PhishGuard: Machine Learning-Powered Phishing URL Detection

Saydul Akbar Murad[1], Nick Rahimi[1]*, and Abu Jafar Md Muzahid[2]
[1]*School of Computing Sciences & Computer Engineering, University of Southern Mississippi, Hattiesburg, USA*
[2]*Faculty of Computing, University Malaysia Pahang, Pahang, Malaysia*
E-mail: saydulakbar.murad@usm.edu, nick.rahimi@usm.edu, mrumi98@gmail.com

***Abstract*—Phishing is a major threat to internet security, targeting human vulnerabilities instead of software vulnerabilities. It involves directing users to malicious websites where their sensitive information can be stolen. Many researchers have worked on detecting phishing URLs, but their models have limitations such as low accuracy and high false positives. To address these issues, we propose a machine-learning model to detect phishing URLs. To detect these malicious URLs, we use a dataset of over 500K entries collected from the Kaggle website. The dataset is used to train five supervised machine-learning techniques, including K-Nearest Neighbors (KNN), Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), and Random Forest (RF). The aim is to improve the performance of the classifier by studying the features of phishing websites and selecting a better combination of them. To measure the performance, we considered three parameters: accuracy, precision, and recall. The LR technique yielded the best performance, demonstrating its efficacy in detecting phishing URLs.**

***Index Terms*—Phishing URL, Machine Learning, KNN, SVM, Logistic Regression.**

I. Introduction

Phishing is a form of online deception where fraudsters fabricate fraudulent websites or emails that seem genuine to dupe users into divulging sensitive information, such as credit card details or login credentials [1]. In recent times, phishing has emerged as one of the most prominent cybersecurity menaces. Phishing attacks can be very effective because they exploit human vulnerabilities rather than technical vulnerabilities. These attacks often use social engineering techniques to trick users into giving away sensitive information, such as login credentials and personal information. Phishing URL detection refers to the process of identifying and blocking URLs (Uniform Resource Locators) that lead to phishing websites [2]. This involves using various techniques to analyze URLs and their associated web content to determine whether they are legitimate or malicious.

As a result, many organizations and individuals have become more aware of the dangers of phishing and have taken steps to protect themselves. One of the most effective ways to prevent phishing attacks is to detect and block phishing URLs. Phishing URL detection research has been ongoing for several years, and it has become increasingly important as phishing attacks have become more sophisticated. Researchers have developed various techniques and algorithms to identify phishing URLs based on different characteristics, such as the URL structure, domain reputation, and content analysis. Some common techniques used in phishing URL detection include:

- **Blacklisting:** This involves maintaining a list of known phishing URLs and blocking access to them.
- **Machine learning:** This involves training models to identify patterns in phishing URLs and using those models to detect new phishing URLs.
- **URL analysis:** This involves analyzing the structure and content of URLs to identify suspicious or malicious characteristics.
- **Domain reputation analysis:** This involves analyzing the reputation of the domain associated with a URL to determine whether it is likely to be malicious.
- **Content-based analysis:** This involves analyzing the content of a webpage associated with a URL to determine whether it is likely to be malicious.

All the techniques mentioned for detecting phishing websites have their limitations. For instance, the blacklist technique relies on an up-to-date and comprehensive list of known malicious URLs or IP addresses, but attackers can easily create new phishing websites or move their operations to new IP addresses, making it difficult to maintain an accurate blacklist. URL analysis may not be effective at detecting phishing attacks that use legitimate websites as a platform. For example, an attacker may create a phishing email that links to a legitimate website but directs the user to enter sensitive information on a fake login page. In this case, the URL analysis may not detect any malicious activity, as the website itself is legitimate. The drawback of domain reputation analysis is that it focuses on the domain name itself, rather than the content of the website. This means that even if a domain has a good reputation, it may still be hosting phishing content or other malicious activity. In Content-based analysis, attackers can use techniques such as URL cloaking or obfuscation to hide the true destination of a link or to make the phishing content appear legitimate. This can make it difficult for content-based analysis to detect phishing websites.

