Journal of Circuits, Systems, and Computers
Vol. 33, No. 2 (2024) 2450031 (44 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218126624500312

A Feature Extraction Approach for the Detection of Phishing Websites Using Machine Learning*

Sri Charan Gundla[†], M. Praveen Karthik[‡],
Middi Jashwanth Kumar Reddy[§], Gourav[¶] and Ashutosh Pankaj[‖]

School of Computer Science and Engineering,
Vellore Institute of Technology, Katpadi,
Vellore 632014, Tamilnadu, India
[†]*gundlasricharan@gmail.com*
[‡]*praveenkarthik.m2020@vitstudent.ac.in*
[§]*middijashwanth.kumar2020@vitstudent.ac.in*
[¶]*gourav.2020@vitstudent.ac.in*
[‖]*ashutoshkumar.pankaj2020@vitstudent.ac.in*

Z. Stamenkovic

University of Potsdam,
Institute for Computational Science,
An der Bahn 2, Potsdam 14476, Germany

The IHP-Leibniz-Institut für innovative Mikroelektronik,
Im Technologiepark 25, Frankfurt (Oder) 15236, Germany
stamenkovic@uni-potsdam.de
stamenko@ihp-microelectronics.com

S. P. Raja

School of Computer Science and Engineering,
Vellore Institute of Technology, Katpadi,
Vellore 632014, Tamilnadu, India
avemariaraja@gmail.com

Received 11 April 2023
Accepted 17 June 2023
Published 3 August 2023

In this growing world of the internet, most of our daily routine tasks are somehow connected to the internet, from smartphones to internet of things (IoT) devices to cloud networks. Internet users are growing rapidly, and the internet is accessible to everyone from anywhere. Data phishing is a cyber security attack that uses deception to trick internet users to get their content and information. In this attack, malicious users try to steal personal data such as login credentials, credit card details, health care information, etc.,

*This paper was recommended by Regional Editor Tongquan Wei.
[†]Corresponding author.

of the users on the internet. They exploit users' sensitive information using vulnerabilities. Information stealers are known as phishers. Phishers use different techniques for phishing. One of the most common methods is to direct the users to a false website to enter their login credentials and their details on these phishing sites. Phishing websites look like the original websites. Phishers use these details to get access to the user's accounts and hijack them for monetary purposes. Many internet users fall for this trap of phishing sites and share their personal and sensitive details. In this paper, we will analyze and implement machine learning (ML) techniques to detect phishing attacks. There are different methods to identify phishing attacks, one of them is by checking the uniform resource locator (URL) address using ML. ML is used to teach a machine to differentiate between phishing and original site URLs. There are many different techniques to overcome this attack. This research paper aims to provide accurate and true phishing detection with less time complexity.

Keywords: Phishing attacks; machine learning; classification; accuracy; detection; URLs; features.

1. Introduction

The Internet has become a basic need for everyone in this digital world. Every day, a large amount of data is exchanged through the internet. There are multiple use cases of the internet from e-commerce sites to net banking. Password-attacks, SQL injection attacks, phishing attacks, and malware attacks are the most common types of cyber-attacks on the internet.

Net banking is one of the most used services on the internet. In the previous year, there was a rapid increase in net banking usage in Europe by 51% and by 60% in the United States of America.[1] On the other hand, there was a 36% increase in data phishing attacks on the internet and a rise of 220% in phishing sites.[1] In the 1900s, America Online introduced the word "phishing". Every single day, thousands of new websites are being created and a lot of them have a login page where users enter their details. Most of these websites take the users' email addresses and password credentials as well as other details like their phone numbers. If the website is an e-commerce website, when a user tries to order items from it, they will enter their credit or debit card details along with the CVV number. Here the phisher gets all the sensitive information of the user and using this, the phisher can do anything he wants. It is impossible to keep an eye on all these websites because their lifespan is limited. If one of the phishing sites is put down by the government, the next moment a new site will be created by the phishers. Therefore, it is the user's responsibility to be careful while entering their credentials on any website.

The main objective of these phishing attacks is getting the users' login credentials and their sensitive data like credit card details and health care information. One of the most common and famous phishing attacks is done through email. In this attack, the phisher sends fake emails that make it look like the mail came from a reputable source. These emails contain a hyperlink. On clicking these links, the user will be navigated to fake websites, these websites look like the original reputable sites. Then the user fills in their login credentials and their sensitive information. Through these, the phishers get access to the user's profile and their sensitive data. This is how a phisher, through a simple hyperlink, gets all the details of an internet user.

Phishers mainly focus on employees working in multi-national companies like Facebook, Amazon, and Google and wait until an employee clicks the hyperlink. Phishing has a major role in economic losses. According to these companies, they already have lost \$100 million till 2017.[1] The Gmail phishing attack targeted 1 billion Gmail users on the internet in 2017.[1] This attack has made a major dent in the brand's image. Phishers are taking benefit of the digital marketing done by these brands. Users see many advertisements on different platforms like Facebook, YouTube, etc. Some of these advertisements are real but most of them are fake advertisements like getting an Apple iPhone 14 at 25,000 rupees, most users know that these advertisements are scams, but all users are not the same. Many users get trapped by these fake advertisements.

There is no single approach to detect all data phishing sites because the one who has made one fake website can also make many other fake websites. Machine learning (ML) is a smart and intellectual approach to handle these issues. In ML, we can use different features to distinguish between phishing uniform resource locators (URLs) and legitimate URLs.

There are many different techniques we have used in this research to handle this data phishing attack. The techniques used are decision tree, multilayer perceptron (MLP), XGBoost, random forest (RF), support vector machine (SVM), k-nearest neighbor (kNN), and Naïve Bayes (Gaussian NB). These techniques use different features to distinguish between phishing sites and legitimate sites.

A phishing attack happens due to a lack of security in the system. Phishers use loopholes present in the security system to attack the user. Based on previous research, these attacks are categorized into three categories: physical attacks, synthetic attacks, and semantic attacks. Data phishing comes under semantic attack because in this attack sensitive information and data of the user are targeted.

Stealing personal sensitive data is not the only goal of a phisher. For example, if a user has clicked a malicious link and the link was directed to download malware in the user's system, this malware phisher can transfer all the money of the user to his bank account.

According to statistical data from 2019, 15% of the internet users who were attacked successfully, have an increased chance that they will be attacked at least one more time in the same year. In the year 2017, one-third of data leaks were through data phishing.[6] The major concern is that phishing attacks are increasing exponentially and becoming more advanced. Therefore, to overcome this, it is important to create a phishing detection system.

According to the security threat report of 2019, in a set of 170 URLs, at least one of the URLs is a phishing URL. In a report from 2018, 64% of MNCs and organizations have experienced data phishing attacks.[7] Companies have lost 1% of their customers due to these phishing attacks.[7] Companies are creating a lot of software to blacklist these phishing attacks; Google has made Google safe-browsing. The Netcraft Anti-Phishing toolbar is another example.[22] But still, users are phished by phishing attacks. These blacklisting techniques are not a fail-proof way to handle this problem. For the phisher, if one website is blocked, the next moment, they will create another. Therefore, it is important to implement an approach to recognize phishing sites.

1.1. *Preliminary knowledge*

The goal of phishing is to gather a user's information. The attackers get access to private and sensitive user data for their financial gain. Phishing typically entails the transmission of spam emails and the creation of duplicate websites, and it affects numerous industries, including e-commerce, online business, banking, and digital marketing. Websites that are similar to the original. Most phishing websites employ the same user interface and URL as legitimate websites. Many strategies, including blacklists, algorithms, and others, have been advocated for recognizing phishing websites. Despite this, the number of victims is increasing exponentially due to insufficient security technologies. Phishing attacks are more likely to succeed on the internet due to their anonymity and lack of regulation.

Social networks have become a platform where people can meet virtually. Unfortunately, phishing attacks happen when people connect through social networks. Phishing is a criminal offense that puts a user's privacy at risk, can be used to spread malware, and often steals sensitive information. Phishing can be done in several ways, such as by sending fake instant messages or emails, by pretending to be an online bank, auction, or payment site, or by sending people to fake web pages that look like login pages for real sites.

In 2019, phishing attacks escalated considerably, as reported by the APWG, which detected a total of 180,768 phishing websites that year.[18] Furthermore, a proof point poll found that social media users were at a higher risk of falling victim to phishing attacks. Spoofing real identities, such as a respectable website, is used in a phishing attack on the web. If a malicious website can collect any user information, it may then direct the user to other fraudulent websites, where even more of their private data may be stolen. To accomplish this, replica websites are designed to look like the original so that their duplicate nature goes undetected.

Phishing attacks cause a lot of damage to the economy, intellectual property, and the security of the country. Phishing hurts businesses like online shopping and banking. There are several ways to protect users from phishing attacks, such as the heuristic, rule-based, and supervised ML approaches. Supervised ML algorithms are used a lot for classification, and out of all the ways to find phishing websites, they are the most popular.

1.2. *Literature review*

Phishing attacks can happen in various ways like emails, messages, and many more. Various techniques have been developed to stop and control the problem of phishing. The difference between normal messages or emails to the ones that are phished is the way of representation of the emails or messages. There are some features which can be used to find phished messages and one needs to know how to use the respective selected features based on the categorization of phished texts.

Ojewumi *et al.*[1] developed an ML-based phishing identification and detection system. This system's name is PhishNet, it is a Google Chrome extension. This system extracts a total of 14 features from the website URL. These websites are taken from PhishTank.[12] Algorithms used for phishing detection are SVM, kNN, and RFs. This system is developed through the following steps: data assembly, feature extraction, model structure and preparation, model evolution, rule uprooting, and development of PhishNet. In this research, RF has performed the best with 98.35% accuracy and with a 100% true positive rate, and the second is the kNN model with an accuracy of 93.39%. PhishNet is a good solution to avoid phishing attacks.

Moedjahedy *et al.*[2] used two public datasets for this experiment. The first dataset, which was released in 2018, was used to choose legal websites with Alexa and General Archive URLs, as well as phishing domains with PhishTank and all OpenPhish URLs. Webpages were collected during two different sessions for two years for building this dataset. Python programs and GNU Wget utility took the webpages from the two sessions. Datasets are used to download some related resources of the webpages such as images, and JavaScript along with the entire HTML texts to make sure about the appearance of the downloaded webpages in the browser. After the process of download is completed, the dataset is examined to find any broken web pages. If any such webpages are found among phishing or legal datasets, those webpages are eliminated from the dataset. After eliminating the broken webpages, screenshots of all web pages are taken for examination and the filtering process. The first set contains 48 features, and 10,000 data, where data is balanced between phishing and legitimate webpages. The second dataset was created in 2021, and it uses Alexa and Yandex to gather URLs for a nonphishing dataset by taking URLs from the top domain. Also, by removing duplicates and inactive URLs, the phishing data URLs are gathered from PhishTank and OpenPhish. This second dataset contains 87 features which consist of 11,430 data, and it is balanced between phishing and nonphishing. The dataset is used for testing and training purposes in the ratio of 70% and 30%. Eliminating unimportant features and selecting the features that will be used in testing are the objectives of feature selection. There are two ways to go about this process. The first approach entails ranking each of these traits using a variety of criteria, comparing the results, and choosing the best features. The next strategy is to choose an insignificant determination of qualities without examining performance degradation. Recursive feature elimination (RFE) was used to select the final 10 features for this study. In RFE, a model is built by removing some features, then the model is built back up with those features that are still present and modeling the model's correctness.

