

A Heterogeneous Dataset–Driven Ensemble Learning Framework for Malicious URL Detection

Parman Sukarno¹, Syahrulanuar Ngah²

¹Telkom University, Indonesia

²University Malaysia Pahang, Malaysia

Article Info

Article history:

Received Feb 12, 2026

Revised Mar 02, 2026

Accepted Mar 13, 2026

Keywords:

Decision Tree

Ensemble Learning

Gradient Boosting

Malicious URL

Random Forest

ABSTRACT

Modern cyberattacks are increasingly associated with phishing campaign, malware distribution, and website defacement, which are often delivered through malicious Uniform Resource Locator (URL) originating from diverse source. This paper examine malicious URL detection using an ensemble learning framework evaluated on large scale heterogeneous dataset composed of URL aggregated from multiple public threat intelligence source. The dataset include benign, phishing, malware, and defacement URL, thereby reflecting real world variability in attack pattern and data distribution. Three ensemble based classifier, namely Decision Tree (DT), Random Forest (RF), and Gradient Boosting (GB), are evaluated with respect to detection accuracy and computational efficiency. In addition to classification performance, this study present a detailed analysis of training and detection time in order to identify most suitable model for practical deployment. Experimental results indicate that the DT model achieves a training time of 4.14 seconds with macro and weighted accuracies of 94.11% and 91.71%, respectively, and a per category detection time of 0.2162 seconds. The RF model attains macro and weighted accuracies of 93.64% and 90.94%, with training and detection times of 9.73 seconds and 0.2420 seconds, respectively. Although the GB model exhibits the longest training time of 45.38 seconds, it achieves the fastest per category detection time of 0.2151 seconds. Despite its comparatively lower overall accuracy of 92.48% for macro averaging and 89.42s% for weighted averaging, the rapid inference capability of GB makes it a strong candidate for real time malicious URL detection in heterogeneous cybersecurity environments.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Parman Sukarno,

Telkom University,

Indonesia.

Email: psukarno@telkomuniversity.ac.id

1. INTRODUCTION

The rapid expansion of digital technologies has enabled efficient communication, collaboration, and global financial activity. At the same time, this growth has led to a substantial increase in cyber threats [1]. Among the many types of attack, malicious Uniform Resource Locator (URL) have become major delivery channel for malware, phishing activity, and large-scale data breach [2]. Because such attack can be executed within very short time interval, timely detection and mitigation of malicious URL is essential for maintaining security of modern digital system [3].

Conventional method for malicious URL detection commonly depends on blacklist-based technique or predefined rule. Although these ways are effective for known threat, they exhibit clear limitation when facing new or previously unseen attack pattern [4]. Adversary frequently alter URL structure and delivery strategy to bypass static detection rule, which reduce effectiveness of traditional method in rapidly changing threat environment [5]. As a result, machine learning based

detection technique has received increasing attention due to their ability to learn complex pattern and adapt to novel attack. Previous studies report that machine learning model can outperform traditional approach in detection accuracy while remaining efficient enough for real time use [6], [7].

In recent year, ensemble learning method, particularly Decision Tree (DT), Random Forest (RF), and Gradient Boosting (GB), have been widely applied in cybersecurity research [8]. DT algorithms are often selected due to their interpretability and low computational cost, which support rapid decision making. RF enhance robustness by combining multiple decision tree with diverse structure, thereby reducing overfitting and improving generalization across different data distribution [9]. GB further improve performance by iteratively focusing on sample that are difficult to classify, which is especially beneficial for complex and imbalanced dataset [10]. Prior work has shown that these ensemble model can achieve strong detection performance with acceptable computational requirement for practical deployment [11].

Despite this development, most existing studies evaluate ensemble model using limited or relatively uniform malicious URL dataset, or they focus mainly on detection accuracy without jointly examining computational efficiency. In addition, influence of dataset heterogeneity, which arise from multiple data source, diverse attack category, and varying URL pattern, has not been sufficiently investigated. This shortcoming limit relevance of previous finding to real world scenario, where detection system must process large and diverse data stream under strict time constraint [12], [13], [14]. To address this limitation, this study examine ensemble based malicious URL detection using large scale heterogeneous dataset introduced by Siddharta [15]. This dataset consists of URL from multiple public sources and includes benign traffic as well as phishing, malware, and defacement category.

This study proposed several important contributions to field of malicious URL detection. First, it employs large scale heterogeneous malicious URL dataset aggregated from multiple public sources and cover benign, phishing, malware, and defacement URL. Diversity of data source and attack category introduce substantial variability in URL structure and threat behavior, enabling more realistic evaluation environment that help to enhance detection robustness and generalization beyond what is achievable with homogeneous dataset. Second, the study systematically evaluates performance of ensemble learning model, including DT, RF, and GB, under identical experimental condition, highlighting their main strength and limitation when operating on heterogeneous data distribution. Third, besides classification accuracy, this work claims a comprehensive analysis regarding training and inference time, which offer important insight about trade off among detection effectiveness and computational efficiency. Finally, based on the empirical result obtained, this study identifies and determine most suitable ensemble learning approach for real time malicious URL detection by jointly consider accuracy, response latency, and robustness within heterogeneous cybersecurity environment.

2. RELATED WORKS

Since its emergence, the Internet has supported extensive digital innovation while also enabling new form of cybercrime. URL represent basic mechanism for accessing website content. However, they are frequently exploited by attacker to distribute malware, conduct phishing activity, and compromise sensitive information [16], [17]. As attack strategy continue to be more sophisticated, conventional detection approach such as blacklist-based mechanism and rule-based filtering has become increasingly ineffective against newly generated and obfuscated malicious URL [18], [19]. These limitations have encouraged adoption of machine learning based technique for malicious URL detection [20].

Although traditional machine learning model have shown promising result, they often experience overfitting problem and sensitivity to data variation when applied in dynamic cyber threat environment. Therefore, ensemble learning has emerged as effective alternative by combining multiple classifiers to improve robustness, accuracy, and adaptability [21], [22]. Algorithm including DT, RF, and GB are widely used due to their robust classification performance and relatively low computational cost, that make them suitable for practical cybersecurity deployment [23], [24].

Several studies have reported effectiveness of ensemble learning method for malicious URL detection. Ensemble models have capabilities to capture complex URL pattern and adapt to changing attack behavior by using feature derived from URL structure, host information, and content related attribute [25], [26]. For example, Arjun *et al.* [27], reported accuracy of 91% using RF on ISCX URL 2016 dataset, while Babu *et al.* [28] achieved accuracy of 97.2% using GB on dataset mainly focused on phishing attack. Comparative evaluation by Singh *et al.* [29], further demonstrated that ensemble model has better performance compared to individual classifier, with reported accuracy ranging between 94.8% and 97.2%.

Among various algorithm in ensemble learning, RF, DT, and GB have proven to be most effective. DT provide simpler and more operational basic threat evaluation [24], [29]. Mixed approach, such as random forest and gradient boosting, contribute to suppressing overfitting and strengthening robustness of model [30]. The accuracy reported for this class of algorithm by Singh *et al.* [29], ranged from 94.8% for GB to 97.2% for RF. Comparative study consistently demonstrate that ensemble outperform individual classifier [27]. Computational execution time become important consideration when model is expected to be implemented in real system with limited resource. However, to the best of our knowledge, investigation into this aspect is still very little. In addition to that, robustness of machine learning model depends heavily on diversity and quality of dataset used for training [31], [32].

Despite these encouraging finding, most existing studies rely on homogeneous or single source dataset, such as ISCX URL 2016, Kaggle collection, or manually constructed dataset derived from limited number of public repository. Although these dataset are valuable, they do not fully reflect heterogeneity of real world environment, where malicious URL originate from diverse platform and exhibit varied attack characteristic, structure, and data distribution. As a result, robustness and generalization capability of ensemble model under heterogeneous data condition remain insufficiently examined. In addition, while detection accuracy is widely reported in previous work, computational execution time, particularly training duration and per URL inference time, has received limited attention. This aspect is critical because execution efficiency is key requirement for real time malicious URL detection system operating under resource constraint.