To overcome those limitations, machine learning and deep learning techniques have been widely used for phishing URL detection, allowing for the creation of more accurate models that can detect even previously unseen phishing URLs. In [3], Zamir, Ammara, et al. introduced a framework for identifying

979-8-3503-2759-5/23/$31.00 ©2023 IEEE
DOI 10.1109/CSCE60160.2023.00371

phishing websites through a stacking model. In this process, various feature selection techniques such as information gain, gain ratio, Relief-F, and recursive feature elimination (RFE) are utilized to analyze the characteristics of the phishing dataset. In [4], the authors evaluated and compared several machine learning techniques to predict phishing websites. The dataset used in their study was obtained from the PhishTank website, consisting of roughly 11,000 sample websites. For testing purposes, 10% of the samples were utilized. In another paper [5], the authors proposed three separate deep-learning techniques for the identification of phishing websites. These techniques include long short-term memory (LSTM) and convolutional neural network (CNN) for comparative purposes, as well as an LSTM-CNN hybrid approach. Al-Tamimi, Y., and M. Shkoukani. [6] created a model utilizing machine learning algorithms, specifically the decision tree and random forest.

Previous research in the field of machine learning for predicting phishing URLs had limitations, such as a narrow focus on accuracy without considering other performance metrics, as well as small sample sizes in some studies. In light of these limitations, we proposed a novel machine-learning model specifically designed for predicting phishing URLs. The objectives of this research article are as follows:

- Developed a machine learning model that can accurately and efficiently identify phishing websites or URLs.
- Minimize the false positives and false negatives rate.
- Improve the security of online systems and protect users from falling victim to phishing attacks.

The remainder of the paper is divided into the following sections: In *Section II*, a literature review is presented. *Section III* of the document covered methodology. Results and discussion were maintained in *Section IV*. Finally, *Section V* addresses the paper's concluding observations.

II. Literature Review

Numerous methods have been suggested to counter the menace of phishing attacks, many of which involve extracting phishing features. However, some of these techniques require running the page, which can be resource-intensive and time-consuming. The authors of [1] proposed an intelligent system that serves as an extension to internet browsers, providing an additional functionality of detecting phishing websites. The system is designed to automatically alert the user when a phishing website is detected. The algorithm used in the system is limited to the random forest algorithm. In [7], the authors employed two classifiers, Random Forest and Support Vector Machine (SVM), to detect phishing elements. The objective was to help users determine whether a given link was a legitimate website or a phishing website. However, the study did not include any performance metrics such as accuracy or precision. To address this gap, in this paper [8] analyzes the common attributes exhibited by phishing websites and develops a model to detect such websites. The study employed five machine learning algorithms and achieved good accuracy, especially with the use of Artificial Neural Network (ANN). However, the dataset was split only into 80% and 20%, which may not provide a comprehensive evaluation of the model's performance. In contrast, this paper [9] proposes a framework for detecting phishing websites using random forest ensemble techniques. The approach involves combining random forest with k-means clustering to capture feature correlation. However, the study is limited to evaluating the framework using a dataset containing only 5000 samples. The researchers in this study [10] performed cross-validation to test the effectiveness of the model, as well as to assess the correlation between the features. They used Logistic Regression to determine the significance of the features and also tested the Multinomial Naïve Baye classifier. The results revealed that the Logistic Regression classifier had the best accuracy compared to other classifiers. *Table I* provides a summary of different studies that have used various machine learning algorithms to classify or predict outcomes for a particular task or problem. Each study is represented by a row in the table, and the columns represent different aspects of the study, such as the algorithms used, the performance metrics evaluated, and the size of the training and testing datasets.

The first column lists the references for each study. The second column specifies the machine learning algorithms used in each study. The third column indicates whether the study reported accuracy as a performance metric. The fourth column indicates whether the study reported precision as a performance metric. The fifth column indicates whether the study reported recall as a performance metric. A checkmark (✓) in this column means that the study reported accuracy/precision/recall, while a cross (✗) means that it did not. The sixth column shows the size of the training and testing datasets used in each study. The percentage values indicate the proportion of the dataset used for training and testing, respectively. In some studies, the size of the dataset is not reported (N/A).

