Machine Learning-Based Phishing Attack Detection

Sohrab Hossain[1], Dhiman Sarma[2]*, Rana Joyti Chakma[3]
Department of Computer Science and Engineering, East Delta University, Chittagong, Bangladesh[1]
Department of Computer Science and Engineering, Rangamati Science and Technology University, Rangamati, Bangladesh[2,3]

***Abstract*—This paper explores machine learning techniques and evaluates their performances when trained to perform against datasets consisting of features that can differentiate between a Phishing Website and a safe one. This capability of telling these sites apart from one another is vital in the modern-day internet surfing. As more and more of our resources shift online, one vulnerability and a leak of sensitive information by someone could bring everything down in a connected network. This paper's objective through this research is to highlight the best technique for identifying one of the most commonly occurring cyberattacks and thus allow faster identification and blacklisting of such sites, therefore leading to a safer and more secure web surfing experience for everyone. To achieve this, we describe each of the techniques we look into in great detail and use different evaluation techniques to portray their performance visually. After pitting all of these techniques against each other, we have concluded with an explanation in this paper that Random Forest Classifier does indeed work best for Phishing Website Detection.**

Keywords*—Phishing attack; phishing attack detection; phishing website detection; machine learning; random forest classifier

I. Introduction

Phishing Attacks are the most common ways of attack in the digital world these days. Any method of communication can be used to target an individual to trick them into leaking confidential data in a fake environment, which can later be used to harm the sole victim or even an entire business depending on the attacker's intent and the type of data leaked.

Phishing attacks, while dangerous, can be avoided by simply creating awareness and developing habits of staying alert and continuously being on the lookout when surfing through the internet and only clicking links after verifying if the source of the links is trustworthy at all. There are also tools such as browser extensions that notify users when they have entered their credentials on a fake site, possibly having their credentials transferred to a user with malicious intent. Other tools can also allow networks to lock down everything and allow access to whitelisted sites to provide extra security while compromising some convenience on the user side [1].

In a related study, five main reasons have been stated behind users falling into traps of phishing attack schemes:

- Lack of knowledge about URLs.
- Lack of knowledge about trusted websites.
- Lack of visibility of full web addresses due to the redirection or hidden URLs.
- Lack of time for analyzing URLs, and accidental entries of some web pages.
- Lack of capability of telling phishing web pages apart from legitimate ones.

One example of such an attack would be the attack in 2016, known as the Bangladesh Bank Cyber Heist. Security Hackers issued thirty-five fraudulent instructions via the SWIFT network to illegally transfer almost 1 billion US dollars from the Federal Reserve Bank of New York account that belonged to Bangladesh Bank. Out of these 35 instructions, 5 of them successfully transferred 101 million dollars, with 20 million traced to Sri Lanka and 81 million traced to the Philippines. Fortunately, the Federal Reserve Bank of New York was able to block the remaining thirty transactions. Without this block, another 850 million dollars would have been lost. And it was possible all thanks to noticing a misspelled instruction that raised suspicions among the authorities. The money transferred to Sri Lanka was all recovered, but from the US$ 81 million transferred to the Philippines, only US$ 18 million was recovered. Most of the money transferred to the Philippines were collected into four personal accounts [2].

The method of this attack has been suspected to be a Dridex malware. It specializes in stealing bank credentials by using macros set up in a Word or Excel document. Windows users can fall victim to such an attack if they open email attachments in Word or Excel, containing such a macro, which once activated on opening these documents, begin downloading Dridex, which then infects computers and sets up the stage for a banking theft. A knowledgeable and alert employee or a software aiding in detecting such an attack would have helped immensely in this event [3].

Machine learning algorithms are widely used to detect hidden patterns in the dataset. The most common algorithms are K-nearest neighbor, decision trees, random forest, and support vector machine [4]. In addition, belief rule-based expert system can mine rules from the dataset [5] [6].

In this paper, we focus on training machine learning models that can detect phishing web pages apart from real web pages. We analyze each of these models and state our findings and research in this paper to allow for others to have a clear understanding of the performance of these models when trained for this purpose. Of course, data preprocessing is very crucial for the models to work as they did in our case, and that is an essential part of the procedure. Papers from other researchers contributed immensely to our research, and we hope our paper will do the same by providing a collection of our findings regarding Phishing Detection using Machine Learning in this paper.

*Corresponding Author

The remaining of the paper is organized as follows. In Section II, we reviewed the literature, followed by presenting the proposed methodology in Section III. The empirical results of the proposed approach are explained in Section IV, followed by Section V where a conclusion and further research scopes are discussed.

II. Literature Review

A. Types of Phishing Attacks

1) Algorithm-Based phishing: Attackers access sensitive information from a website's database by employing different algorithms V. Shreeram, M. Suban, P. Shanthi, K. Manjula proposed an anti-phishing detection method that would detect phishing hyperlinks with the help of the rule-based system that is formulated from the genetic algorithm (GA). A phishing link is detected if it matches the ruleset that is created by GA, which is stored in a database [7].

2) Deceptive phishing: This technique involves supplying clients with malicious links via emails and redirecting them to malicious websites where they are likely to enter sensitive information. Huajun Huang, Junshan Tan, Lingxi Liu gives a thorough overview of a deceptive phishing attack and different anti-phishing techniques. They present the different methods used by phishers and the advantages and disadvantages of the different countermeasures used [8].

3) URL phishing: Hackers can inject hidden links that redirect to malicious pages into the URL, where one may not expect to find one. Mohammed Nazim Feroz, Susan Mengel, proposes a method to detect URL phishing with URL ranking. They classify the URLs by their lexical and host-based features and categorizes and rank the URLs using the online URL reputation services [9].

4) Hosts file poisoning: Replacing hostnames in the host records can override the usual process of DNS servers trying to retrieve actual IP addresses from beyond the network. This technique can poison the records and allow valid URLs that are meant to lead to secure sites lead to malicious pages instead, due to compromised IP associations in the server. Saeed Abu-Nimeh, Suku Nair, proposes a new attack that can bypass security toolbars and phishing filters by using DNS poisoning. They use spoofed DNS cache entries to create fake results and successfully attack four renowned security toolbars and the phishing filters of three popular browsers without being detected [10].

5) Content injection phishing: Data collection is achieved in this technique by concatenation of malicious sections within a real website. Jussi-Pekka Erkkil presents the different methods by which phishing techniques can trick a person. A list of several strategies is listed that can detect phishing. The paper proposes that the company adapt effective protocols to keep their security features up to date [11].

6) Clone phishing: Duplicating already sent emails and attaching a malicious link into it can allow for a successful attack on an unsuspecting user. Ahmad Alamgir Khan proposed a new method where websites use One Time Password and User-machine Identification system to combat phishing attacks. Webservers will send a one time password to a user by SMS or email and create an encrypted token for the device after the user inputs the password [12].

