Innovative Machine Learning Model for Malicious URL Detection

Nur Irdina Binti Hassan
Faculty of Computing and Informatics
Multimedia University
Malaysia
1191202351@student.mmu.edu.my

Tea Boon Chian
Faculty of Computing and Informatics
Multimedia university
Malaysia

Tan Wei Chong
Faculty of Computing and Informatics
Multimedia University
Malaysia

***Abstract*— Malicious URLs promote scams, attacks, and fraud, posing significant risks when clicked. This research explores machine learning techniques to combat these threats. The research reviews current research and employs an ensemble approach for improved detection accuracy. Top models will be combined such as Decision Tree, Random Forest, AdaBoost, and XGBClassifier, Voting Classifier with hard voting will also be used. The model's performance is evaluated with accuracy, precision, recall, and F1-score, demonstrating its effectiveness in real-world scenarios.**

Keywords— malicious URL detection, feature extraction, machine learning, ensemble model, voting classifier.

I. INTRODUCTION

The number of malicious Uniform Resource Locator (URL), which are web addresses created by hackers to carry out scams and attacks, is increasing as the online world evolves. According to a report by Cybersecurity Malaysia [22] covering the first half of 2023, 85.23% of cyber incidents in Malaysia involved malware, while 14.77% were related to phishing, including phishing URLs, IP addresses, and domains. The rise in phishing attacks highlights the urgent need for a more advanced defence system. Traditional methods, like manually updating lists of known malicious URLs, are not enough to keep up with the constant stream of new threats. To address these challenges, researchers are turning to machine learning. Machine learning uses data-driven insights to identify complex patterns and traits that suggest malicious intent, allowing for the detection of new and evolving malicious URLs.

In addition to employing machine learning, the online security community has established blacklisting services to aid in the identification of harmful websites. The blacklist functions as a repository containing URLs flagged as malicious. While URL blacklisting has demonstrated effectiveness in certain cases, it remains susceptible to manipulation by attackers who can easily modify components of the URL string to evade detection. Consequently, numerous malicious sites evade blacklisting, either due to their novelty or lack of assessment. Another method for identifying malicious URLs is the heuristic approach, an enhancement of the blacklist method that relies on signatures to establish correlations between new URLs and the signatures of known malicious ones.

While these methods suffice for discerning between malicious and benign URLs, they possess inherent limitations. For instance, the blacklist method falls short in safeguarding against zero hour phishing attacks, as it only identifies 47–83% of new phishing URLs within a 12 hour time frame, thereby failing to categorise new URLs. Additionally, both methods are susceptible to bypassing through obfuscation techniques, such as the generation of a vast number of URLs using algorithms designed to circumvent blacklist and heuristic methods.

Despite these drawbacks, many anti-phishing companies employ the blacklist method due to its simplicity. Consequently, leveraging machine learning (ML) for malicious URL detection has emerged as an effective strategy, particularly in detecting newly created URLs and facilitating model updates. Recent research has delved into Deep Learning models that automatically detect newly formed URLs and extract relevant features. These features encompass lexical, content-based, and network-based attributes, all contributing to the efficacy of Machine Learning algorithms in categorising URLs as malicious or benign.

A. Problem Statement

In the field of cybersecurity, detecting malicious URLs is a challenge that requires the creation of a novel machine learning model that can accurately and efficiently identify malicious URLs based on their distinctive features. Traditional defences frequently fail in this dynamic environment due to cyber adversaries' persistent evolution of new and sophisticated malicious URLs. Despite numerous research aimed at countering this threat, the adaptability of malicious hackers continues to outsmart the existing defence mechanisms. One worrying issue is that new malicious URLs are always being discovered, which makes building a strong training model with a large amount of training data an unattainable objective. Relying solely on large datasets for training is impractical due to the sheer volume and diversity of newly emerging threats. Furthermore, existing approaches to URL classification often involve manual feature engineering, prerequisite knowledge of binary code analysis, or labour-intensive reverse engineering processes. These manual interventions not only contribute to the complexity of the classification model but also introduce a significant time overhead. As malicious URLs evolve rapidly, the time-consuming nature of these processes becomes a crucial impediment to the development of an agile and adaptive deep learning model.

B. Objectives

- To study the existing work on malicious URLs detection using machine learning models to be applied to this research's model.
- To construct a high-performance machine learning model.
- To perform malicious URL classification using ensemble models for better detection and categorizing them into different types such as malicious or benign.

- To evaluate the model's performance when implementing a balanced and imbalanced dataset.

II. LITERATURE REVIEW

A. Ensemble Learning

Ensemble learning combines multiple machine learning models to improve performance and robustness. Alsaedi et al. [1] demonstrated the efficacy of cyber threat intelligence (CTI) features extracted from Google searches and Whois websites in enhancing detection performance. They proposed a two-stage ensemble learning model combining the Random Forest (RF) algorithm for preliminary classification and the Multilayer Perceptron (MLP) for final decision-making. This model achieved a 7.8% increase in accuracy and a 6.7% reduction in false-positive rates compared to conventional URL-based models, although high dimensionality of CTI features could lead to noisy and sparse data, affecting deep learning models like CNNs and SDLs.

Indrasiri, Halgamuge, and Mohammad [2] introduced a novel ensemble machine learning model architecture aimed at achieving high prediction accuracy with minimal error rates compared to other robust machine learning models. The approach combines supervised and unsupervised techniques for detection. The study employed seven classification algorithms, one clustering algorithm, two ensemble techniques, and two standard datasets containing 73,575 and 100,000 URLs. Two test modes, percentage split, and K-Fold cross-validation were utilised for experimentation and final predictions. The study developed mechanisms to determine the optimal heuristic based threshold value for word splitting, tune hyperparameters for each classifier, select prominent features, propose a robust ensemble model named Expandable Random Gradient Stacked Voting Classifier (ERG-SVC), analyse dataset clusters using k-means clustering, assess the gradient boost classifier (GB) with different parameters, and introduce a lightweight preprocessor to reduce computational cost and preprocessing time. With initial experiments conducted on 46 features, the number was reduced to 22 features. The GB classifier achieved the highest prediction accuracy of 98.118% with the least number of NLP-based features. Additionally, the stacking ensemble model and ERG-SVC voting ensemble model outperformed other approaches, demonstrating reliable prediction accuracy rates of 98.23% and 98.27%, respectively, in detecting malicious URLs.

Al-Haija and Al-Fayoumi [3] introduce an inventive machine learning based detection system tailored for identifying Malicious URLs. The proposed system operates via a dual-layered detection mechanism. Initially, URLs undergo classification as either benign or malicious using a binary classifier. Subsequently, the URLs undergo further classification into five distinct classes based on their features: benign, spam, phishing, malware, and defacement. Four ensemble learning approaches, namely the ensemble of bagging trees (En_Bag), the ensemble of knearest neighbours (En_kNN), the ensemble of boosted decision trees (En_Bos), and the ensemble of subspace discriminator (En_Dsc), are investigated in this study. The efficacy of these approaches is assessed using the ISCX-URL2016 dataset, renowned for its comprehensive and contemporary collection of uniform resource locators. This dataset enables the detection and categorization of malicious URLs based on attack type and lexical analysis. Standard machine learning evaluation metrics, including accuracy, precision, recall, F Score, and detection time, are utilized to evaluate the performance of the developed approaches. The experimental findings underscore the superior performance of the ensemble of bagging trees (En_Bag) approach over other ensemble methods. Additionally, the ensemble of the k-nearest neighbour (En_kNN) approach demonstrates the highest inference speed among the evaluated models. Comparative analysis reveals that the En_Bag model outperforms state-of-the-art solutions in both binary classification and multi-classification tasks, achieving accuracy rates of 99.3% and 97.92%, respectively.

