1 = \operatorname{argmax}\{p_{\alpha_{p1}}, p_{\beta_{p1}}\}. \tag{15}$$

The class with the highest probability was selected as the predictive class. If $p_{\alpha_{p1}}$ had the highest probability score, then

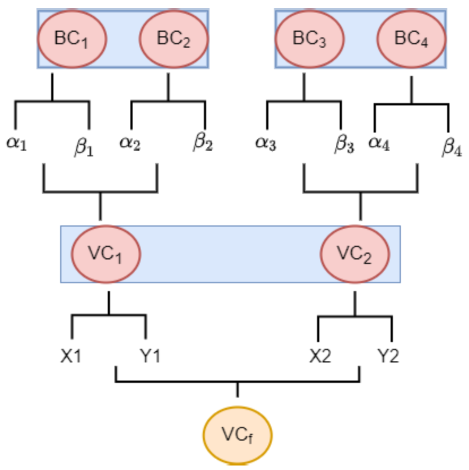


FIGURE 11. ***ERG-SVC*** **architecture elaboration using 2 classifier pairs.**

"legitimate" would the final class

$$VC_1 = p_{\alpha_{p1}}. \tag{16}$$

The same flow was then carried out for BC_3 and BC_4 to find the highest probability class from BC_3 and BC_4 for the second pair (*Pair 2*)

α_3 = probability of BC_3 for predicted legitimate class
β_3 = probability of BC_3 for predicted phishing class
α_4 = probability of BC_4 for predicted legitimate class
β_4 = probability of BC_4 for predicted phishing class.

Again, the soft voting criterion used the average probabilities of each class. Accordingly, it computed *Pair 2* average probabilities of each target class

$$\text{Probability of legitimate class } (\textit{Pair 2}) = p_{\alpha_{p2}} = \frac{\alpha_3 + \alpha_4}{2} \tag{17}$$

$$\text{Probability of phishing class } (\textit{Pair 2}) = p_{\beta_{p2}} = \frac{\beta_3 + \beta_4}{2}. \tag{18}$$

Hence, as highlighted in Figure 11, the combination of BC_3 and BC_4 provide it's voting classifier prediction for Pair 2 (VC2) as

$$VC_2 = \text{argmax}\{p_{\alpha_{p2}}, p_{\beta_{p2}}\}. \tag{19}$$

Assuming that the probability of the phishing class was higher than that of the legitimate class, the final prediction by VC_2 for the phishing class was given by

$$VC_2 = p_{\beta_{p2}}. \tag{20}$$

Then, the final prediction using the voting classifier (VC_f) was made using the probability score for each target class by VC_1 and VC_2 where the probability of VC_1 for the predicted legitimate class was equal to X1, and the probability of VC_1 for the predicted phishing class was equal to Y1.

probability of VC_1 for predicted legitimate class $= X_1$

probability of VC_1 for predicted phishing class $= Y_1$.

Once again, soft voting criteria used average probabilities for compute X_1 and Y_1

$$X_1 = (p_{\alpha_{p1}} + p_{\alpha_{p2}})/2 \tag{21}$$

$$Y_1 = (p_{\beta_{p1}} + p_{\beta_{p2}})/2. \tag{22}$$

Similarly for VC_2

probability of VC_2 for predicted legitimate class $= X_2$

probability of VC_2 for predicted phishing class $= Y_2$.

VC_f makes final prediction using average of VC_1 and VC_2 probabilities as

$$\text{probability of legitimate class} = P_X = \frac{X_1 + X_2}{2} \tag{23}$$

$$\text{probability of phishing class} = P_Y = \frac{Y_1 + Y_2}{2}. \tag{24}$$

Final predicted class (VC_f) was selected using the considering the highest probability

$$VC_f = \text{argmax}\{P_X, P_Y\}. \tag{25}$$

Assuming P_X provides the highest probability and then this model (*ERG-SVC*) selected the final predicted class as legitimate

$$VC_f = P_X. \tag{26}$$

We used the weighted voting mechanism for the final prediction of the proposed model. Weighted average ensembles provide better capability and contribute to predictions. The final prediction was accomplished with the use of weight voting techniques in our *ERG-SVC* model. After improving the weights of the Layer 2 base classifiers, the proposed technique uses the weighted voting combination rule to aggregate the final output from Layer 2 classifiers. Hence, the final prediction by the four base classifier combinations with our proposed model is in a legitimate class according to the example elaborated.

J. CLUSTERING

This study used a clustering algorithm to distinguish the data points and assign data points to their groups. This was done for all sets of data points. Theoretically, data points are a set of data that belong to different categories or groups with somewhat different properties or features or both. The set of data that belongs to the same category or group has identical properties and characteristics. To find these similarities, a k-means algorithm was used. The easiest clustering algorithm is k-means and, therefore used for visualizing the clusters. Only *DataSet*1 was used to compare the results before and after the feature reduction process.

K. DATA ANALYSIS

In this study, we utilize the malicious URL detection accuracy using three different methods (single base machine learning model, ensemble model, cluster analysis), and evaluate and compare the final outputs once all the experiments are completed in order to make decisions. We use a Windows 10 computer with Processor Intel(R) Core(TM) i7-6500U CPU @ 2.50 GHz, 2601 MHz, 2 Core(s), 4 Logical Processor(s), 8 GB DDR3 RAM and Jupyter NoteBook and Pycharm, which are third-party applications.

III. RESULTS

This section summarizes the outcomes and the aim of this study, which was to build up a robust ML model to detect phishing URLs. Various classification, ensemble, and clustering algorithms were used to measure the rate of detecting malicious URLs. Two test modes, six performance evaluation methods of classification algorithms, and two evaluation metrics of clustering and model computation time were used by composing three different datasets for our evaluation process.

A. BEST N MODULE

The best N module was executed using two test modes (*k-fold cross-validation* and *percentage split*) while composing both *DataSet*1 and *DataSet*2 to obtain the optimal N value for each classifier. As per the results shown in Table 6, it was observed that the same N value was not provided for each classification algorithm, and the results of the two datasets were likely similar for each tested classifier using both test modes. Some algorithms obtained the same N value for all datasets, whereas others obtained more than one N value (*RF* and *AdaBoost*). Hence, the best N value was selected after the N values from both methods were analyzed using both datasets for each classifier.

Table 7 compares the prediction accuracies using N values found by our experiments and using $N = 7$. It was noticed

TABLE 6. Observations of best *N* values after executing two test modes using *DataSet*$_1$ and *DataSet*$_2$.