Motivated by the large scale heterogeneous dataset introduced by previous research [15], which aggregate malicious URL from multiple public source and cover diverse attack category, this study aim to address these limitation. Unlike earlier work, this paper evaluate ensemble learning model using heterogeneous dataset and provide joint analysis of detection accuracy and computational execution time. This approach offer more realistic and deployment oriented assessment of DT, RF, and GB model for malicious URL detection.

Table 1. Existing Model Evaluation Results

Reference	Accuracy (%)	Execution Time (sec/URL)	Algorithm	Dataset
R. K. Singh <i>et al.</i> (2025) [29]	96.5	Not Available	Decision Tree	Not Available
	97.2	Not Available	Random Forest	
	94.8	Not Available	Gradient Boosting	
Shantanu <i>et al.</i> (2021) [30]	92.66	Not Available	Random Forest	Kaggle: Christian Urcuqui (2017), "Malicious and Benign Websites".
Sahingoz <i>et al.</i> (2019) [20]	97.02	Not Available	Decision Tree	Create their own dataset with PhishTank (2018), Ebbu (2017), and Phishing Dataset (2017).
	97.98	Not Available	Random Forest	
Marchal <i>et al.</i> (2014) [33]	95.22	4.2 seconds/URL (All algorithm average time)	Random Forest	Create their own dataset with PhishTank and DMOZ
Arjun <i>et al.</i> (2023) [27]	90	Not Available	Decision Tree	ISCX-URL-2016
	91	Not Available	Random Forest	
	83	Not Available	Gradient Boosting	
Babu <i>et al.</i> (2024) [28]	96.1	Not Available	Decision Tree	Mendeley Data: Hannousse, Abdelhakim; Yahiouche, Salima (2020), "Web page phishing detection"
	96.7	Not Available	Random Forest	
	97.4	Not Available	Gradient Boosting	
Our Research		Available	Decision Tree, Random Forest, and Gradient	The heterogeneous dataset is compiled from five reputable public

Boosting

sources: ISCX-URL-2016, Malware Domain Blacklist, PhishTank, PhishStorm, and a GitHub repository maintained by Faizan.

Table 1 summarizes representative studies that apply ensemble learning algorithms to malicious URL detection. The reported result indicate that DT, RF, and GB model consistently achieve high detection accuracy across range of dataset. However, more detailed review of the literature reveal two important limitation. First, most existing studies rely on homogeneous or single source dataset, such as ISCX URL 2016, Kaggle repository, or narrowly focused phishing dataset. Although these dataset are useful for comparative evaluation, they do not adequately reflect diversity of real world malicious URL, which originate from multiple platform and exhibit varied attack characteristic. Second, execution time, including training cost and per URL detection latency, is rarely reported, even though it is critical requirement for real time deployment. As a result, accuracy metric alone are insufficient to assess practical feasibility and robustness of these model in heterogeneous operational environment.

Although several studies report detection accuracy exceeding 95%, direct comparison with result of this study should be made with caution. Many prior works evaluate ensemble learning model using homogeneous dataset derived from single source or limited to specific attack category, such as phishing. In contrast, the dataset applied in this study gather malicious URL from multiple public repositories and include benign, phishing, malware, and defacement category. This level of heterogeneity presents greater variation in URL structure, feature distribution, and attack pattern, which make classification task more demanding. As a result, performance outcome obtained from heterogeneous dataset offer more realistic and rigorous evaluation of model robustness and generalization in real world deployment setting.

3. RESEARCH METHOD

The primary objective of this study is to evaluate and compare ensemble learning method for malicious URL detection using large scale heterogeneous dataset. The dataset is obtained from reputable public repository on Kaggle platform and aggregate URL from multiple public source. It include four class, namely benign, phishing, malware, and defacement. This heterogeneity introduce diverse URL characteristic and attack pattern that closely reflect real world deployment condition. The overall research methodology, illustrated in Fig. 1, follow structured five step process that is designed to preserve dataset diversity while still enabling reliable model evaluation.

In the first step, data preprocessing is conducted to remove duplicate and empty record, followed by encoding and transformation of numerical and categorical feature. The second step address feature extraction and selection, where key discriminative attribute, including URL length, usage of URL shortener, HTTPS usage, and presence of IP address, are identified and encoded. To maintain practical relevance, the original class distribution is preserved, reflecting non uniform data distribution that are commonly observed in operational system. The third step apply tree based ensemble learning algorithm, specifically DT, RF, and GB. In the final step, model performance is evaluated using standard classification metric such as accuracy, precision, recall, and F1 score, together with detailed analysis on training and inference execution time. This structured evaluation enable comprehensive assessment regarding suitability of each model for real time malicious URL detection in heterogeneous environment.

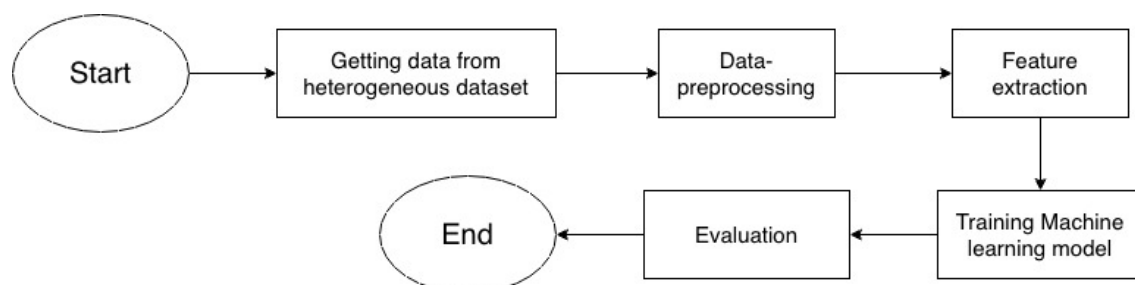


Figure 1: Process System Diagram

3.1. Getting the Data

The first step illustrated in Figure 1 involve dataset preprocessing and preparation. A public dataset provided by previous research [15] is used, which consist of 651,191 URL entry. Among this data, 428,103 are labeled as legitimate or benign, 96,457 correspond to defacement, 94,111 are associated with phishing, and 32,520 represent malware. The dataset is compiled from five reputable public source, including ISCX-URL-2016, Malware Domain Blacklist, Phishtank, PhishStorm, and a GitHub dataset repository maintained by Faizan. All the sources are then combined to construct dataset that is both diverse and representative.

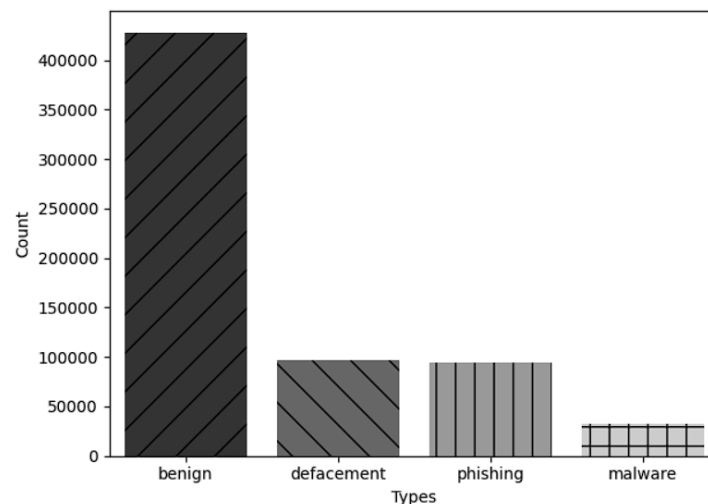


Figure 2. URL Data Distribution by Category

Following data collection, preprocessing is performed to remove duplicate record and empty entry. Duplicate removal was conducted at URL string level after standard normalization procedure, including lowercasing and trimming of whitespace character. However, advanced canonical URL normalization (such as protocol harmonization, removal of “www” prefix, path normalization, or query parameter standardization) was not applied. Therefore, deduplication was based on standardized string matching rather than full canonical equivalence. Each record is then transformed to ensure consistent representation of numerical and categorical feature, including URL length, presence of IP address, use of HTTPS, and utilization of URL shortening service. Figure 2 present class distribution of the dataset, where benign class is significantly more dominant compared to malicious class. This imbalance distribution shows non uniform class distribution, which introduce significant challenge during training of malicious URL detection model.