Kalabarige et al. [6] introduced a novel multilayered stacked ensemble learning technique designed for enhanced phishing detection. The proposed model has several layers, each of which uses its estimators' predictions as an input for its subsequent layer. This creates a cascading effect that improves detection accuracy. The architecture of the proposed model embodies layer-wise stacked ensemble learning, where each layer comprises a designated number of estimators. The process unfolds in two phases: layers with learners/estimators and a meta-learner. Initialization of the model precedes the development of layers, with each layer housing the requisite number of estimators, culminating in the addition of the meta-learner as the final layer. Subsequently, model training and prediction are executed. In stacked ensemble learning, the output of one layer serves as input for the subsequent layer, enhancing the model's predictive capabilities. Estimators such as Random Forest, Logistic Regression, and K Nearest Neighbours are employed within each layer, operating in parallel, while estimators between layers function sequentially. Ultimately, the outputs of all estimators across layers are stacked and fed into the meta-learner for the generation of predicted outputs. The proposed Multi-Layer Stacked Ensemble Learning Model (MLSELM) comprises four phases: (1) Input phase, where the phishing dataset is ingested, (2) Data balancing phase, as described in Section III-B, and (3) Implementation of MLSELM, outlined in Section III-C. After testing on a variety of datasets, the suggested model performs admirably, with accuracy rates with a range from 96.79% to 98.90%. The datasets from Mendeley 2018 (D2), Mendeley 2020 (D3, D4), and UCI (D1) are used for the evaluation. Remarkably, the model obtains an accuracy of 98.9% with the D2 dataset and a detection rate of 97.76% with the D1 dataset. Moreover, evaluation using the D3 and D4 datasets produces accuracy results of 96.79% and 98.43%, in that order. Comparative analysis against baseline models reveals the superior performance of the proposed model in terms of accuracy and F-score metrics across various datasets.

Krishna, Lavanya, Kola, Pedaprolu, Basha and Rao [7] proposed a model that focuses on identifying legitimate and illegitimate URLs through a feature extraction-based stacking approach. In this model, features such as hostname length, path length, fd_length, tld_length, and various counts related to special characters and elements within URLs are extracted to facilitate prediction. The model utilises a preprocessing step to tokenize URLs, followed by feature extraction, vectorization using techniques like TF-IDF and Word2vec, and ensemble stacking for final predictions. Through a comparison of accuracy, recall, precision, and F1-score metrics with other state-of-the-art models, the proposed approach showcases its effectiveness. Notably, the model demonstrates improvement through ensemble stacking, although some deviations in precision, recall, and F1-score are observed. The study utilises a dataset sourced from Kaggle

and Open Phish, containing URLs labelled as benign or malicious. With 17 extracted features, the model performs stable and enhanced prediction, highlighting the significance of feature engineering and ensemble learning techniques in URL classification. An accuracy of 99.74% was achieved by the proposed model.

Al-Mekhlafi, Mohammed, Al-Sarem, Saeed, Al-Hadhrami, Alshammari, Alreshidi, and Alshammari [8] proposed a model harnessing genetic-based ensemble classifier techniques. Initial training involves classifiers such as AdaBoost (AB), Bagging (BA), GradientBoost (GB), Random Forest (RF), XGBoost (XGB), and LightGBM (LGBM) to identify significant characteristics of malicious URLs and provide an initial assessment of ensemble classification performance without optimization. Genetic Algorithm (GA) optimization is then applied to enhance model accuracy by selecting optimal classifier parameters. The optimized classifiers are organized using the stacking method during the ranking stage to construct an ensemble classifier. Testing data, collected separately, is subsequently utilized. The methodology extracts features using a Python script with various libraries, focusing on abnormality-based, domain-based, bar-based, HTML, and JavaScript features. Experiments conducted on a UCI ML Repository dataset involve oversampling techniques to address class imbalance, particularly utilizing SVM-SMOTE for minority class augmentation. Performance evaluation metrics including classification 32 accuracy, recall, precision, F-score, false-negative rate (FNR), and false-positive rate (FPR) assess the ensemble model's effectiveness, aligning with common practices in PW detection system evaluation. The model achieves 97.16% detection accuracy, demonstrating that the use of a genetic algorithm leads to improved performance and better predictive capabilities.

Sameen, Han, and Hwang [10] present PhishHaven, an innovative phishing detection system that capitalises on lexical analysis for feature extraction. The system introduces URL HTML Encoding to enhance the effectiveness of lexical analysis and offers real-time classification against existing methods. Addressing the challenge posed by tiny URLs, PhishHaven introduces a URL Hit approach, providing a solution to a prevalent open problem in the field. Notably, the system employs an unbiased voting mechanism for final URL classification, mitigating the risk of misclassification in instances of tied votes. PhishHaven's implementation includes a multithreading approach to facilitate real-time detection, enhancing the speed of ensemble-based machine learning models. Theoretical analysis underscores PhishHaven's consistent performance in detecting tiny URLs and its ability to achieve 100% accuracy in identifying future AI-generated phishing URLs based on specific lexical features. Experimental validation conducted with a benchmark dataset containing 100,000 phishing and normal URLs affirms PhishHaven's exceptional performance, yielding an accuracy rate of 98.00%. This surpasses the capabilities of existing lexical-based and human-crafted phishing URL detection systems.

Mondal et al. [11] present an advanced phishing detection methodology that harnesses the power of ensemble learning. The approach integrates multiple classifiers to predict class probabilities for URLs, introducing an innovative thresholding technique to filter decisions made by these classifiers. Through this process, decisions are amalgamated based on corresponding class probabilities, culminating in the selection of the class label with the highest probability as the final decision for unlabeled URLs. Moreover, the system can detect non-malicious vs. malicious URLs with the highest accuracy compared to other models, that is, 99:91% for dataset-I and 97:98% for dataset-II.

Subasi and Kremic [19] introduce ensemble learners in the paper, proposing a novel approach to advance phishing prevention and combat attacks. By adopting an ensemble classifier model utilizing publicly available datasets, it offers an intelligent and automated method to identify phishing websites. AdaBoost and Multiboost emerge as the proposed ensemble learners, significantly enhancing the performance of phishing detection algorithms. Ensemble models exhibit notable improvements in classification accuracy, F-measure, and ROC area metrics. Experimental findings demonstrate the potential to detect phishing pages with an impressive accuracy of 97.61% using the presented model.