Mughaid *et al.*[3] categorized the methodology into different phases: dataset collection, dataset pre-processing, and usage of ML techniques. Models have been proposed to classify emails established on three datasets that have distinct features and all the models are built with completely different functions. The results show that newly proposed models are effective at identifying phishing emails better than the currently used detection techniques. Three datasets are used for this experiment. Using three datasets with various attributes was motivated by the fast evolution of phishing attack plans, which makes it more difficult to distinguish between phishing email attacks, classify phishing emails, and determine how features affect a model's ability to identify phishing emails. The second phase is the training phase. The classifier model, which incorporates an ML technique, uses a 0.70 dataset for training to categorize the data as real or phishing mail. After we complete both phases, the model will determine if a mail is legal, or spam based on the algorithm. To train and verify the effectiveness of detecting phishing emails using the grouped features, seven algorithms were used. The purpose for choosing all these different algorithms is the diverse preparation methodology that is used to find all the guidelines and the system of testing and learning. The seven algorithms are locally-deep SVM, logistic regression, SVM, boosted decision tree, averaged perception, neural network, and decision forests. The initial dataset utilized in this study had 5,25,754 instances, 8,351 of which are phishing emails, and 5,17,402 of which are genuine emails. The percentage of legitimate emails is nearly 98.5% and the dataset is imbalanced. This dataset contains 22 features. To verify the balance, a random sample was chosen which includes 8,351 legal and 8,400 phishing incidents. Then the data is distributed into 0.70 training and 0.30 testing. Another dataset is used with different features which consists of 10,000 instances with an equal number of phishing and legal emails. There are over 50 features in this dataset. This data is split into 70% training and 30% testing. The third dataset contains 2,500 valid emails and 500 emails that are spam. All the numbers and URLs were transformed into strings with the prefixes NUMBER and URL. When numbers and URLs have been converted, the data is divided into 70% train and 30% test groups.

Jain *et al.*[4] proposed an ML-based anti-phishing approach that only uses client-side features. This method analyzes the hyperlink information extracted from the website's source code to determine whether it is a legitimate website or not. The hyperlinks features were divided into 12 different categories. Various classification algorithms along with a dataset of 2,544 phishing and legitimate websites, this proposed phishing detection strategy was evaluated. According to the results, logistical repression outperformed other methods in phishing website identification. The suggested method has a high true positive rate of more than 98.4% and a low false positive rate of only 1.52%, making it very effective in identifying phishing websites. To improve the efficiency of phishing website detection, six new features have been suggested. These characteristics show the connection between a webpage's URL and its content. The websites are converted into a document object model (DOM) tree to extract the linking features. A web crawler is used to automatically collect the webpages' hyperlinks. During the extraction process, all the relative connections will be swapped out for their hierarchically referred-to absolute links. Each website's feature vector is produced following the feature extraction procedure. Training and testing datasets are built after features from phishing and nonphishing webpages are extracted. A binary classifier divides the websites into the phishing or legitimate category. The binary classifier properly recognizes the requested website when the crawler creates the feature values in response to a request for a new website.

Pavithra *et al.*[5] proposed an approach that involved three steps: preparation of the dataset, usage of ML techniques during various situations, and evaluation of performances. 80 features were selected for this analysis. 70% of the data among these 80 attributes is for training, and 30% is for testing. Three chosen ML classifiers helped execute the suggested model. Results were compared and reviewed to determine which method is more accurate. The calculation of the true-positive and false-positive rates was done. The dataset was collected and 80% of the data was used for training and 20% for the phase of testing. The algorithms are applied to the datasets and results were recorded and compared. Of all the algorithms, three algorithms produced the highest accuracy, and those algorithms are RF, Naïve Bayes, and SVM. The process of categorization is done and the malware and benign applications in the dataset were classified. After analyzing all the manifest.xml files, the malicious keywords can form a dataset.

Mehanovic *et al.*[6] utilized a phishing websites database for their research. The database carries 11,215 records and 21 features. Of these 11,215 samples, 1,185 samples have been deemed nonfraudulent and the remaining 10,030 of them are regarded as phishing websites. For the feature selection process, the Weka tool and its algorithm have been used. Some of the used algorithms are relief attribute evaluator, one R attribute evaluator, info gain ratio attribute evaluator, symmetric uncertainty attribute evaluator, and gain ratio attribute evaluator. The ranker search method is used for all these algorithms. ML is defined as the application of algorithms to build models that make predictions based on specified input data. Classification is one of the methods. The model is trained using the training set. The testing set is utilized to evaluate the effectiveness of the model. To perform the detection of phishing websites, they are using kNN, decision tree, and RF classifiers. Every attribute from the feature selection process was utilized for the testing process, and two test options were also used: percentage split and 10-fold cross-validation. In terms of percentages, the data was divided as follows: 66% for the training set and 34% for the testing set, and for the evaluation process, the accuracies were measured and the obtained accuracy for these algorithms was 100%. Since 100% accuracy was obtained for the algorithms, they checked the value of accuracy when the number of attributes is reduced. The following model phase included properties that are at a high position as selected by all feature selection methods used. Although it took much less time to create this model than the initial model, the accuracy remained 100%. Using percentage split, RF required 2.88 s to build the model and 3.05 s to build using 10-fold cross-validation in the first experiment. As for the second experiment, the time required was 0.02 and 0.16, 2.64 and 1.14 s were the times required for the first and second models respectively for kNN.

Gupta *et al.*[7] proposed a methodology for detecting phishing URLs using lexical features, which is based on ML in a real-time environment. The aim of this research is developing a detection system that will detect phishing URLs which is based on ML that helps users to verify the legitimateness and maliciousness of the URLs without taking a long time. A mechanism is developed when a user visits that URL it will take out the feature from the URL in the form of vectors. Once the vectors are extracted, they are prepared and added to several other ML algorithms to validate the URL. The objectives of this approach are reliable and real-time security, low response time, automation of phishing website detection, and scalability. The URL is divided into different partitions. The obtained lexical information from feature extraction is then charted to corresponding features and an instance of URL is prepared and it consists of all the data required to label the URL as phishing or legitimate. Once the validation is done and the URL is found to be legitimate, then access will be given to the client to use, or else the website will be blocked. This phishing detection approach was evaluated using four ML algorithms. They are RF, logistic regression, kNN, and SVM. Python has been used for this experiment and the reason behind this is that python offers many libraries for ML problems. For this experiment, they used Scikit-learn,[17] SciPy, Pandas, and Matplotlib. A dataset called ISCXURL-2016[13] from the information vault of Canada is used for the evaluation of this approach. This dataset contains spam records, legitimate URLs, phishing sites, and many more. 10,000 benign and 9,964 phishing URLs were extracted for this evaluation and the number of benign and phishing URLs are nearly the same. The pre-processing process is done once the dataset is sorted out. Scaling is another process that is necessary to keep the data scaled into an acceptable range. Scaling on information is done to maintain the change of the features in the similar reach. Once the processing of the dataset is done and it is standardized, the data is divided into a training set and a testing set. The training set gets 15,971 instances whereas 3,993 instances have gone to the testing set. The most accuracy was by RF with 99.7% and it produced a false positivity rate of just 0.53% which indicates the reliability of this approach. Logistic regression performed the worst among all the approaches with a false positivity rate of 3.67%.

Thabtah *et al.*[8] examined classifiers and their rules using ML. They proposed that website features are the key to constructing an automated anti-phishing system. The quantity of features linked with the website is large so the features will be pre-processed to pick the features which are most effective. The effectiveness of features is measured using different methods like information gain, correlation analysis, and others. After feature selection, the classification algorithms can be applied to the features, and compare the results to come up with the system. The ML algorithms for classification used in the paper are decision trees, probabilistic methods, rule-based classification, SVMs, neural networks, bagging, self-organization maps, instance-based learning, and others. Of all these algorithms, they focused on rule-based classification. For the experiment analysis, a phishing dataset was taken which consists of 11,000 websites, and 30 features are utilized. The websites for this experiment were gathered with the help of an online scripting tool that focuses its analysis on the rate of detection, the number of regulations applied, and the time duration in which models are constructed. After data is processed, an ML algorithm is used to derive the anti-phishing model called a classifier. The classifier derived will be added to any search engine and is utilized to predict the type of websites being surfed by the user. Associative classification is an approach to get classifiers using methods like association rule discovery. The approach they have taken depends on two thresholds called minimum confidence and user minimum support. The support of the features depends on the quantity of appearances made in the data training set and confidence is denoted as a measure of feature frequency that is included together with the target class. If any of the features are more frequently used than the minimum level of support, they will become common features. Classification based on associations (CBA) and multi-class classification based on association rules (MCAR) are two associative classification methods that were evaluated on a PhishTank dataset to find the potential in identifying phishing sites. A dataset with more than 1,000 instances and 27 features was used and CBA along with classifiers of 4 types was applied using the WEKA tool. The aim was to create a phishing detection tool for browsers that could be used as accurately as feasible. New classifiers were created and were compared to the traditional classifiers present with the help of another dataset which contained nearly 2,000 instances and 16 features to find the accuracy. The outcomes demonstrated that the newly developed classifiers performed more effectively than the already existing classifiers such as CBA and decision trees.

In their study, Zamir *et al.*[9] proposed using principal component analysis (PCA) with various ML algorithms, such as RF, neural network, bagging, SVM, Naïve Bayes, and kNN, to identify phishing websites. They used performance evaluation measures to assess the classifiers' efficacy in detecting phishing websites. However, the drawback of this approach is that it relies on RFE, a feature reduction technique that eliminates the weakest features from a dataset.

Dutta *et al.*[10] created a framework that used recurrent neural networks (RNNs) with long short-term memory (LSTM) to distinguish between malicious and trustworthy URLs. A directed acyclic graph can be converted into a pure feedforward neural network if an RNN has a finite input; otherwise, it is a directed cyclic graph that cannot be converted. LSTM solves RNNs' gradient problem. The performance of LSTM is enhanced using many gates. The heuristic and ML based approaches required characteristics or labels and both supervised and unsupervised learning techniques to predict the authenticity of an URL. Like the ML technique, proactive phishing URL detection analyzes URLs to determine if they are trustworthy or fraudulent. Phishing sites are identified using conventional techniques like blacklists and whitelists, but their effectiveness is decreasing due to the exponential growth of web domains. Phishers use a range of platforms to collect user information, including email, websites, instant messaging, forum postings, phone calls, and text messages.

Almseidin *et al.*[11] used Google's image search to find phony identities for their study's verification and ML techniques to find the right logo image. They considered the domain name as the logo's identity because of the unique connection between them. To distinguish between legitimate and fake web pages, they compared the domain name that Google returned with the one from the web page inquiry. To clarify features and create an accurate classifier model, the researchers applied five ML approaches, including SVM, Naive Bayes, decision trees, *k*NN, random trees, and RFs. They calculated the false positive (FP), true positive (TP), false negative (FN), and true negative (TN) values as well as the F1-measure and accuracy using SVM on the collected features, and they discovered that the accuracy was 96%. The researchers gathered a dataset of 200 URLs for valid and phishing web pages from Phish-Tank and Yahoo Directory; the learning dataset consisted of 600 phishing web pages and 400 legitimate web pages. They examined the remaining websites after training 100 genuine and 100 fraudulent web pages. The outcomes of the experiments demonstrated that the suggested method produced excellent detection accuracy and decreased detection time. While the performance of the MLP algorithm was found to have less accuracy and time taken compared to the J48 algorithm, hybrid techniques obtained good accuracy and efficiency rates.

Table 1 describes the accomplishments and contributions of existing works.

1.3. *Motivation and justification of the work*

1.3.1. *Motivation*

With a focus on the most popular feature selection techniques for addressing a variety of issues and improving the performance and efficacy of phishing datasets, this research intends to give a study of the current techniques for identifying phishing websites that make use of ML algorithms. To improve the usefulness of the data set and shorten the time required to create the models, we will also use feature selection for an existing new phishing data set. We will then evaluate various ML techniques to see which one is more effective.

This research paper is about using supervised ML[28–31] to find phishing websites. Stacking the best algorithms to improve classification accuracy. Measures of performance such as accuracy, F1-score, recall, and precision, are used to evaluate the performance of all classifiers.

Table 1. Accomplishments and contributions of existing works.