3.2. Data Pre-Processing

As mentioned in Section 3.1, dataset preprocessing comprises cleansing process by removing duplicate and empty row. The dataset is then divided into training and testing subset using 80%–20% split. The original class distribution is preserved at data level, and no resampling technique such as oversampling or undersampling are applied; therefore, inherent imbalance characteristic rests unchanged. However, to mitigate potential bias toward majority class during model optimization, class weighting is applied at algorithm level during training. Specifically, class weight is processed inversely proportional to class frequency in training data, allowing minority class to exert greater influence on impurity reduction and loss optimization without altering empirical distribution. As input to the model, categorical feature is transformed into numerical representation using One Hot Encoding. All of them are subsequently normalized using MinMaxScaler, with scaling parameter derived from training set and applied to testing set to prevent data leakage.

The 80%–20% hold out strategy was selected to maintain large and statistically representative training set while preserving sufficiently large independent testing set (approximately 130,000 sample). Given overall dataset size of 651,191 URL, the testing subset is

kept substantial for reliable performance estimation. However, this study used single randomized split rather than cross validation, and therefore the reported result reflects one deterministic experimental configuration.

3.3. Feature Extraction

After pre-processing, the feature extraction from each URL is shown in Figure 3. The extracted features are as follows:

- URL length (number of characters)
- Main domain (extracted via tldextract)
- Frequency of special characters (e.g., @, -, ?)
- HTTPS protocol presence
- Number of digits and letters in the URL
- Use of URL shortening services (detection via regex)
- IP address presence (IPv4/IPv6/hexadecimal)

All these features serve as input variables for the next analysis and modeling steps. To handle class imbalance, class weighting was applied to the training data prior to training the model.

3.4. Ensemble Learning for Malicious URL Detection

Ensemble learning algorithm are widely used to detect malicious URL by utilizing potential feature in the dataset, such as number of dots in the URL, total length of URL, and multidimensional feature [34]. As shown in Figure 1, after feature extraction process, the next step is training of ensemble learning model. This research proposes Random Forest, Decision Tree, and Gradient Boosting as ensemble learning algorithm. These three algorithms are selected based on their capability to recognize complex pattern of URL feature and learn to distinguish between safe website and harmful malicious site.

3.4.1. Decision Tree

As shown in Figure 4, Decision Tree is supervised machine learning algorithm that resemble an inverted tree structure. The relationship between node characterizes decision flow, while leaf node denotes the output variable. Decision Tree is considered as one of optimal model for solving various classification and regression problem [35]. This algorithm can also be applied on small to medium size dataset [36]. In addition, this algorithm is able to handle nonlinear data and perform detection in fast manner. However, it is prone to overfitting problem and may require longer training process [37].

3.4.2. Random Forest (RF)

As shown in Figure 5, RF is supervised machine learning method based on ensemble learning concept, that allow this model to solve complex problem by combining multiple classifiers in order to improve accuracy. RF produce average predicted value based on series of decision tree built on different data subset to enhance detection accuracy [38]. RF is known to be robust against outlier and can handle them in effective manner. While RF is relatively more resistant to noise, it needs longer training time compared to decision tree because it build many trees. Furthermore, this process also requires greater computing power and computational resource [39].

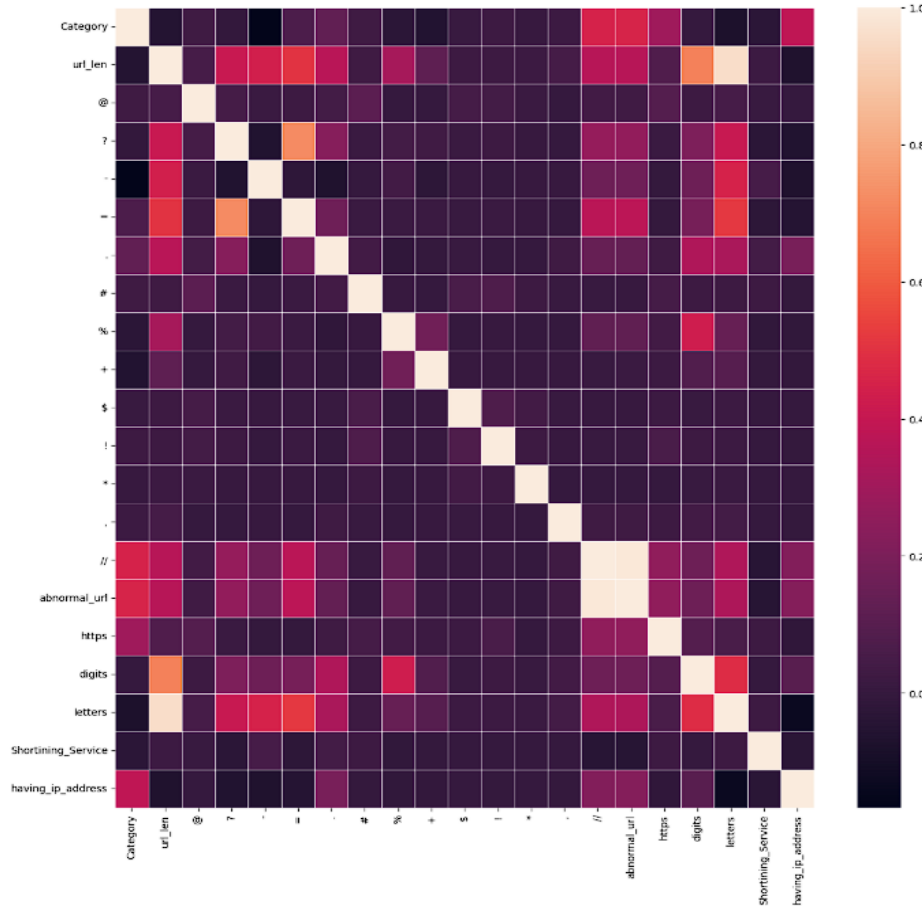


Figure 3. Feature extraction from each URL

3.4.3. Gradient Boosting

Gradient Boosting (GB) is one of the most effective tree-based ensemble method. Label or weight of GB model are stored in leaf node, which make the model easier to interpret as shown in Figure 6. GB apply residual based classification method obtained from previous iteration [40], by selecting best accuracy level from each feature. Challenge related to feature variation are effectively addressed by using GB method. In its application, this algorithm syndicates several weak model such as Decision Tree to gradually improve overall prediction performance [41].

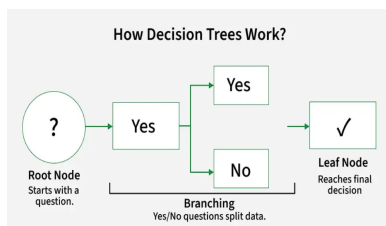


Figure 4: Decision Tree Flow Diagram [26]

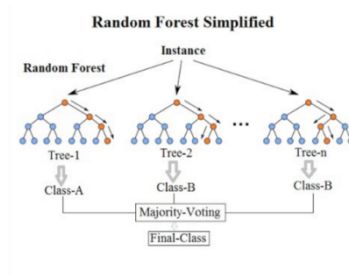


Figure 5: Random Forest Flow Diagram [27]

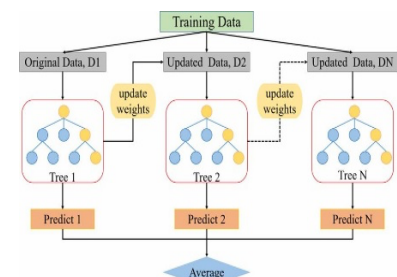


Figure 6: Gradient Boosting Flow Diagram [28]

3.5. Evaluation Metrics

After applying a machine learning model to a dataset, the next step is model evaluating to predict model performance in the real world. The experimental results in this paper were obtained using a Lenovo Gaming 3 laptop with these specifications: AMD Ryzen 5 5600H processor, 16GB of DDR4-3200 SO-DIMM RAM, and NVIDIA GeForce RTX 3050 4GB GDDR6 graphics.

The metrics used to assess the results of the experiments in this study are Accuracy, Precision, Recall, and F1-Score [29]. The parameters include:

- True Positive (TP): when the model detects positive, and the result is correct.
- True Negative (TN): when the model detects negative, and the result is correct.
- False Positive (FP): when the model detects positive, but the result is negative.
- False Negative (FN): when the model detects negative, but the result is positive.