Algorithm	N value using Method 1		N value using Method 2		Estimated best N value
	Dataset 1	Dataset 2	Dataset 1	Dataset 2	
Decision tree	3	3	3	3	3
Random forest	7	7	5, 6, 7	7	7
XgBoost	5	5	5	5	5
AdaBoost	4, 6	4	4, 6	4	4
Gradient Boost	7	7	7	7	7
KNN	4	3, 4	4	4	4
Logistic Regression	6	5,6,7	6	6, 7	6

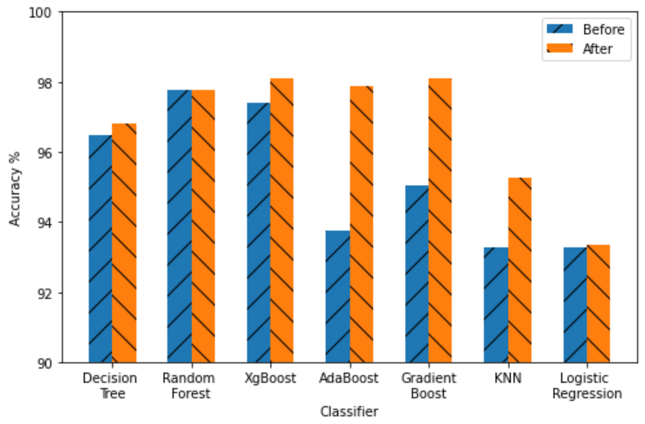


FIGURE 12. Comparison of accuracies for each classifier before and after *HyperParameter Tuning*.

that the accuracy values were somewhat higher (approximately 0.3%) when experimented N values were used instead of $N = 7$. For places like banks or any financial organization, an increase in accuracy of even 0.1% is significant because cyber-attack can have severe consequences. Hence, a 0.3% increase was considered to be remarkable.

B. HYPER PARAMETER TUNING

Hyperparameter tuning was performed next to the best N module. Figure 12 compares prediction accuracies before and after parameter tuning. The analysis determined that the accuracy rates of some algorithms (*AdaBoost*, *GB*, *KNN*) increased by a considerable percentage, whereas the rest of the algorithms showed only a tiny percentage increase.

C. FEATURE SELECTION PROCESS

Certain undesirable features that had only a minor effect on final accuracy scores were removed by the sequential feature selection process, as shown in Table 5. No constant features were found for the subdatasets with N values (N = 3, 4, 5, 6, 7), and for all sub datasets, quasiconstants and duplicated feature counts were same. However, it was found that the correlated feature counts (correlation >0.8) were different for some sub datasets (datasets with N = 3, 4, 7). Even so, the counts were identical for other sub datasets. The Analysis of Variance (*ANOVA*) test and *CHI-squared test* also eliminated almost the same features from all sub datasets after the experiments.

Table 7 shows the shift in accuracy score from the initial state (before the best N module was executed) to the end of the feature selection process. A marked increase in accuracy scores was noticed after hyperparameter tuning and the addition of eight new features. After the feature reduction process was completed, it was found that even after eliminating more than 20 features for all classifiers, prediction accuracies were not severely reduced. Table 7 shows the accuracy values after each feature reduction technique for every classifier. The initial 46 features were reduced by up to 27 for some subdatasets (N = 3, 7) and by 26 for other subdatasets (N = 4, 5, 6). By contrast, after diminishing features it was noticed that the *KNN* accuracy improved by approximately 1%, and the *GB* classifier also raised its accuracy, while the accuracy of the other five algorithms was reduced by a small proportion. The highest accuracy reduction percentage was 0.2% in *LR*.

D. SELECT BEST INDIVIDUAL ML MODEL

Table 8 lists the top four classifiers that had the highest prediction performance after the feature selection process; these were more than 97% accurate and had less fluctuation compared with other classification algorithms. Further investigation and attempts were made to optimize the training of ML models with optimal features by eliminating more features using the feature importance of each classification algorithm. The goal was to preserve the prediction accuracy with a reduction percentage maximum of 0.5%. This was crucial to recognize highly significant primary parameters. Before this process was done, the accuracy of the predictions was compared using *DataSet*1 and *DataSet*2 with the final feature counts of each algorithm after the initial feature selection process. Table 8 shows the results of the experiments. Based on the performance, it was found that the *GB* classifier provided a higher prediction accuracy with the least number of features (22 features) for both datasets relative to other algorithms. The *GB* model was therefore chosen as the best individual ML model. This introduced a simpler preprocessor to acquire the most accurate predictions by extracting the least number of features from the dataset. It took approximately 109 s to run the model with 46 features; however, that was reduced to 57 sec after the optimal feature reduction process had been completed. This equates to a computation time reduction of approximately 48%, which is a remarkable result. In this case, even after reducing the time by 57 sec, the computational time is slightly high because of the very large dataset used.

E. LIGHTWEIGHT PRE-PROCESSOR

Preprocessing time is one of the key variables in obtaining real-time results. In accordance with our trials, a lightweight preprocessor was proposed after the optimal feature selection process. Calculations showed that the preprocessing time was reduced by 7% when the lightweight preprocessor included 22 features, in contrast to the initial preprocessing time that included 46 features. The time taken to extract domain age was not considered because it depended on the internet speed at that moment. Figure 13 shows the final simplified preprocessor with features extracted (shown in orange boxes) at each stage, and Table 9 briefly shows, the

TABLE 7. Classification results (Accuracy %) in each experimental level and feature selection process for each classifier and highest value in each level is highlighted.

Analysis	*Decision tree*	*Random forest*	*XgBoost*	*AdaBoost*	*GradientBoost*	*KNN*	*Logistic Regression*
Before best N	94.360	**96.362**	94.991	90.068	91.537	91.166	87.751
After best N	94.488	**96.362**	95.076	90.177	91.533	91.185	87.798
After new features	96.481	**97.769**	97.385	93.76	95.047	93.284	93.27
After hyper parameter	96.818	97.779	98.086	97.87	**98.091**	95.282	93.34
Feature reduction methods	*Decision tree*	*Random forest*	*XgBoost*	*AdaBoost*	*GradientBoost*	*KNN*	*Logistic Regression*
STEP 1 : After removing constatnts	96.818	97.779	**98.086**	97.87	98.012	95.257	93.376
STEP 2 : After removing quasi constants	96.739	97.822	97.825	97.802	**98.007**	95.255	93.389
STEP 3 : After removing constants	96.725	97.807	97.865	97.815	**98.005**	95.306	93.391
STEP 4 : After removing correlated	96.695	97.795	97.616	97.826	**98.094**	96.008	93.333
STEP 5 : After *ANOVA* test	96.68	97.815	97.605	97.758	**98.102**	96.188	93.246
STEP 6 : After *CHI-square* test	96.691	97.761	97.643	97.783	**98.130**	96.189	93.103

TABLE 8. Top four ML models with an accuracy of greater than 97% after removing unwanted features with final feature counts for *DataSet*1 and *DataSet*2.