Reference	Simplicity	Automation	Scalability	Ease of use	Main contributions
Ref. 1	✓	✓	✓	✓	Google Chrome extension called PhishNet which provides 98.35% accuracy and 100% TP rate.
Ref. 2			✓		Used two different datasets with 48 and 87 URL based features respectively and used RFE to select the best 10 features.
Ref. 3	✓		✓		Used diverse preparation methodology to train and verify the effectiveness of detecting phishing emails using the grouped features. Showed the difference in ML model performance using balanced and imbalanced datasets.
Ref. 4		✓		✓	Used web crawler to extract hyperlinks and applied 12 categories of hyperlink features which yielded a TP rate of 98.4% and FP rate of 1.52%. The classifier is trained using feature vectors and can recognize new URLs.
Ref. 5	✓		✓	✓	Used 80 features for identification of phishing URLs and training the classifier. Found the best models to be RF, Naïve Bayes and SVM. Results were compared to find the best model.
Ref. 6	✓	✓	✓	✓	Utilized 21 features in the dataset. Used WEKA Tool for feature selection and used different attribute evaluator algorithms to achieve 100% accuracy. Followed two testing methods: percentage split and 10-fold cross-validation and created highly efficient models.
Ref. 7			✓		Used feature extraction algorithms to obtain lexical information from the URL and data scaling is done to bring it to an acceptable range. Different categories of features are utilized to predict whether the website is legitimate or not. Yielded high accuracy and efficiency.
Ref. 8		✓		✓	Constructed an automated anti-phishing system and features were selected using information gain, correlation analysis, etc. Used rule-based classification and associative classification.

(*Continued*)

Table 1. (*Continued*)

Reference	Simplicity	Automation	Scalability	Ease of use	Main contributions
Ref. 9	✓			✓	Utilized RFE for feature selection and proposed using PCA along with other classification algorithms to increase the effectiveness of ML models to identify phishing websites.
Ref. 10	✓		✓	✓	Created a framework to distinguish legitimate and phishing URLs using RNN and LSTM. LSTM performance is enhanced using many gates. Has proactive phishing URL detection but is dependent on whitelists and blacklists.
Ref. 11			✓	✓	Used the relationship between logo and domain name to distinguish between legitimate and phishing sites. Calculated the FP, TP, FN, TN, accuracy, and F1-Score using SVM to classify URLs. Found that this method was more accurate and reduced detection time.

1.3.2. *Justification*

Several methods[23–27] have been developed to stop phishing attempts and guarantee users' online security. It is difficult to detect and remove forged emails and URLs. Lexical analysis has been suggested as a proactive strategy to prevent phishing by spotting fraudulent URLs. An examination of obfuscation strategies has been done to lessen the consequences of dangerous URLs, and an effective feature set has been presented to classify them. For this procedure, the age of the domain of the URLs is necessary. Moreover, features and methods have been used to identify phishing websites by classifying data sets using labels. Numerous techniques have been devised to stop phishing scams and guarantee customer safety online. It can be difficult to recognize and remove forged emails and bogus URLs. Lexical analysis has been used to identify bad URLs as part of a proactive strategy to reduce phishing. The classification of dangerous URLs using an effective feature set has been suggested, and the consequences of obfuscation methods have been examined. For this strategy, the domain age of URLs is necessary. Furthermore, tools and methods have been applied to identify phishing websites utilizing classification methods using labeled data sets.

1.4. ***Contributions***

- The research uses important and precise features for more accuracy and less noise.

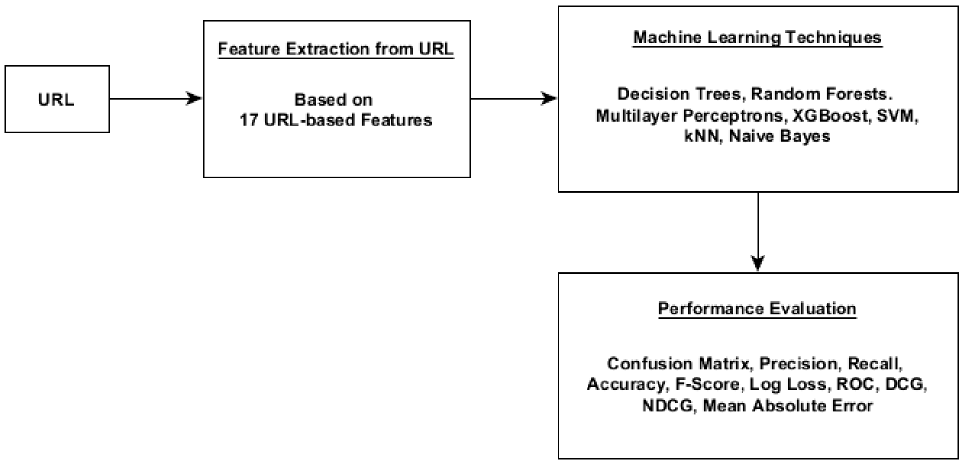

URL
Feature Extraction from URL
Based on
17 URL-based Features
Machine Learning Techniques
Decision Trees, Random Forests.
Multilayer Perceptrons, XGBoost, SVM,
kNN, Naive Bayes
Performance Evaluation
Confusion Matrix, Precision, Recall,
Accuracy, F-Score, Log Loss, ROC, DCG,
NDCG, Mean Absolute Error

Fig. 1. Outline of research.

- The ML model can detect new websites using the training from the dataset and does not rely on blacklisted website databases.
- The response time of detection of the websites is less.
- The proposed approach requires less computational power and fewer resources.
- The feature extraction algorithms use third-party records like WHOIS and DNS for constantly updated data. The model can be used without the internet if we ignore the WHOIS and DNS features.

1.5. *Outline of the research work*

Figure 1 presents the outline of the research work and the steps followed.

1.6. *Organization of the paper*

The rest of the research paper is organized as follows: Section 2 explains the methodology of the paper, Sec. 3 describes the experimental design of the proposed approach and the respective results, Sec. 4 presents the performance evaluation and comparison of the results, Sec. 5 concludes the research paper and finally, Sec. 5 proposes future works that will be explored.

2. Methodology

Our research follows the feature selection approach to detect phishing websites using a website's URL. A dataset of legitimate and phishing URLs is taken. We used 17 URL-based features to label each URL as either "0" = legitimate or "1" = phishing website. This labeled dataset is used to train different classification techniques. The trained models are evaluated using various performance metrics.

2.1. *Decision trees classifier*

A decision tree is an acyclic-directed graph, where each leaf node maintains a class label, each branch denotes an outcome of the test, and each internal node denotes a test on an attribute. Often employed in classification issues, a decision tree is a sort of learning algorithm which is supervised and has a pre-defined goal variable. Decision trees excel at solving categorization problems. Decision trees can handle both continuous input–output variables and categorical outcome variables. In this procedure, the most important splitter or differentiator among the input factors divides the sample into two or more homogenous groups. Categorical variable decision trees and continuous variable decision trees are the two varieties of decision trees, depending on the type of target variable.

A form of C4.5 decision tree algorithm called J48 is used for classification tasks; it makes use of a collection of training samples that are categorized. It builds the decision tree with the training dataset. The feature that efficiently splits each node's set of values into several new subgroups with the amount of information gain and thus identifies potential new nodes in the tree. The clarity with which decision trees depict, and the ability to consider the connections and interactions between the elements is one of its key properties.

Merits of the decision trees classifier are that it is versatile—capable of handling both categorical and numeric data variables with ease, nonparametric—has no assumptions concerning the spatial distribution or classifier structure, little requirement for data cleaning—it continues to be unaffected by the missing numbers and outliers, making the cleaning procedure simple, ease of use for data exploration—it is one of the quickest methods for identifying the relationship between two or more variables as well as the most important variable. Users may design new features and variables with the use of a decision tree. These new characteristics will be stronger at predicting the desired variable. It may be used throughout the data exploration phase as well. Demerits of this classifier are over fitting—which can be corrected with pruning or placing constraints on model parameters, and that it is not optimal for continuous variables—as information loss occurs during the

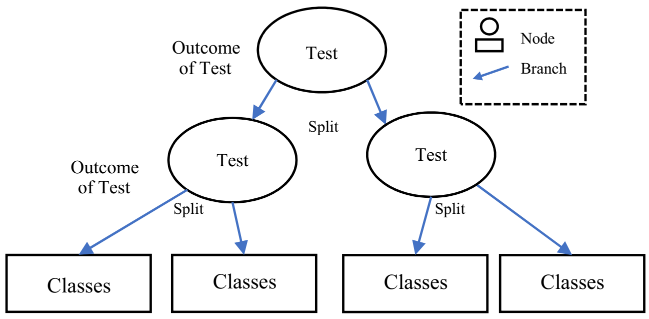


Fig. 2. A binary decision tree.

categorization process. Decision trees are extremely memory-efficient in comparison to other techniques. A decision tree can also be defined as a classifier that generates decision rules. To determine the best course of action for achieving a goal, operational research frequently uses decision trees, more especially decision analysis. Figure 2 shows the working of a decision tree.

The steps to build a decision tree and classify the test dataset using it are

```
(a) Define function BuildTree (node, depth):
    (i) If the stopping criteria are met (e.g., depth is equal to max_depth or the
        node is pure), then
        1. Mark the node as a leaf node
        2. Assign the majority class of the samples in the node as the predicted
           class
        3. Return
    (ii) Otherwise
        1. Calculate the best feature and split point that maximize the information
           gain or Gini impurity
        2. Split the node into two child nodes based on the split criteria
        3. For each child node:
           • Call the function BuildTree recursively with the child node and increment depth by 1.
(b) Initialize the root node of the decision tree.
(c) Call the function BuildTree with the root node and depth set to 0.
```

Prediction with the built model:

```
(a) For each test sample x in the testing dataset
    (i) Start from the root node of the decision tree
    (ii) Traverse the tree based on the split criteria until reaching a leaf node
    (iii) Assign the predicted class of the leaf node as the predicted class label for
          the test sample x
```

2.2. *RFs classifier*

A classification technique based on the decision tree algorithm is called RF. It is ideal for large datasets since it can contain many dataset variables; during the training phase, it creates a variety of decision trees. The fundamental idea behind this strategy is to mix several decision trees in order to arrive at the result rather than depending simply on one decision tree. Each tree is based on a set of predetermined traits that are chosen at random. The results from each tree are voted on by the majority to classify the data. A subset of the training data set is used to train RF. It uses a technique known as bootstrap and aggregation. Using the RF has the advantage of resolving the over-fitting issue that frequently arises when utilizing individual decision trees. Yet, because the process of creating the forest is *ad hoc*, there is no repeatability. RF can manage both regression and classification tasks.

Developing pseudo-samples is the initial step in building an RF, from which decision trees are eventually created. As a result, n elements are taken from an n-element set for each pseudo sample, and any object may be drawn with a chance of $1/n$. A decision tree is then constructed using each pseudo-sample as the basis. The process of choosing pseudo-samples is analogous to bagging. Every node in the decision tree is repeated as many times as there are items allocated to it. Hence, there is no trimming done when the trees are created. In contrast to the conventional way of creating that tree, RFs use characteristics at random when choosing the test for each node (individually). The split at that node will be generated taking those qualities into account. As a result, each segment works under a separate set of rules. Collection of characteristics so that regardless of other randomization, m characteristics (later considered during the division's selection) are drawn without return from all conceivable p characteristics of the learning set's objects. A significant issue is choosing the value for the m-parameter. The outcome for classification-related issues will be the decision tree that produced the most frequent result overall.

$$RFfi_i = \frac{\Sigma_{j \in \text{alltrees}} normfi_{ij}}{T}. \tag{1}$$

The average of all determined feature importance will be utilized to determine the feature importance for each of the decision trees

$$fi_i = \sum_{j:\text{nodes } j \text{ splits on feature } i} s_j C_j, \tag{2}$$

where RF is the RF function, while fi_i is the significance of a feature i, and sj is the samples that reach node j which belongs to class Cj one of T number of trees.