B. Phishing Website Detection Techniques

1) Blacklist filter: Blacklists can be maintained to block recorded unwanted sites from reaching the client's machine. These filters can be applied in different security measures like DNS servers, firewalls, email servers, etc. A blacklist filter maintains a list of elements like IP addresses, domains, IP netblocks that are commonly used by phishers. Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, Kevin Tyers uses a scalable framework to test the effectiveness of browser blacklist filters. Their study concluded that most blacklist filters in mobile browsers failed to combat phishing attacks and are more vulnerable [13]. Mohsen Sharifi, Seyed Hossein Siadati, proposes a new method that will create a blacklist generator and keep a timely track of phishing website blacklists. Their techniques yield an accuracy of 91% and 100% in detecting real pages and phishing websites, respectively [14].

2) Whitelist filter: Unlike a Blacklist, Whitelist filters allow recorded website URLs, schemes, or domains to make it through to the client machine and block all other unrecorded sites. A whitelist, contrary to a blacklist, maintains a list of all legitimate websites. A. Belabed, E. Aïmeur, A. Chikh proposes a method that combines the whitelist approach with machine learning. A support vector classifier is used to filter further the websites that are not blocked by the whitelist filter [15]. Linfeng Li, Marko Helenius, and Eleni Berki conducted tests that compared the effectiveness of blacklist and whitelist anti-phishing toolbars. Their study did not find a significant difference in performance between both toolbars but encourages that toolbars be more instructive in helping users identify phishing websites [16].

3) Pattern matching filter: Checks whether or not individual tokens or sequences of data is contained within a given list of data by using a pattern matching technique. Rahamathunnisa Usuff, N. Manikandan, U.S. Kumaran, and C. Niveditha propose a method that uses pattern matching to detect phishing websites. A database of blacklist and whitelist that contains malicious URL patterns and original URL patterns is used to match with the user requested URL [17].

C. Machine Learning-Based Methods

1) Malicious domain detection: Machine Learning models are being trained to optimize their capabilities of detecting Phishing pages, one of the most common forms of cyberattacks. Nitay Hason, Amit Dvir, and Chen Hajaj propose a robust feature selection mechanism that creates better malicious domain detection models. All of the data are collected from 5000 legitimate URLs and 1350 harmful URLs. The models created are robust to different malicious abnormalities and show the effectiveness of models trained on features [18]. Hossein Shirazi, Bruhadeshwar Bezawada,

Indrakshi Ray shows concern about the large number of training features and types of datasets used and suggests that the domain name is much better and useful detecting phishing websites. Their learning model detects unknown live phishing URLs with an accuracy of 99.7% [19]. Krzysztof Lasota, Adam Kozakiewicz proposes a study that shows the similarity of different malicious domain name creations. The main task for detecting malicious behaviors was to detect similarity based on sets of domain names, URL names, and hostnames [20].

2) Email spam filtering: Emails are screened through various scoring techniques based on thousands of rules set to predict their probability of being an actual spam email. If the evaluated probability is beyond the acceptable range, then the email is blocked via the spam filter. Phishers use spam emails to direct a client to their malicious webpage and steal data. Andronicus A. Akinyelu1 and Aderemi O. Adewumi research about the effectiveness and use of random forest classifier in developing a phishing email classifier by extracting pertinent phishing email features from a dataset of 2000 phishing and ham emails. The proposed machine learning models shows a classification accuracy of 99.7% with low false positives and negatives [21]. Tushaar Gangavaraapu, C.D. Jaidhar, and Bhabesh Chanduka focus on the proper ways of extracting features from spam email content and behavior-based features, the features necessary in detecting spam emails, and on the selection of an important feature set. Their proposed machine learning model based on their selected features yields a constant accuracy of 99% in spam emails [22]. Table I illustrate the advantages and limitations of existing phishing detection researches. In Table I, we observed that most of the researches consider a small number of features and datasets. In this research, we try to overcome the limitations observed from Table I by increasing the number of features and dataset volume.

TABLE I. COMPARISON OF MACHINE LEARNING BASED PHISHING DETECTION SYSTEMS

Description	Pros	Cons	Ref.
Detects phishing attacks by using a whitelist filter.	* Pages that bypass the whitelist filter are filtered again by Support Vector Machines. * Maintains accuracy of whitelist filter by using a personalized whitelist.	* Limited dataset of 850 pages. * Unable to detect the attachment of DNS spoofs to legitimate web pages. * High False positive rate.	[23]
Implement a comment spam detection mechanism that can be used as a browser plugin and remove spam comments.	* Balances dataset by applying WEKA filters to get the best suitable features. * Spam detection classifier can accommodate new features and detect new classes of spam content.	* Does not do well with a random dataset without applying a supervised resample filter.	[24]
Proposes a machine learning-based method that can detect whether a web page exhibits phishing attacks.	* Proposed method is based on an easy to acquire feature vector that does not require additional computation.	* Only uses 10 features for detection. * Limited dataset of 1353 instances.	[25]
Uses feature selection to identify important features that categorize phishing and legitimate websites.	* Feature selection highly improves the accuracy score after implementation. * Use of feature selection reduces computational time.	* 14 features. * limited dataset (200 legitimate URL and 1400 phishing URL) * May not work properly with datasets of equal URLs of legitimate and phishing web pages.	[26]
Builds a system using machine learning that can classify websites using URLs.	* Can be used to build a rule-based system with associative rules to classify URLs.	* 9 features for each URL * All features are discrete. * Limited dataset (1353 URLs)	[27]
Proposes a learning-based aggregation analysis mechanism to decide page layout similarity, which is used to detect phishing pages.	* Automatically trains classifiers to determine web page similarity from CSS layout features, which does not require human expertise.	* Method is lightweight as it only takes one class of features, CSS structure. * Limited by the size of the dataset and distribution of samples.	[28]
This research uses a new attribute called the "domain top page similarity" to improve the efficiency of a machine learning-based phishing detection model.	* Increases f-measure and reduces the error rate. * Proves that with better features, the detection rate is much higher and can be implemented in future works.	* The model is highly dependent on the accuracy of the features.	[29]
This paper proposes a real-time anti-phishing system that uses seven classification algorithms and natural language processing-based features (NLP)	* Independence from language and third party services. * Huge dataset of legitimate and phishing data. *Real-time execution. * Can detect new websites because of NLP features.	* Machine learning-based systems cannot correctly utilize such a vast dataset.	[30]
Performs an extensive measurement of squatting phishing, where the phishing pages impersonate target brands at both the domain and content level.	* Uses features from visual analysis and optical character recognition. * Open sourced tool. * Uses evasive behaviors of phishing pages to build classifiers.	* Unable to detect phishing pages that use cloaking. * Only focuses on popular brands. * The classifier cannot be compared with other phishing tools like CANTINA and CANTINA+.	[31]
Uses features from HTML content, JavaScript code and URLs to build a classifier that can detect malicious web pages and threat types.	* Diverse features. * High accuracy score. * Highlights features that are necessary to extract.	* Limited dataset (2500 URLs) * Classifier may not do well with large datasets.	[32]

III. Phishing Website Detection

In this section, we explain our proposed data-driven phishing website detection system—the dataset obtained from the online repository of Mendeley. Parallel coordinates, pearson and shapiro ranking, and principal component analysis are used for feature extraction. We use KNN, decision trees, random forest, SVM, and logistic regression to detect phishing websites.