B. Supervised Learning

Dsouza [5] proposed a model that utilises Logistic Regression and Support Vector Machine (SVM), to classify URLs as either credible or malicious. It operates in both traditional and distributed computing environments, leveraging tools like the Hadoop Distributed File System (HDFS) for data storage and Spark MLlib for distributed processing. The methodology involves ingesting data into HDFS, processing it using Spark MLlib, training the model with supervised learning algorithms on labelled datasets, and evaluating its performance using various metrics like precision, recall, F1-score, and accuracy. The dataset used in the study contains 2.4 million URLs with lexical and host-based features extracted, enabling the model to learn patterns indicative of malicious URLs. By evaluating the model's performance across different data sizes and distribution methods, the study aims to provide insights into the effectiveness of machine learning techniques in URL classification tasks. The advantage of the proposed model lies in its utilisation of both Logistic Regression and Support Vector Machine algorithms, enabling robust URL credibility determination. Leveraging distributed machine learning environments like HDFS and Spark MLlib enhances the scalability and efficiency of processing large datasets, ensuring effective handling of URL classification tasks. However, the complexity of implementing and fine-tuning the model parameters, as well as the computational resources required for training and evaluation, may pose challenges and limitations, particularly for organizations with limited infrastructure or expertise in distributed computing.

In the study by Alkhudair, Alassaf, Khan, and Alfarraj [13], the emphasis lies in detecting malicious URLs employing four machine learning algorithms: Random Forest, J48 Decision Tree, BayesNet, and K Nearest-Neighbor (KNN). The research initiates by preprocessing the dataset sourced from various origins, eliminating noisy data and null content length columns. Subsequently, it utilizes 10fold cross-validation to divide the data for training and testing, ensuring robust model assessment. Feature selection employs CfsSubsetEval (CSE) to identify the most pertinent attributes, resulting in the identification of six key features. The classifiers undergo training and evaluation based on

performance metrics such as accuracy, recall, and precision. Random Forest exhibits the most favorable performance, attaining accuracies of 96% and 95% in two test phases, surpassing other classifiers.

Rashid, Mahmood, Nisar, and Nazir [16] proposed a method that involves several systematic steps: automatic web page collection using GNU Wget and Python scripts, including the download of HTML documents and related resources alongside screenshot capture; preprocessing to eliminate duplicates, error pages, and extract features from both internal (URL, HTML) and external (third-party services) sources; feature extraction encompasses vocabulary, host, and word-based attributes, with Weka's "StringtoWordVector" aiding in automatic vectorization; Principal Component Analysis (PCA) addresses high dimensionality; and classification employs support vector machines (SVMs) to effectively differentiate phishing and legitimate web pages based on learned patterns. Comparative analysis against previous methods reveals the superior performance of the proposed technique, particularly in accuracy for classification algorithms, indicating its potential for robust phishing detection in cybersecurity contexts. Overall, the experimental results demonstrate that the suggested method performs best when combined with the Support vector machine classifier, correctly identifying 95.66% of legitimate websites from phishing ones with only 22.5% of the innovative features.

Korkmaz, Sahingoz and Diri [18] proposed a robust technique for detecting phishing websites. The model heavily relies on URL analysis, with features such as domain, subdomain, Top Level Domain (TLD), protocol, directory, file name, path, and query, all playing crucial roles. The study emphasises the significance of effective URL features in improving classification accuracy, alongside considerations of third-party service usage, site layout, CSS, and meta-information. By focusing solely on URL analysis, the model is designed for efficient classification without the need for third-party services or content analysis, making it expedient in real-time applications. The dataset utilised includes phishing and non-phishing URLs sourced from Phistank.com and open datasets, facilitating comprehensive analysis and comparison across different scenarios. Feature extraction, conducted meticulously across hostname, domain, and path sections, yields 58 features, with the most significant 48 selected for model training to enhance accuracy. The study evaluates various machine learning algorithms including Logistic Regression, K-Nearest Neighbourhood, Decision Tree, Naive Bayes, XGBoost, Random Forest, and Artificial Neural Network, highlighting their individual strengths and limitations. Experimental results, validated through 10-Fold Cross Validation, demonstrate the model's efficacy in phishing detection, providing valuable insights into the optimal features and algorithms for robust cybersecurity applications. While the model benefits from comprehensive URL analysis and efficient real-time classification, challenges include complexity and computational resources required for implementation, along with potential limitations in detecting sophisticated phishing attempts that exploit other aspects of web content. Further refinement may be necessary to address these limitations and ensure robust performance across various phishing scenarios. Overall, the highest result was 94.59% obtained from using Random Forest classifier on Dataset 1.

C. Deep Learning

Z. Wang, Ren, Li, B. Wang, Zhang, and Yang [4] introduced a malicious URL detection model harnessing convolutional neural networks (CNNs), comprising three primary modules: the vector embedding module, dynamic convolution module, and block extraction module. The vector embedding module utilizes word embedding based on character embedding to represent the URL sequence as a vector, streamlining subsequent processing. Employing advanced techniques, it extracts both phrase and character-level information from the URL to optimize vector expression. The dynamic convolution module automatically extracts features from input data using 1D convolution, folding, and dynamic pooling, adjusting parameters based on input length and current convolutional layers. Simultaneously, the block extraction module segregates various URL fields such as subdomain names and domain suffixes, enhancing the model's input by distinguishing between generic and national top-level domains. The detection process involves sequentially extracting domain components and training the model with embedded sequences, followed by DCNN processing and feature extraction. The model combines outputs from different branches, incorporating information from various URL fields, thus enhancing detection accuracy and recall without misclassifying benign websites. This comprehensive approach underscores the model's effectiveness in leveraging both automated and manual feature extraction to optimise malicious URL detection. The model has achieved an accuracy of 98% in detecting malicious URLs.

Afzal, Asim, Javed, Beg and Baker [14] proposed a hybrid deep-learning approach named URLdeepDetect for time-of-click URL analysis and classification to detect malicious URLs. URLdeepDetect incorporates a range of techniques to analyse the semantic and lexical features of URLs comprehensively. It employs semantic vector models, leveraging the semantic meaning encoded within the URL's structure and content. Additionally, the framework utilizes URL encryption to enhance the detection of malicious patterns within URLs, thereby improving classification accuracy. The classification process within URLdeepDetect is facilitated by two main mechanisms: supervised learning using Long Short-Term Memory (LSTM) networks and unsupervised learning via k-means clustering. LSTM networks are well-suited for sequential data processing tasks, making them ideal for analysing the sequential nature of URLs. On the other hand, k-means clustering enables the unsupervised grouping of URLs based on shared characteristics, allowing for effective classification without labelled training data. The paper emphasises the effectiveness of URLdeepDetect in accurately distinguishing between benign and malicious URLs, thereby providing a robust defence against cyber threats originating from malicious online content. URLdeepDetect achieved an accuracy of 98.3% and 99.7% with LSTM and k-means clustering, respectively.