Algorithm	**Feature Count After six Techniques**	**Accuracy of Dataset 1**	**Acuuracy of Dataset 2**	**Feature Count After Reducing More Features**	**Accuracy of Dataset 1**	**Accuracy of Dataset 2**
Random forest	27	97.761	94.255	25	97.757	94.10
XgBoost	26	97.643	93.93	24	97.413	93.812
AdaBoost	26	97.783	93.385	24	97.49	93.284
Gradient Boost	27	**98.13**	**94.322**	**22**	**98.118**	**94.11**

preprocessing steps and Python libraries and functions used for the proposed lightweight preprocessor.

F. PERFORMANCE EVALUATION

Previous experiments concluded that the *GB* classifier granted the best individual ML model by analyzing prediction accuracies using $DataSet_1$ and $DataSet_2$. Even so, for assessing the final version model, different extra measures might have been helpful. One method used to pick the most suitable prediction model is known as Receiver Operating Characteristics (ROC) curves. Therefore, this analysis used precision, recall, recall (true positive rate), precision, accuracy, and the region under the ROC curve.

As described in sections III-A and III-B, the prediction accuracies of all classification algorithms were analyzed after selecting the best N, adding new features, hyperparameter tuning, and executing the feature selection module. The k-fold cross-validation (when $k = 10$) was used for each classifier for previous evaluations. In terms of prediction accuracy, in accordance with the previous analysis, the *GB* classifier outperformed the accuracy by 98.118%. Table 10 shows the *accuracy* value change over various k-fold values, and Table 11 shows the Area Under the Receiver Operating Characteristic Curve (*ROC-AUC*) value change over various k-fold values. It was observed that accuracy values and the *ROC-AUC* did not change by a large percentage when the k-fold values increased. Hence, the k-fold value was set at 10 for further experiments for each algorithm.

The results in Table 7 show that six algorithms (*RF*, *DT*, *XgBoost*, *GB*, *AdaBoost*, *KNN*) performed better with respect to the model accuracy. On the other hand, compared to other algorithms, the *LR* algorithm was the least worth testing because its final accuracy score was 93%; very low relative to that of the other algorithms. According to our assumption at the initial stage of this analysis, *LR* might not be useful for achieving a higher accuracy rate due to the several overlaps found within this test. This shows that our assumption functioned as expected. Table 12 shows that the *GB* classifier outperformed the others in all important evaluation metrics (precision, recall, F1 score). The computation time of *GB* was slightly high, however, was negligible considering the size of the large dataset. Hence, the *GB* classifier was confirmed as the best individual ML model.

G. GRADIENT BOOSTING ANALYSIS

When considering the theoretical background of *Gradient Boosting* also known as *GBDT* (*Gradient Boost Decision tree*), it is capable of using both classification and regression problem statements and is built with three primary components (*loss function*, *weak learner*, *additive model*). One of

TABLE 9. Lightweight pre-processor steps and required Python libraries and functions.

Steps	Libraries	Function and Services
Step 1: Collect the URL dataset (phishing and legitimate)	Manual	Json.loads ()
Step 2: Split the URL into its main parts	Automated/tldextract	extract ()
Step 3: Extract features using URL	Automated/urllib/third party services	whosislookup (service), Alexa database (service), request. urlopen ()
Step 4: Extract features using top level domain	Automated/tldextract	extract ()
Step 5: Extract features using domain	Automated/tldextract	extract ()
Step 6: Extract features using sub domain	Automated/tldextract	extract ()
Step 7: Extract features using URL path	Automated/tldextract	extract ()
Step 8: Split domain, sub domain and path by special characters and extract raw words	Automated/regex	re. split ()
Step 9: Extract random words from the raw word set	Automated/Gibberish detector,2015	avg_transition_probe ()
Step 10: Remove digits of the remain words	Automated/regex	re.sub ()
Step 11: Select the word length greater than 7 from the remain words in raw word set	Automated	-
Step 12: Split the words in to sub words	Automated	-
Step 13: Select only sub words which are greater than 3 and others are removed	Automated	-
Step 14: Extract features from spitted words	Automated/regex	re.sub ()
Step 15: Extract features from word length less than 7	Automated/regex	Re.sub ()

TABLE 10. Accuracy value change over K-fold value.

Algorithm	Fold = 10	Fold = 20	Fold = 30	Fold = 40	Fold = 50	Fold = 60
Decision tree	96.69	96.74	96.73	96.73	96.72	96.73
Random forest	97.76	97.81	97.81	97.91	97.98	97.95
XgBoost	97.64	97.71	97.71	97.71	97.72	97.71
AdaBoost	97.78	97.79	97.80	97.80	97.71	97.74
Gradient Boost	**98.118**	**98.12**	**98.12**	**98.14**	**98.13**	**98.13**
KNN	96.18	96.23	96.23	96.24	96.24	96.24
LGR	93.10	93.10	93.10	93.10	93.10	93.10

TABLE 11. ROC_AUC value change over K-fold value.

Algorithm	Fold = 10	Fold = 20	Fold = 30	Fold = 40	Fold = 50	Fold = 60
Decision tree	96.69	96.71	96.73	96.73	96.72	96.73
Random forest	99.3	99.34	99.32	99.38	99.35	99.32
XgBoost	99.66	99.68	99.68	99.68	99.68	99.68
AdaBoost	99.38	99.39	99.44	99.44	99.4	99.46
Gradient Boost	**99.74**	**99.74**	**99.74**	**99.74**	**99.74**	**99.74**
KNN	98.65	98.68	98.68	98.68	98.69	98.68
LGR	93.10	93.10	93.10	93.10	93.10	93.10

the other important parameters used to calculate the quality of a split is "criterion". Three options have been suggested as values for this parameter and the default value for criterion is *Friedman_MSE*. Other options are *MSE* and *MAE* where *MSE* represents "mean squared error" and *MAE* represents "mean absolute error". We try to verify how the accuracy and *ROC-AUC* scores differ by replacing three criterion parameter options with two options (*deviance. exponential*) for the loss function. We follow the same procedure with $DataSet_2$, $DataSet_3$ and compare the results.

TABLE 12. Classification model performance using confusion matrix (Weighted Average). Test mode 10 fold cross validation.