Merits of this technique are—both classification and regression may be done with it, the capability of handling big data sets with many dimensions, the ability to manage thousands of variables in addition to a few hundred, clearly distinguish the important ones from the others (which is why it is regarded as a crucial dimensionality reduction technique), can quickly and efficiently predict the missing data and even when fed a big amount of data, retain accuracy offers several techniques for balancing faults in the collection of data. The properties are also applicable to unlabeled data. It can thus work independently. It substitutes the existing data samples with fresh ones. This is referred to as bootstrap sampling. Various settings and values for the seeds can be altered to fine-tune the model. Reasons against its use are—regression performance is inferior to classification performance. It frequently fails to produce accurate, consistent predictions of nature. In the case of regression, it cannot generate predictions outside the range of the given training dataset. If the sample data are sufficiently noisy, the data may be over-fitted. Since it is not possible to accurately predict the model's performance, it may be used as a "black box" method for statistical modelers.

Common uses of the RF algorithm include forecasting market trends, market penetration, and making medical diagnoses. It has the potential to dramatically improve the effectiveness of corporate operations and scientific research when used for classification and regression tasks. Figure 3 gives a pictorial representation of the process via flowcharts.

The steps to build an RF classifier model are:

(a) For k within the range of maximum number of decision trees allowed

- (i) Randomly select a bootstrap sample $(X_{\text{train}}k, y_{\text{train}}k)$ from the training dataset
- (ii) Initialize a decision tree T_k with maximum depth
- (iii) For each node n in T_k
    1. Randomly select the required number of features
    2. Split the node n by finding the best feature and split point that maximizes the information gain or Gini impurity
    3. Create two child nodes based on the split and assign the samples that satisfy the split condition to the corresponding child nodes
    4. If the stopping criteria are not met (e.g., maximum depth not reached), repeat steps (i)–(iii) for each child node

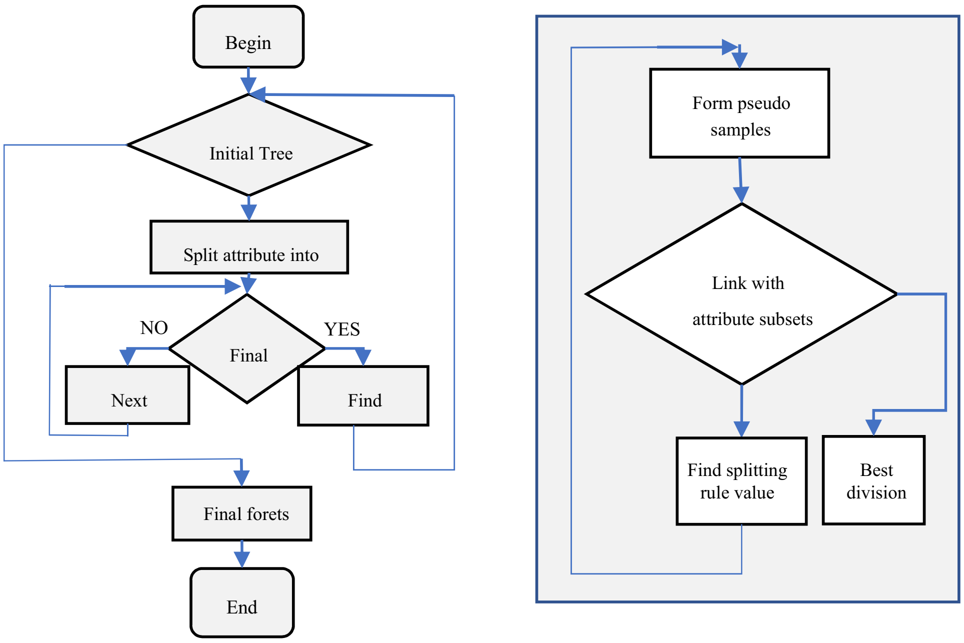


Fig. 3. RF construction.

```
            5. If the stopping criteria is met or the node is pure, mark the node as a
               leaf node and assign the majority class of the samples in the node as
               the predicted class
      (iv) Add the decision tree $T_k$ to the RF ensemble
(b) Prediction
      (i) For each test sample $x$ in the testing dataset
            1. Initialize an empty list for predictions
            2. For each decision tree $T_k$ in the RF ensemble
                  • Obtain the predicted class label from $T_k$ for the test sample $x$
                  • Append the predicted class label to predictions
            3. Return the predicted class labels in predictions
```

2.3. *MLPs*

The most common and widely used artificial neural network is an MLP. Similar to a neural network, MLP is made up of numerous interconnected parts. Input layer, hidden layer, and output layer are the three layers that make them up. A hidden layer acts as the MLP's computational engine, while the input layer is used to receive the signal. The output layer generates a conclusion about the input. It is often used to solve supervised learning issues. To do this, it is trained on a set of input–output pairs and taught how to correlate and depend on them.

All the neurons in the hidden layer are fully linked to all the neurons in the previous input layer and the neurons in the subsequent output layer because neural networks are coupled. To get the best outcomes, a neural network repeatedly adjusts the weights and biases of each layer as it learns. An MLP minimizes the loss function, such as cross-entropy, while training using the supervised learning method known as back propagation. It adjusts settings using an optimizer (weight and bias). An MLP differs from a linear perceptron by having many layers and nonlinear activation. A fundamental type of deep neural network is an MLP.

MLP differs from a linear perceptron because of its multiple layers and nonlinear activation. The creation of MLP models works nicely using TensorFlow. You must fine-tune the weight and bias of every iteration in an MLP. This indicates that when reducing the loss function, the weight and bias change continuously until they stabilize. With TensorFlow, you can therefore construct variables for the weight and bias. We can provide them with starting values that are randomly distributed or consist entirely of 0 or 1 s. Placeholders must have a certain kind of value and a specific form. The neural network perceptron is only a method of implementing the hyperplane that the perceptron describes. If one has a sample of data, we may compute the weight values in advance and then input them into the perceptron to get the output values. To train neural networks, we often employ online learning, where we get individual instances rather than the entire sample. We want the network to alter its parameters after each occurrence and gradually adapt over time.

Some unique features and merits of this technique: There is no need of keeping the training sample and intermediate results after optimization in external memory. In certain cases, we may choose a simpler technique where we do not need to keep the entire sample and solve a challenging optimization problem on it. Alternative techniques like SVMs may be extremely expensive with huge samples. The sample distribution may not be constant since the problem might change over time, making it impossible to select a training set in advance. For instance, we may put in place a speech recognition system that adjusts to the user. The system might undergo physical modifications. For instance, a robotic system's parts may deteriorate, or its sensors could stop working properly.

With online learning, we write the error function on individual instances rather than the entire sample. Beginning with arbitrary initial weights, we tweak the parameters slightly after each cycle to reduce error while keeping in mind what we've already learned. If this error function can be differentiated, gradient descent can be used. Regression errors on the single instance pair with index a are

$$(x^a, r^a), \quad \text{is } E^a(w|x^a, r^a) = 0.5(r^a - y^a)^2 = 0.5[r^a - (w^T x^a)]^2 \tag{3}$$

and for $b = 0, \ldots, d$, the online update is

$$\Delta w_b^a = \eta(r^a - y^a)x_b^a, \tag{4}$$

where $y = wtx$, $w = [w_0, w_1, \ldots, w_d]t$ and $x = [1, x_1, \ldots, x_d]t$ are augmented vectors to add the input and bias weight, and η is the learning factor, which is slowly reduced in time for stochastic convergence. This is known as stochastic gradient descent

$$\text{Update} = \text{Learning Factor} \cdot (\text{Desired Output} - \text{Actual Output}) \cdot \text{Input}. \tag{5}$$

If the desired output and the actual output match, no update is necessary. When the gap between the planned output and the actual output widens, the size of the update grows after it is finished. The update is considered positive if the input is positive and negative if it is negative, as is the case when the actual output is less than expected. As a result, the difference gets less while the actual production gets bigger. If the actual output exceeds the intended output, the update, which lowers the actual output and brings it closer to the desired output, is negative if the input

Table 2. Input and output for the AND function.

x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1

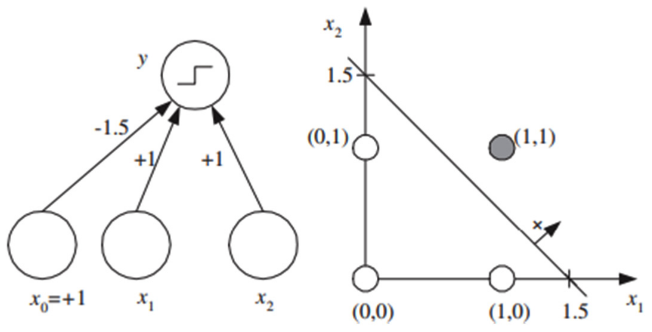


Fig. 4. The perceptron that implements AND and its geometric interpretation.[14]

is positive and positive if the input is negative. Table 2 shows the input and output for AND function. Figure 4 presents the geometric interpretation of a perceptron that implements AND.

The process involved in developing an MLP classification model:

(a) Initialize the weights and biases for each layer randomly
  - (i) For each hidden layer l
    1. Initialize the weight matrix W_l with random values
    2. Initialize the bias vector b_l with random values
  - (ii) Initialize the weight matrix W_{output} with random values
  - (iii) Initialize the bias vector b_{output} with random values

(b) Perform training iterations
  - (i) For each iteration within the provided range
    - Forward propagation
      1. Set the input layer activation A_0 as X_{train}
      2. For each hidden layer l
         - (i) Compute the linear transformation Z_l
         - (ii) Apply the activation function g to Z_l to compute the activation A_l
      3. Compute the output layer linear transformation Z_{output}
      4. Apply the softmax activation function to Z_{output} to compute the predicted probabilities A_{output}.
    - Backward propagation
      1. Compute the gradients at the output layer
      2. Compute the derivative of the loss function with respect to Z_{output} using the cross-entropy loss and softmax activation
         For each hidden layer l
      1. Compute the gradients
      2. Compute the gradients for the weights and biases

```
        3. Compute the gradients for the output layer weights and biases
      • Update the weights and biases using gradient descent: For each hidden layer $l$
        1. Update the weights and biases
        2. Update the output layer weights and biases
(c) Perform prediction on the testing dataset
    (i) Set the input layer activation $A_0$ as $X_{\text{test}}$
    (ii) For each hidden layer $l$
        1. Compute the linear transformation $Z_l$
        2. Apply the activation function $g$ to $Z_l$ to compute the activation $A_l$
    (iii) Compute the output layer linear transformation
    (iv) Apply the softmax activation function to $Z_{\text{output}}$ to compute the predicted probabilities $A_{\text{output}}$
(d) Return the predicted class probabilities for the testing dataset
```

2.4. *eXtreme Gradient Boosting*

A parallel tree boosting method called eXtreme Gradient Boosting (XGB) can be used to improve the accuracy of boosting models even more. The XGBoost's main approach is to develop ensemble-based algorithms by combining ensemble learning with decision trees. The ensemble learning idea has an impact on the training process for subsequent tree generation in the application of the XGBoost algorithm by employing the residual result from the prior training process as a new threshold for constructing new trees. This approach reduces the overfitting probability that was obtained by growing a new tree. When the maximum number of repeats is reached, the final output value is the outcome. The XGBoost algorithm develops a strong learner through additive learning by integrating multiple learners.

Unique features of XGB are Regularization—unlike the standard GBM implementation, which lacks regularization, XGBoost contains regularization. The degree of overfitting can be controlled through regularization. Because of this, XGBoost is also known as the "regularized boosting approach". Parallel processing: GBM lacks parallel processing, hence XGBoost is significantly quicker than conventional boosting. Parallelizing a serialized or sequential operation might also increase its performance. Only an older or prior decision tree can be used as the foundation for a new decision tree. Thus, each core may be used to start or develop a new tree. It is a flexible technique as a result. Hadoop and XGBoost are both compatible. Tree-pruning—when a GBM detects a negative loss in the split, it frequently stops dividing nodes. As a result, it is seen as a greedier algorithm. In contrast to GBM, XGBoost can divide until the max depth of the value is reached before beginning the pruning method, which involves removing the splits and moving backward when a gain cannot be found. Built-in cross-validation—cross-validation is a built-in feature of XGBoost and is executed after each repeat. As a result, determining the ideal number of boosting repetitions for a given run becomes quite simple. This is not the case with GBM since you must do a grid-search and the process only tests a limited set of data. Flexibility: XGBoost is far more adaptable than GBM since it enables users to customize and optimize any assessment criteria and optimization objectives they choose. This makes it extremely strong since it essentially removes the field's restrictions and makes anything conceivable.