A. Dataset

The phishing webpage dataset contains 48 features that are obtained from the online repository of Mendeley. The total number of websites is 1000, where 5000 phishing and 5000 legitimate websites. The class label 0 indicates a phishing website and 1 a legitimate website.

B. Feature Extraction and uses

For feature extraction, we used parallel coordinates, pearson and shapiro ranking, and principal component analysis. We used parallel coordinates to visualize and analyze our dataset and PCA to reduce the dimensionality of our dataset. We have explained our features in Table I, Table III, and Table IV. In Table II, a total number of 27 lexical features are descrived like NumDots, SubdomainLevel, PathLevel, and so on. A total number of 15 host-based features are explained in Table III. In Table IV, a set of 8 correlation features are shown with data types and description.

C. Classifiers

We deploy KNN, decision tree, random forest, extra trees, SVM, and logistic regression in our system.

1) K-Nearest Neighbors (KNN): We calculated the distance using the Euclidean method from equation (1),

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + \cdots + (x_n - x'_n)^2} \quad (1)$$

Our KNN model is based on equation (2),

$$P(y = j \mid X = x) = \frac{1}{K} \sum_{i \epsilon A} I(y^{(i)} = j) \quad (2)$$

Our dataset has 48 features and a Class label where 0 indicates a phishing website, and 1 indicates a legitimate website. When given an unknown sample, KNN will first measure the distance of the unknown sample with its neighbors by using Euclidean distance. The number of neighbors that it will check will be the value of K that can be chosen by setting the value of "n_neighbors." The distances will be measured by taking in the features of the samples that are in the dataset. The majority class of the neighbors that are the closest will be then assigned to the unknown sample.

TABLE II. List of Url Features in Lexical Feature Group

Feature	Data Type	Description
NumDots	Numeric	The number of dots In the URL.
SubdomainLevel	Numeric	Determines the number of subdomain levels.
PathLevel	Numeric	Determining the level of the path in the URL.
UrlLength	Numeric	Length of each URL used in the dataset. The length contains the number of letters or symbols used to create the URL.
NumDash	Numeric	Total number of dash in a URL.
NumDashInHostname	Numeric	The number of dashes in a hostname
AtSymbol	Boolean	Total number of '@' symbol in the URL.
TildeSymbol	Boolean	Total number of tilde '~' symbol in the URL.
NumUnderscore	Numeric	Number of underscores'_' used in the URL.
NumPercent	Numeric	Total number of percent symbol present in the URL.
NumQueryComponents	Numeric	Total number of query components.
NumAmpersand	Numeric	Total number of '&' character.
NumHash	Numeric	Total number of '#' character.
NumNumericChars	Numeric	The total number of numeric characters.
NoHttps	Boolean	Check if there is a HTTPS in the URL.
RandomString	String	Set of Characters that are random.
IPAddress	Boolean	Check if the hostname of the URL uses the IP address.
DomainsInSubDomains	Boolean	Determines if TLD or CCTLD is in the subdomain of URL.
DomainsInPaths	Boolean	Determines if the website link has used TLD or CCTLD.
HttpsInHostname	Boolean	Determines if HTTPS is disorderly in the hostname of the URL.
HostnameLength	Numeric	Length of hostname which includes all the characters and symbols.
PathLength	Numeric	Length of all paths in each URL.
QueryLength	Numeric	Length of query in the URL.
DoubleSlashInPath	Boolean	Checks if there is a double slash in the path.
NumSensitiveWords	Numeric	Checks if there are any sensitive words like secure, sign in, login, etc.
EmbeddedBrandName	Boolean	Checks if there is the name of a brand in the domain.
PctExtHyperLinks	Float	Checks the percentage of external hyperlinks in the source code.

TABLE III. LIST OF URL FEATURE IN THE HOST-BASED FEATURE GROUP

Feature	Data Type	Description
PctExtResourceUrls	Float	Checks the percentage of URL external resources in the source code.
ExtFavicon	Boolean	Checks if the favicon is installed from a different hostname.
InsecureForms	Boolean	Will see if the action in forms follow the HTTPS protocol.
RelativeFormAction	Boolean	Checks if the action form contains a relative URL.
ExtFormAction	Boolean	Checks if the action form contains an external URL.
AbnormalFormAction	Boolean	Checks if the action form contains an abnormal URL.
PctNullSelfRedirectHyperlinks	Float	Check the percentage of hyperlinks that have an empty value and also if it has an auto directing value.
FrequentDomainNameMismatch	Boolean	Checks if the URL, when accessed, shows a mismatch in the frequent domain name.
FakeLinkInStatusBar	Boolean	Checks if there are any fake link in status bar that lures the user towards unsafe websites.
RightClickDisabled	Boolean	Check if the right-click option has been disabled in the URL.
PopUpWindow	Boolean	Checks if the URL contains any pop up windows when opened or accessed.
SubmitInfoToEmail	Boolean	Checks whether a URL requires you to submit your information to email.
IframeOrFrame	Boolean	Check if the given URL has used iframes or frames.
MissingTitle	Boolean	Check if there are any missing title.
ImagesOnlyInForm	Boolean	Checks if there are only images in the action form.

TABLE IV. LIST OF URL FEATURES IN CORRELATED FEATURE GROUP

Feature	Data Type	Description
SubdomainLevelRT	-1, 0, 1	Checks if the subdomain levels are correlated.
UrlLengthRT	-1, 0, 1	Checks if the URL lengths are correlated.
PctExtResourceUrlsRT	-1, 0, 1	Checks if the percentage of external URL is correlated.
AbnormalExtFormActionR	-1, 0, 1	Checks the relationship of different abnormal action forms in the URL.
ExtMetaScriptLinkRT	-1, 0, 1	Checks the correlation of meta script links
PxtExtNullSelfRedirectHyperlinksRT	-1, 0, 1	Checks the correlation of the percentage of self-directed hyperlinks.
Class_label	0, 1	Identifying the 2 classes of Phishing and Real Website.

2) Random forest: We used Gini importance to calculate a node's importance for each decision tree. This was based under the assumption that the tree is binary, and so each node has at most two children. For the elimination of branches in the tree, we used the equation (3),

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)} \quad (3)$$

For calculating the importance of each feature on a decision tree, we used the equation (4),

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k all\ nodes} ni_k} \quad (4)$$

These can be normalized afterward to a value between 0 and 1 by the equation (5),

$$normfi_i = \frac{fi_i}{\sum_{j all\ features} fi_j} \quad (5)$$

And the sum of the feature's importance value on each tree is calculated by the equation (6) and divided by the total number of trees.

$$RFfi_i = \frac{\sum_{j all\ trees} norm\ fi_{ij}}{T} \quad (6)$$

A random forest classifier consists of a large number of decision trees that work as an ensemble. At first, it will create a bootstrap dataset of size "N" that will randomly take samples from our dataset. A random forest can then use these bootstrap samples to create a tree. For example, if our training data was [a, b, c, d, e, f], we might give one of our trees the following list [a, b, b, c, f, f]. It should be noticed that both samples are of the same size, and "b" and "f" are repeated in the bootstrap dataset because we sample with replacement. After taking in the samples from the bootstrap dataset, it begins to build trees by first choosing a root node. Random forest differs from decision trees because it uses a method called Feature Randomness. This means that when it comes to choosing a root node for a random tree forest will only allow the trees to choose a root node from a subset of features. The Gini impurity is measured among these subsets of features, and the lowest score will be used as the root node, and the different subsequent nodes are chosen in the same way. After creating the trees, the random forest classifier is ready to make predictions. It will take an unknown sample from our test dataset and run the sample among all of the trees. All of the individual trees give a class prediction, and the class that has the most votes will be the class of the unknown sample. One of the main reasons random forest classifier does well with large

datasets is because it maintains diversity between models by using bootstrap aggregation and feature randomness.