Algorithm	Time	Precision	F1 - Score	Recall
Random forest	9.8 s	96.58	97.49	98.42
Decision tree	806 ms	95.81	96.55	97.31
KNN	13.5 s	95.09	96.26	97.46
XgBoost	5.49 s	96.89	97.72	98.55
AdaBoost	23.12 s	96.38	97.30	98.25
Logistic Regression	820 ms	92.15	93.16	94.19
Gradient Boost	57.2 S	**97.85**	**98.23**	**98.61**

Figures 14 and 15 show that the lowest accuracy was achieved when using the mean absolute error (*MAE*) as a criterion in comparison to those using *MSE* and *Friedman MSE* as criteria for all three datasets with "davience" as the loss function. Furthermore, that difference was very similar when using the option "exponential" as a loss function. Moreover, the same kind of output occurred when the ROC-AUC score was measured by replacing *MAE*, *MSE*, and *Frindman MSE* as the criterion parameter. Figures 16 and 17 show the results of *ROC-AUC* after changing *loss function* and *criterion* parameter.

By analyzing Figures 14, 15, 16, 17, it might be said that *MAE* is not a good option for a *GB* classifier for use as the criterion to obtain the best results. When considering the theoretical background of *gradient boosting*, it provides a prediction score using minimum squares, and if the accuracy must be increased, deviance or an exponential can be used as a loss function and *MSE* or *Frindman MSE* as a criterion.

H. ENSEMBLE MODEL

Table 13 shows the accuracy scores of the top six ensemble models using stacking model (model 1), whose accuracy

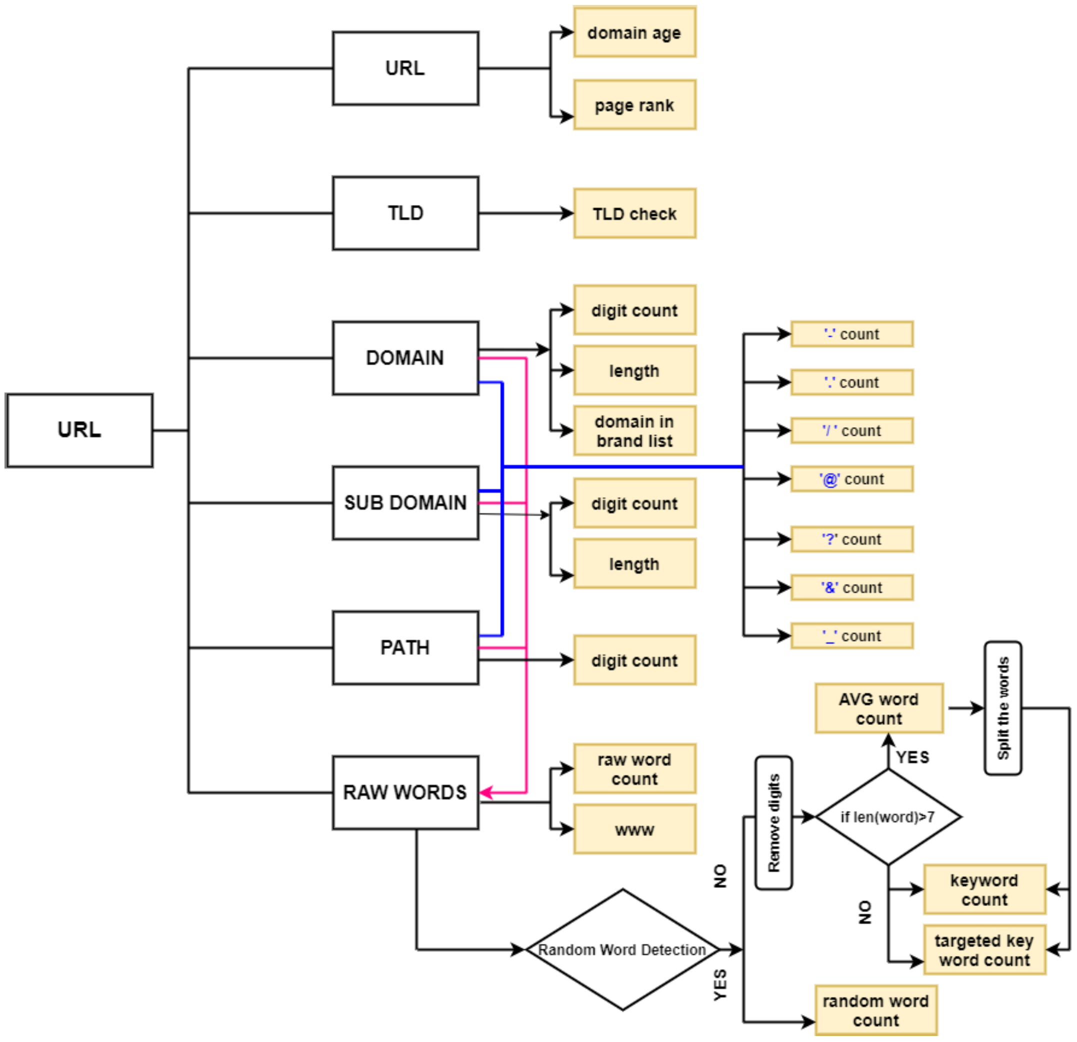


FIGURE 13. Proposed lightweight pre-processor.

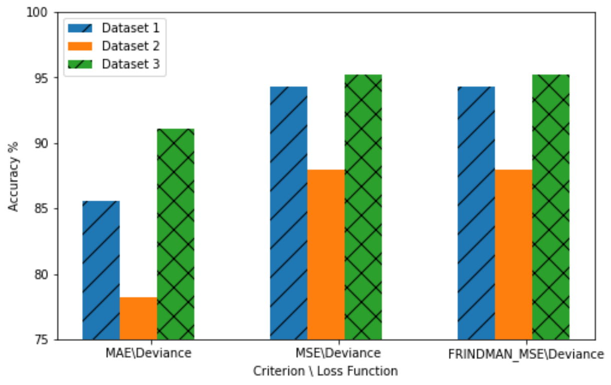


FIGURE 14. Observation of the accuracy value of *Gradient Boost* model change when "Loss Function" = "deviance" and "Criterion" with all options.

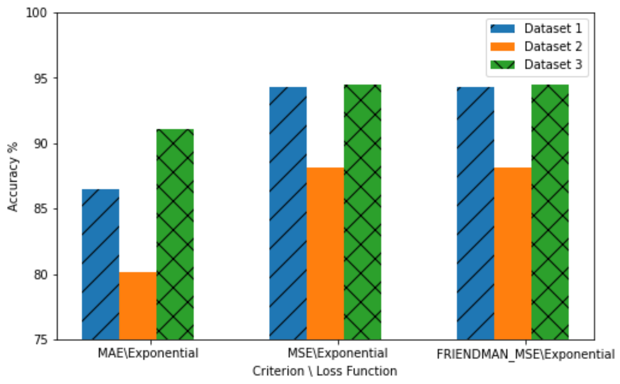


FIGURE 15. Observation of the accuracy value of *Gradient Boost* model change when Loss Function = "exponential" and "Criterion" with all options.

scores were higher than 98% using a *stacking classifier* out of 122 different ensemble models. For each stacked ensemble model, a *GB* algorithm was used as the meta classifier. According to the results obtained, the combination of *LR* and *GB* as base classifiers with the meta classifier (*GB*), offer a 98.23% accuracy score, outperforming all other ensemble models that use stacking.