XGBoost can tolerate missing values better than GBM, hence, it is a better alternative than GBM. Another value that is not the observation may be provided by the user and sent as a parameter. As XGBoost can find a missing value on every node and can also learn the route, it will attempt to complete various jobs. Ability to continue current model—the XGBoost model may also be restarted from any previous run's last iteration. The steps for creating a classification model using the XGBoost algorithm are explained in Fig. 5.

An XGBoost model can be created by building a tree followed by ensemble learning. The steps for creating and XGB model are

(a) Initialization is done by anticipating that the 0th tree equals 0
(b) All leaf node gain values are calculated and traversed until the largest gain score relative to the root node is obtained to determine the splitting mode
(c) Up until the gain score is negative or until another halting condition is met, assign the current binary leaf node set with values from the previous step's calculation

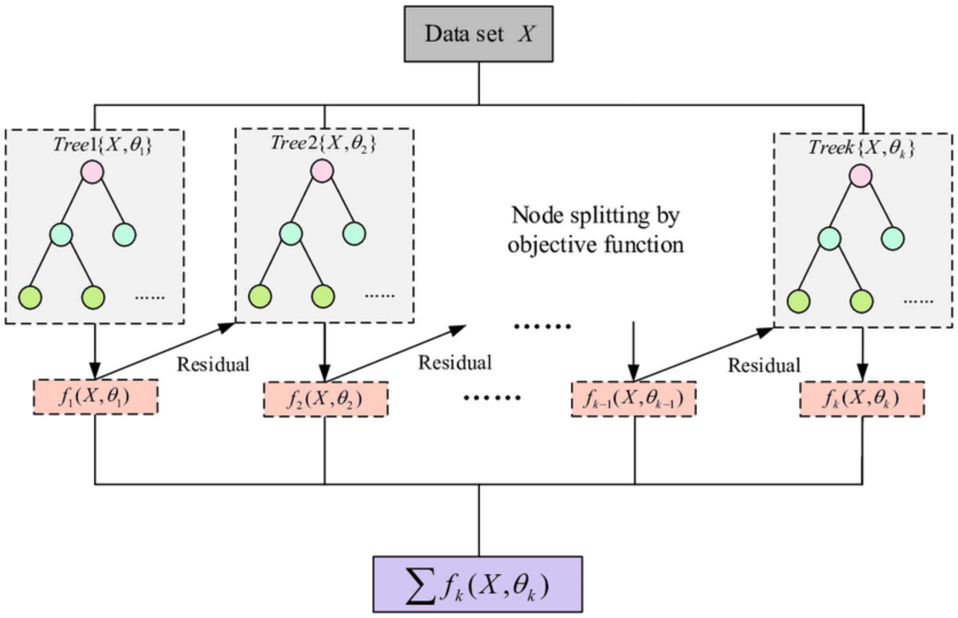


Fig. 5. XGB.[24]

(d) Determine the expected value for the entire leaf node
(e) If the value is larger than the threshold, create a new tree with the most recent forecast result as the threshold
(f) Repeat until the maximum number of trees is reached
(g) Compute the outcome by adding up the output values from the most recent node

2.5. *SVMs*

SVM is a linear model classification algorithm that uses planes, hyperplanes, and straight lines to define decision boundaries in a spatial plane. Then, according to whatever class division they belong to, new observations are categorized.

Predictions are made in a binary classifier using the general formula

$$\hat{x} = y[0] \times z[0] + y[1] \times z[1] + \cdots + y[a] \times z[c] + p > b, \tag{6}$$

where y is the regression coefficient,

$z[i]$ is the feature,
p is the intercept,
$\hat{x}$ is the regression prediction and
b is the bias term.

The formula's prediction feature is shown by the "greater than b" at the conclusion, which indicates that values bigger than b belong to one class and those less than or equal to b belong to another where b is the dividing bias.

SVMs are new learning methodologies that were developed in the theory of VC bounds and the structural risk minimization (SRM) framework. To put it more specifically, the SVM conducts SRM as opposed to classical adaptation algorithms, which operate in an L1 or L2 norm and minimize the absolute value of an error or an error square. This results in a model with a reduced VC dimension. The predicted likelihood of error is low when the VC dimension of the model is low, which results in a strong performance on unobserved data (good generalization). A good model generalizes well rather than one that excels in training data pairs. Figure 6 depicts the optimal hyperplane separating points in a two-dimensional place.

The SVM algorithm seeks to define the optimal line or judgement boundary that can divide n-dimensional space into groups in order to quickly categorize fresh data points in the future. This best option boundary is known as a hyperplane. To quickly classify the new piece of information into the appropriate category soon, the SVM's algorithm will probably create the best line or choice limit that can divide an n-layered region into categories. The optimum option limit is a hyperplane. The extreme foci/vectors that direct the development of the hyperplane are chosen by SVM. The calculation is known as the SVM, and these extreme situations are called help vectors.

The URL samples for the two classes are stored in an altered feature space by SVM using a map it constructs from the feature set and a hyperplane, based on

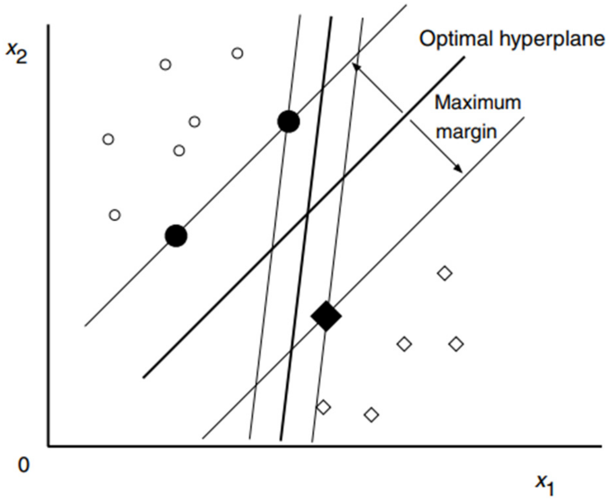


Fig. 6. Optimal separating hyperplane in a two-dimensional space.[16]

data from the training dataset and a specified alteration ($\theta : Rs \rightarrow F$). An SVM is trained for a two-class problem so that the direct decision function maximizes generalizability. Namely, the one-dimensional (l_m) feature space z is mapped from the m-dimensional input space x. Next, in z, the quadratic programming problem is resolved to create the best separation hyperplane between two classes.

SVMs employ a linear separation hyperplane to build a classifier with a maximum margin in the most straightforward pattern recognition tasks. The learning issue is transformed into a restricted nonlinear optimization problem to do that. In this situation, the constraints are linear, and the cost function is quadratic (i.e., there is a need to solve a quadratic programming problem).

The steps for training the SVM model

(a) Initialize an empty list called to store the support vectors
(b) Initialize an array called with zeros, where each element represents the Lagrange multiplier for each training example
(c) Initialize the bias term b as zero
(d) Compute the Gram matrix G of the training dataset using the kernel function

$$G[i, j] = K(X_{\text{train}}[i], X_{\text{train}}[j])$$

(e) Repeat the following steps until convergence
   (i) Select two Lagrange multipliers, randomly
   (ii) Compute the error for these multipliers
   (iii) Update the multipliers if the KKT conditions (complementary slackness, primal feasibility, dual feasibility, stationarity) are not satisfied

(f) Select the support vectors based on nonzero alpha values

$$\text{support\_vectors} = X_{\text{train}}[\text{alpha} > 0]$$

The steps for predicting the class labels for the testing dataset

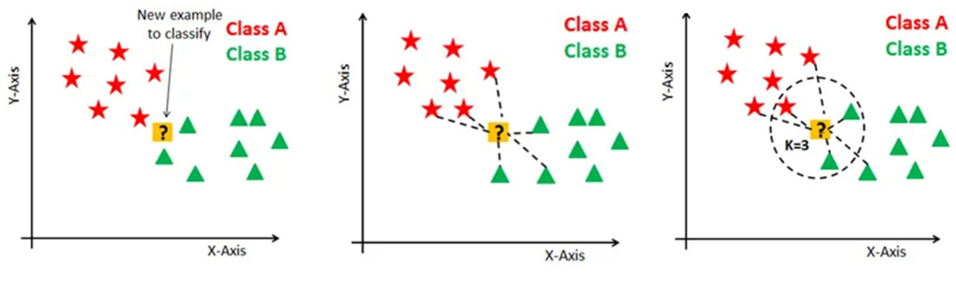
(a) Initialize an empty list called to store the predicted class labels
(b) For each testing example in X_{test}
   (i) Compute the decision function
   (ii) Classify the testing example based on the sign of the decision

2.6. *kNNs*

The kNNs classification technique involves training models straightforwardly by only storing the dataset. By finding the datapoints that are most similar to the unique data point in question and placing it in the average class of those data points, it generates new predictions for that data point. The kNNs method evaluates just one neighbor in its most basic version and simply applies that neighbor's class to the new observation. The k in "k-Nearest Neighbors" stands for any number of neighbors that can be considered.

kNN is a method that may be used for classification and regression problems; however, it is most frequently employed for classification issues. Although the algorithm is easy to understand and does not make any assumptions about the input, its memory usage is high, and the prediction process may be slow.

An example of classification using kNN is shown in Fig. 7. We need to choose which of the two potential classes—red and green—the new yellow instance belongs. It is computed how far the new instance is from its neighbors. A prediction has been made using most of the nearest neighbors. When given test samples, kNN uses distance measurements to seek in n-dimensional space for the k-closest neighbor. Classification choices are made using the k-closest neighbor. Figure 7 conveys the process of the kNN algorithm.



Initial Data

Calculating Distance

Voting for Labels, Finding Neighbors

Fig. 7. kNN algorithm.[6]

In this type of classifier, the conditional distribution of B given A is calculated, and the observation is assigned to the class with the highest probability. The kNN classifier initially selects the points in the training data that are nearest to x_0, denoted by m_0, given a positive in-k-nearest integer k and a test observation x_0. The percentage of points in m_0 with answer values equal to d is how the conditional probability for class d is then calculated. It then calculates the proportion of points in m_0 whose response values equal d to represent the conditional probability for class d

$$P(B = d|A = x_0) = 1/K \left(\sum_{c \in m_0}^{i} P(y_i = d) \right). \tag{7}$$

The test observation x_0 is then classified using kNN's Bayes rule into the class with the highest probability. The kNN process is explained in the pseudocode below

(a) Calculate the Euclidean distance between the current point and the rest of the points
(b) Repeat for the rest of the points
(c) Sort the distances in ascending order
(d) Take the first a distance values and find the a points that correspond to the distances
(e) Let a_i be the number of points of the ith class out of the total a points
(f) If a_i is greater than a_j, when i and j are not the same, add x to class i

2.7. *Naïve Bayes classifier* (*Gaussian NB*)

A straightforward probabilistic classifier based on conditional probability is Naive Bayes. This classification technique propagates strong independence between features using the fundamental Bayesian theorem. The frequency of characteristics and their interactions are determined when the corpus is considered. One class label is used. The correlation of all ignored qualities is regarded as independent in NB.

