Phishing Website URL's Detection Using NLP and Machine Learning Techniques

Article *in* Journal of Artificial Intelligence · December 2023

CITATIONS
0

READS
238

2 authors:

Dinesh Kalla
Colorado Technical University
25 PUBLICATIONS **228** CITATIONS
SEE PROFILE

Sivaraju Kuraku
University of the Cumberlands
18 PUBLICATIONS **193** CITATIONS
SEE PROFILE

Journal on Artificial Intelligence

Tech Science Press

DOI: 10.32604/jai.2023.043366

ARTICLE

Phishing Website URL's Detection Using NLP and Machine Learning Techniques

Dinesh Kalla[1,*] **and Sivaraju Kuraku**[2]

[1]Department of Computer Science, Colorado Technical University, Colorado Springs, CO, 80907, USA

[2]Department of Computer Science, University of the Cumberlands, Williamsburg, KY, 40769, USA

*Corresponding Author: Dinesh Kalla. Email: kalladinesh@outlook.com

Received: 30 June 2023 Accepted: 01 November 2023 Published: 18 December 2023

ABSTRACT

Phishing websites present a severe cybersecurity risk since they can lead to financial losses, data breaches, and user privacy violations. This study uses machine learning approaches to solve the problem of phishing website detection. Using artificial intelligence, the project aims to provide efficient techniques for locating and thwarting these dangerous websites. The study goals were attained by performing a thorough literature analysis to investigate several models and methods often used in phishing website identification. Logistic Regression, K-Nearest Neighbors, Decision Trees, Random Forests, Support Vector Classifiers, Linear Support Vector Classifiers, and Naive Bayes were all used in the inquiry. This research covers the benefits and drawbacks of several Machine Learning approaches, illuminating how well-suited each is to overcome the difficulties in locating and countering phishing website predictions. The insights gained from this literature review guide the selection and implementation of appropriate models and methods in future research and real-world applications related to phishing detections. The study evaluates and compares accuracy, precision and recalls of several machine learning models in detecting phishing website URL's detection.

KEYWORDS

Cybersecurity; artificial intelligence; machine learning; NLP; phishing detection; spam detection; phinshing website URLs

1 Introduction

Phishing websites have become a significant cybersecurity concern in today's digital realm. As online activities proliferate across e-commerce, banking, and social media, the threats posed by these deceptive websites have magnified. They are designed to masquerade as legitimate entities to steal users' sensitive information, leading to devastating consequences like monetary losses, data breaches, and jeopardized user privacy [1,2].

Machine learning emerges as a promising solution in the ongoing battle against such deceptive attacks. Traditional rule-based approaches need help to stay updated with the ever-evolving strategies employed by attackers, often proving ineffective in real-time threat scenarios. On the other hand, machine learning, with its ability to learn from vast datasets and discern patterns, provides a proactive

defense mechanism against phishing attacks [3]. Its adaptability allows it to anticipate emerging phishing strategies and offer preemptive solutions.

Numerous studies have delved into employing machine learning for phishing detection [4]. These techniques harness the power of predictive analytics, Machine learning, and Natural Language Processing (NLP) to enhance the accuracy and efficiency of phishing detection systems. However, the rapid evolution of phishing strategies calls for a more in-depth exploration and analysis of state-of-the-art machine learning methods to ensure continued effectiveness [5].

In the digital age, the integrity and safety of user data have become paramount. With every innovation in online security, there emerges a counterforce attempting to exploit the very vulnerabilities these measures seek to protect. As phishing attacks continue to evolve, our approach to their detection and prevention must likewise advance. Machine learning stands out as a beacon of hope in this landscape. By continuously updating and refining its knowledge from diverse data sources, it offers a dynamic counter to phishing's adaptive nature [5]. The future of cybersecurity hinges on our ability to seamlessly merge human intuition with computational prowess. As this study delves deeper into machine learning's potential, it serves as a testament to our commitment to protect users from the subtle, yet dangerous, nuances of phishing, reinforcing trust in the digital domain we increasingly depend upon.

This study aims to bridge this gap by comprehensively reviewing the latest machine learning models and techniques employed for phishing website detection. We assess each model's strengths, weaknesses, and applicability, drawing insights that could shape future research directions and real-world applications [6,7]. By integrating advanced machine learning techniques, the goal is to build robust defenses that not only detect but also anticipate and counteract phishing threats, thereby fostering a safer digital ecosystem.

2 Related Models and Methods

Phishing is an offense in which attackers pretend to be trustworthy websites or other organizations to trick consumers into disclosing private information such as usernames, passwords, or financial information. Phishing websites are made to seem authentic and reliable, and they often utilize methods like URL manipulation, social engineering, or email spoofing to deceive unwary consumers [5]. Phishing assaults may have serious repercussions, including money losses, identity theft, and weakened cybersecurity. The body of knowledge on phishing website detection has significantly advanced, emphasizing machine-learning approaches to improve detection efficacy and accuracy. Machine learning algorithms have shown that they can identify phishing websites by extracting patterns and attributes from massive datasets. Based on various traits and signs, these algorithms may categorize websites as authentic or fraudulent [3]. ML models and AI are often used to identify phishing websites and are examined in this research study.

This research study focuses on the examination of several machine learning models and techniques that are commonly employed to identify phishing websites. Exploring these methodologies, researchers aim to deepen our understanding of effective measures for combating the pervasive threat of phishing, bolstering the defenses against this nefarious practice and safeguarding individuals and organizations alike. Through continued advancements in machine learning and the collaborative efforts of cybersecurity experts, we can strive towards a safer digital landscape where the risks of phishing attacks are mitigated and trust can flourish. Several machine learning models and techniques that are often used to identify phishing websites are examined in this research study.

Several Machine learning-based approaches can be utilized to enhance the efficiency of anti-phishing systems. This is achieved by implementing an enhanced predictive model emphasising the optimal selection of feature vectors extracted from online elements like URLs, webpage properties, and webpage behaviour. The methodology involves an incremental component-based system that presents the feature vectors to the predictive models, utilizing both Support Vector Machine (SVM) and Naïve Bayes (NB) algorithms. These algorithms were tested on phishing and benign datasets, achieving an impressive accuracy of 99.96%.

The challenge most anti-phishing systems face, as highlighted by the authors, is the considerable computational overhead, susceptibility to zero-day attacks, and high false rates. Despite the encouraging accuracy rates achieved by many machine learning models in phishing detection, the efficacy of such models can be impeded by the selection and performance of feature vectors. These challenges can be addressed by developing an optimized feature selection module. This module prioritizes extracting the most pertinent features from URLs, webpage properties, and behavioral metrics, ensuring that the resultant feature vectors presented to the predictive model are streamlined and effective.