In addition to those four metrics, we also used three more: micro average, weighted average, and execution time.

Accuracy is the ratio between correct detections and the total number of detections generated by a model. The calculation of accuracy in assessing the effectiveness of a classification model's performance is described in Equation 1. This is done by comparing the number of accurate detections, namely true positives and true negatives, with the total number of detections that include true positives, true negatives, false positives, and false negatives [42], [43].

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \times 100\% \quad (1)$$

Precision indicates the proportion of positive cases that a model accurately identifies, indicating its reliability. Precision is calculated in Equation 2 by comparing the total number of correct positive detection to all positive detections generated by the model [42], [43].

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall measures the model's ability to identify true positive cases. Equation 3 provides the formula for calculating Recall, which compares the number of correct positive detections (True Positives) to the total number of actual positive cases (True Positives + False Negatives) [42], [43].

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

The F1-Score combines Precision and Recall. This value is used to balance the two because, as Precision increases, Recall usually decreases, and vice versa [24,25]. Equation 4 shows the F1-Score formula, which is calculated by taking the harmonic mean of Precision and Recall, providing a more balanced assessment of models with an imbalance between the two [42], [43].

Micro Average or Micro averaging treat all detection across class as single binary classification problem, where total true positive (TP), false positive (FP), and false negative (FN) from all class are aggregated before calculating evaluation metric. Equation 5 show the micro average precision formula, Equation 6 show the micro average recall formula, Equation 7 show the micro average F1 Score formula, and Equation 8 show the micro average accuracy formula. This method is unweighted and equalize influence of each instance, making it suitable for imbalanced dataset because it does not favor larger class [44].

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

$$Precision_{micro} = \frac{\sum_{i=1}^{Category} TP_i}{\sum_{i=1}^{Category} (TP_i + FP_i)} \quad (5)$$

$$Recall_{micro} = \frac{\sum_{i=1}^{Category} TP_i}{\sum_{i=1}^{Category} (TP_i + FN_i)} \quad (6)$$

$$F1 - Score_{micro} = 2 \times \frac{Precision_{micro} \times Recall_{micro}}{Precision_{micro} + Recall_{micro}} \quad (7)$$

$$Accuracy_{micro} = \frac{\sum_{i=1}^{Category} TP_i}{Total\ number\ of\ samples\ in\ dataset} \quad (8)$$

$$Metrics_{Weighted} = \sum_{i=1}^{Category} \frac{Number\ of\ true\ instances_i}{Total\ Number\ of\ true\ instance} \times Metrics_i \quad (9)$$

$$Total\ Execution\ Time = T_{training} + T_{testing} \quad (10)$$

Weighted Average is one of the evaluation metrics used for machine learning model, especially in multiclass classification case. Equation 9 is formula for calculating weighted average, which calculate metric such as precision, recall, or F1 score by weighting contribution of each class based on number of true instance (support) for that class. This method provides more weight to class with higher number of samples, making it useful for imbalanced dataset where it is important to reflect original distribution of the dataset [44].

Execution time assess overall computational efficiency by integrating training time with training performance. In real world scenario, execution time play crucial role, especially in resource constrained environment. This makes execution time become important aspect in balancing speed and efficiency [44], [45]. The execution time calculation is shown in Equation 10.

Testing time was measured as total wall clock time required to generate prediction for the complete testing dataset using pretrained model. The model was loaded one time and inference was performed sequentially across all test sample. Class wise testing time were obtained by computing proportional inference time relative to number of samples in every category. The per URL inference time was calculated by dividing total testing time by total number of test sample. All measurement were conducted under identical hardware condition without applying additional optimization technique or batching strategy.

4. RESULTS AND DISCUSSION

This section provides the test results. The testing is performed on the Malicious URL dataset. As mentioned above, the dataset is splitted into two subsets namely training and testing. Testing was conducted using three ensemble learning models such as DT, RF, and GB. The relationship between the max_depth parameter in the DT algorithm and the accuracy results on the training dataset and testing dataset is shown in Figure 7. In the DT algorithm, max_depth refers to the parameter that limits the maximum depth of the tree, which has been defined as the longest path from the root to a leaf node. This parameter controls the depth of the model to prevent overfitting, where the model fits too well to the training data and produces poor performance on new data. The possibility of failure to capture complex data patterns is caused by a depth that is too low. So max_depth can help balance bias and variance so that it can improve model generalization [46]. As the tree depth increases, there is an increase in accuracy on the two datasets.

It should be noted that even though accuracy on the training data reached 93%, the difference between training and testing accuracy remained relatively small, indicating that the model still maintained good generalization ability. The result is expected as a tree with a larger number of layers to recognize more complex patterns in the data. As depth increased, accuracy significantly improved. It should be noted that even though accuracy on the training data reached 93%, the difference between training and testing accuracy remained relatively small, indicating that

the model still maintained good generalization ability. In a condition of max_depth less than 10, the model suffers from underfitting indicated by low accuracy because of limited capacity. The accuracy of the model increases as more layers are added to the tree. The experimental results indicate that the optimal max_depth value is between 15 and 20. The model achieves acceptable accuracy without adding unnecessary complexity within such max_depth range.

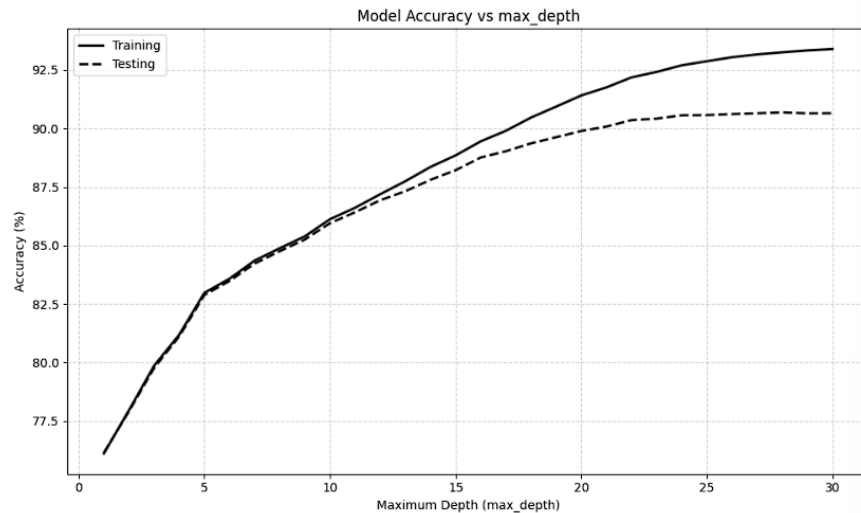


Figure 7. Comparison of Decision Tree Model Accuracy with max depth

Figure 8 shows the relationship between the max_depth and execution time for training and testing phases of the DT model. Training execution time significantly shows longer times as the tree depth increases. The increases in execution time come from the larger size of trees to match more patterns that cause increment of computational resources and time requirements. One can see that the tree depth is proportional with training times.

Based on the evaluation of Figures 7 and 8, the most appropriate max_depth value to maintain a balance between accuracy and efficiency is in the range of 15 to 20. Within this range, the model is able to achieve sufficient level of accuracy without becoming too complex or consuming excessive processing time. In this study, a max_depth of 15 was applied to identify malicious URLs, which provides a good compromise between detection effectiveness and training execution time.