This text classification classifier may be a reward-based variation of keyword filtering. The rules for making decisions are described as

$$\begin{aligned}\emptyset_{k|y=1} &= p(x_j = k|y = 1)\\ &= \left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \text{ and } y^{(i)} = 1\} + 1}{\left(\sum_{i=1}^{m} 1\{y^{(1)} = 1\} n_i\right) + |V|} \right),\end{aligned} \tag{8}$$

$$\begin{aligned}\emptyset_{k|y=1} &= p(x_j = k|y = 0)\\ &= \left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \; and \; y^{(i)} = 0\} + 1}{\left(\sum_{i=1}^{m} 1\{y^{(1)} = 1\} n_i\right) + |V|} \right),\end{aligned} \tag{9}$$

$$\emptyset_{y=1} = \left(\frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}{(m)} \right). \tag{10}$$

$\theta_{x|y=1}$ assesses the probability that a specific feature in a phishing URL will be the kth word in the dictionary; $\theta_{x|y=0}$ assesses the probability that a specific feature in a legitimate URL will be the kth word in the dictionary. θ_y assesses the probability that any URL will be a phishing URL. m is the number of URLs in the training set. The entire dictionary contains V words. Training, $\theta_{x|y=1}, \theta_{x|y=0}\theta_y$ are calculated and for testing, $p(x|y = 1)\ p(y = 1)$ is compared to $p(x|y = 0)\ p(y = 0)$. To avoid underflow errors, logarithms are used. A given URL is classified as benign or phishing according to the following equation[20,21]:

$$\log p(x|y = 1) + \log p(y = 1)[\log p(x|y = 0) + \log p(y = 0)]. \tag{11}$$

The error rate of the Bayes classifier is the lowest when compared to other classification algorithms. The Bayes classifier will always select the class, which is biggest in terms of Bayes error, hence the error rate for $X = x_0$ will be 1-max$_j$ $\Pr(Y = t)$ Typically, the total Bayes error rate is provided by

$$1 - E(\max_j \Pr(Y = j \mid X), \tag{12}$$

where the probability for all potential values of X is averaged by the expectation (E is the expectation value, P is probability, and Y is another observation given X occurs).

The advantages of this classifier over others are that it is quick to train and classify, handles discrete and real data well, is insensitive to irrelevant features, and can handle streaming data well. A drawback is that it assumes the independence of features.

A generalized pseudo-code for carrying out classification using Naïve Bayes is given by

(a) Read values from train dataset D
(b) Compute the standard deviation and mean of the predictor variables in every class
(c) Compute the probability of each value of the predictor variable set using the gauss density equation until the probability of all predictor variables $(f_1, f_2, f_3, \ldots, f_n)$ are calculated
(d) For each class in the set of classes, compute the probability
(e) Select the highest probability

3. Experimental Design

3.1. *Dataset and pre-processing data*

To create a phishing website detection model, we first need to train it using a dataset. The dataset we have used for this research is the "ISCXURL-2016" dataset[13] which has been created and verified by the University of Canada, Brunswick. The file contains different datasets containing defacement, malware, phishing, spam, and benign URL links. For legitimate URLs, we have taken the benign URLs dataset which contains 10,000 links. For phishing links, we have used the website PhishTank,[12] which contains a list of phishing websites that are updated periodically to keep the internet safe. The latest 10,000 phishing links have been fetched from the site and stored in a comma-separated values file. The same number of benign and phishing links have been taken to avoid bias. From these datasets, 5,000 random URLs each were taken and compiled into a collective dataset of 10,000 URLs.

Raw data may contain missing values or incorrect features which will affect the model's accuracy. Data preprocessing is a technique used to remove any deviations and biases from the dataset. There are many steps involved such as cleaning data, data integration, transforming data, and reduction. This decreases the percentage of false negatives and false positives. After the data preprocessing has been performed on the dataset, feature selection and extraction algorithms are applied to the data in the dataset. These algorithms help in checking each instance for multiple identifiers of a phishing URL and reduce false positives. Each URL in the dataset is labeled "0" (legitimate) and "1" (phishing) for each feature that is being tested. Standard scaling is used to conclude if the URL is legit or malicious depending on the outcomes of all the features. The dataset is divided into 8,000 instances for training and 2,000 instances for testing the model.

3.2. *Libraries utilized*

The python[19] libraries used to implement our phishing detection models and feature extraction algorithms are

(1) Pandas: Pandas was used to perform the analysis and manipulation of data as well as reading CSV files
(2) NumPy: NumPy was used for array and matrix manipulation
(3) Scikit-Learn: Scikit-Learn was used for the implementation of ML techniques like Decision Trees, RF, MLPs, XGBoost Classifier, SVM, kNN, and Naive Bayes Classifier and for creating confusion matrices
(4) Seaborn: Seaborn was used for visualization of statistical data and plotting graphs
(5) Matplotlib: Matplotlib was used for plotting graphs and analyzing the distribution of data and the relationship between different data features
(6) Mlxtend: Mlxtend was used for the visualization of confusion matrices

3.3. *Feature extraction algorithms*

There are 6 main components of any URL, namely the protocol, subdomain, domain-name, path, query, and parameters. The protocol determines how data is transferred between a host server and a client. Some widely used protocols are HTTP/HTTPS, FTP, and SMTP. A domain name is a human-readable name that references the IP address of a host. Path points to the location or directory where

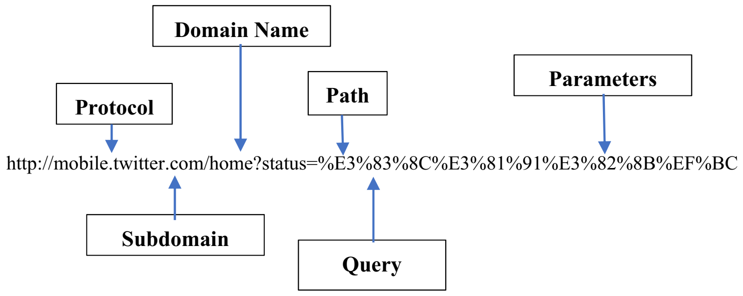

Domain Name
Protocol
Path
Parameters
http://mobile.twitter.com/home?status=%E3%83%8C%E3%81%91%E3%82%8B%EF%BC
Subdomain
Query

Fig. 8. Anatomy of an URL.

the requested file exists in the server. A sub-domain is simply a subpart of the domain. Queries are followed by a question mark in a URL and are requests by the users to fetch something. Figure 8 depicts the anatomy of an URL.

The feature extraction algorithms proposed for this research aim to dissect and analyze the differences between phishing and legit URLs. Various lexical features of URLs have been identified to detect phishing URLs. We have applied many algorithms to detect these features of the URL and determine whether the URL belongs to a phishing site or not. Contrary to other papers, we have not only used features based on the address bar but also domain-based features based on the HTML and JavaScript of the web pages. The feature extraction algorithms are explained.

(a) Address bar based features

(i) Domain of the URL: The domain is the primary part of any URL by which users identify a website. By extracting the domain, we check if the website is legitimate. Algorithm 1 extracts the domain name that is in the URL.

Algorithm 1. Get the URL's Domain-Name.

```
Input: URL
Output: Domain of URL
Procedure: getDomain (URL)
            Store domain and subdomains in D
            If regular expression of domain matches
                Store domain name
            End if
            Return domain
Store domain
```

(ii) IP address in the URL: Almost every website that is commonly used on the internet consists of domain names, subdomains, path, and protocol consisting of strings. The usage of an IP Address rather than a domain name of the site is a sign that the website might be a phishing website. Algorithm 2 checks if there is an IP address in the URL.

Algorithm 2. Check for IP Address in URL.

```
Input: URL
Output: IP Feature (0 = Legitimate or 1 = Phishing)
Procedure: presenceIP (URL)
        Try
              Check for IP_Address
              Seip_flag = 1
        Except
              Setip_flag = 0
        Return ip_flag
Store ip_flag
```

(iii) "@" in the URL: Any legitimate website does not contain the "@" symbol in it. Algorithm 3 checks if there is an "@" symbol in the URL. Usage of this symbol makes it so that the browser ignores everything before it and the actual address is often followed by the "@" symbol.

Algorithm 3. Check for "@" Symbol in URL.

```
Input: URL
Output: "@" Feature (0 = Legitimate or 1 = Phishing)
Procedure: haveAtSign (URL)
          If "@" sign is present in URL
             Set at = 1
          Else
             Set at = 0
          Return at
Store at
```

(iv) Length of the URL: The phishers tend to utilize longer URLs to conceal suspicious parts present in the address bar. Algorithm 4 computes the URL length and if it exceeds 54, it is classified as a phishing site.

(v) Depth of the URL: The URL is tokenized to get different parts of the URL, delimited by "/". A higher count of URL tokens is generally seen as a sign of a phishing website. Algorithm 5 calculates the URL's depth by computing the number of sub-pages present in URL.

(vi) Redirection: The presence of "//" in an URL indicates the redirection to another website. It usually comes after the protocols, HTTP and HTTPS,

Algorithm 4. Categorizing URL based on length.

```
Input: URL
Output: Length Feature ($0 = Legitimate$ or $1 = Phishing$)
Procedure: URL_Length (URL):
        If length of URL is greater than or equal to 54
            length_flag $= 0$
        Else
            length_flag $= 1$
        Return length_flag
Store length_flag
```

Algorithm 5. Returns number of "/" in URL.

```
Input: URL
Output: Depth (Number of "/")
Procedure: getDepth (URL)
        Split the URL at each "/" and store in S
        Initialize depth $= 0$
        For each part in range of S
            If length of part in $S != 0$
                $depth = depth + 1$
        Return depth
Store depth
```

in the URL. If the beginning of URL is HTTP then the "//" should be present in the URL's sixth position and if the beginning of URL is HTTPS then the "//" should be present in the URL's seventh position. Algorithm 6 checks if there is a "//" in the URL and if it is found in any other position, the URL is classified as phishing.

Algorithm 6. Redirection "//" in URL.

```
Input: URL
Output: Redirection Feature ($0 = Legitimate$ and $1 = Phishing$)
Procedure: redirection (URL)
        Store the position of "//" in pos
        If pos is greater than 6
            If pos is greater than 7
                return 1
            Else
                return 0
        Else
            return 0
```

(vii) Presence of "HTTP/HTTPS" in the URL's domain name: The phishers may sometimes attach an "HTTPS" token in the URL's domain area to fool the users. Algorithm 7 checks if there is an "HTTP/HTTPS" in the URL's domain area.

Algorithm 7. Existence of HTTP/HTTPS Token.

```
Input: URL
Output: HTTPS Feature (0 = Legitimate and 1 = Phishing)
Procedure: httpDomain (URL)
        Store the domain of the URL in domain
        If "https" is present in domain
            return 1
        Else
            return 0
```

(viii) Usage of URL shortening services: URL Shortening is a commonly used method to reduce an URL's length and still take the user to the required webpage. This method uses HTTP redirection to achieve this. Not all shortened URLs are phishing sites but for the sake of this research, we assume it to be a phishing site. Algorithm 8 checks if the URL is using a shortening service.

Algorithm 8. Shortening Service in URL.

```
Input: URL
Output: Shortening Feature (0 = Legitimate and 1 = Phishing)
Procedure: shortURL (URL)
        Store shortening service URL's regular expressions in shortening_service
        Search for the shortening_service in URL and store in match
        If match:
            return 1
        Else
            return 0
```

(ix) Prefix/Suffix "−" in domain of URL: The hyphen symbol "−" is not often utilized in URLs. The phishers add "−" as the suffix or prefix to URLs to make them appear legitimate to users. Algorithm 9 checks if there is a "−" symbol in the URL.

Algorithm 9. Check for "–" prefix/suffix in the URL.

```
Input: URL
Output: Hyphen Feature (0 = Legitimate and 1 = Phishing)
Procedure: Hyphen (URL)
        If "–" is present in the URL
            return 1
        Else
            return 0
```

(b) Domain-based features

(i) DNS record: The identity of a website is cross-checked with the WHOIS database. If the website's identity is unrecognized in the WHOIS database or if the website has no record, the URL is deemed as a phishing site.