At the same time, the proposed ensemble model architecture using voting classifier with *GB* and *RF* had a 98.27% prediction score, and *GB* with *LR*, *XgBoost*, and *AdaBoost*

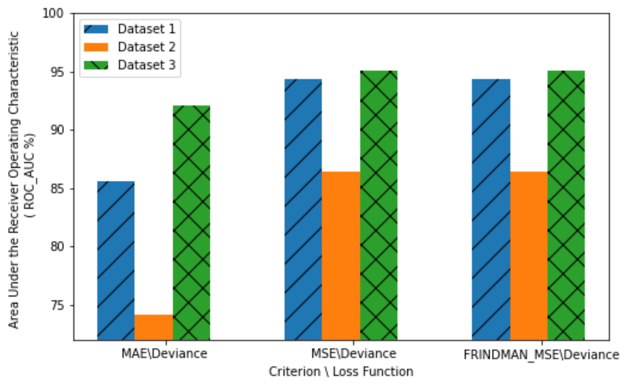


FIGURE 16. Observation of the *ROC_AUC* value of *Gradient Boost* model change when loss function = "deviance" and "Criterion" with all options.

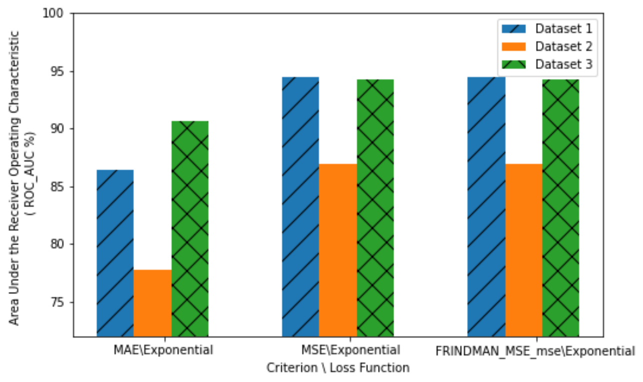


FIGURE 17. Observation of the *ROC_AUC* value of *Gradient Boost* model change when loss function = "exponential" and "Criterion" with all options.

TABLE 13. Accuracy scores which provides more than 98% with relevant classifier combinations using *Model 1*.

Group	XGB	ABC	KNN	DT	RF	LGR	GBC	Accuracy
1	-	-	-	-	-	yes	yes	98.23
2	-	-	-	-	yes	-	yes	98.19
3	yes	-	-	-	-	yes	yes	98.18
4	yes	-	-	-	yes	-	yes	98.17
5	yes	-	-	-	-	-	yes	98.17
5	yes	-	-	-	yes	yes	yes	98.13

classifiers also having high accuracy scores of greater than 98%. Even so, *GB* with *KNN* and *DT* had the lowest accuracy in this experiment. Table 14 shows all the prediction accuracy scores using tested pairs with ensemble Model 2.

I. CLUSTERING

K-means clustering was used as the unsupervised ML algorithm, in which all the visualization diagrams were created using it. The *silhouette* and *elbow method* analyses were used to determine the optimal number of clusters for the dataset. This study's analysis had the best outputs when $K = 4$ using the elbow analysis with 46 features as shown in Figure 18. The same number of cluster K value was provided using the same dataset with 22 features as well, shown in Figure 19.

TABLE 14. Accuracy scores of all possible classifier pairs with gradient boost using ensemble *Model 2*.

Group	GBC	RF	DT	KNN	XGB	LGR	ABC	Accuracy
1	yes	yes	-	-	-	-	-	98.27
2	yes	-	yes	-	-	-	-	96.74
3	yes	-	-	yes	-	-	-	96.65
4	yes	-	-	-	yes	-	-	98.18
5	yes	-	-	-	-	yes	-	98.11
6	yes	-	-	-	-	-	yes	98.15

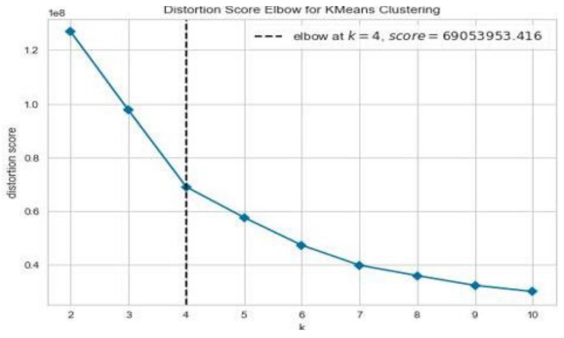


FIGURE 18. Elbow analysis using 46 features.

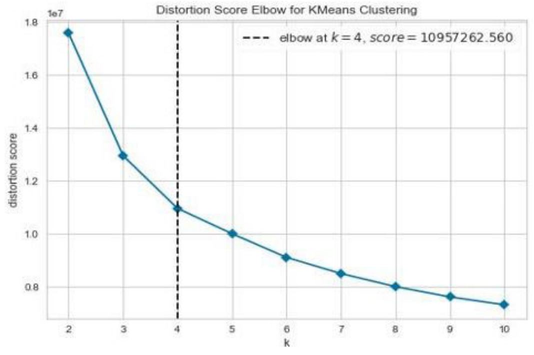


FIGURE 19. Elbow analysis using 22 features.

Silhouette analysis also shows the best number of clusters as four as shown in Figures 20 and 21. It is observed that, a 22 feature set provides the optimal division into four clusters. In Figure 21, we observe four clusters and conclude them be phishing, legitimate, suspicious emails and certain anomalies. Furthermore, it is noticeable that two of the clusters are large and roughly the same in size. From these characteristics, it is concluded that these two clusters are phishing and legitimate. This is because the $DataSet_1$ was used for this experiment and we are aware that the phishing and legitimate URL counts are almost the same. Therefore, we can assume Figure 21 provides a correct cluster output.

J. PERFORMANCE COMPARISON

Table 15 shows the performance comparison of our model with other models in considered papers.

K. STATISTICAL SIGNIFICANCE TEST

The statistical t-test was used to evaluate the significance of our proposed model. It determines whether the difference in

TABLE 15. Performance comparison of our method with other approaches in literature.