III. Methodology

In this section, we explain how our proposed model works for detecting phishing URLs. To conduct the experiment, we employed five robust machine learning algorithms, which were executed using Python. These algorithms include Logistic Regression, KNN, Decision Tree, Random Forest, and SVM, all of which used the default settings. *Figure 1* describes the total working procedure of this experiment.

A. Dataset Description

The dataset was obtained from Kaggle [11], which is a reliable platform for datasets, and it contains 549,346 entries. The given dataset consists of a pair of columns, wherein the first column enlists distinct URLs, and the second column denotes the labels. The prediction column comprises two categories: Good (72%), indicating that the URLs are legitimate and do not possess any malicious content, while Bad (28%) signifies that the URLs are deceptive and carry malicious content, making them phishing sites. *Table II* showcases a sample dataset.

TABLE I
ANALYSIS OF PREVIOUS RESEARCH WORK.

Ref.	Algorithms	Accuracy	Precision	Recall	Training & Test size
Alswailem, Amani, et al. [1]	RF	✓	✗	✗	80% / 20 %
Helmi et al. [7]	RF, SVM	✗	✗	✗	N/A
Ameya Chawla [8]	RF, DT, LR, KNN, ANN	✓	✓	✓	80% / 20%
Mohammad A. Alsharaiah et al. [9]	RF	✓	✓	✓	70% / 20%
Zongo et al. [10]	NV, LR	✓	✗	✗	80% / 20%
Proposed Model	SVM, LR, RN, DT, KNN	✓	✓	✓	Nine Sample

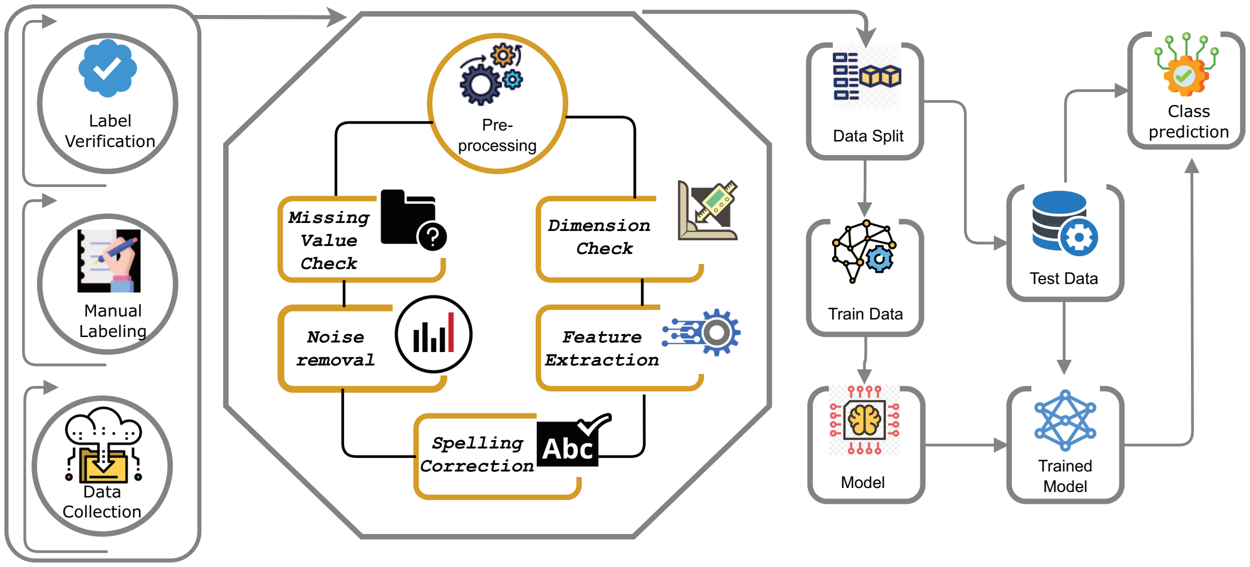

Fig. 1. Machine Learning Workflow: From Data to Model.

TABLE II
URL CLASSIFICATION: LEGITIMATE VS. MALICIOUS.