Despite the comprehensive description of the methodology and results in the paper, there appears to be a need for more explicit detail concerning the computational overhead of the proposed model. Computational overhead is crucial in real-world applications, especially when deploying solutions in memory-constrained environments or when real-time detection is required. For a holistic assessment of the model's efficiency, it would be imperative to consider metrics like model size, inference time, and memory consumption during training and inference processes. By evaluating these metrics, one can determine the viability of implementing the proposed model in practical, real-world settings.

2.1 Logistic Regression

The logistic regression model estimates the probability of an event (in this case, phishing) using the logistic function, which maps the linear combination of input features (x) to a probability (p) [8].

$$p(x) = \frac{1}{1 + e^{-(x-u)/s}} \tag{1}$$

where

u is location parameter (Midpoint curve p(u) = ½)

s is scale parameter for sigmoid function is also called as activation function for logistic regression.

$$p(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

where

e = base of natural algorithms.

The equation below represents logistic regression.

$$y = \frac{e^{(b_0 + b_1 x)}}{1 + e^{(b_0 + b_1 x)}} \tag{3}$$

where

X = input value, y = predicted output, b_0 = intercept term, b_1 = coefficient for input x.

2.2 K-Nearest Neighbors

KNN calculates distances between the target sample and its neighboring samples and assigns the most prevalent class label among the K nearest neighbors. KNN is versatile and easy to implement [9].

Given two feature vectors with numeric value

$$\mathrm{A} = (a_1, a_2, \ldots, a_n) \text{ and } \mathrm{B} = (b_1, b_2, \ldots, b_n) \tag{4}$$

Below is the distance measure (Euclidean Distance) formula where R_i is the range of the ith Component:

$$d = \sqrt{\sum_{i=1}^{n} \frac{(a_i - b_i)^2}{R_i^2}} = \sqrt{\frac{(a_1 - b_1)^2}{R_1^2} + \frac{(a_2 - b_2)^2}{R_2^2} + \ldots + \frac{(a_n - b_n)^2}{R_n^2}} \tag{5}$$

2.3 Decision Tree

Decision tree algorithms create a tree-like structure to classify instances based on hierarchical decisions. Each internal node represents a decision based on a specific feature, while each leaf node represents a class label. Decision trees are intuitive, interpretable, and can handle categorical and continuous features. However, they are prone to overfitting and may need help with complex relationships [10]. Decision trees partition the feature space based on a series of hierarchical decisions. The classification process involves traversing the tree from the root to a leaf node based on the feature values. The decision tree uses the ID3 (extension of D3) algorithm. ID3 follow a rule if entropy is zero branch is the leaf node, and if it is greater than zero, it needs further splitting. Entropy avails the probability of a definite outcome to decide how the node should branch. Entropy is mathematically intensive due to the log functions used in it.

$$\text{Entropy} = \sum_{i=1}^{C} -p_i * log_2(p_i) \tag{6}$$

where

p_i represents relative frequency and c represents number of classes.

Information gain is another attribute used to measure how well a attribute divides the training example according to target classification. Developing a decision tree is all about identifying an attribute that returns high information gain and low entropy.

$$\text{IG (T/X)} = \text{Entropy(T)} - \text{Entropy (T,X)} \tag{7}$$

Gini index is used which is the formula used to decide how nodes on a Decision tree branch.

$$\text{Gini} = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{8}$$

where

p_i represents relative frequency and c represents number of classes.

2.4 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees. Each tree is trained on a random subset of data and features. The final classification is determined by aggregating the predictions of individual trees. Random Forest mitigates overfitting and improves generalization [11]. It is robust to noisy data and capable of handling high-dimensional feature spaces. However, it

may be computationally expensive. When using RF to solve regression issues, it uses mean squared error, which calculates each node distance from the actual value. It helps in deciding which branch is the efficient decision for the forest.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2 \tag{9}$$

where

N = datapoints

f_i = Value returned by model

y_i = Actual value of Data point

When RF performed on classification data Gini index is used which is the formula used to decide how nodes on a Decision tree branch.

$$\text{Gini} = 1 - \sum_{i=1}^{C}(p_i)^2 \tag{10}$$

where

p_i represents relative frequency and c represents number of classes.

Entropy avails the probability of definite outcome in order to make decision on how the node should branch. Entropy is mathematically intensive due to log functions used in it.

$$\text{Entropy} = \sum_{i=1}^{C} -p_i * log_2(p_i) \tag{11}$$

2.5 Support Vector Classifier

SVC is a binary classification model that separates instances using a hyperplane with the maximum margin. It maps the data to a higher-dimensional space and finds an optimal hyperplane that maximizes the distance between the two classes. SVC effectively handles high-dimensional data and can capture complex relationships through kernel functions [12]. However, it may be sensitive to the choice of hyperparameters and may suffer from long training times for large datasets.

The SVC finds the optimal hyperplane that maximizes the margin between the two classes. The decision function for classification is given by:

$$\text{f(x)} = \text{sgn}(w^T\text{x} + \text{b}) \tag{12}$$

where

w = $\sum\alpha$i $y_i x_i$, αi is zero for all cases and $y_i \in \{1,-1\}$ are the labels.

2.6 Linear Support Vector Classifier

Linear SVC is a variant of SVC that utilizes a linear kernel function. It is particularly suitable for linearly separable data. Linear SVC is computationally efficient, scales well to large datasets, and generalizes well [13]. However, it may need help with nonlinearly separable data. Linear SVC is generally faster to converge large datasets when compared to support vector classifiers. Linear SVC reduces squared hinge loss, whereas SVC reduces regular hinge loss. LSVC uses one *vs*. rest multiclass reduction, while the SVC classifier uses one *vs*. one.

2.7 Naïve Bayes

Naïve Bayes is a probabilistic classifier based on Bayes' theorem and feature independence assumption. It calculates the posterior probability of a class label given the input features. Naïve Bayes classifiers are computationally efficient, require minimal training data, and can handle high-dimensional feature spaces [14]. However, they may oversimplify relationships among features due to the independence assumption.

Previous research in phishing website detection has employed these models and methods to varying degrees of success. For example, Machan utilized logistic regression to detect phishing websites based on URL features [15]. KNN and decision tree algorithms are applied to analyze website characteristics for detection [16]. Random Forest can identify phishing websites using website content and URL features. The literature review highlights the utilization of various machine learning models and methods for phishing website detection. Each model/method has strengths, limitations, and relevant research supporting its application. Bayes model is simple to build and useful for large datasets. It is known to outperform sophistication classification methods, and Bayes theorem provides a way of computing posterior probability p(c/x) from p(c), p(x/c) and p(x).

$$p\left(\frac{c}{X}\right) = \frac{\mathrm{p}\left(\frac{\mathrm{x}}{\mathrm{c}}\right)\mathrm{p}\,(\mathrm{c})}{2p\,(x)}$$

Here P(C) and P(X) are prior probability of class and predictor, respectively.