N	Method	Technique	System details	Programming language	No of data sets	Data size	Initial feature count	Final feature count	Third party features	Results	Data pre-processing time (Sec)	Computation time (Sec)
1	Feng et al. (2018) [19]	ML	Core i3 processor, 8 GB RAM, 64-bit Windows 7	Python	1	11,055	30	30	>2	97.71%	-	-
2	Yang et al. (2019) [32]	DL	64 GB RAM, a E5-2683 v3 CPU, and GTX 1080ti GPUs	-	2	44,835 for training, 1,965,964 for pre-processing	48	48	>2	98.99%	4000	298
3	Babagoli et al. (2019) [18]	ML	-	-	1	11,055	30	20	>2	96.32%	-	-
4	Sindhu et al. (2020) [15]	ML	-	-	1	11,055	30	30	>2	97.45%	-	-
5	Zamir et al. (2020) [16]	ML	Core i7, 8th Gen, 8 GB RAM	-	1	11,055	32	32	>2	97.4%	-	105.32
6	Wang et al. (2020) [33]	DL	-	-	1	2,456	30	30	>2	95.47%	-	-
7	**Our Method (ERG-SVC)**	**ML**	**Core (TM) i7-6500U, 4th Gen, 2.5 GHz, 8 GB DDR3 RAM**	**Python**	**3**	**75,000**	**46**	**22**	**2**	**98.27%**	**57**	**170**

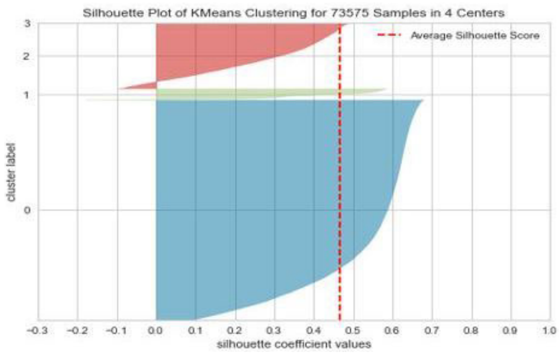


FIGURE 20. Silhouette analysis with 46 features.

the performance of the proposed *ERG-SVC* model is statistically significant. There are two hypotheses to legitimize the test: (I) Null hypothesis (*Ho*), where: the mean difference between paired observations is zero between the proposed models and other models; and(II) Alternative Hypotheses (*Ha*), where: the mean difference between paired observations is not zero.

The null hypothesis is rejected with respect to the obtained *p*-values shown in Table 16 in each time ($p < 0.05$) in favour of the alternative hypothesis based on the multiple t-tests between the proposed *ERG-SVC* model and other best five models (stacking classifier model, *RF*, *XgBoost*, *AdaBoost*, *GB*) using the same dataset (Confidence level: 5%). Moreover, internal consistency was measured using $Cronbach's\ \alpha$,

TABLE 16. Statistical t-test *p*-values (significance) with a single dataset.

Proposed Model	Paired Model	p-value (significance)
ERG-SVC	*stacking*	0.044
ERG-SVC	*GB*	0.033
ERG-SVC	*RF*	0.000
ERG-SVC	*XgBoost*	0.000
ERG-SVC	*AdaBoost*	0.000

which demonstrated that the *ERG-SVC* model has a higher α value than other models that were tested using different classifiers. Significance values shown in Table 16 for each paired t-tests verify that our proposed model (*ERG-SVC*) performs better than other models.

IV. DISCUSSION

Our proposed study, *GB*, outperformed other algorithms as an individual model with a prediction accuracy of 98.118%. In addition, *GB* demonstrates the best results for all other performance metrics. The model accuracy was improved in three levels (best *N* module, hyperparameter tuning, adding new features) and the final model was optimized by reducing undesirable features while maintaining the highest accuracy level.

The significance of the proposed model is that the use of only a few features for the final predictions can attain higher prediction accuracy than the model proposed by [10]

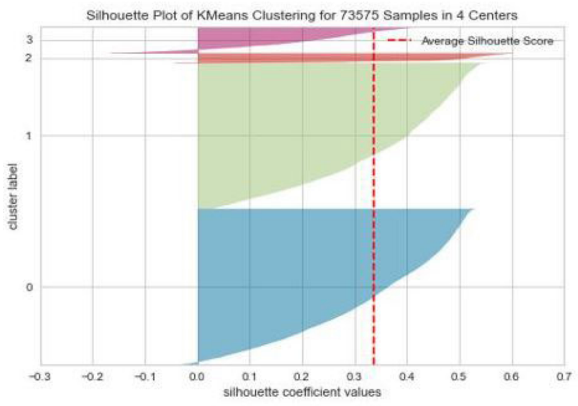


FIGURE 21. **Silhouette analysis with 22 features.**

that uses 40 features. Also, the *ERG-SVC* ensemble technique achieved the best outcome using 22 features. The proposed ensemble model using a *stacking classifier* achieved a 98.23% prediction accuracy rate, while the *ERG-SVC* model with the *voting classifier* achieved a 98.27% accuracy score using 22 features. The proposed model has few additional advantages. This is an entirely dynamic and expandable model which allows the addition or removal of voting classifier objects with respect to the user requirement. In terms of results, this has high precision, recall and F1-score and a significantly higher rate of success in detecting a new phishing URL, which was tested and validated using two datasets. Surprisingly the described ensemble architecture could be applied for any classification problem statement from any domain without fail since the soft voting classifier uses the maximum probability as the voting mechanism and its inbuilt dynamic nature. Nonetheless, a few trials should be conducted to determine the optimum classification algorithm for a specific application. Two third-party service-based features (*domain age* and *page rank*) were used along with other features, however, [10] also used two third-party-based features. *Random forest* outperformed their methodologies, and our experiment found that the *RF* was computationally less time consuming than *GB*.

Another study [34] achieved a 96.5% accuracy score using an *RF* with 12 features; however, used third-party services-based features and a very small sample size with 9,000 URLs. Our study achieved a higher success rate than that of [15], [16], [18], [19], [33], and [34] using a large dataset. It is difficult to conclude that the least feature count always provides a high accuracy score using a single dataset. The resultant accuracy may vary drastically for different datasets with the same set of features. In our analysis, experiments were done using two datasets and the best feature count that suited both datasets to obtain high accuracy was selected. An accuracy of 97.5% was achieved by using 13 features from $DataSet_1$; however, the accuracy was reduced drastically when the same feature set was used with $DataSet_2$. Nonetheless, a feature count of 22 produced the best results for both datasets. Although deep learning model in [32] outperformed ours in terms of accuracy, it suffers from a few drawbacks. In comparison to our methodology, their data pre-processing and model training times were significantly longer, and used a smaller dataset to quantify training time. At the same time they employed a very high-powered computer for their research, (as seen in the Table 15), compared to our computer specifics.