3) Support vector machines: We used the equation (7) to calculate the loss function for our support vector machine,

$$min_w \lambda \left|| w ||\right|^2 + \sum_{i=1}^{n} \left(1 - y_i \langle x_i, w \rangle\right)_+ \quad (7)$$

For calculating gradients, we used the equation (8),

$$\frac{\partial}{\partial w_k} \lambda \left|| w ||\right|^2 = 2\lambda w_k$$

$$\frac{\partial}{\partial w_k}(1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0, & if\ y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & else \end{cases} \quad (8)$$

By using SVM, we plot each data item as a point in n-dimensional space (where n is the number of features and in our dataset it is 48) with the value of each feature being a value of a specific coordinate. After that SVM finds a hyperplane or a decision boundary that can properly differentiate between the classes. An optimal hyperplane is one where it has equal and maximum distance between two data points, which are considered as support vectors. SVM is very easy to apply when the data points can be easily divided by a linear line, but it is rare to find such datasets in the real world. This is where the kernel trick of SVM comes to work. One of the reasons why SVM works well with our large dataset is that it can work in infinite dimensions. The best part is that the kernel does not necessarily generate the infinite dimensions but simulates the lower dimension data so as if they are working in infinite dimensions. The kernel is very useful here because it can make a non-separable problem into a separable problem by adding more dimensions to it, and the number of dimensions depends on the number of features each sample has; some of the kernels that we found compelling are Linear Kernel, Polynomial Kernel, and the Radial Basis Function (RBF) kernel.

4) Logistic regression: Logistic regression is based on the linear regression, where a line is plotted its axes for a given dataset.

The conditional probability function we used gives a binary output for the variable Y as a function of X. Any unknown parameters in the function are estimated by maximum likelihood. The conditional probability is calculated by using equation (9).

$$\Pr(Y = 1 \mid X = x) = \log \frac{p(x)}{1-p(x)} = \beta_0 + x.\beta \quad (9)$$

We also used equation (10) for the sigmoid function,

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \quad (10)$$

Equation (11) is the cost function,

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_0(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_0(x^{(i)}))] \quad (11)$$

We calculate the gradient by using the equations (12), (13), (14), and (15).

$$J = -\frac{1}{m}[\sum_{i=1}^{m} y_i \log h_i + (1 - y_i) \log 1 - h_i] \quad (12)$$

$$\frac{\partial J}{\partial \theta_n} = -\frac{1}{m} \cdot [\sum_{i=1}^{m} \frac{y_i}{h_i} \cdot h_i^2 \cdot x_n \cdot \frac{1-h_i}{h_i} + \frac{1-y_i}{1-h_i} \cdot -h_i^2 \cdot x_n \cdot \frac{1-h_i}{h_i}] \quad (13)$$

$$\frac{\partial J}{\partial \theta_n} = -\frac{1}{m} \cdot [\sum_{i=1}^{m} x_n \cdot (1 - h_i) \cdot y_i + x_n \cdot h_i \cdot (1 - y_i)] \quad (14)$$

$$\frac{\partial J}{\partial \theta_n} = \frac{1}{m} \cdot x_i \cdot [\sum_{i=1}^{m} h_i - y_i] \quad (15)$$

IV. Result Analysis

A. ROC Curve

Now let us look at our ROC curves of different models.

Fig. 1 shows the ROC curve of the support vector machine. The X-axis indicates the false positive rate, and the Y-axis indicates the True positive rate. The AUC value for this is 0.97.

Fig. 2 shows the ROC curve of the non-uniform support vector machine. The X-axis indicates the false positive rate, and the Y-axis indicates the True positive rate. The AUC value for this is 0.96.

Fig. 3 shows are the ROC curve of the linear support vector machine. The X-axis indicates the False Positive rate, and the Y-axis indicates the True positive rate. The AUC value for this is 0.98. This is the highest and best one so far. We can see the steepness in the curve is much closer to the top-left position of the plot.

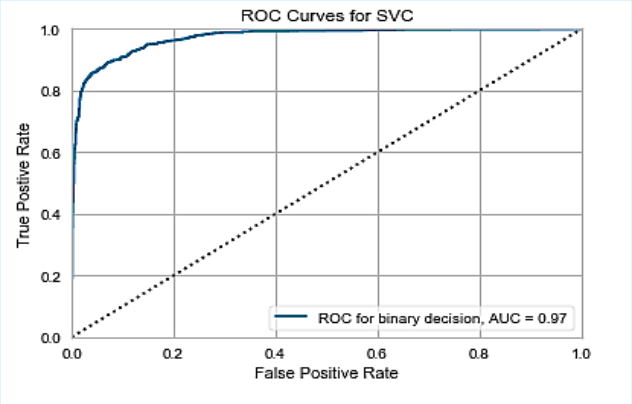

Fig. 1. Curves for SVC.

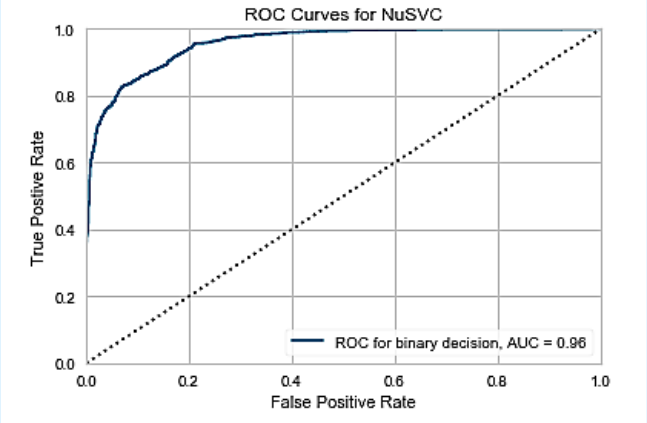

Fig. 2. ROC Curves for NuSVC.