URL	Label
www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrcmd=_home-customer&nav=1/loading.php	bad
anonymeidentity.net/remax./remax.htm	bad
lastminutevillas.net/	good
lasvegasbroker.com/	good

B. Data Pre-processing

Data pre-processing is the process of preparing and cleaning raw data to be used in machine learning models. This involves several steps, including data cleaning, missing value check, data transformation, and data reduction. We have used all the above pre-processing steps in our experiment. First, we have checked whether there are any missing values or not. In this dataset, we do not find any missing values.

Data cleaning is another important step that involves removing any errors or inconsistencies in the data, such as missing values, duplicates, or outliers. This ensures that the data is accurate and reliable for analysis. We do not get any missing values for this dataset. But, we get a good number of duplicate values like 42151. Before going to splitting the dataset into train and test, we remove those duplicate data first. To check the outliers in the URL dataset, we applied outlier detection and handling techniques to improve the accuracy and reliability of the machine-learning model.

As we work on the URL, text tokenization is an important process for this kind of data. Text tokenization is the process of breaking down text into smaller units, called tokens, which can then be used as input for machine learning models. Tokens are usually words, but they can also be phrases, symbols, or other types of units depending on the task at hand. Tokenization is a critical step in machine learning tasks that involve text analysis. The main goal of text tokenization is to convert raw text data into a format that can be easily processed by machine learning algorithms. There are several methods for text tokenization. In this experiment, we have used Word Tokenization which involves breaking down the text into individual words or terms.

We have used another pre-process technique called text stemmed. Text stemming is a natural language processing technique used to reduce words to their base or root form, which is often referred to as the “stem.” The purpose of stemming is to reduce the dimensionality of the text data and to group together different forms of the same word, which can help improve the accuracy of certain text analysis tasks, such as information retrieval or sentiment analysis.

TABLE III
THIS TABLE REPRESENTS TRAINING AND TESTING SAMPLE SIZES.

Training Size	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Test Size	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1

Stemming algorithms typically work by removing suffixes from words, which can include tense, pluralization, and other grammatical variations. There are several stemming algorithms available, including the Porter stemmer, Snowball stemmer, and Lancaster stemmer. In this experiment, we used Snowball stemmer here. The Snowball stemmer works by applying a set of rules to a word to transform it into its base form. The rules are based on patterns of word endings, prefixes, and suffixes. The stemmer applies the rules in a step-by-step manner until it arrives at the root form of the word.

C. Split the dataset

In machine learning, splitting a dataset involves dividing it into two or more distinct subsets to train and evaluate a machine learning model. Typically, this involves creating a training set and a testing set. The training set is used to teach the model, while the testing set is used to assess its performance on new, unseen data. This approach is crucial to avoid overfitting the model to the training data and ensure its ability to generalize to novel data. Here are the steps we use to divide the dataset into training and testing sets:

- First, we imported the necessary libraries like pandas, scikit-learn, etc. and load the dataset into a pandas dataframe.
- Next, to divide the data into two separate sets, namely the training set and the testing set, we utilized the "train-test-split" function from the scikit-learn library. The purpose of creating the training set was to train the machine learning model, while the testing set was utilized to assess the model's performance.
- Once the data was divided, we utilized the training set for training the machine learning model, while the testing set was used to assess the model's performance.

A common split is to use 70% of the data for training and 30% for testing. But in this experiment, we have used nine training samples and nine testing samples that are shown in *table III*. We used nine different training and testing sizes in order to investigate the impact of varying dataset sizes on the machine learning model's performance. By using a range of training and testing size combinations, we can analyze how the model's accuracy, precision, recall, or other performance metrics change with different proportions of data used for training and testing.

D. Implemented Model

In this experiment, we have used five robust machine learning algorithms. For all algorithms, we find a good result. Comparing all five algorithms, logistic regression shows the best result. To measure the performance of algorithms, we have utilized three performance parameters. *Table IV* represent the details of implemented algorithms.