With regard to the ensemble models, although they delivered the best output, they were computationally expensive relative to the individual model. The accuracy difference between the individual model and the ensemble models was only approximately 0.15%. However, in the cybersecurity field even a 0.1% percentage could lead to severe consequences. Recent research [16] using a stacking ensemble technique achieved 97.3% as the highest accuracy with 28 features along with third-party-based features. Both the proposed ensemble models in this study achieved an approximately 98.25% accuracy rate with 22 features. The model proposed by [16] took 105.2 seconds to run a model with 11,000 URLs, while our model took 170 sec for 75,375 sets of URLs.

One of the foremost additional findings observed from $DataSet_1$ and $DataSet_2$ on the *GB* classifier was the effect of the *criterion* parameter on the final outcome. $DataSet_3$ was used to further verify that finding, and it was noticed that both accuracy and *ROC-AUC* values markedly declined with *MAE* as the criterion, while *MSE* and *Friedman MSE* achieved the best results for all three datasets. That said, more datasets must be tested to confirm the finding for *MAE*. Furthermore, it is worth testing this finding with regression problem statements.

V. LIMITATIONS AND DRAWBACKS OF THE PROPOSED MODEL (ERG-SVC)

This model, like all others, has some limitations and drawbacks. This information would facilitate the creation of a better model by overcoming the current model's constraints.

1) *Limitations of the feature extraction process* - Since our model involves two third-party-based attributes (domain age and PageRank), extracting outlined features may take longer due to the necessity to connect with distant services maintained by third parties. If the web service cannot access such services or they are unreachable, the model may produce slightly different predictions.
2) *Complexity of the architecture* - In nature, ensemble models require slightly longer times to perform compared to simple models.
3) *Cloud deployment cost* - Additional costs would be consumed if the model needs to be deployed on a cloud platform concerning the model retraining conditions.

VI. FUTURE DIRECTIONS

More research on this topic must be undertaken to build solutions without third-party services such as WhoIs lookup-based features. This would have a marked effect on the computation time. In this research, two third-party features were also used, and it is frequently advised that third-party dependencies should be ignored for ML models. Currently, intruders are hosting phishing campaigns on publicly accessible websites by leveraging their weaknesses and using various phishing tools. Deployment of a phishing page on a compromised domain provides various favourable circumstances to cybercriminals. A hacker does not require a web hosting server to install a phishing website. Hence, more research must extract certain valid text-based features from URLs to identify phishing URL patterns. It is better to conduct a more comprehensive analysis using the whole phishing message (URL, HTML, content, images, attachments and others) and introduce a robust hybrid solution. In future research, attention must be paid to the security of the application after deployment, whether it is on the web or standalone. In those domains, a specific challenge is that intruders are continually using fresh tactics against security measures. Algorithms and methods that continuously adjust to changes, for instance, phishing URL attributes must overcome those scenarios. Furthermore, more experimentation must be done using the proposed *ERG-SVC* model to determine whether it can provides accurate detection results for other cyber-security related incidents such as malware, network anomalies, and IoT attacks.

VII. CONCLUSION

The main focus of this study was to propose a robust ensemble ML model with a high prediction accuracy rate. To improve the accuracy, various experiments were carried out using datasets to determine the optimal heuristic-based threshold values for each algorithm. A well-organized feature selection technique was used to eliminate unwanted features and finally develop a lightweight preprocessor including 22 features. The experimental results showed that *GB* outperformed other models, with a 98.118% accuracy rate and a low error rate as an individual model. An ensemble model was developed that used a voting classifier (*ERG-SVC*) and outperformed all other models, with a 98.27% prediction accuracy rate. Hence, the proposed individual and ensemble models provided higher accuracy than other existing approaches. Most of the features used for this study were URL-based, although two third-party features were also used. Those third-party features risked complicating the model, whereas client-side features could simplify it. Furthermore, compromised websites might provide inaccurate data as third-party services-based features. Therefore, the goal of detecting phishing websites using only client-side features is a motivation for further research and development.

A. AUTHOR CONTRIBUTION

Pubudu L. Indrasiri and Malka N. Halgamuge conceived the study idea and developed the analysis plan. Pubudu L. Indrasiri wrote the initial article. Malka N. Halgamuge helped to prepare the figures, tables and finalizing the manuscript. All authors read the manuscript.

REFERENCES

[1] Y. Chai, Y. Zhou, W. Li, and Y. Jiang, "An explainable multi-modal hierarchical attention model for developing phishing threat intelligence," *IEEE Trans. Dependable Secure Comput.*, early access, Oct. 12, 2021, doi: 10.1109/TDSC.2021.3119323.

[2] R. Valecha, P. Mandaokar, and H. R. Rao, "Phishing email detection using persuasion cues," *IEEE Trans. Dependable Secure Comput.*, early access, Oct. 8, 2021, doi: 10.1109/TDSC.2021.3118931.

[3] M. N. Halgamuge, "Optimization framework for best approver selection method (BASM) and best tip selection method (BTSM) for IOTA tangle network: Blockchain-enabled next generation industrial IoT," *Comput. Netw.*, vol. 199, Nov. 2021, Art. no. 108418.

[4] Q. Li, M. Cheng, J. Wang, and B. Sun, "LSTM based phishing detection for big email data," *IEEE Trans. Big Data*, early access, Mar. 12, 2020, doi: 10.1109/TBDATA.2020.2978915.

[5] (2008). *Phishing Activity Trends Reports*. [Online]. Available: https://apwg.org

[6] Z. Wang, H. Zhu, and L. Sun, "Social engineering in cybersecurity: Effect mechanisms, human vulnerabilities and attack methods," *IEEE Access*, vol. 9, pp. 11895–11910, 2021.

[7] V. Mullet, P. Sondi, and E. Ramat, "A review of cybersecurity guidelines for manufacturing factories in Industry 4.0," *IEEE Access*, vol. 9, pp. 23235–23263, 2021.

[8] D. Lee, D. Kim, M. K. Ahn, W. Jang, and W. Lee, "Cy-through: Toward a cybersecurity simulation for supporting live, virtual, and constructive interoperability," *IEEE Access*, vol. 9, pp. 10041–10053, 2021.

[9] M. Volkamer, K. Renaud, B. Reinheimer, and A. Kunz, "User experiences of TORPEDO: Tooltip-powered phishing email detection," *Comput. Secur.*, vol. 71, pp. 100–113, Nov. 2017.

[10] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Syst. Appl.*, vol. 117, pp. 345–357, Mar. 2019.

[11] G. Tuysuzoglu and D. Birant, "Enhanced bagging (eBagging): A novel approach for ensemble learning," *Int. Arab J. Inf. Technol.*, vol. 17, no. 4, pp. 515–528, Jul. 2020.