Singh, Singh and Pandey [15] proposed a phishing detection system using deep learning techniques. The proposed phishing detection system employs a Convolutional Neural Network (CNN) architecture to analyse URLs and distinguish between phishing attempts and legitimate web addresses. Initially, the URLs undergo pre-processing steps such as concatenation, tokenization, padding, and sequencing to prepare them for input into the neural network. The URLs are then converted into dense vectors using pre-trained word embeddings like GloVe. These embeddings are fed into the CNN, which consists of convolutional layers applying filters of various sizes to capture different patterns within the URL sequences. Pooling layers may down-sample the feature maps, and dense layers perform non-linear transformations and feature aggregation. The output layer, typically utilising a softmax activation function, provides the final classification decision, indicating whether a URL is phishing or legitimate. Evaluation metrics such as precision, recall, F1-score, accuracy, and support are used to assess the model's performance, ensuring effective cybersecurity against phishing attacks. The proposed phishing detection model leveraging Convolutional Neural Networks (CNNs) offers several advantages, notably achieving a high accuracy rate of 98.00% in identifying phishing URLs without the need for extensive feature engineering.

Yang, Zheng, Wu, B., Wu, C., and Wang. [20], introduced an integrated method for detecting phishing websites that merges Convolutional Neural Networks (CNNs) and Random Forests (RFs) to predict URL legitimacy without accessing web content or depending on third-party services. Character embedding techniques are utilized to transform URLs into fixed-size matrices, establishing a consistent data structure conducive to phishing website detection. The method incorporates an enhanced CNN network tailored to extract URL features across different layers, enabling detailed detection capabilities. The character embedding method enhances detection accuracy by capturing small variations in URLs, common in phishing attempts. The CNN network architecture comprises seven layers, including convolutional, pooling, and linear layers, culminating in a SoftMax classifier for final prediction. Additionally, the ensemble classification approach employs multiple RF classifiers to classify multi-level features extracted from different CNN layers, further enhancing accuracy and generalisation. This strategy optimises the phishing detection model's performance by leveraging the strengths of both CNNs and RFs while minimising reliance on external services or web content access. The main advantage of this paper is its strong generalisation ability, independent from third-party service and language independent. However, while achieving high accuracy rates of 99.35% and 99.26% on the dataset and benchmark data, respectively, the model would take a longer time to train, and the model cannot determine whether the URL is active or not.

The model proposed by Somesha, Pais, Rao, and Rathour [17] for detecting phishing websites integrates various deep learning methodologies, including Deep Neural Networks (DNN), Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN). The model architecture encompasses three primary stages: feature extraction, feature selection, and classification. Features are extracted from three sources: URL obfuscation, hyperlink-based characteristics, and third party-based information. The authors utilize Selenium with Python, HTML parsers, and BeautifulSoup for feature extraction and employ the Information Gain (IG) mechanism for feature selection. Specifically, the model extracts feature such as dots in hostnames, URL length, and presence of HTTPS for URL obfuscation; domain presence in anchor links, broken links ratio, and common page detection ratio for hyperlink-based features; and Alexa rank-based features for third party-based characteristics. IG ranks features based on their relevance to phishing detection. The model is subsequently trained and cross-validated. Further analysis was conducted to evaluate performance, with the proposed model achieving an accuracy of 99.52% for DNN, LSTM with 99.57%, and CNN with an accuracy of 99.43%.

D. Feature Extraction

The algorithm proposed by Khukalenko, Stopochkina, and Ilin [9] presents a novel approach to URL classification, focusing on the extraction and analysis of various features to distinguish between reliable and phishing URLs. The model utilises address-based features such as IP address usage, dot count, digit count, and special character count, alongside network features including resolved IP count, name server count, and name server IP count. The dataset used comprises 96,018 instances, evenly split between reliable and phishing URLs, sourced from the Aalto University Datasets service. Python was employed for processing, analysis, training, and model evaluation. The assessment of models, performed using cross-validation with k=5, focused on precision, recall, and AUC metrics. The results indicate that XGBoost and Random Forest consistently yielded the highest performance across all types of features, with SVM performing the poorest. Specifically, for lexical features of the address string, kNN and Random Forest demonstrated superior estimations across all three parameters, while SVM exhibited the lowest performance. The study compared various classifiers using two sets of features: lexical and network features. For lexical features, the best performing model was kNN with an AUC of 0.813, while for network features, XGBoost excelled with an AUC of 0.872. The overall model, incorporating all features, achieved the highest AUC of 0.884 using XGBoost. A new model was created by stacking three models: two pre-trained kNN and XGBoost models, along with a Transformer model, yielding a top-performing Logistic Regression model with an AUC of 0.927.

Jalil, Usman, and Fong [12] introduced an effective machine learning framework for predicting phishing URLs without the need for webpage visits or third-party services. The proposed methodology entails three primary phases: dataset preprocessing, feature engineering, and model evaluation. The feature engineering process is crucial for phishing detection, as it significantly influences the method's performance. Rather than relying on extensive feature extraction methods, which can be time-consuming, the study opts to extract features directly from the URL string, ensuring

a faster and more efficient process. Initially, 100 features are extracted from various parts of the URL, and then the top 30 features are selected using domain knowledge and the ReliefF technique. These features are classified into three categories: full URL-based features, host-based features, and path section of the URLbased features. Special characters-based features, count/presence-based features, and suspicious words-based features are utilised to identify phishing URLs. Moreover, brand name-based features and entropy-based features are introduced to further enhance the detection process. The paper employs a TF-IDF technique to match brand names in URLs and calculates Shannon's entropy to assess the randomness in URLs. The extracted features are then normalised using MinMaxScaler to ensure uniformity across the dataset. Following dataset division into 70% for training and 30% for testing the model, the study assesses the performance of eight machine learning classifiers: Naïve Bayes, Logistic Regression, Support Vector Machine, K-Nearest Neighbor, Decision Tree, AdaBoost, Multilayer Perceptron, and Random Forest. Various evaluation measures are employed to determine the most efficient classifier for phishing URL detection. The proposed method achieves the highest accuracy of 96.25% and 94.65% on the Kaggle datasets compared to the benchmark datasets.

III. THEORETICAL FRAMEWORK

The preceding section discusses the existing work on malicious URLs detection using different types of machine learning models with different approaches. In this chapter, a detailed explanation with respect to the main theoretical concepts will be presented. This section also contains further discussion of the approach that is best suited to the system.

A. Feature Extraction

In this section, a detailed explanation of the feature extraction approach to analyse various features of the URLs will be outlined. Each feature contributes to a holistic understanding of the URL's nature. Although individually it might not be definitive, when there is a pattern, it can be indicative of malicious intent.

.