P(x/c) is Probability of predictor given class.

P(c/x) is posterior probability of class C given predictor (x, attribute).

3 Architecture of ML Models Testing and Methodology

In this research model implementation, the first stage is to gather a dataset containing an adequate amount of phishing and legitimate website URLs along with the tagging. The phishing site prediction dataset is taken from the Kaggle website and contains over half a million entries. The dataset contains two attributes one attribute contains a website URL, and the second attribute contains a label of the URL. The label column has 2 categories Fake and Legitimate. A fake website URL is a phishing site containing malicious stuff, whereas a Legitimate website URL is not a phishing website and will not contain any malicious content. The dataset contains around 150000 fake website URL's and 360000 Legitimate website URLs. Fig. 1 shows the ML model implementation which helps to analyze deviation in Accuracy, precision and recall along with different training percentage data.

3.1 Data Pre-Processing

At this stage, testing and analyzing the dataset will happen, and it will check for any imbalance in the data. During this phase, it will check whether duplicate and null values are in the datasets. If any unbalance in the dataset is found, it is better to clean it during this phase. During the preprocessing stage, shuffling the dataset will take place, and only a certain amount of data will be chosen. The sample data set should be in the same distribution as the entire dataset for better results. If necessary, the dataset can be updated with labels, further dividing it into labels and features.

3.2 Split Data into Train & Test Datasets

During this phase, data is split into train and test datasets. The training dataset is used to train ML models, and the test datasets will be used to test the trained data models. The train and test datasets

will be the subsets of the main dataset. The test dataset should be large enough to provide statistically valuable results and represent the dataset as a whole. The model should be trained with train data and avoid test data to provide accuracy. The training set should also be significantly large as ML models are required to be trained by large datasets.