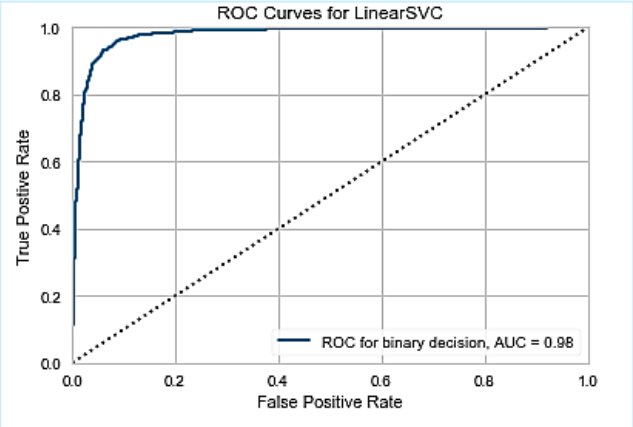

Fig. 3. ROC Curves for LinearSVC.

Fig. 4 shows the ROC curve of KNN. The X-axis indicates the false positive rate, and the Y-axis indicates the True positive rate. Here the AUC of class 0 (phishing website) is 0.94, and class 1 (real website) is 0.94. The AUC of the macro and micro average of the ROC curve is also 0.94.

Fig. 5 shows the ROC curve of Logistic Regression. The X-axis indicates the False Positive rate, and the Y-axis indicates the True positive rate. Here the AUC of class 0 (phishing website) is 0.96, and class 1 (real website) is 0.96. The AUC of the macro and micro average of the ROC curve is also 0.96.

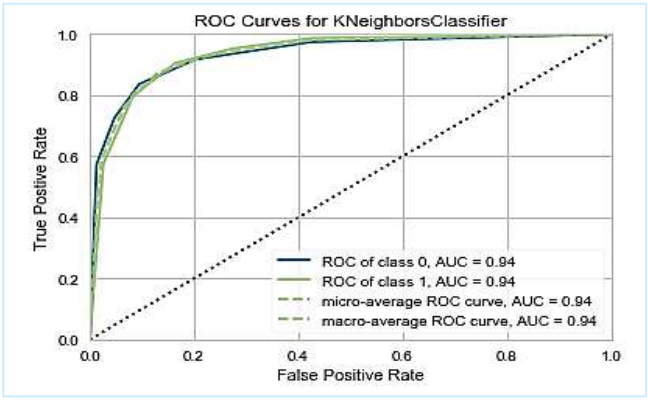

Fig. 4. ROC Curves for KNeighborsClassifier.

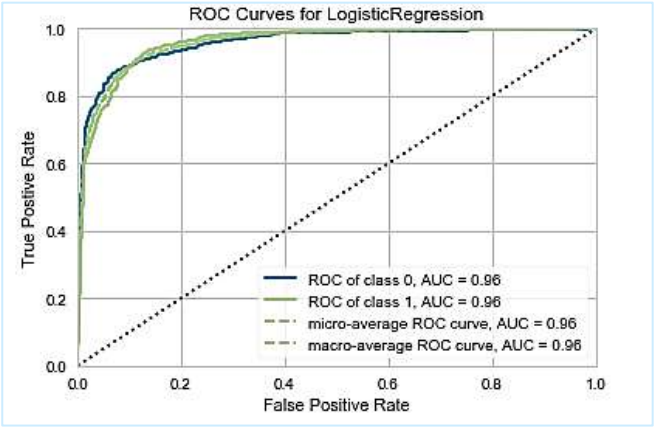

Fig. 5. ROC Curves for LogisticRegression.

Fig. 6 shows the ROC curve of stochastic gradient descent (SGD). The X-axis indicates the false positive rate, and the Y-axis indicates the True positive rate. The AUC value for this is 0.97.

Fig. 7 shows the ROC curve of logistic regressionCV. The X-axis indicates the false positive rate, and the Y-axis indicates the True positive rate. Here the AUC of class 0 (phishing website) is 0.98, and class 1 (real website) is 0.98. The AUC of the macro and micro average of the ROC curve is also 0.98.

Fig. 8 shows the ROC curve of the bagging classifier. The X-axis indicates the false positive rate, and the Y-axis indicates the True positive rate. Here the AUC of class 0 (phishing website) is 0.99, and class 1 (real website) is 0.99. The AUC of the macro and micro average of the ROC curve is also 0.99. This is the best ROC curve so far.

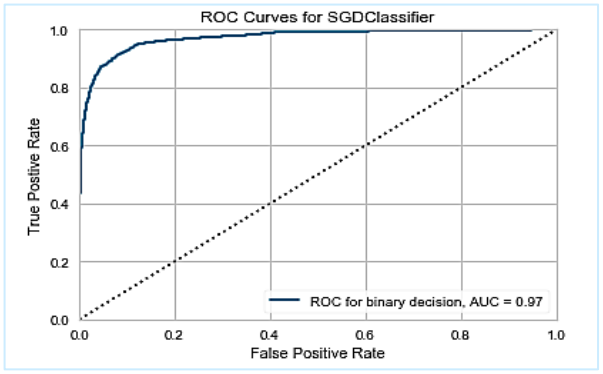

Fig. 6. ROC Curves for SGD Classifier.

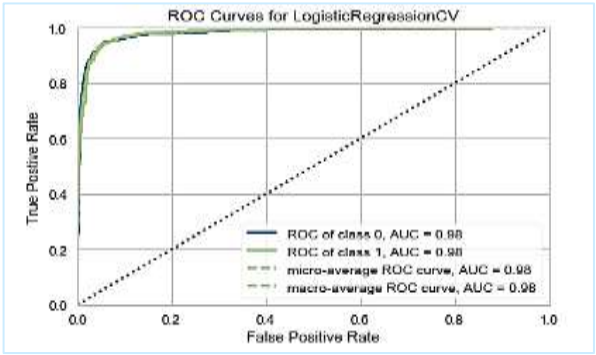

Fig. 7. ROC Curves for Logistic Regression CV.

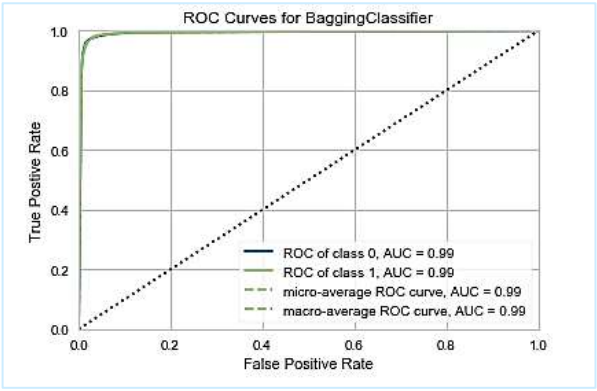

Fig. 8. ROC Curves for Bagging Classifier.

Fig. 9 shows the ROC curve of the extra trees classifier. The X-axis indicates the False Positive rate, and the Y-axis indicates the True positive rate. Here the AUC of class 0 (phishing website) is 1.00, and class 1 (real website) is 1.00. The AUC of the macro and micro average of the ROC curve is also 1.00. This is the best ROC curve so far. We can see that the steepness of the curve is at the most top left corner.