E. Performance Evaluation

In this experiment, we have considered a total of three evaluation metrics, which are explained as follows:

Accuracy: Accuracy, which measures the correctness of predictions made by a machine learning model, is determined by the proportion of correct predictions over the total number of predictions made. It is typically represented as a percentage, with perfect accuracy (100%) indicating that the model made no mistakes in its predictions. This definition can be formalized using *equation 1*.

$$Accuracy = \frac{TP + TN}{TP + TN.FP + FN} \tag{1}$$

Precision: Precision is a metric used to evaluate the performance of a machine learning model's positive class predictions. A high precision score indicates that the model has a low false positive rate, meaning it is unlikely to incorrectly classify a negative instance as positive. Therefore, precision provides an insight into how precise or accurate the model is when predicting positive instances. *Equation 2* explains the precision.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall: Recall tells us how well the model can identify positive instances in the dataset. A high recall score indicates that the model has a low false negative rate, which means that it is unlikely to miss a positive instance. *Equation 3* defines the recall formula.

$$Recall = \frac{TP}{TP + TN} \tag{3}$$

IV. RESULT AND DISCUSSION

In this experiment, we have utilized a total of five machine learning algorithms such as KNN, Decision Tree, Random Forest, SVM, and logistic regression. For all five algorithms, we have considered three performance metrics like accuracy, precision, and recall. For all five algorithms, we got a good result. For the training and testing, we divided the dataset into nine trains and test data sizes. We got the result for each train and test data.

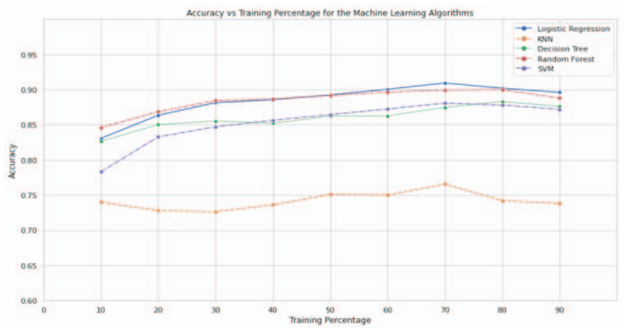

Fig. 2. Accuracy VS Training Percentage for the Machine Learning Algorithms.

TABLE IV
DETAILED TRAINING PARAMETERS OF IMPLEMENTED ALGORITHMS.

Name of Algorithms	*Description*	*Initial Parameters*
K-Nearest Neighbors [12], [13]	KNN is a machine learning technique that involves identifying the k closest data points in the training set to a new data point. The algorithm then assigns the new data point to the class or value that is most common among its k closest neighbors. If the task is classification, the class labels of the k closest neighbors are used to determine the class label of the new data point. If the task is regression, the average of the values of the k closest neighbors is used to predict the value of the new data point. The value of k is an important parameter in KNN because it determines the flexibility of the decision boundary. A smaller value of k provides a more flexible boundary, but it is more susceptible to noise and outliers. Conversely, a larger value of k provides a smoother boundary but is more likely to misclassify data points that lie near the boundary between classes.	n_neighbors, weights, leaf_size
Decision Tree Classifier [14], [15]	The Decision Tree Classifier is a machine learning approach that constructs a model resembling a tree by splitting the data repeatedly based on features that have the best discriminatory power between classes. The algorithm generates internal nodes representing attributes or features of the data and leaf nodes representing class labels. It chooses the most informative feature to divide the data based on certain measures (such as Gini impurity or information gain) and recursively divides the data until it meets a stopping criterion. The stopping criterion may be a limit on the maximum depth of the tree or a minimum number of data points assigned to a leaf node.	criterion, max_depth, max_features
RandomForest Classifier [16]	The Random Forest Classifier is a machine learning technique that constructs an ensemble of decision trees known as a "forest." Each tree is trained on a random subset of data and features. The algorithm begins by selecting random subsets of the training data and features, which are then utilized to train a decision tree. This procedure is iterated multiple times to produce a set of decision trees, each of which produces a prediction. In the classification task, the final prediction is obtained by aggregating the individual predictions of all the trees, usually through a majority vote.	n_estimators, max_depth, max_features
SVM [17], [18]	Support Vector Machines (SVM) use a technique that seeks to identify the best hyperplane in a dataset to maximize the separation margin between different classes. SVM is particularly useful in situations where the data cannot be separated linearly and needs a nonlinear transformation to achieve precise classification. The algorithm applies a kernel function to map the data into a higher-dimensional space, where it can be effectively separated. It then determines the optimal hyperplane in this transformed space.	kernel, gamma, degree
Logistic Regression [19]	Logistic Regression is a statistical approach that allows us to predict the likelihood of a binary outcome based on one or more predictor variables. It is frequently used in machine learning for binary classification tasks, where the goal is to predict whether an observation belongs to a particular class. Logistic regression uses a logistic or sigmoid function to model the association between the independent variables and the probability of the binary outcome. The sigmoid function produces a probability value between 0 and 1 by taking any real number input. The model parameters are estimated through maximum likelihood estimation, which involves identifying the parameter values that maximize the likelihood of observing the training data given the model.	Penalty, Maximum iterations, Class weight