Algorithm 10. Check DNS Record.

```
Input: URL
Output: DNS Feature (0 = Legitimate and 1 = Phishing)
Procedure: DNS (URL)
        Set dns = 0
        If URL present in WHOIS database
            Store URL in domain_name
        Else
            dns = 1
Store dns
```

(ii) Website traffic: The traffic of a website is used to measure its popularity by checking the website visitors count and how many pages are visited by each of them per session. The lifespan of phishing websites is small, so they are not recognized by the Alexa database. Even in worst-case scenarios, legitimate sites were ranked at 100,000 in Alexa. So, if the site is found to have no traffic or if it is unacknowledged by the Alexa database, it is deemed as a phishing site i.e., the domain's rank is less than 100,000.

(iii) Age of domain: The age of a domain can be found using the WHOIS database. Since most phishing websites are short-lived, we consider the age of a legitimate website to be at least 6 months. The age is calculated as the difference between the creation time of a website and its expiration time.

(iv) End period of domain: The remaining time of a domain is calculated by computing the difference between the creation time of a website and its

Algorithm 11. Check Website Traffic.

```
Input: URL
Output: Web Traffic Feature (0 = Legitimate and 1 = Phishing)
Procedure: web_traffic (URL)
        Try
            Store URL in url
            Query the Alexa database for rank of website and store in rank
        Except
            return 1
        If rank is less than 100,000
            return 1
        Else
            return 0
```

Algorithm 12. Calculate Domain's Age.

```
Input: Domain Name
Output: Domain Age Feature (0 = Legitimate and 1 = Phishing)
Procedure: domainAge (domain_name)
        Get creation date of URL and store in creationDate
        Get expiration date of URL and store in expirationDate
        Try
            Convert creationDate to Year-month-date format
            Convert expirationDate to Year-month-date format
        Except
            return 1
        If creationDate or expirationDate is empty
            return 1
        Else if creationDate or expirationDate is a list
            return 1
        Else
            Calculate age_of_domain as expiration_date–creation_date
        If age_of_domain < 6
            age = 1
        Else
            age = 0
return age
```

expiration time using the WHOIS database. We considered the end period of a legitimate site as less than or equal to 6 months.

Algorithm 13. Check Domain's End Period.

```
Input: Domain Name
Output: Domain End Period Feature (0 = Legitimate and 1 = Phishing)
Procedure: domainEnd (domain_name)
        Get expiration date of URL and store in expirationDate
        Try
            Convert expirationDate to Year-month-date format
        Except
            return 1
        If expirationDate is empty
            return 1
        Else if expirationDate is a list
            return 1
        Else
            Get current day's date and store in current_date
            Calculate end_period as expirationDate − current_date
        If end_period < 6
            end = 0
        Else
            end = 1
return end
```

(c) HTML and JavaScript-based features

All the following feature extraction algorithms use the "requests" library and check the response for the corresponding feature.

(i) iFrame redirection: iFrame can be used by phishers to embed harmful links or buttons that can redirect the user to another harmful webpage. Algorithm 14 checks the website's HTML markup for iFrame tags and classifies the URL as legitimate or phishing.

(ii) Status bar customization: Sometimes the legitimacy of a site can be recognized by hovering the mouse over the status bar. By analyzing the site for JavaScript mouseover events, we can check if the site is the original one or a phishing site.

(iii) Disabled right click: Unless the website is for examinations or has copyright-protected content, typically right-click is enabled on every website. It is very suspicious if the right click has been disabled on a regular website without any reason. Algorithm 16 checks the status of the right click and classifies the website as legitimate or phishing.

Algorithm 14. Detecting iFrame Redirection.

```
Input: Response
Output: iFrame Feature (0 = Legitimate and 1 = Phishing)
Procedure: iframeRedirect (response)
        If the website response is empty
            return 1
        Else
            If the < iframe > tag or < frameBorder > tag is found in website response
                return 0
            Else
                return 1
```

Algorithm 15. Check Mouseover effect on Status Bar.

```
Input: Response
Output: Mouseover Feature (0 = Legitimate and 1 = Phishing)
Procedure: hoverMouse (response)
        If the website response is empty
            return 1
        Else
            If status bar mouseover event is found
                return 0
            Else
                return 1
```

Algorithm 16. Checking if Right Click is disabled.

```
Input: Response
Output: Right-Click Feature (0 = Legitimate and 1 = Phishing)
Procedure: disabledRightClk (response)
        If the website response is empty
            return 1
        Else
            If any right click disable eventhandler is found
                return 0
            Else
                return 1
```

(iv) Website forwarding: Website forwarding is used to trick users into entering phishing sites using URL Redirection and aims to collect personal information and login credentials. Algorithm 17 checks the number of redirects from the website upon access. If the number is greater than 2, we classify it as phishing.

Algorithm 17. Checking the number of forwarding.

```
Input: Response
Output: Forwarding Feature (0 = Legitimate and1 = Phishing)
Procedure: webforwarding (response)
        If the website response is empty
            return 1
        Else
            If the number of redirections is less than or equal to 2
                return 0
            Else
                return 1
```

For example, let us consider an URL: http://abc12345.example.com/img/abcd/efghi/. Applying feature extraction to the URL we get the following labels

Have_IP - 0, Have_At - 0, URL_Length - 0, URL_Depth - 3, Redirection - 0, https_Domain - 0, TinyURL - 0, Prefix/Suffix - 0, DNS_Record - 0, Web_Traffic - 1, Domain_Age - 0, Domain_End - 1, iFrame - 1, Mouse_Over - 1, Right_Click - 1, Web_Forwards - 1.

We apply ML techniques to classify the URL based on these labeled features and predict the outcome i.e., phishing or legitimate, e.g., "1" i.e., a phishing website.

4. Performance Evaluation

4.1. *Performance metrics*

- Precision: Precision is used to calculate the number of URLs that were predicted as positive which is correct. Precision is the number of phishing URLs predicted correctly out of the total URLs

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{13}$$

where TP is true positive and FP is false positive.

- Recall (or sensitivity or TP rate): Recall is utilized to calculate the number of actual positives that were predicted properly. Recall is the total number of phishing URLs that are classified as phishing and the number of legitimate URLs that are correctly classified as legitimate.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

where TP is true positive and FN is false negative.

- F1-score: F1-score is used to measure the accuracy of the dataset. It is calculated as shown below. It is useful when computing average rates

$$\mathrm{F1-score} = 2 \times \frac{(\mathrm{Precision} \times \mathrm{Recall})}{(\mathrm{Precision} + \mathrm{Recall})}. \tag{15}$$

- Accuracy: Accuracy is the number of properly classified sites out of the total sites in the dataset

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \tag{16}$$

where TP is true positive, TN is true negative, FP is false positive and FN is false negative.

- Log Loss: Log Loss indicates how close the predicted value is to the actual value

$$\mathrm{Log\ Loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij} (\log(P_{ij})), \tag{17}$$

where N is the number of samples, M is the number of attributes, x_{ij} indicates whether the ith sample belongs to the jth class, and P_{ij} indicates the probability of the ith sample belonging to the jth class.

- AUC-ROC: A receiver operating characteristic (ROC) curve is used to plot the performance of a classification model based on an FP rate and a TP rate at different thresholds. The area under the curve (AUC) is calculated

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{18}$$

where TN is true negative and FP is false positive.

- DCG: The discounted cumulative gain (DCG) is used to measure ranking quality. Mostly used for information retrieval applications like search engines

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{rel_i}{\log_2 (i+1)}, \tag{19}$$

where p is a particular rank position and rel_i is the graded relevance of the result at position i.

- NDCG: NDCG is the normalized DCG. Ideal DCG (IDCG) is arranging the values in descending order by their ranking and calculating their DCG

$$\mathrm{IDCG}_p = \sum_{i=1}^{|REL_p|} \frac{rel_i}{\log_2 (i+1)}, \tag{20}$$

where $|REL_p|$ represents the list of relevant documents in the corpus up to position p, p is a particular rank position, and rel_i is the graded relevance of the result at position i

$$\mathrm{NDCG}_p = \frac{\mathrm{DCG}_p}{\mathrm{IDCG}_p}, \tag{21}$$

where p is a particular rank position.

- Mean absolute error: MAE is the measure of errors between a pair of observations for a feature or URL.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}, \tag{22}$$

where y_i is the actual value for the ith observation, x_i is the predicted value for the ith observation and n is the total number of observations.

- Confusion matrix: Confusion matrix describes how well an ML system performs by presenting the TPs, FPs, FNs, and TNs predicted by the trained model in the form of a matrix. Figure 9 depicts a confusion matrix.

TP: The true prediction rate of the positive samples. The predicted value is positive, and the actual value is also positive.[11]
FP: Negative value incorrectly classified as positive.[11]
TN: The true prediction of negative samples. The predicted value is negative, and the actual value is also negative.[11]
FN: Positive value incorrectly classified as negative.[11]

4.2. *Performance evaluation of classification techniques*

The performance metrics are utilized to evaluate the performance of each classification technique and to determine the best model for different dataset splits.

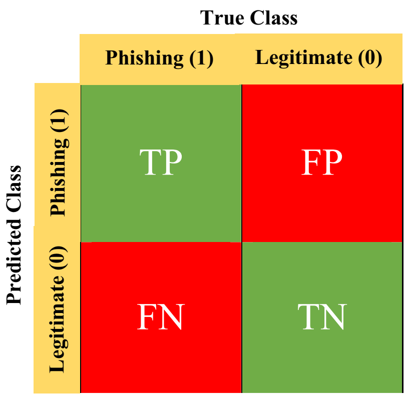


Fig. 9. Confusion matrix.

4.3. *Results and discussion*

Based on the research done, we have implemented 17 different feature extraction algorithms to distinguish between legitimate and phishing websites. The tables contain the performance metrics of each technique ranked in descending order based on their accuracy. In Table 3, XGB provided the best performance with 93.5% accuracy, 89.1% precision, 93.1% F-score, and 93.5% ROC. The technique also had the least log loss and MAE. decision tree had the best precision while both decision tree and RF techniques had the highest NDCG score. Table 4 provides similar results to the 65:35 split dataset but with some key differences. *k*NN technique provides better accuracy than the decision tree technique. The MLPs technique provided the best precision score and the highest NDCG score. XGB is at the top for other metrics. Table 5 provided interesting results with the MLPs technique

Table 3. Performance analysis based on ML techniques and the metrics with a 65:35 training and testing dataset split.

Technique	Confusion matrix		Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	Log loss	ROC	DCG	NDCG (%)	MAE
XGBoost	1735	39	93.5	97.5	89.1	93.1	2.240	0.9345	191.278	99.1	0.065
	188	1538									
MLPs	1732	42	93.4	97.3	89.0	93.0	2.280	0.9334	191.214	99.1	0.066
	189	1537									
Decision tree	1759	15	92.7	99.0	86.0	92.1	2.526	0.9259	191.473	99.2	0.073
	241	1485									
kNN	1731	43	92.6	97.2	87.4	92.1	2.566	0.9250	191.044	99.0	0.074
	217	1509									
RF	1760	14	92.2	99.1	85.0	91.5	2.694	0.9210	191.398	99.2	0.078
	259	1467									
SVM	1759	15	91.5	99.0	83.7	90.7	2.931	0.9140	191.254	99.1	0.085
	282	1444									
Naïve Bayes	1743	31	91.1	97.9	83.7	90.2	3.089	0.9095	190.928	98.9	0.089
	282	1444									

Table 4. Performance analysis based on ML techniques and the metrics with a 70:30 training and testing dataset split.