Fig. 10 shows the ROC curve of the random forest classifier. The X-axis indicates the False Positive rate, and the Y-axis indicates the True positive rate. Here the AUC of class 0 (phishing website) is 1.00, and class 1 (real website) is 1.00. The AUC of the macro and micro average of the ROC curve is also 1.00. This is the same as the Extra Trees classifier. We can see that the steepness of the curve is at the most top left corner. Hence it can be said that the extra trees classifier and random trees classifier has the best ROC curve.

B. Discrimination Threshold

Let us look at the discrimination threshold of our models.

Fig. 11 shows the threshold plot for the support vector machine. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.03. For this threshold, we see that the precision, recall, and f1 score are approximately around 0.89.

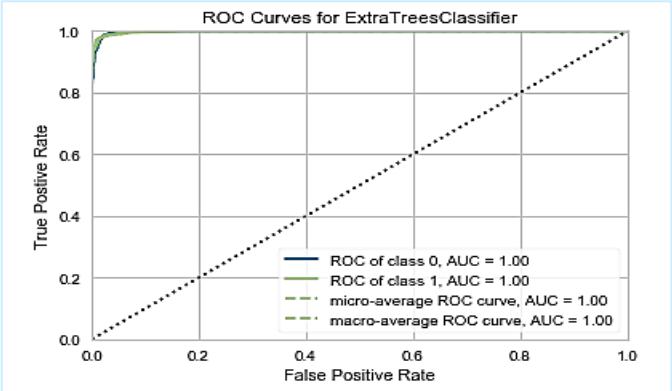

Fig. 9. ROC Curves for ExtraTrees Classifier.

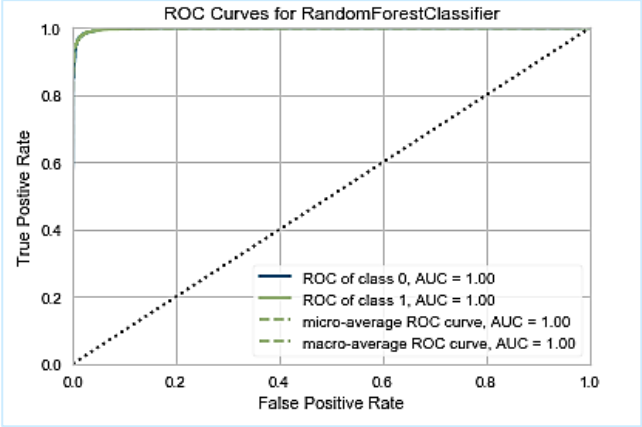

Fig. 10. ROC Curves for Random Forest Classifier.

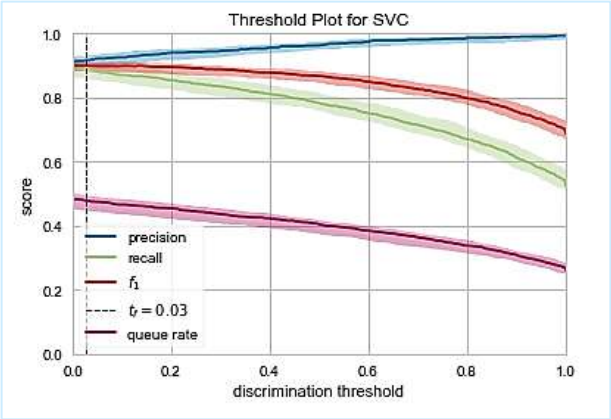

Fig. 11. Threshold Plot for SVC.

Fig. 12 shows the threshold plot for the non-uniform support vector machine. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.00. For this threshold, we see that the precision, recall, and f1 score are approximately around 0.86.

Fig. 13 shows the threshold plot for the linear support vector machine. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.05. For this threshold, we see that the precision, recall, and f1 score are approximately around 0.9.

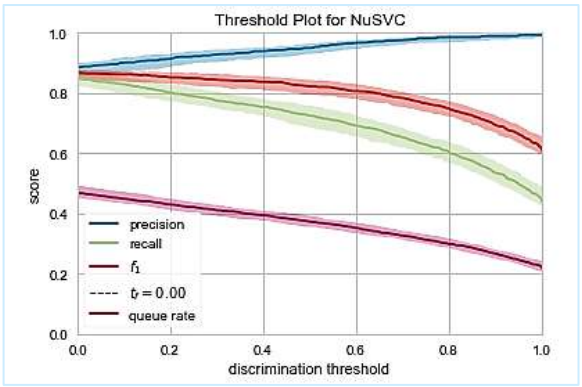

Fig. 12. Threshold Plot for NuSVC.

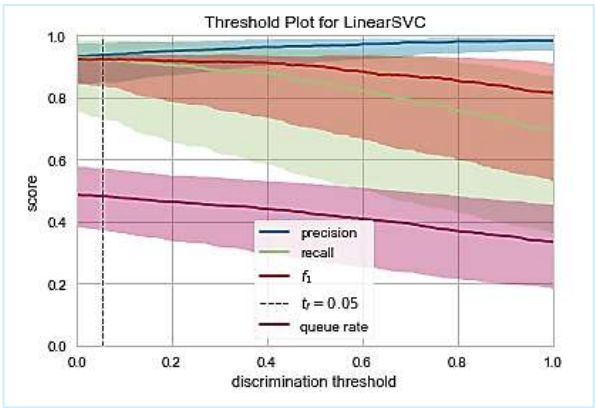

Fig. 13. Threshold Plot for Linear SVC.

Fig. 14 shows the threshold plot for KNN. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.50. For this threshold, we see that the precision, recall, and f1 score are approximately 0.82 to 0.89.

Fig. 15 shows the threshold plot for logistic regression. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.46. For this threshold, we see that the precision, recall, and f1 score are approximately 0.85 to 0.9.

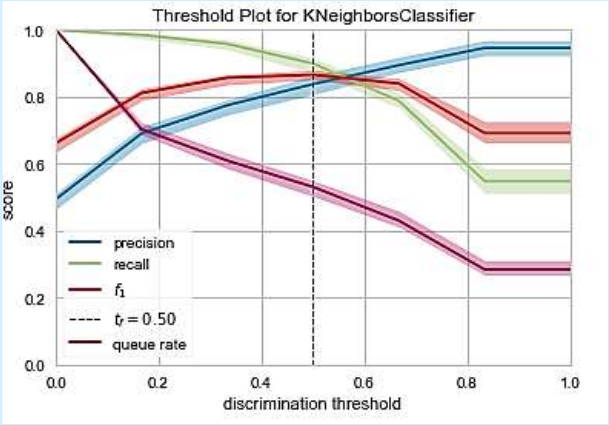

Fig. 14. Threshold Plot for KNeighbors Classifier.