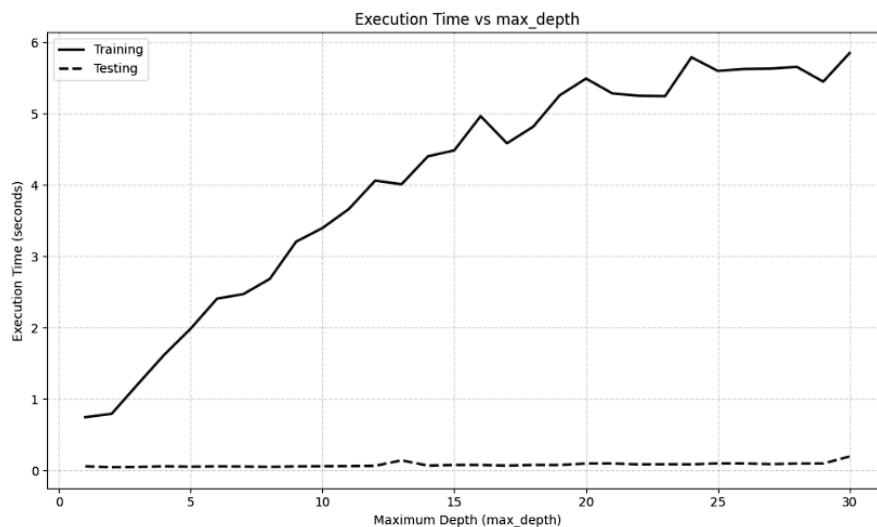


Figure 8. Comparison of Decision Tree Model Execution Time with max depth

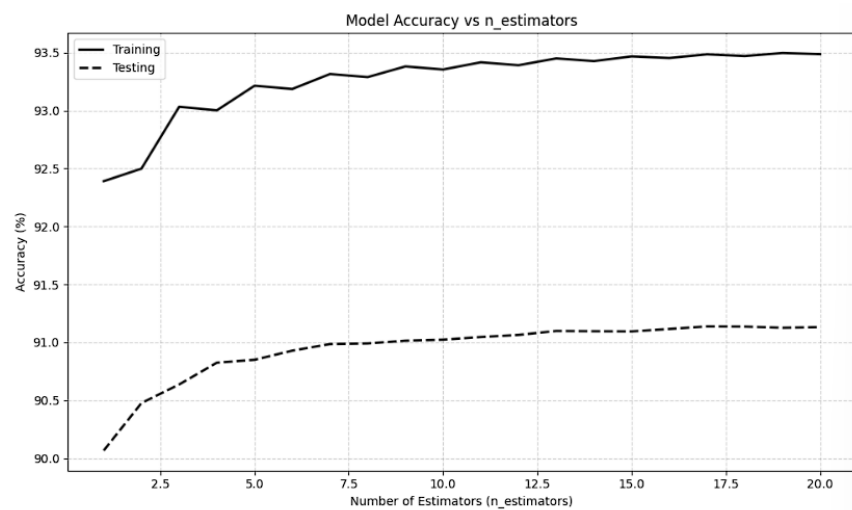


Figure 9. RF Model Accuracy Comparison with different number of estimators

Figure 9 shows how changing the number of estimators affects model accuracy. In RF, the parameter $n_estimators$ denotes how many DT are in the model. Adding more trees are not only able to improve the detection accuracy but also increase the execution time. The model must be a compromise between between accuracy and execution time [47]. The experimental results show that increasing the estimator number from 1 to 5 greatly improves accuracy. After this, the improvement rate slows down, and the accuracy curves begin to level off. Specifically, training and testing accuracies stabilize around 93.5% and 91%, respectively. The experimental results also show that $n_estimators$ value between 10 and 15 are the optimal number of estimators.

The relationship between the number of estimators and model execution time is shown in Figure 10. As the number of estimator increase, the training and testing time gradually increase. This consistent growth rate indicate interaction between model efficiency and complexity. Reducing number of estimators can help to improve execution time, since higher value tend to prolong training and testing process.

In summary, as shown in Figure 9, the best range for $n_estimators$ in term of accuracy is between 10 to 15, where the model accomplishes well without adding unnecessary complexity. Meanwhile, Figure 10 show that execution time increase with higher $n_estimators$ value, highlighting advantage of using fewer estimator for faster training process. Therefore, this study proposes $n_estimators$ value of 10 for malicious URL detection, in order to balance between accuracy and computational efficiency (Optimal: $n_estimators = 10$).

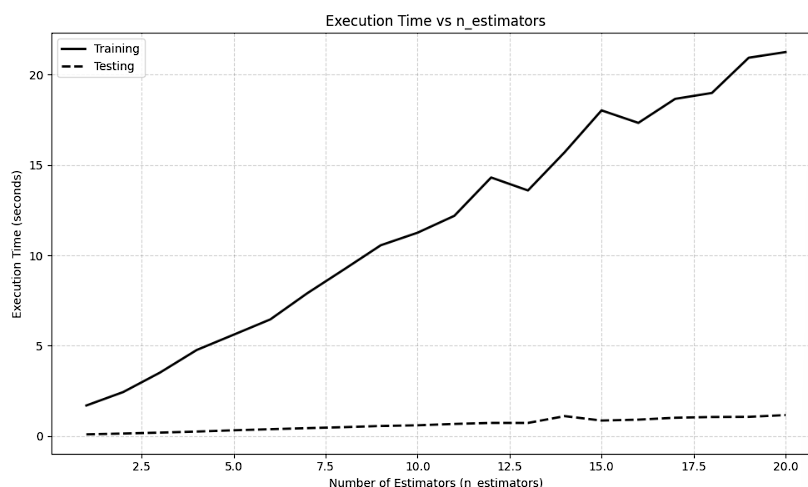


Figure 10. RF Model Execution Time Comparison with different number of estimators

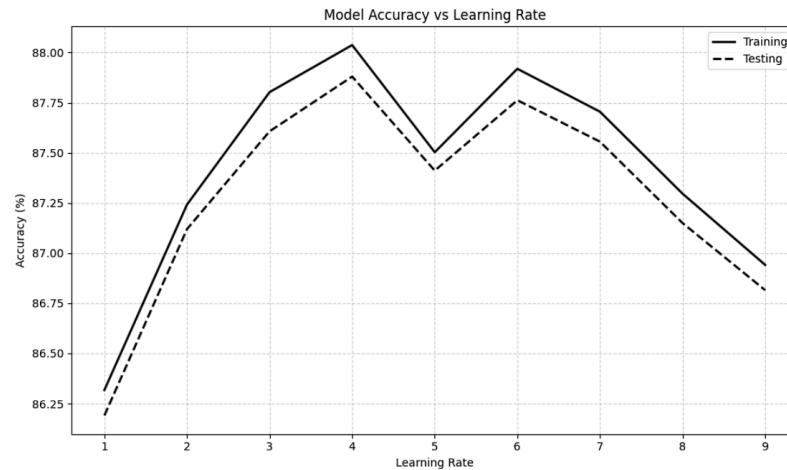


Figure 11. Comparison of GB Model Accuracy

Figure 11 show how changing learning rate affect model accuracy on both training and testing dataset. In GB, the learning rate control how much each new decision tree contributes to final prediction. Smaller value (for example, 0.01 to 0.1) lead to smaller and more gradual learning step, which help the model to learn in more stable manner and reduce overfitting, although it needs more iteration or more tree to reach optimal performance. On the other hand, larger learning rate marks the model adapt too quickly to training data, which can reduce its ability to generalize well [48]. As the learning rate increase from low value up to around 0.4 (corresponding to 4 on the plot), there is noticeable improvement in accuracy. At this point, both training and testing accuracy reach peak value of around 88%, indicating that the model learns effectively. Increasing learning rate beyond 0.4 (or 4 on the plot) cause accuracy to decrease and become more unstable. This indicate that very high learning rate can prevent the model from suitably adjusting its parameter, leading to reduced performance, especially on testing data. Therefore, the best learning rate happens between 0.3 and 0.4 (3 and 4 on the plot), since this range provide highest accuracy with smallest gap between training and testing result.

Figure 12 show how changing learning rate affect model training and testing time. Increasing the learning rate from 0.1 and 0.2 (point 1 and 2 on the plot) to 0.3 (point 3) cause sharp increase in execution time, reaching around 550 second. After this point, increasing learning rate further (above 0.3 or 3 on the plot) lead to shorter execution time. This pattern indicate that higher learning rate advantages the model to adjust faster, thereby speeding up convergence process. Based on this achievement, the best learning rate for reducing execution time is between 0.4 and 0.9 (point 4 to 9 on the plot).