Figure 2 define the accuracy curve for all five machine learning that is implemented in this experiment. The x-axis defines the training percentage, and the y-axis shows the accuracy. This graph shows that when the number of training data is 10%, the accuracy rate is low, and that is increased with the increase of the number of the training sample. For all training data, the KNN performance is worse compared to the other four algorithms. When the number of training data is 70%, the KNN accuracy slightly higher than 70%, and at other times, it's lower than 70%, what's defines low performance. For all training samples, logistic regression accuracy is best. When the training size is 10%, the accuracy is 85% and rose to more than 90% for 70% training data. But when the training size has been increased, the accuracy falls down. In the beginning, the SVM performance was low, but with the increase of training data, it's increased to what is near to 90%. For all algorithms, the common thing is between 60 - 80% training data; the accuracy is best.

Figure 3 represents the precision performance for all implemented algorithms. This graph shows that, for KNN, the precision score is the value that defines the low performance of this algorithm. On the other hand, SVM performance is best for almost all training samples. When the training sample is 10%, the precision is more than 95% for SVM, but with the increase of the training sample, the score falls down. In the beginning, the score was low for logistic regression and random forest, but it increased with the increase of training data. The precision is also good for the decision tree algorithm, but compared with other algorithms, the score is low.

Recall is another performance metric that we have considered in this experiment, and it's shown in *figure 4*. This figure shows that, in the beginning, when the training data was 10%,

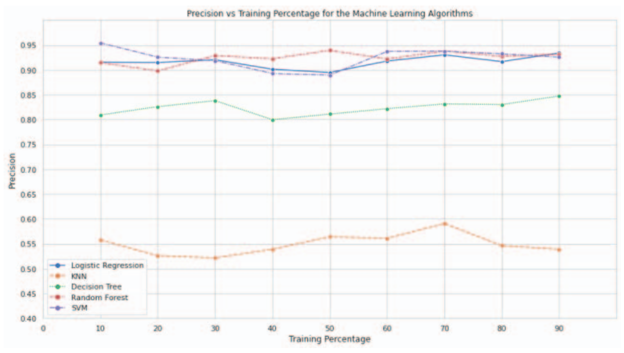

Fig. 3. Precision VS Training Percentage for the Machine Learning Algorithms.

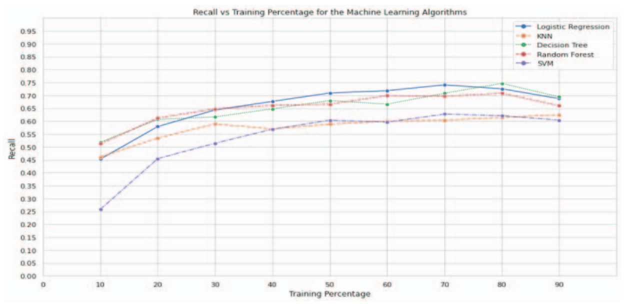

Fig. 4. Recall VS Training Percentage for the Machine Learning Algorithms.

the performance was very low for all algorithms, which is like less than 50%. Especially for SVM, it's very low. But with the increase of training samples, the performance is increased for all algorithms. When the number of training samples is 80%, the recall value was max for decision tree. The recall value was also higher for logistic regression for 70% training data. When the training sample was increased to 90%, the recall performance fell down for all implemented algorithms.