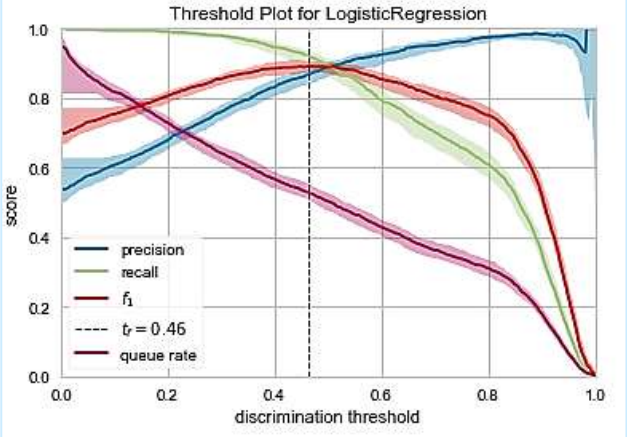

Fig. 15. Threshold Plot for Logistic Regression.

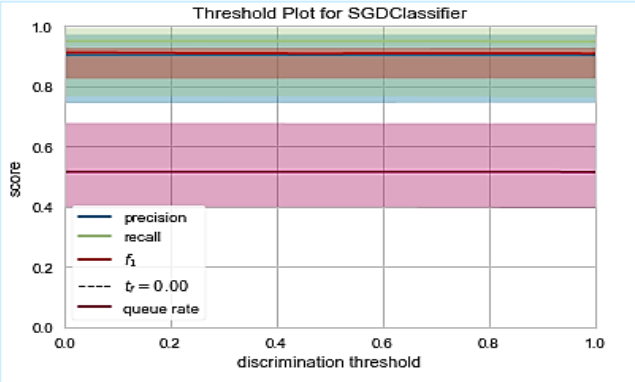

Fig. 16. Threshold Plot for SGD Classifier.

Fig. 16 shows the threshold plot for stochastic gradient descent (SGD). On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.00. For this threshold, we see that the precision, recall, and f1 score are approximately 0.8 to 0.9.

Fig. 17 shows the threshold plot for logistic regressionCV. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.58. For this threshold, we see that the precision, recall, and f1 score are approximately around 0.95.

Fig. 18 shows the threshold plot for Bagging Classifier. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.56. For this threshold, we see that the precision, recall, and f1 score are approximately around 0.98.

Fig. 19 shows the threshold plot for random forest classifier. On the X-axis, we have the discrimination threshold, and on the Y-axis, we have the score. Here we see that the discrimination threshold for this is 0.48. For this threshold, we see that the precision, recall, and f1 score are approximately around 0.99.

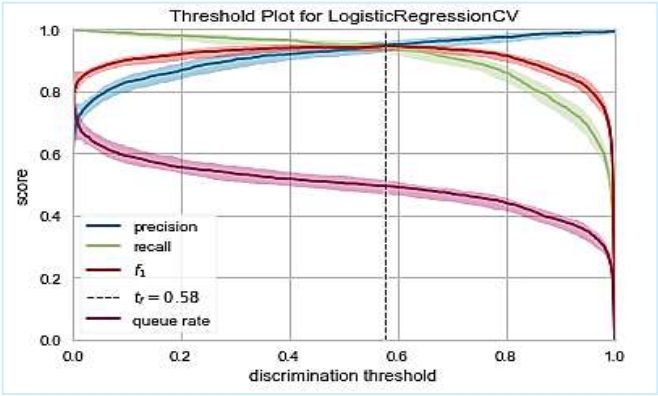

Fig. 17. Threshold Plot for Logistic Regression CV.

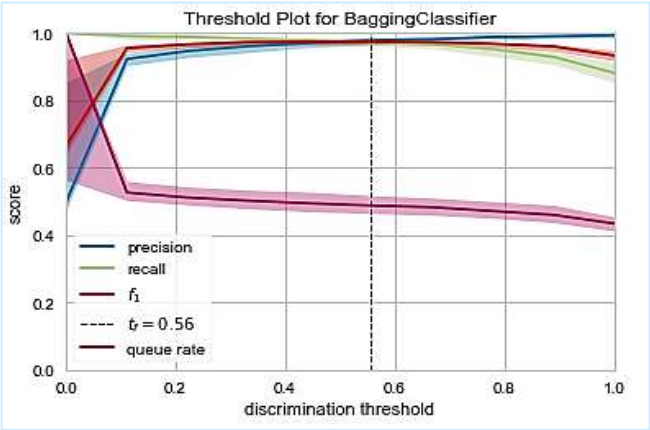

Fig. 18. Threshold Plot for Bagging Classifier.

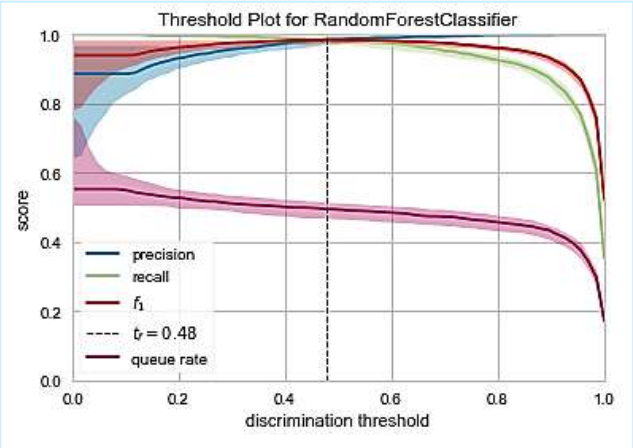

Fig. 19. Threshold Plot for Random Forest Classifier.

V. Conclusion

Our work analyses different machine learning techniques when implemented over a dataset of features regarding websites and their corresponding details that may prove useful to detect a possible phishing website. This document aims to be useful to its readers to provide a conclusive analysis of these methods and to verify our observations regarding the random forest classifier's optimal performance. F1 score for the random forest is 0.99, which indicate that both false positive and false negative rate are in the satisfactory level. The graphs and details we have added to the document aim to help others carry out further experimentation to conclude our work. And we, ourselves, also intend to carry on our work with further modifications to the dataset and applying other machine learning techniques with modified parameters to hopefully open more possibilities in the hopes of improving the world's defenses against the cyber attackers out there. The internet is both fantastic and dangerous. And our work's main objective is to help minimize the danger by addressing a pervasive security issue of the modern world. In this paper, we apply basic machine learning algorithms. In the future, we will deploy deep learning techniques like multilayer perception and artificial neural networks to improve the performance of the detection system.

References

[1] H. Alqahtani et al., "Cyber Intrusion Detection Using Machine Learning Classification Techniques," in Computing Science, Communication and Security, Singapore, 2020, pp. 121-131: Springer Singapore.

[2] T. Bukth and S. S. Huda, The soft threat: The story of the Bangladesh bank reserve heist. SAGE Publications: SAGE Business Cases Originals, 2017.

[3] P. Black, I. Gondal, R. J. C. Layton, and Security, "A survey of similarities in banking malware behaviours," vol. 77, pp. 756-772, 2018.

[4] S. Hossain, A. Abtahee, I. Kashem, M. M. Hoque, and I. H. Sarker, "Crime Prediction Using Spatio-Temporal Data," in Computing Science, Communication and Security, Singapore, 2020, pp. 277-289: Springer Singapore.

[5] S. Hossain, et al., "A Belief Rule Based Expert System to Predict Student Performance under Uncertainty," in 2019 22nd International Conference on Computer and Information Technology (ICCIT), 2019, pp. 1-6.