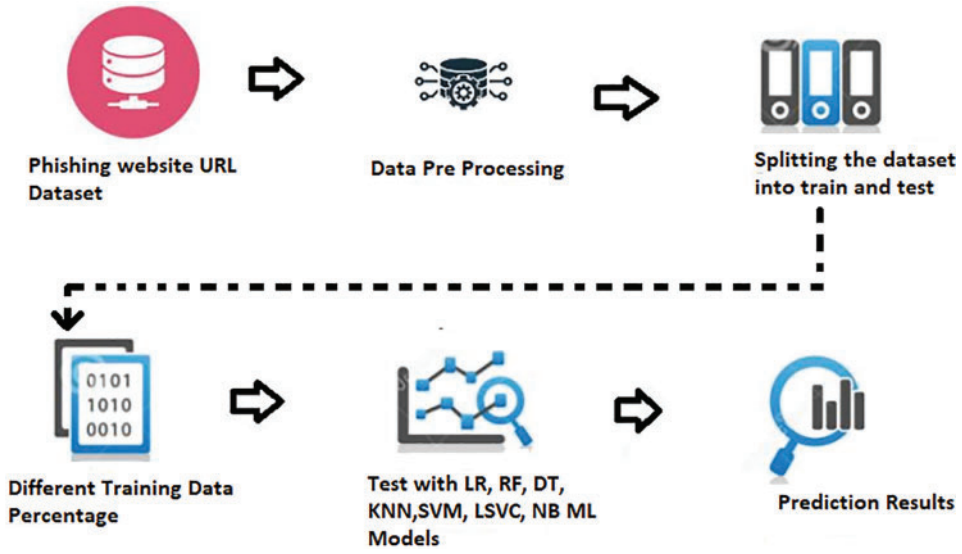

Figure 1: Implementation of ML model testing

3.3 Import and Initialize Models

At this state, we must import and initialize ML models like Logistic Regression, Random Forest, Decision tree, KNN, SVM, Linear SVC, and Naïve Bayes. All the imports will be done using the sklearn package. Below is the sample code for importing ML models from the sklearn package:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import Logistic Regression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier.
```

3.4 Train ML Model with Train Data

After importing the ML models, we need to train the models using different training percentages. While training the model, we must ensure the training dataset is significantly large and different from the testing dataset. The training dataset should also be a subset of the main dataset, and the larger the dataset, the better prediction the ML models will have. Training datasets should always cover all the different and unique types of phishing website URL's.

3.5 Test ML Models with Test Data and Using Different Training Percentage

This is the final stage of the experiment, where testing is done using the test dataset and different training percentages. The test results will contain the ML models' accuracy, precision and recall. We can even generate a confusion matrix at this stage using Python code. At this stage, evaluating the ml models' performance will ensure the ML models work as expected to achieve the highest quality results. Testing ML models requires a huge volume of data and long training cycles. The test dataset is used to validate the ML model's performance. An ML model is considered a working model if it passes the test. If the test results are not as expected, they must be sent back to training, where they will be trained using different datasets containing more phishing site URLs [17]. This stage will be helpful to the organization as it will help in picking up suitable ML models, as phishing is a severe problem for the employees as well as the organization. Fig. 2 represents a confusion matrix at this stage and evaluate the ML models' accuracy, precision and recall.

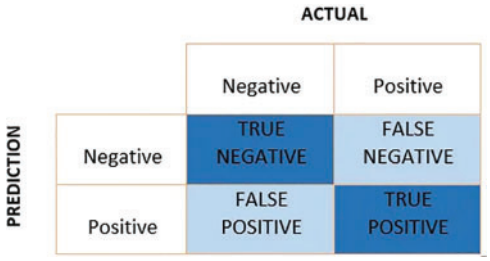

Figure 2: Confusion matrix for machine learning models

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ positive + \ False\ Positive + True\ Negative + \ False\ Negative} \tag{13}$$

$$Precision = \frac{True\ Positive}{True\ positive + \ False\ Positive} \tag{14}$$

$$Recall = \frac{True\ Positive}{True\ positive + \ False\ Negative} \tag{15}$$

3.6 Methodology

Upon realizing the importance of feature engineering, we extracted a combination of structural, lexical, and aggregated features from URLs and evaluated prediction models using combination of qualitative and quantitative methodology. The dataset underwent a preprocessing stage, eliminating any redundant or missing values. Next, using stratified sampling, we divided the data into training and test sets, ensuring that both sets have approximately the same percentage of samples of each target class as the complete set.

To optimize model performance, hyperparameter tuning was conducted for each model. For logistic regression, parameters like regularization strength and the type of regularization were tuned. For KNN, the number of neighbors and the distance metric was optimized. For decision trees, the depth of the tree, minimum samples to split, and the criterion (Gini or Entropy) were adjusted. The models were then evaluated on the test set using accuracy, precision, recall, F1 score, and the Area Under the Curve (AUC) to provide a holistic view of their performance. Post-training, an ensemble approach was explored to combine the strengths of the individual models, aiming to enhance the overall prediction accuracy and reduce the likelihood of overfitting. As mentioned above, the

methodologies provide insights into the models' ability to classify phishing websites and lay the groundwork for practical implementations in real-world cybersecurity frameworks.

4 Data Analysis and Experimental Results

Phishing URL detection refers to identifying and classifying URLs designed to deceive users and steal their sensitive information, such as login credentials, financial data, or personal details. Phishing attacks have become increasingly sophisticated and prevalent in today's digital landscape, posing a significant threat to individuals, businesses, and organizations. Detecting phishing URLs is crucial in preventing users from victimizing these malicious schemes. Various techniques and approaches are employed to analyze the characteristics of URLs and determine their legitimacy or malicious intent. These techniques often leverage machine learning algorithms, natural language processing, and behavioral analysis to identify patterns, anomalies, and indicators of phishing activity [18]. Effective phishing URL detection helps protect users from potential cyber threats, safeguard their privacy and security, and preserve the integrity of online platforms and services. Fig. 3 represents the dataset distribution of web URLs containing legitimate and malicious URL's.

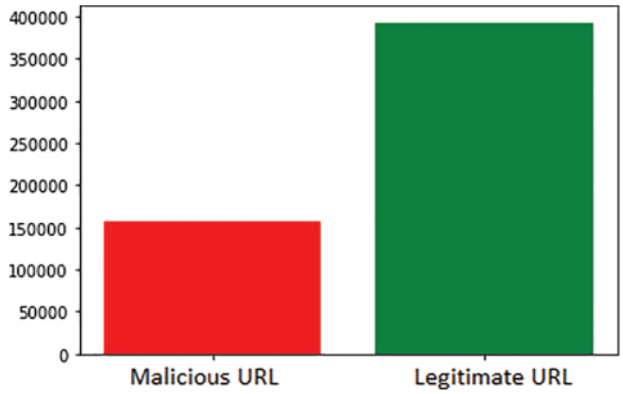

Figure 3: Dataset distribution of legitimate URL and malicious URL

This visualization is an essential tool for data analysis in phishing website detection. Researchers and cybersecurity experts can develop effective algorithms and techniques to identify and flag potential phishing websites by analyzing the common words and their frequencies in both URLs [19]. This data-driven approach enables the development of robust systems to protect users from falling victim to online scams and cyber threats [20].