Technique	Confusion matrix		Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	Log loss	ROC	DCG	NDCG (%)	MAE
XGBoost	1473	36	93.5	97.4	89.3	93.2	2.245	0.9347	169.358	99.1	0.065
	159	1332									
MLPs	1498	11	92.9	99.2	86.4	92.3	2.464	0.9282	169.639	99.3	0.071
	203	1288									
*k*NN	1463	46	92.7	96.6	88.5	92.4	2.510	0.9270	169.083	98.9	0.073
	172	1319									
Decision tree	1490	19	91.9	98.5	84.9	91.2	2.809	0.9182	169.349	99.1	0.081
	225	1266									
RF	1494	15	91.9	98.8	84.6	91.2	2.809	0.9182	169.412	99.1	0.081
	229	1262									
SVM	1493	16	91.2	98.7	83.4	90.4	3.028	0.9118	169.291	99.1	0.088
	247	1244									
Naïve Bayes	1493	16	91.1	98.7	83.2	90.3	3.062	0.9108	169.274	99.1	0.089
	250	1241									

Table 5. Performance analysis based on ML techniques and the metrics with a 75:25 training and testing dataset split.

Technique	Confusion matrix		Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	Log loss	ROC	DCG	NDCG (%)	MAE
MLPs	1167	46	93.6	96.2	91.1	93.6	2.224	0.9363	149.808	99.0	0.064
	115	1172									
XGBoost	1186	27	93.5	97.7	89.5	93.4	2.238	0.9364	150.102	99.2	0.065
	135	1152									
*k*NN	1205	8	92.4	99.3	85.9	92.1	2.625	0.9259	150.268	99.3	0.076
	182	1105									
Decision	1196	17	91.8	98.5	85.5	91.8	2.818	0.9203	150.039	99.1	0.082
tree	187	1100									
RF	1201	12	91.7	98.9	84.8	91.7	2.860	0.9192	150.108	99.2	0.083
	195	1092									
SVM	1197	16	90.8	98.5	83.4	90.8	3.178	0.9102	149.913	99.1	0.092
	214	1073									
Naïve	1193	20	90.6	98.2	83.4	90.6	3.233	0.9086	149.822	99.0	0.094
Bayes	214	1073									

Table 6. Performance analysis based on ML techniques and the metrics with an 80:20 training and testing dataset split.

Technique	Confusion matrix		Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	Log loss	ROC	DCG	NDCG (%)	MAE
XGBoost	970	24	93.8	97.4	90.0	93.5	2.159	0.9377	122.578	99.1	0.062
	101	905									
MLPs	985	9	93.4	99.0	87.7	93.0	2.297	0.9338	122.799	99.3	0.066
	124	882									
*k*NN	961	33	93.2	96.5	89.8	93.0	2.349	0.9322	122.357	98.9	0.068
	103	903									
Decision	985	9	92.4	99.0	85.9	92.0	2.608	0.9248	122.692	99.2	0.076
tree	142	864									
RF	986	8	91.9	99.1	84.7	91.3	2.798	0.9194	122.645	99.2	0.081
	154	852									
SVM	988	6	91.7	99.3	84.1	91.1	2.867	0.9174	122.658	99.2	0.083
	160	846									
Naïve	988	6	91.6	99.3	84.0	91.0	2.884	0.9169	122.652	99.2	0.084
Bayes	161	845									

providing the best accuracy of 93.6%, and the highest F-score and recall scores with the lowest log loss and MAE, while XGB had 93.5% accuracy. *k*NN technique had the highest precision score and highest NDCG score. Observing Table 6, XGB had the best performance with 93.8% accuracy in identifying phishing websites. XGBoost also has the highest recall and F1-score with the lowest log loss and MAE. MLPs is a close alternative, as it has produced similar results with 93.4% accuracy and has the highest NDCG score. Surprisingly, the techniques with the least performance i.e., SVM and Naïve Bayes also have the highest precision of 99.3% compared to the other techniques.

On comparing our research with Jimmy *et al.*,[2] we have found that using an intermediate number of features is necessary to reduce noise in the data and improve accuracy. Their paper tested the performance of ML techniques on datasets with

38, 30, 20, and 10 features and found that the accuracy increased when they lowered the number of features. By using 17 important features, we are maintaining a balance between accuracy and noise in data. The research by Ala *et al.*[3] utilized three different datasets, one with many phishing emails, one with many legitimate emails, and one with a balance in both. They found that using 50 features gave the best performance when data is skewed. Using 20 features was adequate but it was not effective enough for a skewed dataset. In the research by Dželila *et al.*,[6] they concluded that the feature selection optimizes phishing detection immensely. By prioritizing a few important features rather than many, they required less time to build and train the model and increased performance and efficiency.

Our research focuses on creating an effectual solution to help in the detection of phishing attacks with great accuracy and efficiency. Our anti-phishing method utilized various URL-based features to train different ML models and the XGBoost model had the highest accuracy and provided the best performance. However, the developed anti-phishing tool is not foolproof and has some limitations. The HTML and Java based features rely heavily on the source code of the legitimate sites. An attacker can bypass this by copying the source-code of the legitimate website and modifying login portal to steal credentials. If the attacker alters every page resource like the CSS, JavaScript, images, etc., the model will fail to detect the phishing site. The attacker can try to embed elements and objects like JavaScript, images, Flash, etc., rather than utilizing DOM to avoid the detection of HTML code by the detection tool. For these scenarios, our approach might classify a phishing site as legitimate.

5. Conclusion

This research is intended to estimate the performance of various ML algorithms to detect phishing websites by training them utilizing different URL-based features. The ML techniques were implemented, and the results obtained were analyzed using the metrics: precision, sensitivity, F-score, accuracy, log loss, ROC, DCG, NDCG, MAE, and confusion matrix. The ML techniques implemented and examined are decision trees, RFs, MLPs, XGBoost, SVM, *k*NNs, and Naïve Bayes classification algorithms. These techniques provide good accuracy, ROC curve area, DCG, NDCG, precision, and recall with low log loss and MAE. After performance analysis of all the techniques, we conclude that the XGBoost technique provides consistently the best performance compared to the other techniques. MLPs is a close alternative to XGBoost.

Future Works

Attackers are wary of phishing detection tools and mechanisms, so they keep finding new ways to trick innocent users. So, the tool must automatically detect new features in phishing URLs. This is something to be investigated. The techniques have provided good performance but still pale in comparison to some models in other papers. The feature extraction algorithms need to be optimized and new features should be chosen carefully to increase accuracy. The dataset was tested with basic classification algorithms. Training models using deep learning algorithms can provide better results with greater accuracy, so that can be the next step to pursue. The trained model with the best performance can be used to build an extension or software that anyone can utilize with ease to help detect phishing websites.

References

1. T. O. Ojewumi, G. O. Ogunleye, B. O. Oguntunde, O. Folorunsho, S. G. Fashoto and N. Ogbu, Performance evaluation of machine learning tools for detection of phishing attacks on web pages, *Sci. Afr.* **16** (2022) e01165.
2. J. Moedjahedy, A. Setyanto, F. K. Alarfaj and M. Alreshoodi, CCrFS: Combine correlation features selection for detecting phishing websites using machine learning, *Future Internet* **14** (2022) 229.
3. A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar and E. A. Elsoud, An intelligent cyber security phishing detection system using deep learning techniques, *Cluster Comput.* **25** (2022) 3819.
4. A. K. Jain and B. B. Gupta, A machine learning based approach for phishing detection using hyperlinks information, *J. Ambient Intell. Hum. Comput.* **10** (2019) 2015.
5. J. Pavithra and S. S. Samy, A comparative study on detection of malware and benign on the internet using machine learning classifiers, *Math. Prob. Eng.* **2022** (2022) 4893390.
6. D. Mehanović and J. Kevrić, Phishing website detection using machine learning classifiers optimized by feature selection, *IIETA Traitement du Signal* **37** (2020) 563.
7. B. B. Gupta, K. Yadav, I. Razzak, K. Psannis, A. Castiglione and X. Chang, A novel approach for phishing URLs detection using lexical based machine learning in a real-time environment, *Comput. Commun.* **175** (2021) 47.
8. F. Thabtah and F. Kamalov, Phishing detection: A case analysis on classifiers with rules using machine learning, *J. Inf. Knowl. Manage.* **16** (2017) 1750034.
9. A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum and M. Hamdani, Phishing web site detection using diverse machine learning algorithms, *Electron. Lib.* (2020), https://www.emerald.com/insight/content/doi/10.1108/EL-05-2019-0118/full/html.
10. A. K. Dutta, Detecting phishing websites using machine learning technique, *PLoS One* **16**(10) (2021) e0258361.
11. M. Almseidin, A. A. Zuraiq, M. Al-kasassbeh and N. Alnidami, Phishing detection based on machine learning and feature selection methods, *Int. J. Inf. Manage.* **13** (2019) 171.
12. PhishTank, www.phishtank.com.
13. ISCXURL-2016 Dataset, https://www.unb.ca/cic/datasets/url-2016.html.
14. E. Alpaydin, *Introduction to Machine Learning*, (PHI Learning, Cambridge, Massachusetts, 2015).
15. T. H. Saragih, R. Ramadhani, M. I. Mazdadi and M. Haekal, Energy efficiency in buildings using multivariate extreme gradient boosting, *2022 Seventh Int. Conf. Informatics and Computing* (*ICIC*), Denpasar, Bali, Indonesia, 8-9 December 2022, pp. 1–5.
16. S. Abe, *Support Vector Machines for Pattern Classification*, (Springer, London, England, 2010).
17. F. Pedregosa *et al.*, Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.* **12** (2011) 2825.

18. Anti-Phishing Working Group, Phishing activity trends report (2020), https://docs.apwg.org/reports/apwg_trends_report_q4_2019.pdf.
19. G. Van Rossum and F. L. Drake, Python 3 Reference Manual (CreateSpace, Scotts Valley, 2009).
20. G. James, D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, (Springer, New York, 2013).
21. A. K. Jain and B. B. Gupta, PHISH-SAFE: URL features-based phishing detection system using machine learning, in *Cyber Security*, Advances in Intelligent Systems and Computing, Vol. 729 (Springer, Singapore, 2018).
22. Netcraft Anti-Phishing Extension, https://www.netcraft.com/apps/browser/.
23. J. Abualdenien and A. Borrmann, Ensemble-learning approach for the classification of Levels Of Geometry (LOG) of building elements, *Adv. Eng. Inf.* **51** (2022) 101497.
24. R. Guo, Z. Zhao, T. Wang and G. Liu, Degradation state recognition of piston pump based on ICEEMDAN and XGBoost, *Appl. Sci.* **10** (2020) 6593.
25. J. Tanimu and S. Shiaeles, Phishing detection using machine learning algorithm, *Proc. 2022 IEEE Int. Conf. Cyber Security and Resilience*, Rhodes, Greece, 27–29 July 2022, pp. 317–322.
26. A. A. Adzhar, Z. Mabni and Z. Ibrahim, A comparative study on email phishing detection using machine learning techniques, *2022 IEEE Int. Conf. Computing*, Kota Kinabalu, Malaysia, 14–16 November 2022, pp. 96–101.
27. M. Mohammed, K. K. Prasanth and S. V. S. Subhash, Phishing detection using machine learning algorithms, *4th Int. Conf. Smart Systems and Inventive Technology*, Tirunelveli, India, 20–22 January 2022, pp. 921–924.
28. P. Padmalochan, M. A. Kumar and P. Deepak, A novel logo identification technique for logo-based phishing detection in cyber-physical systems, *Future Internet* **14** (2022) 241.
29. P. Pankaj and M. Nishchol, Phish-Sight: A new approach for phishing detection using dominant colors on web pages and machine learning, *Int. J. Inf. Sec.* (2023), https://doi.org/10.1007/s10207-023-00672-4.
30. O. El Kouari, H. Benaboud and S. Lazaar, Using machine learning to deal with Phishing and Spam Detection: An overview, *Proc. 3rd Int. Conf. Networking, Information Systems & Security* (2020), doi: 10.1145/3386723.3387891.
31. M. M. Yadollahi, F. Shoeleh, E. Serkani, A. Madani and H. Gharaee, An adaptive machine learning based approach for phishing detection using hybrid features, *2019 5th Int. Conf. Web Research*, Tehran, Iran, 24–25 April 2019, pp. 281–286.