[6] S. Hossain, D. Sarma, R. J. Chakma, W. Alam, M. M. Hoque, and I. H. Sarker, "A Rule-Based Expert System to Assess Coronary Artery Disease Under Uncertainty," in Computing Science, Communication and Security, Singapore, 2020, pp. 143-159: Springer Singapore.

[7] V. Shreeram, M. Suban, P. Shanthi and K. Manjula, "Anti-phishing detection of phishing attacks using genetic algorithm," 2010 International Conference on Communication Control and Computing Technologies, Ramanathapuram, 2010, pp. 447-450, doi: 10.1109/ICCCCT.2010.5670593.

[8] H. Huang, J. Tan and L. Liu, "Countermeasure Techniques for Deceptive Phishing Attack," 2009 International Conference on New Trends in Information and Service Science, Beijing, 2009, pp. 636-641, doi: 10.1109/NISS.2009.80.

[9] M. N. Feroz and S. Mengel, "Phishing URL Detection Using URL Ranking," 2015 IEEE International Congress on Big Data, New York, NY, 2015, pp. 635-638, doi: 10.1109/BigDataCongress.2015.97.

[10] S. Abu-Nimeh and S. Nair, "Bypassing Security Toolbars and Phishing Filters via DNS Poisoning," IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference, New Orleans, LO, 2008, pp. 1-6, doi: 10.1109/GLOCOM.2008.ECP.386.

[11] Erkkila, J. "Why we fall for phishing." Proceedings of the SIGCHI conference on Human Factors in Computing Systems CHI 2011. ACM, 2011.

[12] Khan, Ahmad Alamgir. "Preventing phishing attacks using one time password and user machine identification." arXiv preprint arXiv:1305.2704 (2013).

[13] A. Oest, Y. Safaei, A. Doupé, G. Ahn, B. Wardman and K. Tyers, "PhishFarm: A Scalable Framework for Measuring the Effectiveness of Evasion Techniques against Browser Phishing Blacklists," 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2019, pp. 1344-1361, doi: 10.1109/SP.2019.00049.

[14] M. Sharifi and S. H. Siadati, "A phishing sites blacklist generator," 2008 IEEE/ACS International Conference on Computer Systems and Applications, Doha, 2008, pp. 840-843, doi: 10.1109/AICCSA.2008.4493625.

[15] A. Belabed, E. Aïmeur and A. Chikh, "A Personalized Whitelist Approach for Phishing Webpage Detection," 2012 Seventh International Conference on Availability, Reliability and Security, Prague, 2012, pp. 249-254, doi: 10.1109/ARES.2012.54.

[16] Li, Linfeng, Marko Helenius, and Eleni Berki. "A usability test of whitelist and blacklist-based anti-phishing application." Proceeding of the 16th International Academic MindTrek Conference. 2012.

[17] Usuff, Rahamathunnisa & Manikandan, N. & Kumaran, US & Niveditha, C.. (2017). Preventing from phishing attack by implementing url pattern matching technique in web. International Journal of Civil Engineering and Technology. 8. 1200-1208.

[18] Hason N., Dvir A., Hajaj C. (2020) Robust Malicious Domain Detection. In: Dolev S., Kolesnikov V., Lodha S., Weiss G. (eds) Cyber Security Cryptography and Machine Learning. CSCML 2020. Lecture Notes in Computer Science, vol 12161. Springer, Cham. https://doi.org/10.1007/978-3-030-49785-9_4.

[19] Hossein Shirazi, Bruhadeshwar Bezawada, and Indrakshi Ray. 2018. "Kn0w Thy Doma1n Name": Unbiased Phishing Detection Using Domain Name Based Features. In Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies (SACMAT '18). Association for Computing Machinery, New York, NY, USA, 69–75. DOI:https://doi.org/10.1145/3205977.3205992.

[20] Lasota K., Kozakiewicz A. (2011) Analysis of the Similarities in Malicious DNS Domain Names. In: Lee C., Seigneur JM., Park J.J., Wagner R.R. (eds) Secure and Trust Computing, Data Management, and Applications. STA 2011. Communications in Computer and Information Science, vol 187. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22365-5_1.

[21] Akinyelu, Andronicus A., and Aderemi O. Adewumi. "Classification of phishing email using random forest machine learning technique." Journal of Applied Mathematics 2014 (2014).

[22] Gangavarapu, T., Jaidhar, C.D. & Chanduka, B. Applicability of machine learning in spam and phishing email filtering: review and approaches. Artif Intell Rev (2020). https://doi.org/10.1007/s10462-020-09814-9.

[23] A. Belabed, E. Aïmeur and A. Chikh, "A Personalized Whitelist Approach for Phishing Webpage Detection," 2012 Seventh International Conference on Availability, Reliability and Security, Prague, 2012, pp. 249-254, doi: 10.1109/ARES.2012.54.

[24] Alsaleh, M., Alarifi, A., Al-Quayed, F., & Al-Salman, A. (2015). Combating Comment Spam with Machine Learning Approaches. 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). doi:10.1109/icmla.2015.192.

[25] Cuzzocrea, A., Martinelli, F., & Mercaldo, F. (2018). Applying Machine Learning Techniques to Detect and Analyze Web Phishing Attacks. Proceedings of the 20th International Conference on Information Integration and Web-Based Applications & Services - iiWAS2018. doi:10.1145/3282373.3282422.

[26] Jeeva, S. Carolin, and Elijah Blessing Rajsingh. "Phishing URL detection-based feature selection to classifiers." International Journal of Electronic Security and Digital Forensics 9.2 (2017): 116-131.

[27] Kulkarni, Arun D., and Leonard L. Brown III. "Phishing websites detection using machine learning." (2019).

[28] Mao, Jian, et al. "Phishing page detection via learning classifiers from page layout feature." EURASIP Journal on Wireless Communications and Networking 2019.1 (2019): 43.

[29] N. Sanglerdsinlapachai and A. Rungsawang, "Using Domain Top-page Similarity Feature in Machine Learning-Based Web Phishing Detection," 2010 Third International Conference on Knowledge Discovery and Data Mining, Phuket, 2010, pp. 187-190, doi: 10.1109/WKDD.2010.108.

[30] Sahingoz, Ozgur Koray, et al. "Machine learning based phishing detection from URLs." Expert Systems with Applications 117 (2019): 345-357.

[31] Ke Tian, Steve T. K. Jan, Hang Hu, Danfeng Yao, and Gang Wang. 2018. Needle in a Haystack: Tracking Down Elite Phishing Domains in the Wild. In Proceedings of the Internet Measurement Conference 2018 (IMC '18). Association for Computing Machinery, New York, NY, USA, 429–442. DOI: https://doi.org/10.1145/3278532.3278569.

[32] T. Yue, J. Sun and H. Chen, "Fine-Grained Mining and Classification of Malicious Web Pages," 2013 Fourth International Conference on Digital Manufacturing & Automation, Qingdao, 2013, pp. 616-619, doi: 10.1109/ICDMA.2013.145.