Fig. 4 illustrates a cloud of words used in legitimate and fake website URLs, with the respective word count represented by colour bars. The red bar represents the number of words in malicious URLs, which amounts to 150,000, while the green bar represents the word count in legitimate URLs, which stands at 400,000.

The figure presents the domains of both fake and genuine websites. The domains associated with legitimate websites typically include terms such as "tryout," "media," "edu," "eira," "ye," "youthleaguesusa," "fmesxcc," "nvinip," "salesecureweb," "index," "html," "end," and others. On the other hand, the domains of fake websites often consist of words like "paypal," "skype," "bin," "login," "cd," "org," "webscrc," "widescreen," "mail," "nobel," "icloud," and so on.

Analyzing the domains used in both types of websites can assist in the development of effective phishing website detection techniques [21]. By identifying patterns and common characteristics in

these domains, cybersecurity experts can enhance their ability to differentiate between genuine and fraudulent websites, enhancing overall online security.

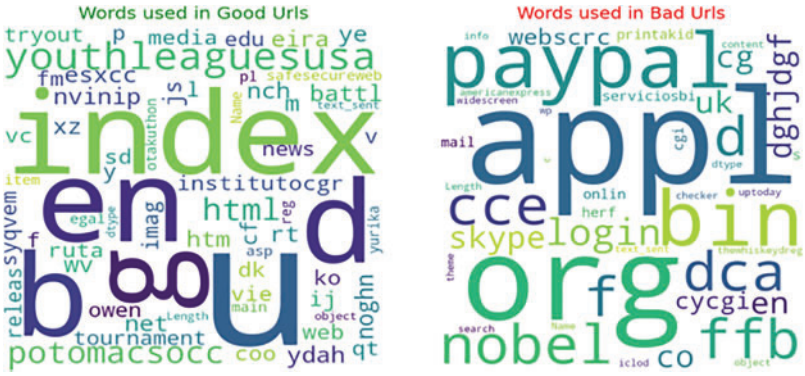

Figure 4: Word cloud of good website and bad website URL's

The top 10 good domains are represented in a bar graph, with the names of the domains on the x-axis and the corresponding count on the y-axis in Fig. 5. The most common good domain is "com" with a count of 290,000, followed by "org" with 45,000, "net" with 15,000, "edu" with 10,000, "ca" with 8,000, "co. uk" with 4,000, "gov" with 1,000, "de," "info," and "com.au" with an equal count. This visualization provides insights into the prevalence of different domain extensions in legitimate websites, aiding in analyzing and detecting good URLs.

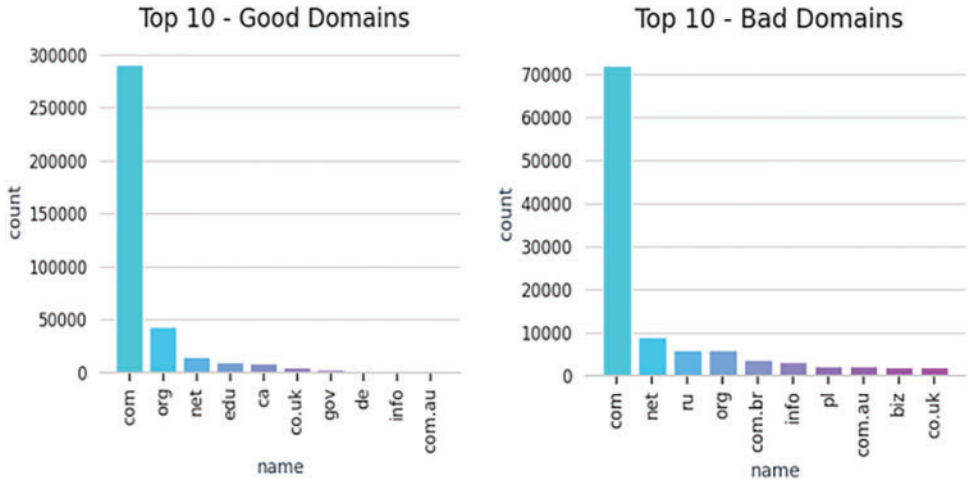

Figure 5: Top 10 good website domains and bad website domains

Similarly, the top 10 bad domains are depicted in a bar graph, showcasing the names of the domains on the x-axis and their respective count on the y-axis. The most common bad domain is "com" with a count of 72,000, followed by "net" with 9,000, "ru" with 6,000, "org" with 5,800, "com.br" with 4,000, "info" with 3,500 and "pl," "com.au," "biz," and "co.uk" each having a count of 2,000. This visualization assists in understanding the prevalence of these malicious domains, allowing for the development of effective countermeasures and phishing detection algorithms.

Table 1 presents the accuracy of various machine learning models at different training percentages. The models included in the table are Logistic Regression, Random Forest, Decision Tree, K-Nearest

Neighbors (KNN), Support Vector Machine (SVM), Linear Support Vector Classifier (LSVC), and Naive Bayes (NB). Each row represents a different training percentage ranging from 10% to 90%, while each column corresponds to a specific machine learning model.

Table 1: Accuracies of different machine learning models

Training %	Logistic regression	Random forest	Decision tree	KNN	SVM	LSVC	NB
10	0.83	0.83	0.82	0.74	0.78	0.87	0.86
20	0.86	0.87	0.85	0.73	0.83	0.88	0.88
30	0.88	0.88	0.85	0.73	0.85	0.90	0.88
40	0.88	0.88	0.85	0.74	0.86	0.90	0.89
50	0.89	0.89	0.86	0.75	0.86	0.91	0.90
60	0.90	0.89	0.87	0.75	0.87	0.91	0.90
70	0.90	0.90	0.87	0.76	0.88	0.91	0.91
80	0.90	0.90	0.87	0.74	0.88	0.91	0.91
90	0.90	0.89	0.88	0.74	0.87	0.90	0.90

The accuracy values show the performance of the models at each training percentage. As the training percentage increases, most models demonstrate an improvement in accuracy. Most models achieve relatively high accuracies at the highest training percentage of 90%. Logistic Regression, Random Forest, Decision Tree, KNN, and SVM consistently perform well across the different training percentages. However, LSVC and NB exhibit slightly lower accuracies than the other models. It is important to note that accuracy is just one metric to evaluate model performance, and other metrics such as precision, recall, and F1 score should also be considered depending on the specific task and dataset.

Table 2 presents the precisions of different machine learning models at varying training percentages. The models included in the table are Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Linear Support Vector Classifier (LSVC), and Naive Bayes (NB). Each row represents a different training percentage ranging from 10% to 90%, while each column corresponds to a specific machine learning model. The precision values indicate the ability of the models to classify positive instances correctly. Similar to the previous table, as the training percentage increases, most models show precision improvements. At the highest training percentage of 90%, Logistic Regression, Random Forest, KNN, SVM, and LSVC consistently achieve high precision scores. Decision Tree and NB also demonstrate relatively high precisions, although slightly lower than the other models. It is worth noting that precision alone may not provide a comprehensive evaluation of model performance, and other metrics like recall, F1 score, and accuracy should be considered depending on the specific context and requirements of the task.

Table 3 displays the recalls of different machine-learning models at various training percentages. The models included in the table are Logistic Regression, Random Forest, Decision Tree, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Linear Support Vector Machine (LSVM), and Naive Bayes (NB). Each row corresponds to a different training percentage ranging from 10% to 90%, while each column represents a specific machine learning model.

Table 2: Precisions of different machine learning models

Training %	Logistic regression	Random forest	Decision tree	KNN	SVM	LSVC	NB
10	0.91	0.89	0.78	0.56	0.95	0.91	0.96
20	0.91	0.91	0.84	0.53	0.92	0.89	0.94
30	0.92	0.93	0.82	0.52	0.92	0.90	0.93
40	0.90	0.92	0.81	0.54	0.89	0.89	0.91
50	0.89	0.94	0.81	0.56	0.89	0.90	0.92
60	0.92	0.93	0.82	0.56	0.94	0.91	0.92
70	0.93	0.93	0.83	0.59	0.94	0.91	0.90