V. Conclusion

We started this research by analyzing previous research work. For this experiment, we have used a secondary dataset that is collected from Kaggle. This dataset contains two columns like, URL and level data. To analyze the dataset, we have used a total of five robust ML Algorithms. This study proposes a framework for detecting phishing websites using an effective machine learning technique that involves preprocessing, data splitting, and classification algorithms. The ensemble-based logistic regression technique outperformed classical classification algorithms, achieving the best results among the compared methods. The pre-processing steps included removing all duplicate values before the data split. The dataset split into nine training and testing sets, and the performance is evaluated for different training and testing sizes. The logistic regression ensemble learning technique achieved the best accuracy of 93.64%. Additionally, the SVM algorithm achieved a precision score of 96% with only 10% training data.

References

[1] Alswailem, Amani, et al. "Detecting phishing websites using machine learning." 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS). IEEE, 2019.
[2] Ravindra, Salvi Siddhi, et al. "Phishing Website Detection Based on URL." International Journal of Scientific Research in Computer Science, Engineering and Information Technology (USRCSEIT) 7.3 (2021): 589-594.
[3] Zamir, Ammara, et al. "Phishing web site detection using diverse machine learning algorithms." The Electronic Library 38.1 (2020): 65-80.
[4] Shahrivari, Vahid, Mohammad Mahdi Darabi, and Mohammad Izadi. "Phishing detection using machine learning techniques." arXiv preprint arXiv:2009.11116 (2020).
[5] Alshingiti, Zainab, et al. "A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN." Electronics 12.1 (2023): 232.
[6] Al-Tamimi, Y., and M. Shkoukani. "Employing cluster-based class decomposition approach to detect phishing websites using machine learning classifiers." International Journal of Data and Network Science 7.1 (2023): 313-328.
[7] Helmi, Rabab Alayham Abbas, Md Gapar Md Johar, and Muhammad Alif Sazwan Bin Mohd Hafiz. "Online Phishing Detection Using Machine Learning." 2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC). IEEE, 2023.
[8] Chawla, Ameya. "Phishing website analysis and detection using Machine Learning." International Journal of Intelligent Systems and Applications in Engineering 10.1 (2022): 10-16.
[9] Alsharaiah, M., Abu-Shareha, A., Abualhaj, M., Baniata, L., Adwan, O., Al-saaidah, A & Oraiqat, M. (2023). A new phishing-website detection framework using ensemble classification and clustering.International Journal of Data and Network Science, 7(2), 857-864.
[10] Zongo, Wend-Benedo Simeon, Boukary Kabore, and Ravirajsinh Sajubha Vaghela. "Phishing URLs Detection Using Machine Learning." Advancements in Smart Computing and Information Security: First International Conference, ASCIS 2022, Rajkot, India, November 24–26, 2022, Revised Selected Papers, Part II. Cham: Springer Nature Switzerland, 2023.
[11] https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls
[12] Ogunseye, Elizabeth Oluyemisi, et al. "Predictive analysis of mental health conditions using AdaBoost algorithm." ParadigmPlus 3.2 (2022): 11-26.
[13] Murad, Saydul Akbar, et al. "AI Powered Asthma Prediction Towards Treatment Formulation: An Android App Approach." Intelligent Automation & Soft Computing 34.1 (2022).
[14] Adhikary, Apurba, et al. "Edge assisted crime prediction and evaluation framework for machine learning algorithms." 2022 International Conference on Information Networking (ICOIN). IEEE, 2022.
[15] Kowsher, Md, Anik Tahabilder, and Saydul Akbar Murad. "Impact-learning: a robust machine learning algorithm." Proceedings of the 8th international conference on computer and communications management. 2020.
[16] Husain, Wahidah, Lee Ker Xin, and Neesha Jothi. "Predicting generalized anxiety disorder among women using random forest approach." 2016 3rd international conference on Computer and information sciences (ICCOINS). IEEE, 2016.
[17] Jain, Tarun, et al. "Machine Learning Techniques for Prediction of Mental Health." 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA). IEEE, 2021.
[18] Prottasha, Nusrat Jahan, et al. "Impact learning: A learning method from feature's impact and competition." Journal of Computational Science (2023): 102011.
[19] LaValley, Michael P. "Logistic regression." Circulation 117.18 (2008): 2395-2399.