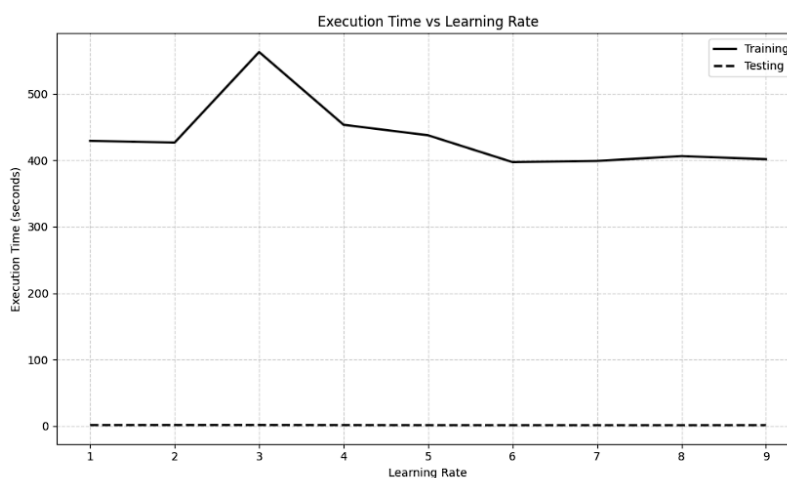


Figure 12. Comparison of GB Model Accuracy

Table 2. DT Model Evaluation Results

Category	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Testing Time (second)	Testing Time/URL (milliseconds)
Benign	88.24	98.59	93.13	90.30	0.2440	0.000570
Defacement	88.64	96.27	92.30	97.60	0.2173	0.002253
Phishing	82.77	34.96	49.15	89.34	0.2141	0.002275
Malware	96.48	80.78	87.93	99.19	0.1893	0.005821
Macro Avg	89.03	77.65	80.63	94.11		
Weighted Avg	91.71	88.14	86.41	91.71		
Total Time (Training)					4.1443	0.006365

The fluctuation in accuracy observed across different learning rate value reflect interaction between shrinkage parameter and number of boosting stage. In tree based Gradient Boosting, the learning rate scale contribution of each newly added tree to the ensemble. Smaller learning rate result in more conservative update and usually require larger number of tree to achieve comparable predictive performance, whereas larger learning rate increase impact of each tree but may reduce generalization capability if it is set too high.

The spike in execution time at learning rate of 0.3 can be attributed to interaction between learning rate and number of boosting iteration used in the model configuration. Since boosting tree are constructed in sequential manner, training time increase proportionally with effective number and complexity of tree required to minimize residual error. Thus, the observed computational overhead reflect inherent sequential tree building process of gradient boosting rather than instability phenomenon that are commonly associated with neural network optimization.

Table 3. RF Model Evaluation Results (max_depth=15 and n_estimators=15)

Category	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Testing Time (second)	Testing Time/URL (milliseconds)
Benign	86.51	99.54	92.56	89.34	0.2893	0.0006756
Defacement	87.99	96.97	92.97	97.57	0.2624	0.0027200
Phishing	91.26	23.38	37.22	88.38	0.1990	0.0021140
Malware	97.67	81.46	88.83	99.25	0.2172	0.0066790
Macro Avg	90.86	75.34	77.72	93.64		
Weighted Avg	87.99	87.28	84.33	90.94		
Total Time (Training)			9.7281			0.01494

Table 4. GB Evaluation Results (learning_rate=0.4 & n_estimators=10)

Category	Precision (%)	Recall (%)	F1-Score (%)	Accuracy (%)	Testing Time (Second)	Testing Time/URL (milliseconds)
Benign	85.22	98.62	91.43	87.68	0.2468	0.0005766
Defacement	83.47	95.65	89.15	96.51	0.2184	0.0022630
Phishing	77.79	14.66	24.67	86.81	0.1961	0.0020830
Malware	93.48	75.45	83.51	98.91	0.1992	0.0061270
Macro Avg	84.99	71.10	72.19	92.48		
Weighted Avg	84.29	84.92	81.07	89.42		
Total Time (Training)					45.3834	0.06971

Table 2 show that DT is able to achieve macro average and weighted average accuracy of 94.11% and 91.71%, respectively. The total execution time is recorded as 4.14 second. In Malware category, the model demonstrates excellent detection performance, where both precision and recall value exceed 96%. In contrast, the recall value for Phishing category is relatively low at 34.96%, which may be caused by model tendency to overfit the training data. The average detection time per category was measured at 0.2162 second.

The RF model, as shown in Table 3, achieved macro average accuracy of 93.64% and weighted average accuracy of 90.94%. The total execution time was 9.73 second, with average detection time per category of 0.2420 second, indicating that it works slower compared to Decision Tree model. Precision and recall value were relatively balanced across all classes, with strong performance observed in Benign and Defacement category. For Phishing category, the recall was

11.58% lower than that of Decision Tree model, but still 8.72% higher compared to recall achieved by GB model.

The GB model, as summarized in Table 4, achieved macro average accuracy of 92.48% and weighted average accuracy of 89.42%. It confirms high precision in most malware category; however, its performance on minority category such as Phishing remain suboptimal, with recall value of only 14.66%. The model needs relatively long execution time of 45.38 second, mainly due to its iterative learning process that progressively strengthen the model from weak learner to strong learner. The average detection time per category was recorded as 0.2151 second. Although GB did not achieve highest accuracy among evaluated model, it suggests slightly faster average detection time compared to both Decision Tree model (by 0.0011 second) and RF model (by 0.0269 second). As result, GB may serve as alternative approach that prioritize quick detection, even though it negotiates some training time and overall accuracy.

Although the macro average and weighted average accuracy of the evaluated model exceed 92%, class wise evaluation reveal significantly lower recall value for phishing category across all algorithms (DT: 34.96%, RF: 23.38%, GB: 14.66%), indicating high false negative rate for phishing URL. From cybersecurity perspective, such degradation in recall is critical because undetected phishing instance may directly expose user to credential theft and financial compromise. This limitation can be attributed to several factor. First, the feature set used in this study mainly consist of lexical and structural URL attribute, such as URL length, frequency of special character, HTTPS usage, digit count, IP address presence, and URL shortening detection. Modern phishing campaign increasingly implements legitimate looking URL structure and HTTPS encryption, which reduce discriminative separability between phishing and benign sample in lexical feature space. As result, significant feature overlap may occur, making it difficult for tree-based ensemble model to construct highly sensitive decision boundary. Second, although class weighting was applied, the dataset stays imbalanced, with benign class having substantially higher support. Consequently, optimization process that prioritize global impurity reduction may implicitly favor majority class performance, thereby increasing likelihood of misclassification for minority class. Third, default probability threshold in ensemble classifier is optimized for overall accuracy rather than attack sensitivity, which may further contribute to false negative prediction in phishing category. These finding indicate that phishing detection within heterogeneous dataset represent particularly challenging classification problem, and while aggregate performance metric appear strong, improved sensitivity toward minority class is required before deployment in high assurance security environment.