80	0.92	0.93	0.82	0.55	0.93	0.90	0.89
90	0.93	0.94	0.85	0.54	0.92	0.92	0.89

Table 3: Recalls of different machine learning models

Training %	Logistic regression	Random forest	Decision tree	KNN	SVM	LSVM	NB
10	0.45	0.49	0.56	0.46	0.26	0.62	0.52
20	0.58	0.60	0.60	0.53	0.45	0.67	0.60
30	0.64	0.64	0.61	0.59	0.51	0.70	0.65
40	0.68	0.65	0.67	0.57	0.57	0.75	0.70
50	0.71	0.70	0.68	0.59	0.60	0.76	0.72
60	0.72	0.68	0.69	0.60	0.59	0.75	0.73
70	0.74	0.70	0.70	0.60	0.63	0.78	0.77
80	0.72	0.70	0.74	0.61	0.62	0.75	0.78
90	0.69	0.67	0.69	0.62	0.60	0.71	0.74

The recall values in the table indicate the models' ability to identify positive instances correctly. As the training percentage increases, most models exhibit recall improvements. At the highest training percentage of 90%, Logistic Regression, Random Forest, Decision Tree, KNN, and LSVM consistently achieve relatively high recall scores [22]. SVM also shows competitive performance in terms of recall, while NB demonstrates slightly lower recalls compared to the other models. It is important to note that recall alone may not provide a complete assessment of model performance, and other metrics such as precision, F1 score, and accuracy should be considered in conjunction with recall to evaluate the overall effectiveness of the models.

Fig. 6 shows the relationship between training percentage and accuracy for various machine learning algorithms. The x-axis represents the training percentage, while the y-axis represents the accuracy. Each algorithm, namely Logistic Regression, Random Forest, Decision Tree, KNN, SVM, LSVM, and NB is represented by a different colored line.

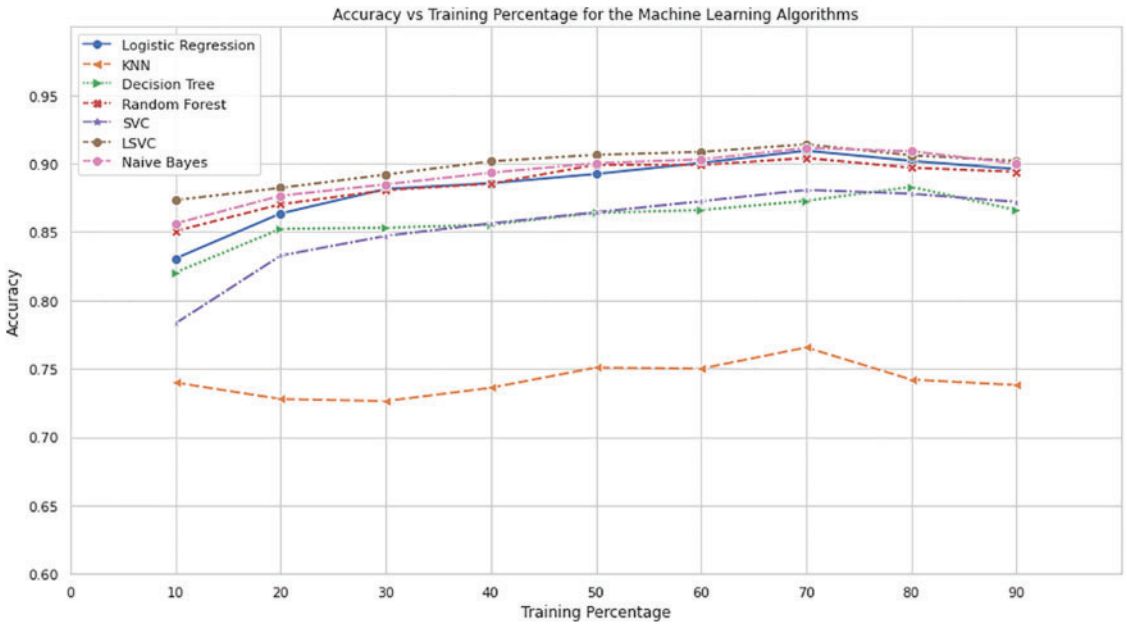

Figure 6: Accuracy *vs*. training percentage for the machine learning algorithm

From the plot, as the training percentage increases, the accuracy of all algorithms generally improves. However, the rate of improvement varies among the algorithms. Initially, the accuracy values for all algorithms are relatively low, but as the training percentage increases, the accuracy also increases.

LSVM and NB consistently achieve higher accuracy values across different training percentages among the algorithms. Logistic Regression, Random Forest, and KNN also significantly improve accuracy with increasing training percentage. SVM and Decision Tree algorithms perform moderately, while NB and LSVM consistently demonstrate higher accuracy.

Fig. 7 illustrates the relationship between training percentage and precision for different machine learning algorithms. The x-axis represents the training percentage, while the y-axis represents the precision. Each algorithm, including Logistic Regression, Random Forest, Decision Tree, KNN, SVM, LSVM, and NB, is depicted with a distinct color.

Analyzing the plot, it becomes evident that the precision values for the algorithms fluctuate across different training percentages. As the training percentage increases, there is a general improvement in precision for most of the algorithms. However, the rate of improvement and the overall precision values differ among the algorithms [23].

NB consistently achieves the highest precision values across various training percentages. LSVM also demonstrates a relatively high precision throughout the range. Logistic Regression, Random Forest, Decision Tree, KNN, and SVM exhibit varying precision values, with Random Forest and SVM showcasing relatively strong performance.

Fig. 8 illustrates the relationship between training percentage and recall for different machine learning algorithms. The x-axis represents the training percentage, while the y-axis represents the recall. Each algorithm, including Logistic Regression, Random Forest, Decision Tree, KNN, SVM, LSVM, and NB, represents a distinct color.

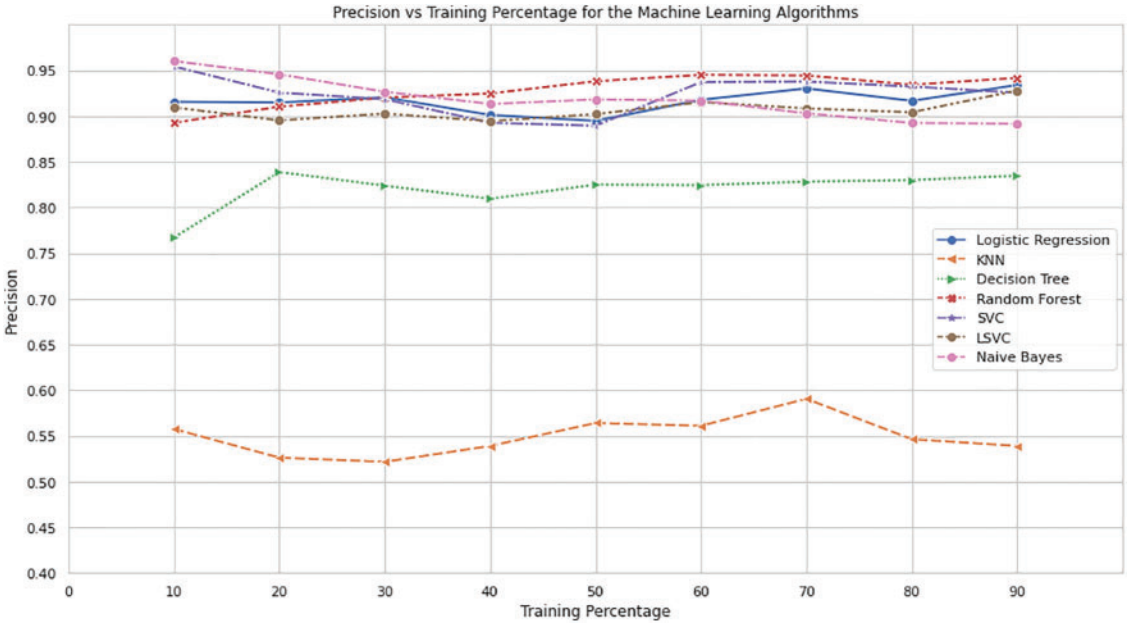

Figure 7: Precision *vs*. training percentage for the machine learning algorithm

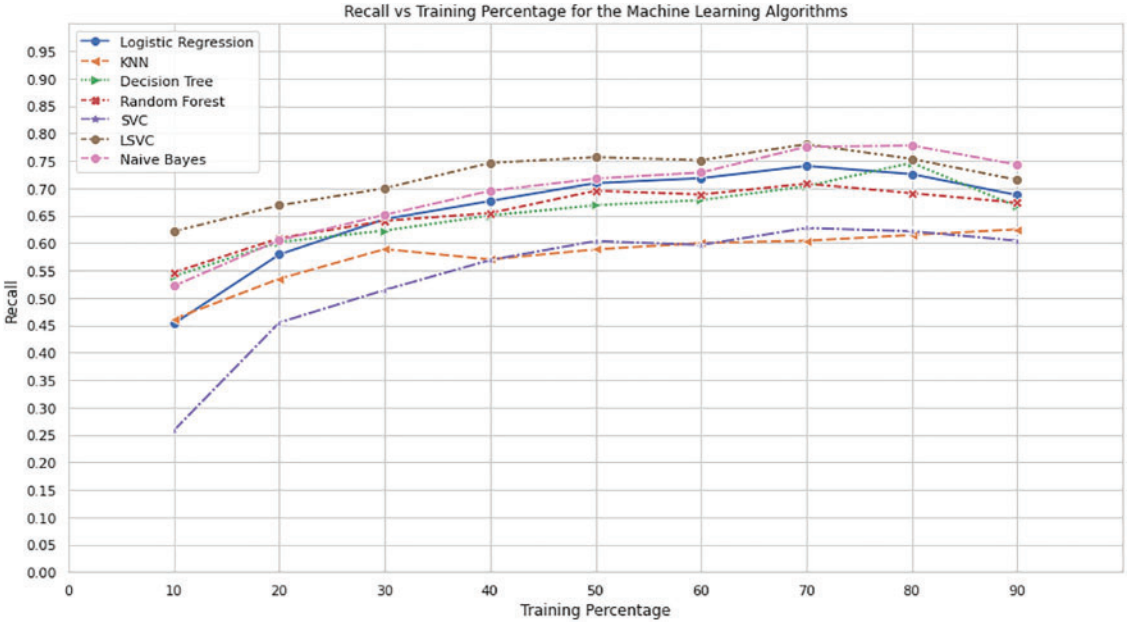

Figure 8: Recall *vs*. training percentage for the machine learning algorithm

Upon analyzing the plot, it becomes apparent that the recall values for the algorithms vary across different training percentages [24]. As the training percentage increases, there is generally an improvement in recall for most of the algorithms. However, the rate of improvement and the overall recall values differ among the algorithms.

LSVM consistently achieves the highest recall values across various training percentages, followed closely by NB. Logistic Regression, Random Forest, Decision Tree, KNN, and SVM exhibit varying recall values, with SVM and Decision Tree showing relatively strong performance.

5 Discussion

The surge in phishing attacks underscores the pressing need for rigorous data analysis techniques for efficient phishing website detection. As cyber attackers employ increasingly sophisticated methods to deceive users, the challenge for cybersecurity professionals is to stay one step ahead. This paper sought to address this challenge, emphasizing the harnessing of machine learning, natural language processing (NLP), and behavioral analysis to evaluate URL characteristics and thereby detect phishing attempts intricately. One crucial insight the paper introduced is the word cloud visualization representing common terminologies in legitimate and phishing URLs [25]. Such visualizations are invaluable; they demystify the structure of URLs and spotlight the lexical distinctions between genuine and deceptive websites. Furthermore, with the burgeoning developments in AI and NLP, platforms like Chatbots and ChatGPT can be harnessed for advanced threat detection mechanisms [26].