[12] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1937–1967, 2020.

[13] A. Rojarath and W. Songpan, "Probability-weighted voting ensemble learning for classification model," *J. Adv. Inf. Technol.*, vol. 11, no. 4, pp. 217–227, 2020.

[14] S. Cui, Y. Yin, D. Wang, Z. Li, and Y. Wang, "A stacking-based ensemble learning method for earthquake casualty prediction," *Appl. Soft Comput.*, vol. 101, Mar. 2021, Art. no. 107038.

[15] S. Sindhu, S. P. Patil, A. Sreevalsan, F. Rahman, and M. S. A. N., "Phishing detection using random forest, SVM and neural network with backpropagation," in *Proc. Int. Conf. Smart Technol. Comput., Electr. Electron. (ICSTCEE)*, Oct. 2020, pp. 391–394.

[16] A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum, and M. Hamdani, "Phishing web site detection using diverse machine learning algorithms," *Electron. Library*, vol. 38, no. 1, pp. 65–80, 2020.

[17] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Comput. Appl.*, vol. 31, no. 8, pp. 3851–3873, Aug. 2019.

[18] M. Babagoli, M. P. Aghababa, and V. Solouk, "Heuristic nonlinear regression strategy for detecting phishing websites," *Soft Comput.*, vol. 23, no. 12, pp. 4315–4327, Jun. 2019.

[19] F. Feng, Q. Zhou, Z. Shen, X. Yang, L. Han, and J. Wang, "The application of a novel neural network in the detection of phishing websites," *J. Ambient Intell. Humanized Comput.*, pp. 1–15, Apr. 2018. [Online]. Available: https://link.springer.com/article/10.1007/s12652-018-0786-3

[20] S. Smadi, N. Aslam, and L. Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," *Decis. Support Syst.*, vol. 107, pp. 88–102, Mar. 2018.

[21] T. Peng, I. Harris, and Y. Sawa, "Detecting phishing attacks using natural language processing and machine learning," in *Proc. IEEE 12nd Int. Conf. Semantic Comput. (ICSC)*, Jan. 2018, pp. 300–301.

[22] A. K. Jain and B. B. Gupta, "Towards detection of phishing websites on client-side using machine learning based approach," *Telecommun. Syst.*, vol. 68, no. 4, pp. 687–700, Aug. 2017.

[23] H. Shirazi, K. Haefner, and I. Ray, "Improving auto-detection of phishing websites using fresh-phish framework," *Int. J. Multimedia Data Eng. Manage.*, vol. 9, no. 1, pp. 51–64, Jan. 2018.
[24] E. Buber, B. Diri, and O. K. Sahingoz, "NLP based phishing attack detection from URLs," in *Proc. Int. Conf. Intell. Syst. Design Appl.* Cham, Switzerland: Springer, 2017, pp. 608–618.
[25] A. Vabalas, E. Gowen, E. Poliakoff, and A. J. Casson, "Machine learning algorithm validation with a limited sample size," *PLoS ONE*, vol. 14, no. 11, Nov. 2019, Art. no. e0224365.
[26] (2018). *NetSec Explained*. [Online]. Available: https://netsecexplained.com/
[27] (2007). *UC Irvine Machine Learning Repository*. [Online]. Available: https://archive.ics.uci.edu/ml/index.php
[28] R. Basnet, S. Mukkamala, and A. H. Sung, "Detection of phishing attacks: A machine learning approach," in *Soft Computing Applications in Industry*. Berlin, Germany: Springer, 2008, pp. 373–383.
[29] M. N. Alam, D. Sarma, F. F. Lima, I. Saha, and S. Hossain, "Phishing attacks detection using machine learning approach," in *Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, 2020, pp. 1173–1179.
[30] F. Klien and M. Strohmaier, "Short links under attack: Geographical analysis of spam in a URL shortener network," in *Proc. 23rd ACM Conf. Hypertext Social Media (HT)*, 2012, pp. 83–88.
[31] K. Demertzis, L. Iliadis, and V.-D. Anezakis, "Aedes albopictus and Aedes japonicus-two invasive mosquito species with different temperature niches in Europe," *Frontiers Environ. Sci.*, vol. 5, p. 85, Dec. 2017.
[32] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15196–15209, 2019.
[33] S. Wang, S. Khan, C. Xu, S. Nazir, and A. Hafeez, "Deep learning-based efficient model development for phishing detection using random forest and BLSTM classifiers," *Complexity*, vol. 2020, pp. 1–7, Sep. 2020.
[34] V. Patil, P. Thakkar, C. Shah, T. Bhat, and S. P. Godse, "Detection and prevention of phishing websites using machine learning approach," in *Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, Aug. 2018, pp. 1–5.

PUBUDU L. INDRASIRI received the B.Sc. degree in computer science from the University of Colombo, Sri Lanka, in 2016, and the M.Sc. degree in network security from Charles Sturt University, Melbourne, Australia, in 2020. He is currently pursuing the Ph.D. degree in artificial intelligence with Deakin University, Geelong, Australia. His research interests include machine learning, deep learning, decentralized deep learning, federated learning, and privacy and security techniques.

MALKA N. HALGAMUGE (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical and Electronic Engineering, The University of Melbourne, in 2007. She is currently a Researcher with the Department of Electrical and Electronic Engineering, The University of Melbourne. She is passionate about research and teaching university students (emerging technologies, the Internet of Things (IoT), blockchain, machine learning solutions, and bioelectromagnetics). She was awarded the Chinese Academy of Sciences President's International Fellowship in 2017; the Incoming Leaders Fellowship from the Australia India Institute @Delhi in 2016; the Next Step Initiative Fellowship in 2015; the Australia-China Young Scientist Fellowship in 2014; the Dyason Fellowship at the University of California (UCLA), Los Angeles, USA, in 2013; the Early Career Researcher (ECR) Award from the Alexander von Humboldt Foundation in 2013; and the Solander Fellowship at Lund University in 2007 and 2008. She was a recipient of the Vice-Chancellor's Engagement Award in 2010 and the Vice-Chancellor's Knowledge Transfer Award in 2008 for her research at The University of Melbourne.

AZEEM MOHAMMAD is currently the National Academic Director of Charles Sturt Study Centers at Melbourne, Sydney, and Brisbane. He is also an Adjunct Associate Professor with over 25 years of experience in international education, including as an Academician. He is the Globally Focused Manager with a strong awareness of educational environments, complex business, technology, and financial management. His research interest focuses on education management, decision making, artificial intelligence, online/web marketing, and data sciences. Other areas of interest include applied managerial ethics and strategic issues in social responsibility. This includes identifying critical success factors in managing the ethical organization, the ethical decision-making process, and the ethics of advertising.