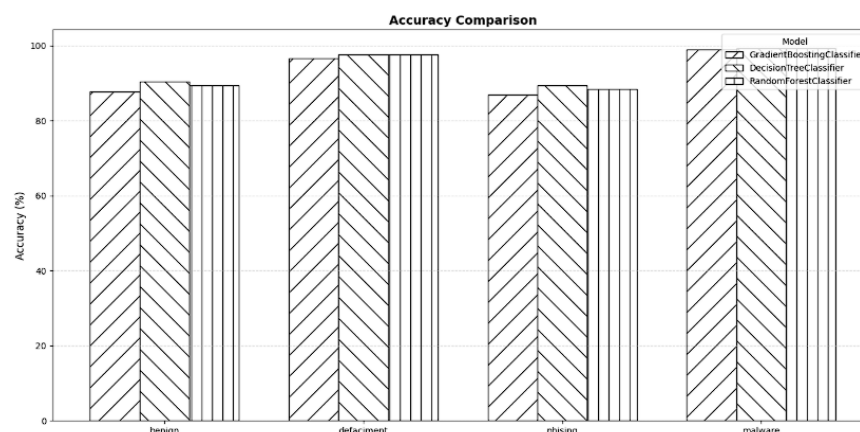


Figure 13: Accuracy Comparison per Category

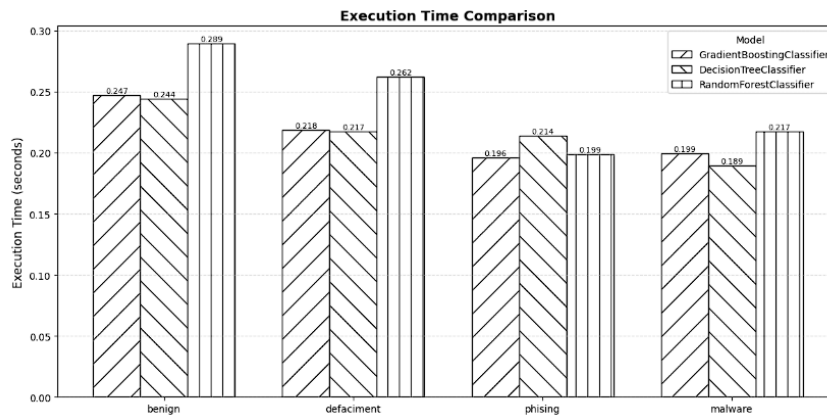


Figure 14: Execution Time Comparison per Category

Analysis of Figure 13 and Figure 14 indicate that DT model consistently succeeds highest accuracy rate across most category, such as Benign, Defacement, Phishing, and Malware. Although DT model sometimes demonstrates slightly longer execution time, its overall performance rests robust. On the other hand, RF model produces competitive accuracy and generally show faster execution speed. In contrast, GB typically demonstrate lowest accuracy among three evaluated model, but it shows to have fastest execution time across some category. Therefore, if computational efficiency is main priority, GB appear to be most optimal choice for this dataset due to its operational speed and acceptable tolerance in accuracy compared to another model.

5. LIMITATION AND FUTURE WORKS

Despite achieving strong macro average and weighted average accuracy, this study present several limitation that need to be acknowledged. Recall for Phishing category remain relatively low across all evaluated model, indicating reduced sensitivity toward socially engineered attack pattern; this limitation likely arise from overlapping lexical characteristic between phishing and benign URL, as well as limited discriminative capability of selected structural feature. In addition, the experimental design rely on single train–test split instead of cross validation, which may constrain statistical generalizability of the reported performance estimation. The study also focus exclusively on classical tree based ensemble method, without evaluating deep learning architecture that are capable of capturing character level or sequential URL representation, which may offer better discrimination for phishing detection. Moreover, threshold optimization and probability calibration are not explored, although such technique could significantly influence detection trade off for minority class. These limitation highlight important direction for methodological refinement in future research.

To improve phishing recall in heterogeneous malicious URL detection, several methodological improvements can be considered in future work. First, decision threshold optimization can be applied on probabilistic output of ensemble model in order to prioritize phishing sensitivity rather than overall accuracy, thereby reducing false negative rate in security critical deployment. Second, more aggressive cost sensitive learning strategy may be introduced by assigning higher penalty to phishing misclassification, which encourage the model to emphasize minority class detection during training process. Third, data level technique such as synthetic oversampling (for example, SMOTE) or adaptive resampling could strengthen representation of phishing instance and improve decision boundary formation in minority class region. Fourth, feature enrichment beyond basic lexical and structural attribute is likely necessary, including character level n gram, entropy-based measure, domain reputation indicator, DNS related attribute, or embedding based representation that better capture semantic similarity between phishing and benign URL. Finally, probability calibration technique such as Platt scaling or isotonic regression may improve confidence estimation and enable more controlled trade off between recall and precision. Overall, these strategies provide systematized roadmap for enhancing phishing detection

robustness while maintaining computational efficiency in heterogeneous cybersecurity environment.

Furthermore, although duplicate URL were removed using standardized string level matching after basic normalization step (such as lowercasing and whitespace trimming), full canonical URL normalization across heterogeneous source was not performed. As result, structurally equivalent URL that differ in protocol usage, subdomain prefix, or query parameter formatting may not be fully consolidated, potentially leaving residual duplicate across different source. In addition, the evaluation depends on single hold out split without repeated random seeding or k fold cross validation. Although the dataset size is relatively large, minority class metric such as phishing recall may still show sensitivity to sampling variability. Future work will incorporate canonical URL normalization procedure, stratified k fold cross validation, and statistical confidence estimation in order to provide more robust assessment of generalization performance and dataset integrity.

6. CONCLUSION

Application of ensemble methods in detecting malicious URLs that focus on the level of classification accuracy and execution time. Ensemble method testing using complex malicious URL datasets is useful in improving the accuracy of model testing. The model performance with the best accuracy was produced by the Decision Tree model with an accuracy value of 94.11% with the fastest execution time of 4.1443 seconds. Decision Tree also got a moderate average detection time of 0.2162 seconds. In the RF model, the resulting accuracy level was 93.64% with an execution time of 9.7281 seconds and almost the same average time. The GB model produces the lowest accuracy of 92.48%. The execution time of this model is also relatively slow at 45.3834 seconds. However, GB shows the fastest average detection time at 0.2151 seconds. This study conducted parameter testing with a maximum testing depth of around 15 for Decision Trees, 10 estimators for RFs, and a learning rate of 0.4 for GB. Research on malicious URL datasets by applying three ensemble methods can be concluded that Decision Trees are superior in terms of accuracy and speed. GB is shown to be the best candidate for the malicious URLs detection model because of its minimal execution time. Despite not having the highest accuracy, the measurement indicates that the GB has robust and acceptable performance to be deployed in a real system implementation. However, detailed class-wise evaluation reveals that phishing recall remains substantially lower than other categories across all models. From a cybersecurity perspective, this implies an elevated false-negative rate for phishing attacks, which represents a critical operational limitation. While overall performance remains strong in terms of aggregate accuracy, further refinement is required to enhance minority-class sensitivity before deployment in high-assurance security systems.

ACKNOWLEDGEMENT

The authors would like to acknowledge that this research is funded by the International Partnership Research Program through grant KWR4.060/LIT06/PPM-LIT/2024 provided by Telkom University and UMPSA. The support from this program has contributed significantly to the completion of this study and the development of the research activities.

REFERENCES

- [1] S. Verma, "Harnessing the strength of digital technologies for cybersecurity," in *Advances in Information Security, Privacy, and Ethics*, IGI Global, 2025, pp. 1–20.
- [2] A. S. Rafsanjani, N. Binti Kamaruddin, M. Behjati, S. Aslam, A. Sarfaraz, and A. Amphawan, "Enhancing malicious URL detection: A novel framework leveraging priority coefficient and feature evaluation," *IEEE Access*, vol. 12, pp. 85001–85026, 2024.
- [3] S. Li and O. Dib, "Enhancing online security: A novel machine learning framework for robust detection of known and unknown malicious URLs," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 19, no. 4, pp. 2919–2960, Oct. 2024.