By delving into domain analyses of URLs, the research has spotlighted characteristic patterns and anomalies in domains that serve as markers for phishing attempts. The presented tables, elucidating the performance metrics of various machine learning models, serve as a tangible testament to the efficacy of data-driven approaches in phishing detection. From the tabulated data, a notable observation is that an increase in training data volume generally correlates with enhanced model performance. These insights not only contribute to the academic realm but also provide practical value for practitioners in the cybersecurity domain. The primary drive behind this research is the relentless proliferation of phishing attacks and the consequent imperative need for advanced detection mechanisms. The paper's main contribution is showcasing how machine learning algorithms, coupled with NLP and behavioral analysis, can offer an innovative and effective solution for distinguishing between genuine and fraudulent URLs. The machine learning models were trained and tested on a system with the following specifications:

Processor: Intel i7-9700K

RAM: 32 GB DDR4

Graphics Card: NVIDIA GeForce RTX 2070

Storage: 1 TB SSD

Software & Libraries: Python 3.8, sci-kit-learn 0.24, TensorFlow 2.4, NLTK 3.5

Development tool—Data lore JetBrains

Compute: 2vCPU core, 4 GB RAM (small cluster).

Moreover, while the paper places significant emphasis on machine learning and NLP techniques for URL analysis, it is vital to recognize the potential advantages of integrating these with other cybersecurity mechanisms. For instance, pairing the data-driven approaches with real-time threat intelligence platforms can create a synergized defense system that continuously evolves with emerging threats. Such integration would allow for more proactive identification of phishing attempts, potentially identifying malicious URLs before they even reach the user. The research also beckons a future exploration into the adaptability and resilience of the machine learning models in real-world conditions. While the current framework is robust in its present state, phishing strategies evolve over time. Thus, ongoing retraining and evaluation of the models using fresh data sets become paramount.

Furthermore, the incorporation of user feedback loops could assist in refining the models. By allowing end-users to report false positives or negatives, we can harness collective intelligence to enhance the system's accuracy. The dynamic nature of cybersecurity calls for adaptable solutions, and the marriage of machine learning, NLP, and user-centric feedback can pave the way for the next generation of anti-phishing strategies.

6 Conclusion

This paper presented various algorithms and machine-learning models for detecting phishing website URLs. The machine learning models experimented on in this paper are Logistic regression, KNN, Decision Tree, Random Forest, SVC, Linear SVC, and Naïve Bayes. All these Machine Learning models are tested with different training percentages, and results show that Linear SVC exhibited good accuracy. Random Forest ML models' precision significantly improved with the increase in training percentage. The Naïve Bayes exhibited a higher true positive rate followed by LSVC, Logistic Regression and other ML models. Logistics regression results are standard and adaptable while considering accuracy, precision, and recall results. KNN and Decision Tree ML Models have lower accuracy, precision, and true positive rates than all other ML models discussed in this paper. Due to the significant increase in the phishing website, these models must be studied, modified and experimented with new phishing trends and techniques.

The algorithms discussed in this paper help spot and avoid possible cybersecurity breaches because they can accurately differentiate between trustworthy and harmful websites. Cybersecurity experts may gain essential insights from developing efficient machine learning-based algorithms for phishing website identification, allowing them to proactively identify and neutralize possible risks. These methods may also be integrated into browsers, email filters, and security programs to provide users with real-time security. By supporting the deployment of machine learning-based detection techniques and increasing awareness of the hazards associated with phishing, organizations may collaboratively collaborate to create a more secure online environment. Various possible paths for future study might be explored. Using ensemble models, which include many machine learning methods, might improve the reliability and accuracy of detection. Incorporating more sophisticated techniques like machine learning and natural language processing may also result in discoveries and advancements in identifying phishing websites. A thorough knowledge of these models' advantages and disadvantages may be obtained by testing and contrasting them on various datasets and situations. This study emphasizes the value of machine learning methods for identifying phishing websites. The concepts and techniques covered in this article provide the groundwork for future developments in the area, opening the door for creative responses to phishing attacks and developing a safer digital environment.

Acknowledgement: We researchers would like to thank Microsoft for providing big data tools to conduct extensive research related to phishing website detection. We want to express our sincere appreciation and our deepest gratitude to Colorado Technical University faculty members for providing guidance in research and writing papers. We also thank the anonymous referee, reviewers, and editors for reviewing our paper. Finally, we sincerely thank the Journal of Artificial Intelligence Tech Science Press for allowing us to publish the paper.

Funding Statement: The authors received no external fundings for this study.

Author Contributions: Dinesh Kalla performed the analysis with the dataset, contributed Azure data engineering tools and conducted research using them. Sivaraju Kuraku helped in experimental design of the study, writing and revision of the paper.