Table 1 Feature Extraction Method and Description

Method	Description
URL Length	Malicious URLs often tend to be longer than benign ones. Attackers use long URLs to hide suspicious parts of the URL from the user such as multiple redirections or encoded information to deceive users.
Domain Extraction	The domain part of a URL can give clues about its legitimacy. Trusted domains are less likely to host malicious content. Malicious URLs often use domains that resemble legitimate ones but with subtle changes or use newly registered domains.
Special Charater Counts	Malicious URLs often include special characters to obfuscate their true destination or to evade detection. High counts of certain special characters can indicate attempts to hide the URL's true nature or add misleading content.
Abnormal URL Detection	An abnormal URL is one that does not contain its own hostname, which can be a sign of phishing or other malicious activity. If a URL's structure is abnormal, it may indicate an attempt to deceive users about the true nature of the website.
HTTPS Detection	HTTPS is a secure protocol. However, its absence does not necessarily indicate malicious intent, but the presence of HTTPS can give a sense of legitimacy. Malicious sites might not use HTTPS, but many do to appear trustworthy. Hence, this feature alone isn't definitive but contributes to the overall analysis.
Digit Count	Malicious URLs often include numbers to obfuscate their intent or because they are auto generated. A higher number of digits can indicate machine-generated URLs or attempts to mimic legitimate ones while including encoded information.
Letter Count	The balance between letters and other characters can indicate if a URL is trying to appear more legitimate or hide its true purpose. A typical benign URL will have a balanced proportion of letters. Deviations might suggest suspicious activity.
Shortnening Service Detection	URL shortening services are often used by attackers to mask the final destination of a URL. URLs using shortening services might be masking their true intent, making it more likely to be malicious.
IP Address Detection	Legitimate websites rarely use raw IP addresses in URLs. Using an IP address can indicate an attempt to bypass filters or mislead users.

B. Imbalanced Classification

Learning classifiers from skewed or unbalanced datasets is a common problem in classification problems, according to Rawat and Mishra [23]. Imbalanced classification refers to a scenario in machine learning where the distribution of classes in the dataset is heavily skewed, with the majority class significantly outnumbering the minority classes. This imbalance poses challenges during model training, as classifiers may prioritize accuracy on the majority class while neglecting the minority classes. To address this issue, various techniques are employed. Resampling methods involve either oversampling the minority class or under sampling the majority class to rebalance the distribution. Algorithmic approaches, such as ensemble methods and boosting algorithms, are adept at handling imbalanced datasets by combining multiple models or adjusting weights. Cost-sensitive learning assigns different costs or weights to different classes to emphasize the correct classification of minority instances.

Additionally, adjusting the classification threshold can improve model performance by increasing sensitivity to minority instances. Hence, imbalanced classification techniques aim to mitigate the adverse effects of class imbalance and ensure effective classification across all classes, making them essential in domains where class distributions are skewed.

C. Balancing Techniques

1) Balanced Random Forest

A Balanced Random Forest (BRF) is a variant of the Random Forest algorithm that addresses the problem of class imbalance in datasets. This is typically accomplished using

resampling techniques applied at each tree-building step. In particular, when constructing each tree, a balanced subset of the data is randomly sampled to ensure that each class is adequately represented. They are particularly useful in domains characterized by class imbalance, such as fraud detection, medical diagnosis, and anomaly detection. One downside of Balanced Random Forest is that it can take longer and use more computer memory compared to regular Random Forest methods. This is because BRF works to balance out the different classes in the data by changing how it picks samples for each tree in the forest. As a result, training the model with BRF might need more time and computer power, especially if the dataset is big or if there's a big gap between the classes. Also, the way BRF balances the data might not always work well with new or unseen data, which could make it harder for the model to correctly classify instances from the less common classes in real-world situations. Despite its ability to handle imbalanced data, it's important to think about the extra time and computer resources BRF needs, as well as its challenges in dealing with new data, when deciding whether to use it.

2) Random Under Sampling

When dealing with imbalanced datasets, especially in the case of classification tasks, where the number of instances in one class significantly outweighs the number of instances in other classes, Random Under Sampling is a famous technique applied to overcome imbalanced datasets as can be seen from research done by Kalabarige et al. [6]. Data points from the majority class are randomly removed or down sampled until the class distribution between majority and minority classes becomes more balanced. By reducing the number of instances in the majority class, Random Under Sampling aims to prevent the model from being biased towards predicting the majority class and helps in improving the performance of classifiers, especially when the imbalance between classes is substantial. However, the disadvantage of Random Under Sampling is that it may lead to a loss of potentially important information that is present in the removed instances. This could affect the overall performance of the model.

3) Random Over Sampling

Kalabarige et al. [6] also applied Random Over Sampling in the proposed model. Random Over Sampling is a technique applied when in a classification task, one class is underrepresented compared to others. In the paper by Sevastianov et al. [21], Random Over Sampling is where instances from the minority class are randomly duplicated or synthesised until the distribution of the classes are more evenly distributed. On the other hand, while Random Over Sampling can overcome the imbalanced dataset, random duplication of instances of the minority classes can lead to overfitting and an increased noise into the dataset. Therefore, to guarantee the efficacy and generalizability of the model trained using Random Over Sampling, rigorous assessment and validation are essential.

D. Voting Classifier

An ensemble learning technique called a voting classifier aggregates the predictions of several different individual models to enhance overall performance. It functions by combining predictions from multiple base classifiers; there are two primary varieties: hard voting and soft voting. Hard voting selects the class with the majority of votes as the final prediction, with each model voting for a particular class. The final decision in soft voting is determined by averaging the predicted probabilities from each model. In this research, a hard voting classifier was implemented to enhance the accuracy of classifying URLs as malicious or benign.

E. Development Tools

For this research, Visual Studio Code served as the primary development tool. Visual Studio Code is a versatile source-code editor compatible with Windows, Linux, and macOS operating systems. It's a widely adopted and free platform renowned for executing Python code, making it an ideal environment for machine learning. Visual Studio Code was chosen as the development platform due to its intuitive interface and robust feature set.

IV. RESEARCH METHODOLOGY

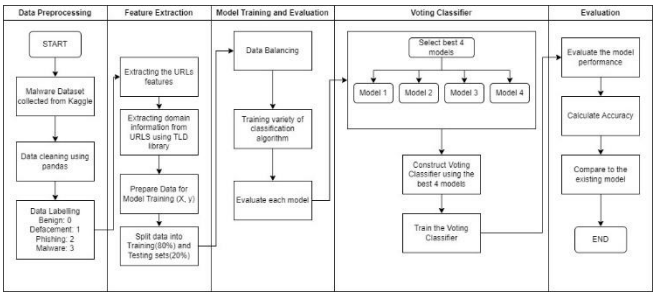


Figure 1 Flowchart of proposed model.

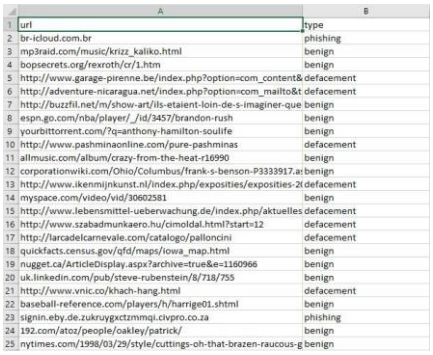
Figure 1 illustrates the flowchart of the proposed model, which is geared towards optimizing the classification process by integrating the most effective classifiers through a voting mechanism.