- [4] D. Orozco-Fonseca, G. Marin, and A. Lara, "Taxonomy of malicious URL detection techniques," in *Lecture Notes in Networks and Systems*, Cham: Springer Nature Switzerland, 2024, pp. 73–81.
- [5] S. Krishnan and M. Naik S, "An efficient framework for phishing URL detection using adaptive drift-aware feature monitoring and online learning," *Security and Privacy*, vol. 8, no. 4, Jul. 2025.
- [6] Sivakumar C, Sagar PL, Swetha K, Raju KA, Kumar GN, Nanditha MN, "Detecting Malicious URLs using Machine Learning," in *2025 5th International Conference on Expert Clouds and Applications (ICOECA)*, IEEE, Mar. 2025, pp. 605–612.
- [7] M. D. Sirajudin, P. Sukarno, and A. A. Wardana, "Malicious uniform resource locators detection using feature extraction and deep learning," in *Lecture Notes in Networks and Systems*, Cham: Springer Nature Switzerland, 2025, pp. 33–47.
- [8] B. V. Pavani, D. Mahitha, and B. U. Maheswari, "Enhancing online safety: Phishing URL detection using machine learning and explainable AI," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jun. 2024, pp. 1–6.
- [9] Karajgar, Manasi Dnyanesh, Sonali Sawardekar, Seema Khamankar, Nalini Tiwari, Manjusha Patil, Vishal Kisan Borate, Yogesh Kisan Mali, and Ashvini Chaudhari, "Comparison of machine learning models for identifying malicious URLs," in *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*, IEEE, Jun. 2024, pp. 1–5.
- [10] M. Isangediok and K. Gajamannage, "Fraud detection using optimized machine learning tools under imbalance classes," in *2022 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2022.
- [11] M. V Pachpatil and A. Agrawal, "Evolution of malicious URL detection: A review of techniques for malicious URL detection and classification," in *Sustainable Artificial Intelligence-Powered Applications*, Cham: Springer Nature Switzerland, 2025, pp. 467–475.
- [12] S. Kailas and R. Roopalakshmi, "'Think Before You Click'-Malicious URL Detection in Cybersecurity: A Systematic Review and Research Roadmap.," *IEEE Access*, vol. 13, pp. 154305–154325, 2025.
- [13] M. Aljabri *et al.*, "Detecting malicious URLs using machine learning techniques: Review and research directions," *IEEE Access*, vol. 10, pp. 121395–121417, 2022.
- [14] J. Islam, C. Patra, P. K. Mani, S. Biswas, D. Giri, and T. Maitra, "Phishing URL detection via machine learning: A comprehensive survey," in *Lecture Notes in Networks and Systems*, in *Lecture Notes in Networks and Systems*, Singapore: Springer Nature Singapore, 2025, pp. 141–159.
- [15] M. Siddhartha, "Malicious URLs Dataset," Kaggle, 2019. [Online]. Available: <https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset>. Accessed: Mar. 4, 2026.
- [16] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through DNS data analysis," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 1–36, Jul. 2018.
- [17] D. L. Hanintya, P. Sukarno, and A. A. Wardana, "Comparing explainable AI framework: Study case on detection of DNS exfiltration attach using neural network," *Procedia Comput. Sci.*, vol. 269, pp. 1022–1032, 2025.
- [18] S. Chaudhari, A. Thakur, and A. Rajan, "An efficient malicious URL detection approach using machine learning techniques," in *Lecture Notes in Electrical Engineering*, Singapore: Springer Nature Singapore, 2024, pp. 485–495.
- [19] Y. Tian, Y. Yu, J. Sun, and Y. Wang, "From past to present: A survey of malicious URL detection techniques, datasets and code repositories," *Computer Science Review*, 2025.
- [20] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri, "Machine learning based phishing detection from URLs," *Expert Systems with Applications.*, vol. 117, pp. 345–357, Mar. 2019.

- [21] V. Dhiman, "Ensemble learning techniques for advanced threat detection in complex data environments for smart education," in *Advances in Educational Technologies and Instructional Design*, IGI Global, 2025, pp. 191–206.
- [22] A. Alabdulatif, "A novel ensemble of deep learning approach for cybersecurity intrusion detection with explainable Artificial Intelligence," *Applied Sciences*, vol. 15, no. 14, p. 7984, Jul. 2025.
- [23] R. Ch, B. Naresh, P. L. Prasanna, N. Chander, E. A. Goud, and P. R. Prasad, "Exploring machine learning algorithms for robust cyber threat detection and classification: A comprehensive evaluation," in *2024 International Conference on Intelligent Systems for Cybersecurity (ISCS)*, IEEE, May 2024.
- [24] A. Kumar and K. Guleria, "Leveraging machine learning algorithms for threat detection using AI-enhanced cybersecurity datasets," in *2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS)*, IEEE, Nov. 2024, pp. 483–488.
- [25] F. A. Ghaleb, M. Alsaedi, F. Saeed, J. Ahmad, and M. Alasli, "Cyber threat intelligence-based malicious URL detection model using ensemble learning," *Sensors*, vol. 22, no. 9, p. 3373, Apr. 2022.
- [26] Q. Abu Al-Haija and M. Al-Fayoumi, "An intelligent identification and classification system for malicious uniform resource locators (URLs)," *Neural Computing and Applications*, vol. 35, no. 23, pp. 1–17, Apr. 2023.
- [27] D. S. Arjun, D. S. Samhitha, and A. Padmavathi A and Hemprasanna, "Detection of Malicious URLs using Ensemble learning techniques," in *2023 IEEE Technology & Engineering Management Conference - Asia Pacific (TEMSCON-ASPAC)*, IEEE, Dec. 2023.
- [28] M. R. Babu, K. S. Raja, K. Ruthvik, P. Yashwanth, and S. Rohith, "Phishing URL detection using gradient boosting: A machine learning approach," *International Journal of Novel Research and Development*, vol. 9, no. 3, 2024.
- [29] R. K. Singh, A. Kumar, and F. Ahmed, "Detection of phishing websites using machine learning algorithms," in *Lecture Notes in Networks and Systems*, Singapore: Springer Nature Singapore, 2026, pp. 147–156.
- [30] Shantanu, B. Janet, and R. Joshua Arul Kumar, "Malicious URL detection: A comparative study," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, IEEE, Mar. 2021.
- [31] Kim, S., Suh, Y. K., & Kim, J., "EXTES: An execution-time estimation scheme for efficient computational science and engineering simulation via machine learning," *IEEE Access*, vol. 7, pp. 98993–99002, 2019.
- [32] L. Baier, N. Kühl, and J. Schmitt, "Increasing robustness for machine learning services in challenging environments: Limited resources and no label feedback," in *Lecture Notes in Networks and Systems*, Cham: Springer International Publishing, 2022, pp. 837–856.
- [33] S. Marchal, J. Francois, Radu State, and T. Engel, "PhishStorm: Detecting phishing with streaming analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, Dec. 2014.
- [34] J. Chen, Z. Hu, and Z. Qian, "Research on malicious URL detection based on random forest," in *2022 14th International Conference on Computer Research and Development (ICCRD)*, IEEE, Jan. 2022.
- [35] S. Bishnoi and B. K. Hooda, "Decision tree algorithms and their applicability in agriculture for Classification," *Journal of Experimental Agriculture International*, pp. 20–27, May 2022.
- [36] S. Patil and U. Kulkarni, "Accuracy Prediction for Distributed Decision Tree using Machine Learning approach," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, IEEE, Apr. 2019.
- [37] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of machine learning algorithms for anomaly detection," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, IEEE, Jun. 2020.

- [38] A. B. Adetunji, O. N. Akande, F. A. Ajala, O. Oyewo, Y. F. Akande, and G. Oluwadara, "House price prediction using random forest machine learning technique," *Procedia Computer Science*, vol. 199, pp. 806–813, 2022.
- [39] A. A. Wardana, P. Sukarno, S. Basuki, and S. B. Utomo, "Federated random forest with feature selection for collaborative intrusion detection in internet of things," *Procedia Computer Science*, vol. 246, pp. 20–29, 2024.
- [40] D. Upadhyay, J. Manero, M. Zaman, and S. Sampalli, "Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 1104–1116, Mar. 2020.
- [41] A. Malinin, L. Prokhorenkova, and A. Ustimenko, "Uncertainty in Gradient Boosting via Ensembles," *arXiv preprint arXiv:2006.10562*, 2020.
- [42] A. A. Wardana, G. Kołaczek, A. Warzyński, and P. Sukarno, "Ensemble averaging deep neural network for botnet detection in heterogeneous Internet of Things devices," *Scientific Reports*, vol. 14, no. 1, p. 3878, 2024, doi: 10.1038/s41598-024-54438-6.
- [43] A. A. Wardana, G. Kołaczek, A. Warzyński, and P. Sukarno, "Collaborative intrusion detection using weighted ensemble averaging deep neural network for coordinated attack detection in heterogeneous network," *International Journal of Information Security*, vol. 23, no. 5, pp. 3329–3349, 2024, doi: 10.1007/s10207-024-00891-3.
- [44] S. Beddar-Wiesing, A. Moallem-Oureh, M. Kempkes, and J. M. Thomas, "Absolute evaluation measures for machine learning: A survey," *arXiv preprint arXiv:2507.03392*, 2025.
- [45] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv: 2004.10934*, 2020.
- [46] M. Azad and M. Moshkov, "Applications of depth minimization of decision trees containing hypotheses for multiple-value decision tables," *Entropy*, vol. 25, no. 4, p. 547, Mar. 2023.
- [47] A. V. Alzuhdi, H. Ar Rosyid, M. Y. Chuttur, and S. Nazir, "Optimizing Random Forest algorithm to classify player's memorisation via in-game data," *Knowledge Engineering and Data Science*, vol. 6, no. 1, p. 103, Oct. 2023.
- [48] C. Ma, X. Qiu, D. J. Beutel, and N. D. Lane, "Gradient-less federated gradient boosting trees with learnable learning rates," *Proceedings of the 3rd Workshop on Machine Learning and Systems*, Apr. 2023.