Availability of Data and Materials: Data is openly available at Kaggle website and has open license Database Contents License (DbCL) v1.0 [26].

Conflicts of Interest: The authors declare that they have no conflict of interest to report regarding the present research.

References

[1] A. Mandadi, S. Boppana, V. Ravella and R. Kavitha, "Phishing website detection using machine learning," in *2022 IEEE 7th Int. Conf. for Convergence in Technology (I2CT)*, Mumbai, India, pp. 1–4, 2022. https://doi.org/10.1109/i2ct54291.2022.9824801

[2] S. Kuraku and D. Kalla, "Emotet malware—A banking credentials stealer," *IOSR Journal of Computer Engineering*, vol. 22, pp. 31–41, 2020.

[3] A. Kulkarni and L. L. Brown, "Phishing websites detection using machine learning," *International Journal of Advanced Computer Science and Applications*, vol. 10, 2019. https://doi.org/10.14569/ijacsa.2019.0100702

[4] D. Kalla and A. Chandrasekaran, "Heart disease prediction using machine learning and deep learning," *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, vol. 13, no. 3, 2023. https://doi.org/10.5121/ijdkp.2023.13301

[5] A. Safi and S. Singh, "A systematic literature review on phishing website detection techniques," *Journal of King Saud University—Computer and Information Sciences*, 2023. https://doi.org/10.1016/j.jksuci.2023.01.004

[6] S. Das Guptta, K. T. Shahriar, H. Alqahtani, D. Alsalman and I. H. Sarker, "Modeling hybrid feature-based phishing websites detection using machine learning techniques," *Annals of Data Science*, 2022. https://doi.org/10.1007/s40745-022-00379-8

[7] D. Kalla, F. Samaah, S. Kuraku and N. Smith, "Phishing detection implementation using databricks and artificial Intelligence," *International Journal of Computer Applications*, vol. 185, no. 11, pp. 1–11, 2023. https://doi.org/10.5120/ijca2023922764

[8] P. Gupta and A. Mahajan, "Phishing website detection and prevention based on logistic regression," *International Journal of Creative Research Thoughts*, vol. 10, pp. 2320–2882, 2022.

[9] T. A. Assegie, "K-nearest neighbor based URL identification model for phishing attack detection," *Indian Journal of Artificial Intelligence and Neural Networking*, vol. 1, no. 2, pp. 18–21, 2021. https://doi.org/10.54105/ijainn.b1019.041221

[10] D. Ahmed, K. Hussein, H. Abed and A. Abed, "Phishing websites detection model based on decision tree algorithm and best feature selection method," *Turkish Journal of Computer and Mathematics Education*, vol. 13, no. 1, pp. 100–107, 2022.

[11] G. Ramesh, R. Lokitha, R. Monisha and N. Neha, "Phishing detection system using random forest algorithm," *International Journal for Research Trends and Innovation*, vol. 8, pp. 510, 2023.

[12] D. Aksu, A. Abdulwakil and M. A. Aydin, "Detecting phishing websites using support vector machine algorithm," *Pressacademia*, vol. 5, no. 1, pp. 139–142, 2017. https://doi.org/10.17261/pressacademia.2017.582

[13] V. Jakkula, "Tutorial on support vector machine (SVM)," 2011. [Online]. Available: https://course.ccs.neu.edu/cs5100f11/resources/jakkula.pdf (accessed on 15/04/2023)

[14] G. Kamal and M. Manna, "Detection of phishing websites using Naïve bayes algorithms," *International Journal of Recent Research and Review*, vol. XI, no. 4, pp. 34–38, 2018.

[15] F. Mbachan, "Phishing URL prediction using logistic regression," 2022. https://doi.org/10.13140/RG.2.2.11606.93767

[16] H. Rajaguru and S. R. Sannasi Chakravarthy, "Analysis of decision tree and K-nearest neighbor algorithm in the classification of breast cancer," *Asian Pacific Journal of Cancer Prevention*, vol. 20, no. 12, pp. 3777–3781, 2019. https://doi.org/10.31557/APJCP.2019.20.12.3777
[17] S. Hutchinson, Z. Zhang and Q. Liu, "Detecting phishing websites with random forest," *Machine Learning and Intelligent Communications*, pp. 470–479, 2018. https://doi.org/10.1007/978-3-030-00557-3_46
[18] M. A. Adebowale, K. T. Lwin and M. A. Hossain, "Intelligent phishing detection scheme using deep learning algorithms," *Journal of Enterprise Information Management*, vol. 36, no. 3, pp. 747–766, 2023. https://doi.org/10.1108/JEIM-01-2020-0036
[19] A. A. Orunsolu, A. S. Sodiya and A. T. Akinwale, "A predictive model for phishing detection," *Journal of King Saud University—Computer and Information Sciences*, vol. 34, no. 2, pp. 232–247, 2022. https://doi.org/10.1016/j.jksuci.2019.12.005
[20] M. Shoaib and M. S. Umar, "URL based phishing detection using machine learning," in *2023 6th Int. Conf. on Information Systems and Computer Networks (ISCON)*, Mathura, India, pp. 1–7, 2023. https://doi.org/10.1109/ISCON57294.2023.10112184
[21] S. Alnemari and M. Alshammari, "Detecting phishing domains using machine learning," *Applied Sciences*, vol. 13, no. 8, pp. 4649, 2023. https://doi.org/10.3390/app13084649
[22] J. Bharadiya, "Machine learning in cybersecurity: Techniques and challenges," *European Journal of Technology*, vol. 7, no. 2, pp. 1–14, 2023. https://doi.org/10.47672/ejt.1486
[23] P. Dhanavanthini and S. S. Chakkravarthy, "Phish-armour: Phishing detection using deep recurrent neural networks," *Soft Computing*, pp. 1–13, 2023. https://doi.org/10.1007/s00500-023-07962-y
[24] P. P. Kumar, T. Jaya and V. Rajendran, "SI-BBA-A novel phishing website detection based on swarm intelligence with deep learning," *Materials Today: Proceedings*, vol. 80, pp. 3129–3139, 2023. https://doi.org/10.1016/j.matpr.2021.07.178
[25] T. Tarun, "Malicious and phishing site URL's, database contents license (DbCL) v1.0," 2020. [Online]. Available: https://www.kaggle.com/datasets/taruntiwarihp/phishing-site-urls/data (accessed on 18/03/2023)
[26] D. Kalla and N. Smith, "Study and analysis of chat GPT and its impact on different fields of study," *International Journal of Innovative Science and Research Technology*, vol. 8, no. 3, pp. 827–833, 2023.