In this research, five primary steps are used in order to create this model which are data preprocessing, feature extraction, model training and evaluation, voting classifier and lastly evaluation. First, in data preprocessing, this stage involves preprocessing the data for feature extraction and modelling. Steps include adjusting the URL format where the code that omits the “www.” prefix from URLs, treating them as subdomains. Then the categories (benign, defacement, phishing, malware) will be mapped to numerical values classification. The NaN values in the dataset will be checked and the distribution of classes is analyzed to understand the data balance. In feature extraction, features are extracted from URL strings such as counts of specific characters like '@', '?', '-', etc. Abnormal patterns in the URL strings and if the HTTPS is used in the URLs are also determined. Other features extracted also include counting the number of digits and letters, detecting the presence of URL shortening services, and detecting an IP address in the URLs. Extracting the domain was also done by using the ‘TLD’ library to extract the primary domain from the URLs. The URL region was also identified by the mapping of TLDs to regions or countries using external databases or APIs for accurate

geographical identification. Moving on to model training and evaluation, imbalanced dataset will be addressed by applying data balancing techniques such as Balanced Random Forest, Random Under Sampling and Random Over Sampling to see which technique gives the best results when the dataset is applied to the model. A variety of classification algorithms such as Decision Tree Classifier, Random Forest Classifier, AdaBoost Classifier, KNeighborsClassifier, SGD Classifier, ExtraTrees Classifier, GaussianNB, and XGBClassifier are then applied. Each model is trained on the prepared dataset and evaluated to get the accuracy. Finally, in the voting classification stage, the top four performing models, based on their accuracy scores, are selected for inclusion in the voting classifier. These models are chosen for their individual strengths and complementary predictive capabilities. This ensemble approach aggregates predictions from these classifiers to make a final decision. For evaluation, cross-validation was employed, specifically using StratifiedKFold with 5 folds. This method ensures each fold maintains the proportion of each class label, providing a robust estimate of model performance across different subsets of the data. A classification report is then done by using a confusion matrix that shows the precision, recall and f1-score.

A. Data Collection

The dataset that will be used in this research will be taken from Kaggle as seen in Figure 2. The dataset from Kaggle consists of 428103 benign, 96457 defacements, 94111 phishing and 32520 malware sites.



	A	B
1	url	type
2	br-icloud.com.br	phishing
3	mp3raid.com/music/krizz_kaliko.html	benign
4	bopsecrets.org/rexroth/cr/1.htm	benign
5	http://www.garage-pirenne.be/index.php?option=com_content&...	defacement
6	http://adventure-nicaragua.net/index.php?option=com_mailto&t...	defacement
7	http://buzzfil.net/m/show-art/ils-etaient-loin-de-s-imaginer-que	benign
8	espn.go.com/nba/player/_/id/3457/brandon-rush	benign
9	yourbittorrent.com/?q=anthony-hamilton-soulife	benign
10	http://www.pashminaonline.com/pure-pashminas	defacement
11	allmusic.com/album/crazy-from-the-heat-r16990	benign
12	corporationwiki.com/Ohio/Columbus/frank-s-benson-P3333917.a...	benign
13	http://www.ikenmijnkunst.nl/index.php/exposities/exposities-2(	defacement
14	myspace.com/video/vid/30602581	benign
15	http://www.lebensmittel-ueberwachung.de/index.php/aktuelles	defacement
16	http://www.szabadmunkaero.hu/cimoldal.html?start=12	defacement
17	http://larcadelcarnevale.com/catalogo/palloncini	defacement
18	quickfacts.census.gov/qfd/maps/iowa_map.html	benign
19	nugget.ca/ArticleDisplay.aspx?archive=true&e=1160966	benign
20	uk.linkedin.com/pub/steve-rubenstein/8/718/755	benign
21	http://www.vnic.co/khach-hang.html	defacement
22	baseball-reference.com/players/h/harrige01.shtml	benign
23	signin.eby.de.zukruygxctzmmqi.civpro.co.za	phishing
24	192.com/atoz/people/oakley/patrick/	benign
25	nytimes.com/1998/03/29/style/cuttings-oh-that-brazen-raucous-g	benign

Figure 2 Details of Kaggle Dataset.

B. Data Preprocessing

In the data cleaning phase, the focus was on preparing the dataset for further analysis and modelling by addressing inconsistencies and missing values. Initially, the dataset was loaded from a CSV file and inspected for any irregularities using functions such as info() and isnull().sum(). This step helped identify missing values and provided an overview of the dataset's structure. Subsequently, specific cleaning tasks were performed to enhance the data quality.

C. Data Labelling

In the labelling process, each URL gets assigned with a simple binary label, which is 0 for benign, 1 for defacement, 2 for phishing and 3 for malware. This tagging helps train the computer models to tell apart different types of URLs. It's crucial to get these labels right because they directly affect how well the ensemble learning system performs. The dataset contains these labelled URLs along with detailed descriptions of their features. Each URL's features are unique characteristics, and they're paired up with the label that tells if it's benign or malicious. These pairs become the training material for the ensemble learning system. The system uses this data to train many models at once, with each model trying to understand the connections between the features and the labels.

D. Feature Extraction

Features within a URL play a crucial role in distinguishing between different types of URLs. The process begins with identifying and extracting various features systematically. Initially, the model categorizes different types of URLs, followed by data labelling. Subsequently, it calculates the length of the URL and extracts its domain. The extraction process includes identifying specific characters like '@', '?', '-', '=', '.', '#', '%', '+', '$', '!', '*', and ','. Additionally, the model identifies abnormal URLs and counts the occurrences of secure HTTPS URLs. It also computes the number of digits and letters within the URL. Further steps involve counting URLs that use shortening services, those containing IP addresses, and finally determining the URL region. This approach ensures comprehensive analysis of URLs, enabling effective differentiation and classification based on their structural and contextual attributes.

E. Model Evaluation

Assessment metrics like accuracy, precision, recall, and F-score will be used to evaluate the model's classification results after it has been trained. TP stands for true positive, FP for false positive, TN for true negative, and FN for false negative. Because these evaluation metrics provide a comprehensive assessment of the specific approach, researchers have utilized them extensively in the field of research (Alsaedi et al. [1] and Dsouza [5]). The following standards will be used to evaluate the proposed model:

Accuracy: The proportion of properly predicted outcomes relative to the total number of forecasts.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (1)$$

Precision: The proportion of true positive predictions achieved by dividing the total number of positive forecasts by the number of true positives.

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

Recall: The proportion of possible positives discovered by the model by dividing true positives by the total number of positives.

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

F1-score: Calculating the weighted average of memory and precision yields the harmonic mean of recall and precision.

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

V. IMPLEMENTATION

This chapter details the implementation of the solution developed to classify URLs based on their potential to be benign, defacement, phishing, or malware. The solution employs various machine learning models combined into an ensemble using a voting classifier to achieve high accuracy and robustness.

A. Development Environment

The implementation is carried out using Python 3.11.17 (‘myenv’) and libraries such as TensorFlow, Keras, NumPy, and Matplotlib. Visual Studio Code serves as the primary development environment due to its versatility and ease of use.

B. Implementation Process

The process of developing the solution includes the section below where the process will start with data preprocessing, data cleaning, data labelling, feature extraction and engineering, data splitting, training individual models, selecting top models, voting models and lastly model saving, loading, and application.

1) Data Preprocessing

Data preprocessing involves preparing raw data for further analysis and modelling. In the model, this step includes reading the dataset from a CSV file (‘malicious_phish.csv’), checking its structure (‘df.head() and df.info()’) , handling missing values (‘df.isnull().sum()’ and ‘df.fillna(0, inplace=True)’), and basic data cleaning operations like removing 'www.' from URLs (‘df['url'] = df['url'].replace('www.', '', regex=True)’. Preprocessing ensures that the data is in a usable format, addresses missing values, and prepares it for subsequent steps such as labelling and feature extraction.

2) Data Cleaning

Data cleaning focuses on identifying and rectifying errors or inconsistencies in the dataset. In the code provided, cleaning steps include checking for missing values (‘df.isnull().sum()’), filling missing values with zeros (‘df.fillna(0, inplace=True)’), and potentially removing duplicate records (‘df.drop_duplicates(inplace=True)’). These steps ensure data integrity and reliability for downstream tasks, such as ensuring no erroneous data affects analysis or model performance.

3) Data Labelling

Data labelling involves assigning categorical labels or numerical values to data points, often crucial for supervised learning tasks. In the provided code, data labelling is exemplified by mapping textual categories ('benign', 'defacement', 'phishing', 'malware') to numerical values ('0', '1', '2', '3') using a dictionary (rem). This step transforms categorical data into a format that machine learning algorithms can interpret, facilitating model training and evaluation.

4) Feature Extraction and Engineering

Feature extraction is a crucial step in preparing data for machine learning tasks, especially in the domain of URL classification. This process involves deriving attributes from raw URL data that serve as inputs for machine learning models to detect between different URL classes. In the context of URL classification, various informative features are extracted from URLs to provide insights into their characteristics. These features include the URL length, presence of specific characters or patterns (e.g.,'@', '?', '-', '=', '.', '#', '%', '+', '$', '!', '*', ','), identification of abnormal URL structures, usage of secure protocols like HTTPS, counts of digits and letters in the URL, detection of URL shortening services. The "Get URL Region" function on the other hand extracts critical features from URLs, including domain names, paths, parameters, and top-level domains, which are then used to enhance machine learning models' ability to classify URLs accurately. By parsing and analysing these components, the function provides valuable context such as domain reputation, presence of sensitive information, and irregular URL structures, aiding in the detection of malicious activities like phishing or malware distribution. Extracting these features enables machine learning models to capture distinctive patterns associated with malicious or benign URLs, facilitating accurate classification and detection of malicious activities on the web. Through feature extraction, the model gains valuable insights into the underlying structure of URLs, enhancing its ability to generalise and make informed predictions.

5) Data Splitting

The pre-processed data was split into training and test sets using an 80-20 split ratio. This ensures that 80% of the data is used for training the models, and 20% is used for evaluating their performance.

6) List of Models

Several classifiers were trained independently, including DecisionTreeClassifier, RandomForestClassifier, AdaBoostClassifier, KNeighborsClassifier, SGDClassifier, ExtraTreesClassifier, GaussianNB, and XGBClassifier.

7) Cross-Validation

Each model was evaluated using 5-fold cross-validation (cv=5). This involves splitting the training data into 5 subsets, training the model on 4 subsets, and validating it on the remaining subset. This process is then repeated 5 times, and the mean accuracy is calculated to evaluate the model's performance.

8) Test Evaluation and Selecting Top Models

The trained models were evaluated on the test set to measure their accuracy. The models are then sorted based on their test accuracy scores. The top 4 models with the highest accuracy were selected for the ensemble

9) Voting Classifier

Firstly, the top 4 models were combined into a Voting Classifier with hard voting. In hard voting, each model casts a vote for a class label, and the class with the majority votes becomes the final prediction. The voting classifier was

evaluated using 5-fold cross-validation to ensure it generalizes well to unseen data. The voting classifier was trained on the entire training set. The pipeline is also generated to have a better understanding of the Voting Classifier architecture as can be seen in Figure 3.

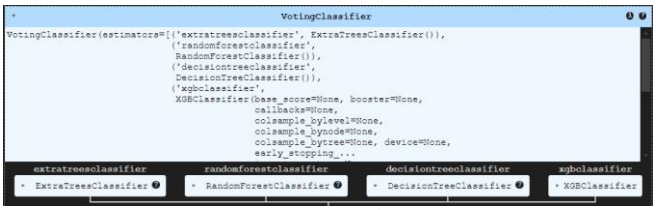


Figure 3 Voting Classifier Pipeline

10) Model Saving, Loading and Application

The trained model is saved using 'joblib', allowing it to be easily loaded and used for future predictions without the need for retraining. This ensures the model's reusability and practicality. The saved model can be loaded back into memory using 'joblib.load()'. This restores the model to its trained state. Once loaded, the model can be used to make predictions on new, unseen data. This involves passing new data points to the model's predict() method to get the predicted class labels.

VI. EVALUATION AND FINDINGS

A. Model Results

The outcome of results for the imbalanced and balanced dataset in the proposed model can be seen in Table 2, Table 3, Table 4 and Table 5. It should also be noted that each class represents categories where Class 0 is for benign, Class 1 for defacement, Class 2 for phishing and Class 3 for malware.

Table 2 Model Result for Imbalanced Dataset

	Imbalanced Dataset			
	Class 0	***Class 1***	***Class 2***	***Class 3***
Precision	0.92	0.95	0.85	**0.97**
Recall	**0.98**	**0.98**	0.58	0.88
F1 – Score	0.95	**0.96**	0.69	0.92
Accuracy	**91.85%**			
Cross-Validation Accuracy	**91.69%**			

Table 3 Model Result for Balanced Dataset with Balanced Random Forest

	Balanced Dataset with Balanced Random Forest			
	Class 0	***Class 1***	***Class 2***	***Class 3***
Precision	0.92	0.95	0.85	**0.97**
Recall	**0.98**	**0.98**	0.58	0.88
F1 – Score	0.95	**0.96**	0.69	0.93
Accuracy	**91.88%**			
Cross-Validation Accuracy	**91.69%**			

Table 4 Model Result for Balanced Dataset with Random Under Sampling

	Balanced Dataset with Random Under Sampling			
	Class 0	***Class 1***	***Class 2***	***Class 3***
Precision	**0.95**	0.92	0.50	0.83
Recall	0.83	**0.97**	0.74	0.91
F1 – Score	0.89	**0.95**	0.60	0.87
Accuracy	**84.42%**			
Cross-Validation Accuracy	**86.25%**			

Table 5 Model Result for Balanced Dataset with Random Over Sampling

	Balanced Dataset with Random Over Sampling			
	Class 0	***Class 1***	***Class 2***	***Class 3***
Precision	**0.95**	0.92	0.50	0.83
Recall	0.83	**0.97**	0.74	0.91
F1 – Score	0.89	**0.95**	0.60	0.87
Accuracy	**87.19%**			
Cross-Validation Accuracy	**92.88%**			

B. Model Performance Conclusion

The proposed model achieved the highest accuracy result of 91.88% by applying Balanced Random Forest. The proposed model also achieved the highest cross-validation mean accuracy of 92.88% by applying Random Over Sampling.

VII. CONCLUSION

In conclusion, the detection of malicious URLs using machine learning techniques has emerged as a critical area of research, driven by the continual evolution and proliferation of new threats in cyberspace. Addressing the challenges posed by the rapid development of malicious URLs remains a paramount concern in modern society, where reliance on web-based services is pervasive. Many cybersecurity experts present a technique to combat the evolution of malicious URLs from causing more harm with its new attacks, mutations and variants. This is a significant barrier to designing an effective and robust malicious URLs detection approach. Therefore, to address this issue, this research uses malicious URLs classification using a voting ensemble learning approach to detect those malicious URLs.

Some challenges faced while doing the early stage of the research is the limited research of applying balanced datasets in Machine Learning for detecting malicious URLs. Moreover, when applying balanced dataset using balancing techniques, the results were lower in accuracy compared to the original imbalanced dataset. This underscores the complexity of addressing class imbalance in malicious URLs. Hence, as the research progressed, further exploration of advanced algorithms, feature engineering strategies and ensemble methods were researched and applied. The final

proposed model achieved a higher accuracy than the proposed model in the early stage and the objective has been achieved as the balanced dataset achieved greater results than the imbalance dataset.

REFERENCES

[1] 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, 2022. [Online]. Available: https://doi.org/10.3390/s22093373

[2] P. L. Indrasiri, M. N. Halgamuge, and A. Mohammad, "Robust Ensemble Machine Learning Model for Filtering Phishing URLs: Expandable Random Gradient Stacked Voting Classifier (ERG-SVC)," *IEEE Access*, vol. 9, pp. 150142-150161, 2021. [Online]. Available: https://doi.org/10.1109/access.2021.3124628

[3] Q. A. Al-Haija and M. Al-Fayoumi, "An intelligent identification and classification system for malicious uniform resource locators (URLs)," 2023. [Online]. Available: https://doi.org/10.1007/s00521-023-08592-z

[4] Z. Wang, X. Ren, S. Li, B. Wang, J. Zhang, and T. Yang, "A Malicious URL Detection Model Based on Convolutional Neural Network," *Security and Communication Networks*, 2021, pp. 1-12. [Online]. Available: https://doi.org/10.1155/2021/5518528

[5] R. Dsouza, "Malicious URL(s) classification," MSc Research Project Cloud Computing, 2020. [Online]. Available: https://norma.ncirl.ie/4138/1/rohandsouza.pdf

[6] L. R. Kalabarige, R. S. Rao, A. Abraham, and L. A. Gabralla, "Multilayer Stacked Ensemble Learning Model to Detect Phishing Websites," *IEEE Access*, 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9843994

[7] D. S. Krishna, T. Lavanya, R. Kola, S. Pedaprolu, S. K. S. Basha, and R. S. Rao, "IEEE Conference Publication," 2020. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10118072

[8] Z. Ghaleb Al-Mekhlafi, B. Abdulkarem Mohammed, M. Al-Sarem, F. Saeed, T. AlHadhrami, M. T. Alshammari, A. Alreshidi, and T. Sarheed Alshammari, "Phishing Websites Detection by Using Optimized Stacking Ensemble Model," *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 109-125, 2022. [Online]. Available: https://doi.org/10.32604/csse.2022.020414

[9] Y. Khukalenko, I. Stopochkina, and M. Ilin, "Machine Learning Models Stacking in the Malicious Links Detecting," *Theoretical and Applied Cybersecurity*, vol. 5, no. 1, 2023. [Online]. Available: https://doi.org/10.20535/tacs.2664-29132023.1.287752

[10] M. Sameen, K. Han, and S. O. Hwang, "PhishHaven—An Efficient Real-Time AI Phishing URLs Detection System," *IEEE Access*, vol. 8, pp. 83425-83443, 2020. [Online]. Available: https://doi.org/10.1109/access.2020.2991403

[11] D. K. Mondal, B. C. Singh, H. Hu, S. Biswas, Z. Alom, and M. A. Azim, "SeizeMaliciousURL: A novel learning approach to detect malicious URLs," *Journal of Information Security and Applications*, vol. 62, 2021, p. 102967. [Online]. Available: https://doi.org/10.1016/j.jisa.2021.102967

[12] S. Jalil, M. Usman, and A. Fong, "Highly accurate phishing URL detection based on machine learning," *Journal of Ambient Intelligence and Humanized Computing*. [Online]. Available: https://doi.org/10.1007/s12652-022-04426-3

[13] F. Alkhudair, M. Alassaf, R. U. Khan, and S. Alfarraj, "Detecting Malicious URL," in 2020 International Conference on Computing and Information Technology (ICCIT-1441), IEEE Access. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9213792

[14] S. Afzal, M. Asim, A. R. Javed, M. O. Beg, and T. Baker, "URLdeepDetect: A Deep Learning Approach for Detecting Malicious URLs Using Semantic Vector Models," *Journal of Network and Systems Management*, vol. 29, no. 3, 2021. [Online]. Available: https://doi.org/10.1007/s10922-021-09587-8

[15] S. Singh, M. P. Singh, and R. Pandey, "Phishing Detection from URLs Using Deep Learning Approach," *IEEE Xplore*, October 2020. [Online]. Available: https://doi.org/10.1109/ICCCS49678.2020.9277459

[16] J. Rashid, T. Mahmood, M. W. Nisar, and T. Nazir, "Phishing Detection Using Machine Learning Technique," *IEEE Xplore*. [Online]. Available: https://doi.org/10.1109/SMART-TECH49988.2020.00026

[17] M. Somesha, A. R. Pais, R. S. Rao, and V. S. Rathour, "Efficient deep learning techniques for the detection of phishing websites," 2020. [Online]. Available: https://doi.org/10.1007/s12046-020-01392-4

[18] M. Korkmaz, O. K. Sahingoz, and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. [Online]. Available: https://doi.org/10.1109/icccnt49239.2020.9225561

[19] A. Subasi and E. Kremic, "Comparison of Adaboost with MultiBoosting for Phishing Website Detection," *Procedia Computer Science*, vol. 168, 2020, pp. 272-278. [Online]. Available: https://doi.org/10.1016/j.procs.2020.02.251

[20] R. Yang, K. Zheng, B. Wu, C. Wu, and X. Wang, "Phishing Website Detection Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning," *Sensors*, vol. 21, no. 24, p. 8281, 2021. [Online]. Available: https://doi.org/10.3390/s21248281

[21] L. A. Sevastianov and E. Y. Shchetinin, "On methods for improving the accuracy of multi-class classification on imbalanced data," in *X International Conference Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems (ITTMM-2020)*. [Online]. Available: https://ceur-ws.org/Vol-2639/paper-06.pdf

[22] Anonymous, "Mid-Year Report Threat Landscape 2023," *Cybersecurity Malaysia*, 2023. [Online]. Available: https://www.cybersecurity.my/data/content_files/26/2511.pdf

[23] S. S. Rawat and A. K. Mishra, "The Best ML Classifier(s): An empirical study on the learning of imbalanced and resampled credit card data," in *2023 Second International Conference on Informatics (ICI)*. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10421691

[24] M. Siddhartha, "Malicious URLs dataset," 2021. [Online]. Available: https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset