Modelling Enhanced Phishing detection using XGBoost

[Data Security and Privacy]

MSc Internship

Cybersecurity

Nishant Nityanand Naik

19138342

School of Computing

National College of Ireland

Supervisor- Ross Spelman

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Nishant Nityanand Naik
Student ID:	19138342
Programme:	MSc CyberSecurity
Year:	2020-2021
Module:	MSc Internship
Supervisor:	Ross Spelman
Submission Due Date:	16-08-2021
Project Title:	Modelling Enhanced Phishing Detection using XGBoost
Word Count:	7064
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on NORMA the National College of Ireland's Institutional Repository for consultation.

Signature:	Nishant Nityanand Naik
Date:	15th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	Q
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	Q
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	Q

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Modelling Enhanced Phishing detection using XGBoost

[Data Security and Privacy]

Abstract

In today's society, where everything is digitized, computers have taken over majority of human activities. Machines have a tendency to execute all of the tasks that humans were capable of, but with a greater load and in a shorter length of time. While there are advantages, the disadvantages overshadow the advantages. Phishing attacks have taken on a new face because to the internet. Phishing prevention has gone a long way as the art of phishing assaults has progressed. Adapting Machine Learning algorithms makes a significant impact in identifying and blocking phishing assaults, which would otherwise be impossible for the human brain to recognize and prevent. This paper proposes a model which is expected to give desired result. The recommended model uses complex Extreme Gradient Boost (XGBoost) algorithm to identify Phishing URLs with extreme accuracy. The result is compared with other ML algorithms like Decision Tree, Random Forest, Multilayer Perceptron, and Support Vector Machines for better analysis.

Keywords- Phishing, Machine Learning, XGBoost, URL

1. Introduction

Phishing is an art of obtaining desired consequences from the victims through socially engineered messages transferred via electronically communicated channels. The attacker may persuade the victims to obtain personal confidential credentials or may even inject the victim's system with a malware eventually benefiting self. According to H. Berghel [1], The first ever notation of phishing was proposed in 1987. A detailed technical paper was presented by Jerry Felix and Chris Hauck at the international HP Users Group, Interex.

Almost a decade later in 1996, the first reported phishing attack was made. One of the top internet providers of that time America Online (AOL) incurred attacks where the phishers created fake AOL accounts and used them to trick users. Further the attackers disguised as employees and used them get credentials from the users. Nothing much changed thereafter in the approach of phishing attacks. In the early 2000's phishing attack began targeting online payment system on E-Gold. End users being the weakest link in the chain, phishers made use of that and entered every domain and sectors which is present on the internet including banks, websites, E-commerce, shopping, social networking sites etc which resulted the losses in terms of billions. The introduction of cryptocurrency made the trade off more secure for the phishers. In the last decade the scale of attack has taken an exponential spike commencement with extraction of RSA tokens to intrude into US, South Korean and other countries. In the year 2020 Anti-Phishing Working Group (APWG) published a report showing there was 70% growth in Phishing attacks hosted on HTTPS timing from the first quarter of 2017 to the second quarter of 2020 making phishing attacks using SSL/TSL with 77.6% as highest [2]. Phishing also entered OTT platforms like

Netflix, Amazon prime, etc eventually leading to loss of innumerable loss of subscriber's data. Lately during the time of global pandemic, the healthcare sector and the education sector experienced huge loss with Ransomware attacks by using simple phishing techniques [3][4].

The versatility in the phishing attacks is vast and hence even the detecting technique. There are number recognition methods such as black list and whitelist-based approach, image-based approach, Machine Learning approach, Fuzzy rule-based approach, and cantina-based approach. With various Machine Learning (ML) algorithms in the present world, it becomes evident on proving which algorithm achieves finest performance without hindering the quality of the result. Present approach is a step forward towards this ever-learning progression and thriving to contribute some valuable information to this field. This paper uses XGBoost i: e decision tree based ensembled machine learning algorithm to enhance the predictive model in detecting phishing websites from a benign one. To evaluate the predictive performance, this model is compared to other models like Decision Tree, Random Forest, Deep Learning, Autoencoder Neural Network and Support Vector Machine. Performance evaluation is done from around ten thousand open-source datasets available on the web.

2. Background Survey and Literature Review

2.1. Background Survey

Phishing attacks can be categorized into eleven types which are Web Trojan, Session Hijacking, Key logger, and Screen capturing, Malware-Based, Deceptive, Host File Poisoning, Data theft, DNS based, Content Injection, Man in the middle, Search Engine. But in a general sense Phishing can be divided primarily into Web Based and Exploit Based phishing where web-based phishing makes the copy of legitimate site in order to trick the user to provide their credentials and exploit based phishing aims on discovering existing vulnerabilities to inject malicious scripts into the host. Paper [5] [6] discuss the various types of phishing attacks, its anatomy and several tactics to mitigate them where the papers brief on all types of mitigation approaches and evaluates their weaknesses. As explained in [5, Tab.VII] and [6, Tab.1], taking into consideration all the criteria, the papers propose modern Machine Learning (ML) algorithms outperforms other detection techniques. Paper [7] emphases the detection approach in a different manner where equal importance is given to User Education and software utilization. A hybrid tactic is applied in software utilization which involves Listing (Black and Whitelisting) alongside Search engine-based approach is done. But the paper lacks to compare the performance of this approach to other methods. In addition, the hybrid method put forth in this paper does not tackle zero-day incident which is a crucial in present day. Comparably, paper [8] is also with the same opinion that both user education and Software Enhancement plays a equal role in mitigating phishing attacks. This paper is purely Literature survey based and provides detailed elaboration on performance classification of detection software like PhishNet, PhishGuard, PhishWish etc with their approach. As illustrated in [8, table.III, p.2115], ML generates promising results with less false positive rate and high accuracy.

2.2 Literature Review

In [9] URL phishing detection is staged using Logistic Regression (LR) with the use of bigrams and host-based features making it straightforward to identifying extensive URLs belonging to the same server. The simplicity of LR makes it difficult to identify complicated URLs. This can be rectified by drawing complex features from the basic ones. Also, this model fails when the data relationships are nonlinear and complex. Missing data also adds a step towards its drawbacks. Static assumptions before training restricts the overall performance. Random Forest (RF) model proposed by Abdulhamit et al. [10] gives better results in classification accuracy, AUC value and F-measures also overcoming high positive errors which may lead to categorising benign websites and phishing. This also adds up to neglecting external dependency and third-party DNS lookups aiding to consistency, high speed, and accuracy. A hybrid classifier of improved RF model and back propagation proposed by Smita et al. [11] shows considerable improvement in the accuracy of results from 87.34% to 97.36%. Moreover, [11] resolves the drawbacks of [12] based on wrapper feature selection which results in time delay and requires extra computational overhead. RF is successfully able to handle huge data and estimate the absent data but leads to creating a generalization error during forest building. However, because of the random character of the forest-building process and the difficulty of interpreting the final model and subsequent findings due to the presence of numerous independent decision trees, RF has a significant downside.

[13] Uses Artificial Neural Network (ANN) with Feature selection and URL transient nature to detect pharming and phishing. The accuracy and precision shows 98.77% and 97.56%. There’s no comparison of this model to others, hence it’s tricky to weigh the results to learn if this methodology is superior to others. On the other hand [14] uses Particle Swarm Optimization (PSO) to develop the results confirming ANN_PSO is better than Back Propagation Neural Networks (BPNN) contemplating Root Mean Square Error (RMSE) and accuracy. The conclusion rates with 97.82% and 95.57 giving ANN_PSO upper hand. To achieve the best results, the study employs various activation function pairs, learning ratios, and other techniques, concluding as it proposes. Nonlinearity in the interconnections provides the NN with computational power, allowing it to compete in learning ability. Because of the customary numerous local minima and delicate regularization, the model requires some expertise even with competitive learning capacity.

Saad and Tariq [15] use Multilayer Perceptron (MLP) method applying 31 features on two hidden layers and 100 neurons. The results are tallied on the basis on Accuracy, Precision, Recall, and F1 score. The findings portray MPD outperforms SVM, DT, RM topping 0.9665, 0.9665, 0.9665, 0.9665, 0.9665 on the mentioned parameters. [16] uses 10 features plus one hidden layer to feed the MLP and assess the outcomes to other NN classifiers based on accuracy and F score. The end results show that MLP has high ranking accuracy with 98.5%.

Saeed et al. [17] compared the predictive accuracy of six classifiers on phishing emails and comes up with considerable conclusion that false positive should be respected more as they can be expensive compared to other factors. The result showed that Random Forest (RF) outperformed others with the error rate of 7.72% but failed with highest false positive rate of 8.29%. The need for decreasing the false positive rate is necessary by adding more features or finding other cost effective yet precise algorithm. Marcos et al. proposed two studies [18][19] in which xgboost is used to detect malicious domains. Passive DNS servers are also highlighted in the article as data sources for performing the action. The suggested approach does not include lexical features and instead rely on DNS traffic to determine its authenticity. While employing lexical characteristics has been shown to improve results, this research achieves equivalent results without them. To balance classes, XGBoost is utilized as a classification algorithm, and the under-sampling approach is employed for training. This paper's methodology includes collecting and extracting datasets and subsets, data labelling with blocklists and allow lists, organizing the data with the XGBoost classifier model, and assessing the results. The algorithm's parameter values were determined via Bayesian optimization with 21 beginning points and 5 iterations. The collected findings demonstrate that XGBoost has a higher average AUC than Decision Tree and

Random Forest, with a value of 0.976. Gualberto et al. [20] use XGBoost to detect email phishing attacks. The paper proposes two methods in which method 1 is based on Document-Term Metrix without Feature Reduction Technique and method 2- is based on Feature Extraction [20, p. 12] . The data is then fed into various algorithms to see which one performs best. Texts from an email's body are extracted (parsed) and sent to pre-processing and DTM. It is then passed through a series of algorithms to obtain the output. The best results were obtained in six of the seven variations of method 2, with the use of the XGBoost algorithm with an average score of 99.65%, which demonstrates a pattern of good scores for its use, based on the appropriate configuration of its hyperparameters according to the proposed scheme for the task. Analysing the results revealed that XGBoost outperformed the others in terms of efficiency.

As previously stated, the proposed model [19] uses only DNS data, which limits the model's feasibility. This can be improved by adapting the model to multiclass classification in order to identify malicious activity in the domain. It is also clear whether or not the addition of lexical features to the proposed model will further tune the results. [20] makes use of distinct email features (quite a different approach than the traditional) to increase precision. The paper focuses solely on the body of the email, ignoring other features such as the IP address, header, and links associated with it. Similarly, word embedding can be used in conjunction with pre-processing to extract syntactic and semantic similarities.

With a careful examination of the articles, it is obvious that Random Forest tends to overfit datasets due to noisy classification. To overcome this difficulty, XGboost uses column sampling, row sampling, shrinkage parameters, and a regularization term. PNN has the same problem in that it requires a lot of memory to store the network and its execution is slow, but XGBOOST outperforms traditional gradient boosting implementations in many ways. Better regularization, which avoids overfitting, high speed, and output due to the parallel nature of tree construction, adaptability due to costume optimization targets and assessment criteria, and integrated methods for handling missing information are just a few of the advantages. As a result, the proposed method employs the XGBoost algorithm to advance phishing detection websites based on URL to determine whether or not it is successful in achieving the desired result.

3. Research Methodology

Research methodology is defined as the cohesive framework based on study, research and experience which guides the researcher to proceed further without deviating from the roadmap. In other words, methodology is the skeleton of any research on which the research further builds. The proposed methodology is inspired from building a Secure Software Development Lifecycle [21] where key aspects like best code practices, architecture overview and reviewed codes are used to build our prototype. In addition, [22] gives a good Software Development Model called Security aware Spiral (SaS) which can applied while further development of this prototype. This model allows enterprises to provide changes in small increments based on continuous iterations resulting in cost effectiveness and fewer complications. Fig.1 shows the proposed methodology that has four phases which continuously aid to build a steady and sustainable version.

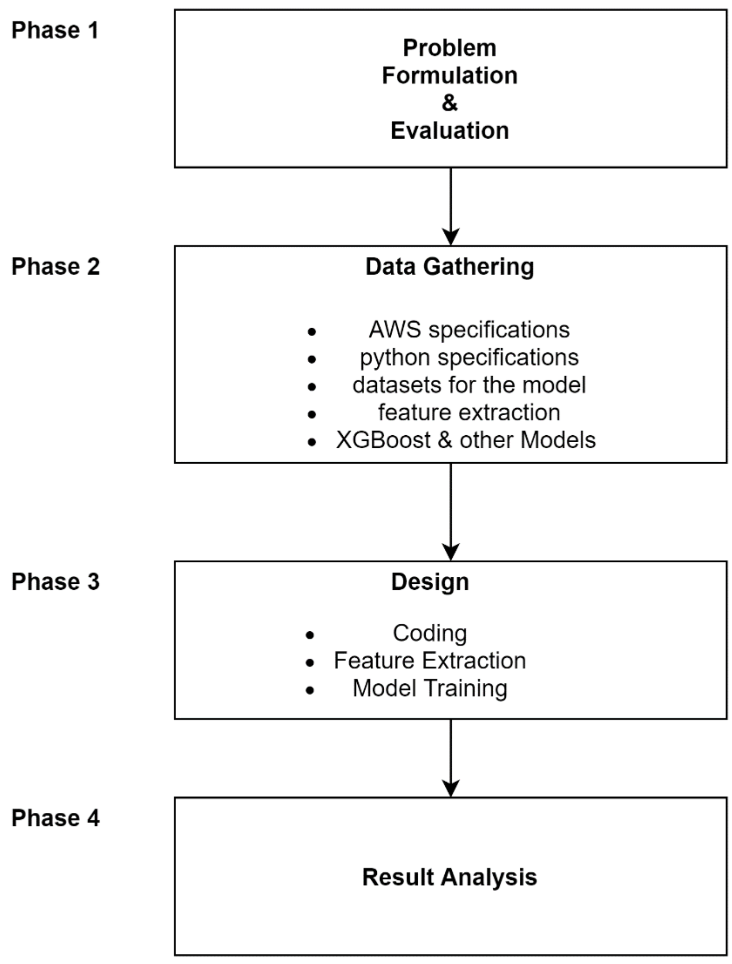

Fig.1 Research Methodology Architecture

3.1 Problem Formulation and Evaluation

A thorough study was made to explore the current technological flaws related to Cybersecurity then decide on the domain and area in which research must be conducted. Assessments [1]-[4] gave sufficient proof that Phishing is on of the most frequent and largescale attacks which happen on the industry in the current time plus monitory loss is the highest from phishing. Artificial Intelligence being the smartest machinery from the humankind, was preferred to proceed further. In this manner, a number of related papers were thoroughly evaluated to establish the research gap and prepare the research problem. Special attention was given to tabulate the approach proposed by various algorithms and their drawbacks. Also feature selection, clustering, classification, and accuracy was considered to further enhance this work. Once Problem Formulation was done, AWS and Python was chosen as the platform because of its elasticity and versatility of already available features.

3.2 Data Gathering

Data gathering or the Requirements stage is the initial process after finalizing the research topic. Even though this is the early stage of building, it’s also the crucial and time-consuming phase. This paper devotes special attention to this aspect of the project because the module's major goal is to study and utilize cutting-edge technology. The current Data Gathering stage is divided into

AWS, Python, Dataset description, Feature Extraction, and models for the experiment. Here each aspect is carefully studied and all the necessary information which is needed for the project is gathered so that the next step (Design) becomes easy. Before deciding each specifics of data gathering, a careful study is made pointing out the advantages of them and then opted to select them.

AWS

Amazon Web Services (AWS) is a secure cloud services platform that helps businesses develop and flourish by providing compute power, database storage, content delivery, and other features. The reason for using AWS is because of its dynamic nature and numerous elements which are already available and there is no issue with the compatibility issues with the system. With enough knowledge about the platform, it becomes quite easy to use and it provides application, ISV's etc to host any application. AWS is quite flexible to select any OS, programming language, web application platforms, database, memory with relatively lower cost price and options to scale up at any point of time with relatively high performance. Also, AWS is secure using end-to-end encryption [23]. A Comparative analysis was made to choose the Operating System for the project and Linux was a clear-cut winner contrasted to Windows. The advantages include superior community support, customization options, reliability, compatibility to all the programming languages, and most importantly privacy and security [24].

Python

Python is certainly the best language for ML because its easy to understand making it quick for data validation and error free. The huge library echo system makes it easy for the developers to perform complex tasks without complicated coding terminologies. The Flexibility of python makes it possible to choose multiple programming styles or merge python with other languages to obtain the result. Platform independence, good community support, good visualization options etc are some of the parameters that makes it optimum for this project [25]. Hence, python is used as the coding language.

Datasets

Datasets are the raw materials required for any decision-making programs. On performing necessary operations on them any conclusion can be made. The availability of a proper dataset is vital and makes the task effortless. There are 2 sets of data required for this project. One is the legitimate URL dataset and the other one is the Phishing dataset. Legitimate dataset is taken from the University of New Brunswick and the later is taken from phishtank which are both opensource.

Feature Extraction

Criteria based on which any decision is made becomes important as the output of these decisions depend solely on the benchmarks put forth. Necessary investigation is made as to which are the features which perfectly distinguish between a benign and phishing URL. These are selected, studied, and used so that the machine can clearly judge, and ideal results are obtained.

Models

Different models have different approach to a proposed problem. The approaches undertaken determines the quality of the result. To ascertain if the preferred model is better, it has to be compared to the other models. Hence, approaches of all the models considered are studied in order to understand the results better and come to a common conclusion.

3.3 Design

After the necessary data is gathered and studied, its time to build the model. The design part of the presented project consists of building the architecture of the model, developing a systematic yet secure codes, and model training. As python is being used for the entire project, essential libraries mainly used for data science are imported and the codes are built or inspired from the web. 17 features are extracted from 4 main categories and embedded in the machine. These features are used to train the various models used in the project. Later, based on the features, our models are trained to classify the data into novel or illegitimate. Finally, the test data is fed into the prototypes and scores are noted down to draw a conclusion.

3.4 Evaluation

The results obtained from the test data is collected and evaluated based on Accuracy, Gain, F-score, Recall. The score from each model is tabulated and matched with the others to see if the proposed model gives better results or not.

This is the overall approach of the project and the details of each pieces are explained in the Design Specifications part.

4. Design Specifications

This section displays the recommended methodology to the research issue and tries to overcome the shortcomings which were highlighted in the previous section. It also includes the specifications on the architectural project plan, the working of the model, suitable equations and diagrams to express the model and supporting evidences to back up the conclusion.

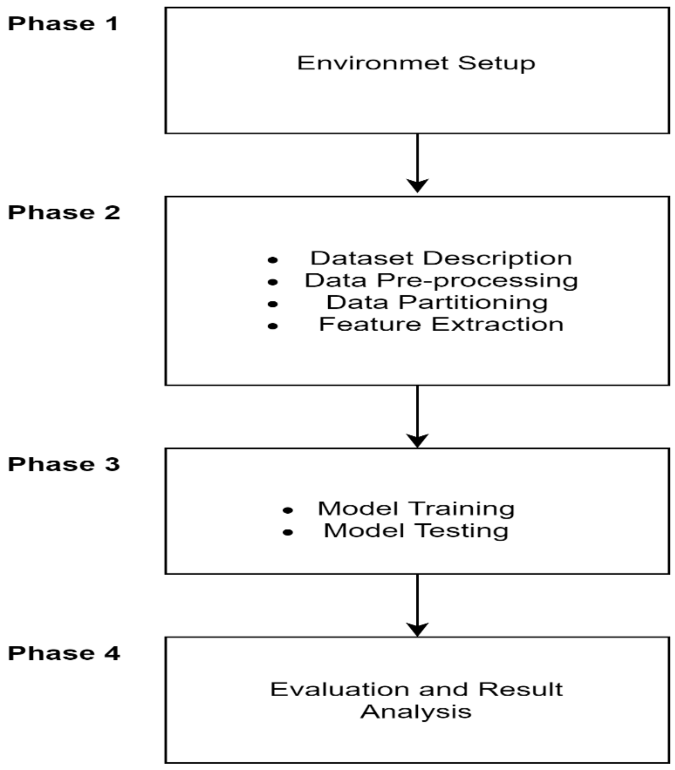

Figure. 2- Architectural Approach

Fig.2 shows the overall architecture of the research. This design is prepared studying the styles used by papers [18]-[20]. Special attention is given to Phase 1 and Phase 2 for flexible use of the resources available on the internet and also to bring wide range of future scope to the investigation.

4.1 Environment Setup

Since this study is decided to perform on virtual public server, AWS platform is chosen. In order to fully understand the working and functions of AWS, a short course was taken from YouTube [26] Initially, a free tier account is made and an EC2 instance is created under N. Virginia (as usually Recommended) with Amazon Linux2 OS. Because this is a basic account, the limited resources that are costless are chosen. Number of CPU is one and with 8Gb of storage capacity (SSD). Public and private IP addresses are assigned default by the server. Inbound traffic is made to pass through port 22 using Secure Shell (SSH) communication making the communication encrypted with password authentication. This instance is connected to windows OS through public and private keys which are separately generated for the project purpose using Putty. The private key generated in stored as a '.pem' file and is later converted to '.ppk' file by 'PuTTYgen'. This '.ppk' file is loaded to putty and connected to the instance by make use of domain name or public IPV4 DNS. Once the connection is secure, it can be saved for future use and operated by the system. In detail specifications of Amazon Linux2 instance is given in Configuration manual for reference.

4.2 Dataset Description and Feature Extraction

The datasets needed for this experiment is taken from two open sources Phishtank [27] and University of New Brunswick [28]. In total ten thousand URLs are used for training and two thousand URLs are used for testing. From the available number of datasets, five thousand URLs are randomly picked from both the sites. Feature extraction is done on both naïve and phishing URLs and the subsequent dataset is stored in a separate file. This file is further used for model training purpose.

It is possible to decide the legitimacy of a URL based on certain features. These features form the deciding factors of this analysis. Hence it is necessary to choose those features which clearly differentiate a valid URL from a phished URL. The features for this model are selected after carefully examining some papers and determining the best among them. In total 18 features are selected from 3 categories and are inspired form [29].

1. Address Bar Features
2. Domain Based Features
3. HTML and Javascript based Features

1.Address Bar Features

- IP Address in the URL

  If an IP address is used as an alternative of the domain name in the URL, such as "http://125.98.3.123/fake.html", users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link "http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html".

Rule: IF $\begin{cases} \text{If The Domain Part has an IP Address } \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

- @ Symbol in the URL

  When the "@" symbol is used in a URL, the browser ignores anything before the "@" symbol, and the genuine address is commonly found after the "@" symbol.

Rule: IF $\begin{cases} \text{Url Having @ Symbol} \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

- Length of the URL

Most of the times, long URLs are kept to hide the questionable part of the website. Like

http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&amp;dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html

To ensure accuracy of our study, we calculated the length of URLs in the dataset and produced an average URL length. The results showed that if the length of the URL is greater than or equal 54 characters then the URL classified as phishing. By reviewing our dataset we were able to find 1220 URLs lengths equals to 54 or more which constitute 48.8% of the total dataset size.

Rule: IF $\begin{cases} URL\ length < 54 \rightarrow feature = \text{Legitimate} \\ else\ if\ URL\ length \geq 54\ and\ \leq 75 \rightarrow feature = Suspicious \\ otherwise \rightarrow feature = \text{Phishing} \end{cases}$

- Depth of the URL
  This feature checks the number of dots (.) in a URL. If the number of dots exceed twice, then its likely do be phished. Example – www.ncirl.ie normally the dot after ‘www’ is neglected because it is a subdomain in itself. So, the number of dots are checked after the main domain name ‘ncirl’ in this case.

$$\text{Rule: IF} \begin{cases} \text{Dots In Domain Part} = 1 \rightarrow \text{Legitimate} \\ \text{Dots In Domain Part} = 2 \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- Redirection “//” in the URL
  The presence of the character "//" in the URL route indicates that the user will be redirected to another website. “http://www.legitimate.com//http://www.phishing.com” is an example of such a URL. We look at the spot where the “//” appears. We discovered that if the URL begins with “HTTP,” the “//” should be placed in the sixth position. If ‘HTTPS’ is present then ‘//’ will be in the seventh position. In this case there’s another ‘//’ after ‘.com’ which clearly signifies it as a phishing URL.

$$\text{Rule: IF} \begin{cases} \text{ThePosition of the Last Occurrence of "//" in the URL} > 7 \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- HTTP/HTTPS in the Domain Name
  The presence of HTTPS is critical in conveying the legitimacy of a website, but it is clearly insufficient. It is recommended that the certificate assigned with HTTPS be checked, including the extent of the trust certificate issuer and the certificate age. Furthermore, we discovered that the minimum age of a reputable certificate is two years by testing our datasets. Some of the certificate enterprises include: GeoTrust, GoDaddy, Network Solutions, Thawte, Comodo, Doster and VeriSign

$$\text{Rule: IF} \begin{cases} \text{Use https and Issuer Is Trusted and Age of Certificate} \geq 1 \text{ Years} \rightarrow \text{Legitimate} \\ \text{Using https and Issuer Is Not Trusted} \rightarrow \text{Suspicious} \\ \text{Otherwise} \rightarrow \text{Phishing} \end{cases}$$

- Using URL shortening service “Tiny URL”
  URL shortening is a technique used on the "World Wide Web" in which a URL can be significantly reduced in length while still directing to the desired webpage. This is accomplished through the use of a “HTTP Redirect” on a short domain name, which links to a webpage with a long URL. The URL “http://portal.hud.ac.uk/” can, for example, be shortened to “bit.ly/19DXSk4”.

$$\underline{\textit{Rule}}\text{: IF} \begin{cases} \text{TinyURL} \rightarrow \text{Phishing} \\ \text{Otherwise} \rightarrow \text{Legitimate} \end{cases}$$

- Prefix or Suffix “-” in the Domain
  In legitimate URLs, the dash symbol is rarely used. Phishers frequently add prefixes or suffixes separated by (-) to domain names to give users the impression that they are dealing with a legitimate website. For instance, see http://www.Confirme-paypal.com/.

Rule: IF $\begin{cases} \text{Domain Name Part Includes } (-) \text{ Symbol } \rightarrow \text{ Phishing} \\ \text{Otherwise } \rightarrow \text{ Legitimate} \end{cases}$

2.Domain Based Features

- DNS Record
  In the case of phishing websites, either the claimed identity is not recognized by the WHOIS database or no records for the hostname are found. If the DNS record is empty or not found, the website is labeled "Phishing," otherwise it is labeled "Legitimate."

Rule: IF $\begin{cases} \text{no DNS Record For The Domain } \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

- Website Traffic
  This feature assesses the popularity of a website by counting the number of visitors and the pages they visit. However, because phishing websites only exist for a short time, they may not be recognized by the Alexa database. Furthermore, if the domain receives no traffic or is not recognized by the Alexa database, it is labelled as “phishing”. Otherwise, it is labelled as "Suspicious".

Rule: IF $\begin{cases} \text{Website Rank} < 100{,}000 \rightarrow \text{ Legitimate} \\ \text{Website Rank} > 100{,}000 \rightarrow \text{Suspicious} \\ \text{Otherwise } \rightarrow \text{ Phish} \end{cases}$

- Age of Domain
  This feature is available from the WHOIS database. Most phishing websites only exist for a short time. We discovered that the minimum age of a legitimate domain is 6 months after reviewing our dataset.

Rule: IF $\begin{cases} \text{Age Of Domain} \geq 6 \text{ months } \rightarrow \text{ Legitimate} \\ \text{Otherwise } \rightarrow \text{ Phishing} \end{cases}$

- End Period of Domain
  This feature is available from the WHOIS database. The remaining domain time is calculated for this feature by finding the difference between the expiration time and the current time. For this project, the end period considered for the legitimate domain is 6 months or less.

Rule: IF $\begin{cases} \text{no DNS Record For The Domain } \rightarrow \text{ Phishing} \\ \text{Otherwise} \rightarrow \text{ Legitimate} \end{cases}$

3. HTML and Javascript based Features

- IFrame Redirection
  IFrame is an HTML tag that allows you to insert another webpage into the one you're now viewing. The “iframe” tag can be used by phishers to make the frame invisible, i.e. without frame borders. Phishers employ the “frameBorder” attribute in this case, which causes the browser to create a visual boundary.

Rule: IF $\begin{cases}\text{Using iframe} \rightarrow \text{ Phishing}\\ \text{Otherwise } \rightarrow \text{ Legitimate}\end{cases}$

- Status Bar Customization
  Phishers may utilize JavaScript to trick visitors into seeing a false URL in the status bar. To get this feature, we'll need to delve into the webpage source code, specifically the "onMouseOver" event, and see if it alters the status bar.

Rule: IF $\begin{cases}\text{onMouseOver Changes Status Bar} \rightarrow \text{ Phishing}\\ \text{It Does't Change Status Bar} \rightarrow \text{Legitimate}\end{cases}$

- Disabling Right Click
  Phishers disable the right-click function with JavaScript, preventing users from viewing and saving the webpage source code. This functionality is handled in the same way as "Hiding the Link with onMouseOver." Nonetheless, we'll look for the event "event.button==2" in the webpage source code and see if the right click is disabled for this functionality.

Rule: IF $\begin{cases}\text{Right Click Disabled } \rightarrow \text{ Phishing}\\ \text{Otherwise} \rightarrow \text{Legitimate}\end{cases}$

- Website Forwarding
  The number of times a website has been redirected is a narrow line that separates phishing websites from authentic ones. We discovered that authentic websites were only routed once in our sample. Phishing websites with this functionality, on the other hand, have been redirected at least four times.

Rule: IF $\begin{cases}\text{ofRedirect Page} \leq 1 \ \rightarrow \text{ Legitimate}\\ \text{of Redirect Page} \geq 2 \text{ And} < 4 \rightarrow \text{ Suspicious}\\ \text{Otherwise } \rightarrow \text{ Phishing}\end{cases}$

4.3 Model training and Testing (proposed model-XGBoost)

The extracted features are stored in a file and this file is then used to train the models. Before training the models, its important to understand how the models work. This section concentrates on in depth working, suitable formulae and diagrams to describe the model.

4.3.1 XGBoost Workflow

XGBoost or Extreme Gradient Boost is a model which is based on Decision Tress ensembled with Machine Learning algorithm that uses extreme Gradient Boosting mechanism. Fig.3 shows how the multi layered mechanism is built to perform.

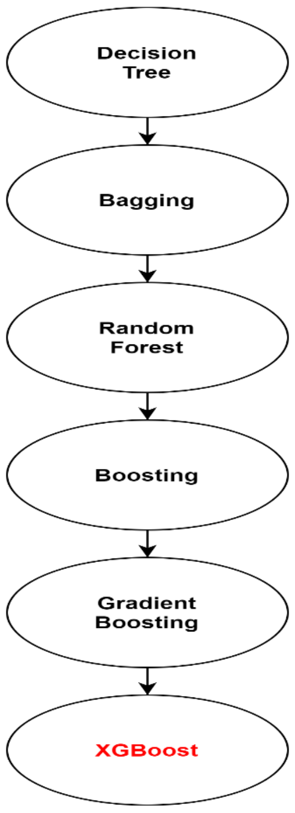

Figure. 3- XGboost Workflow

Decision Tree is a Supervised ML algorithm in which data is constantly split based on certain parameters. These splits are based on the decisions made on the training data. To achieve the desired result, certain criteria are established. A decision tree is a graphical representation of those circumstances.

Bagging or bootstrap aggregation combines each decision tree's collective predictions based on a majority voting mechanism. It combines inputs from all the decision trees to produce a single collective decision**.**

Random Forest is a machine learning technique for solving regression and classification problems. It makes use of ensemble learning, which is a technique that combines many classifiers to solve complex problems. In general, RF consists of many decision trees.

Boosting Models are built in a sequential order to reduce past model errors while maximizing the impact of high-performance models.

Gradient boosting uses the gradient descent approach to reduce error in sequential models.

XGBoost- To eliminate errors, the gradient boosting technique was enhanced using parallel processing, tree pruning, missing value handling, and regularization. It's a powerful combination of software and hardware optimization approaches that deliver higher outcomes in the quickest amount of time while using the fewest computing resources possible.

XGBoost improves Gradient Boosting Machines (GBMs), which use the gradient descent architecture to improve the performance of weak learners (CARTs in general). XGBoost, on the other hand, improves on the base GBM architecture through system optimization and algorithmic enhancements. These two features make XGBoost outperform other algorithms [30].

System Optimization

1. **Parallelization-** To approach the process of sequential tree building, XGBoost employs parallelized implementation. This is possible because the loops used to construct base learners are interchangeable; the outer loop enumerates the leaf nodes of a tree, while the second inner loop computes the features. Parallelization is limited because the outer loop cannot begin until the inner loop is completed. As a result, in order to improve run time, the order of loops is switched using initialization via a global scan of all instances and sorting via parallel threads. By offsetting any parallelization overheads in computation, this switch improves algorithmic performance.
2. **Tree Purning-** The stopping criterion for tree splitting in the GBM framework is greedy by nature and is dependent on the negative loss requirement at the split point. Instead of applying the parameter, XGBoost starts pruning trees backwards by utilizing the 'max depth' parameter. This 'depth-first' method boosts computing performance dramatically.
3. **Hardware Optimization-** This algorithm was developed to make the most efficient use of available hardware resources. This is done through cache awareness, which is achieved by allocating internal buffers in each thread to store gradient statistics. Out-of-core computing makes the most of available disk space while processing massive data sets that are too large to fit in memory.

Algorithmic Enhancement

1. **Regularization-** To prevent overfitting, it penalizes more complex models with both LASSO (L1) and Ridge (L2) regularization.
2. **Sparsity Awareness-** XGBoost naturally admits sparse features for inputs by automatically 'learning' the best missing value based on training loss and handles different types of sparsity patterns in data more efficiently.
3. **Weighted Quantile Sketch**- XGBoost employs the distributed weighted Quantile Sketch algorithm to determine the best split points among weighted datasets.
4. **Cross-Validation**- A cross-validation method is built into each iteration of the algorithm, eliminating the need to explicitly program this search and specify the exact number of boosting iterations required in a single run. Pairs of inputs are

4.3.2 Classification Model (XGBoost Internal Working)

The internal running of this model is derived from [31] where Chen and Guestrin explain the working in detail. The detection of phishing is a supervised learning problem in which the training data xi is used to predict a target variable yi. The inputs to the model are training instances in the pairs of (x1,y1) (x2,y2)….. (xn,yn) where in x symbolizes the feature vector extracted and y symbolizes their respective tags which is either 0 for (legal) and 1 for (phishing) websites. The main aim of the project is to define some particular parameters of the dataset so that the model can use those parameters in future to decide whether the URL is legitimate or not. Each parameter here becomes a tree and adds to the factor for decision making. Though these trees may not give the desired performance however, by combining these trees and boosting them can bring noticeable upscale in the prediction. In XGBOOST, the training data xi is used to predict the target variable yi iteratively until the model's parameters are optimized. The proposed phishing detection model can be represented mathematically as follows

$$\hat{y}_i = \phi(X_i) = \sum_{k=1}^{K} f_k(x_i)\,, f \in F \qquad (1)$$

Where K describes the number of trees and f, the function in function space of F.

Boosted trees are trained using an additive training strategy. At each iteration of the phishing detection process, a new tree is added. The model's final prediction score is calculated by adding the predictive scores of each tree. The end predictive value at t of training can be written as

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_{i=}^{(1)} f_1(x_i) = \hat{y}_1^{(0)} + f_1(x_i)$$

Therefore

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \qquad (2)$$

The most recent tree is created to compensate for the instances of websites that were incorrectly predicted by previous learners. To select the best model for the training data, we must optimize a specific objective function. In this case, we encourage a model to have both high predictive power and to be simple in nature. Minimizing loss function (∅) encourages predictive models, while optimizing regularization (L(φ)) encourages simpler models to have lower variance in future predictions, resulting in more stable predictions.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \qquad (3)$$

Gradient Tree Boosting

XGBOOST uses t boosting iteration to learn a function f(x) that predicts y = f(x) while minimizing a loss function and a regularization term. Similarly, the optimization goal at step t of the training process can be stated as follows:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i{}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t) \qquad (4)$$

The optimization objective using square loss can be written as

$$i = \left(y_i - \hat{y}_i^{(t)}\right)^2 \; but \;\; \hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Adding f_t improves the model, according to equation (2). Second order approximation can be used to optimize the general setting objective, giving rise to.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t) \qquad (5)$$

Where g_i h_i are derived from the loss function. Where

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \text{ and } h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

XGBOOST approximates f(x) using an additive expansion of t regression trees, but instead of minimizing just a lost function, an objective function with two parts is defined, a lost function over the training set and a regularization term to prevent overfitting. Define Ij = {i|q(xi) = j} as the instance set of leaf j. Figure 4 shows how Boosting is done on the Decision Trees, where the average value of each tree is taken.

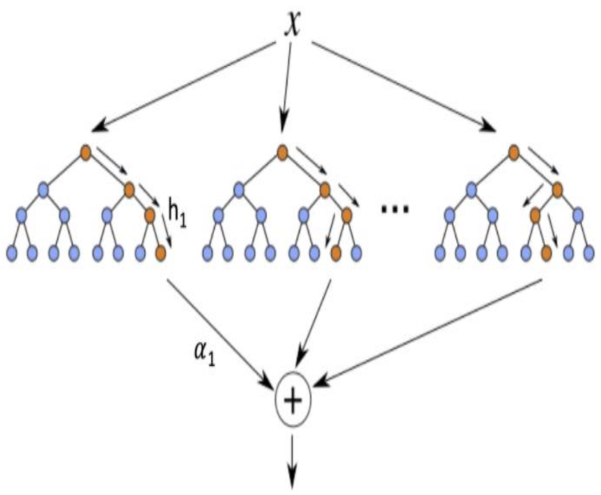

Figure 4: Boosted Decision Trees

Shrinkage, Row subsampling and Column Subsampling

Before adding a new prediction to ft, XGBoost uses the shrinkage parameter to limit the optimal node predictions performed in each iteration t. Furthermore, it employs row and column subsampling. Overfitting can be avoided by using these three parameters [32]. And the corresponding best value is computed by

$$\begin{aligned}\tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n}[g_i f_t(\mathbf{x}_i) + \frac{1}{2}h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T\end{aligned}$$

(6)

For a fixed structure q(x), we can compute the optimal score w ∗ j of leaf j by

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

(7)

Better accuracy can be obtained by selecting an improved tree that optimizes the objective function that was improved in the previous iteration. The optimal score function for a given tree structure is optimal score, and optimal objective reduction measures how good a tree structure is for a given iteration in terms of minimizing the objective function given below.

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

--(8)

Split Finding Algorithms (Basic Exact Greedy algorithm and Approximate Algorithm)

It is extremely difficult to determine the best split in tree learning. To accomplish this, a split finding algorithm known as the exact greedy algorithm is used to specify the overall split. This algorithm is extremely powerful because it greedily enumerates all possible splitting points. As a result, it is impossible to fit all of the data into memory. As a result, the Approximate algorithm is used. Because computing the full tree from a function space is unfeasible, it is diplomatic to include optimal split for practical application [31]. The split can be expressed in the following way:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma \qquad (9)$$

The score on the new leaf, the score on the new right leaf, the score on the original leaf, and the complexity cost of adding another leaf are the components. It is self-evident that if gain is less than γ, we should avoid adding that branch, which is nothing more than pruning.

5. Implementation

As described in the Design Specification, the implementation has 4 parts Environmental Setup, Data Pre-processing, Feature Extraction, Model Training and Result Analysis. And the step-by-step implementation is as follows.

- An EC2 instance is created in AWS with Linux operating system and python 3.8.
- Anaconda 3 and jupyter notebook is downloaded in order to perform the experiment.
- Datasets, both for phishing and legitimate URLs are downloaded and saved.
- Random count of 5000 URLs is selected and stored for the training purposes.
- Feature Extraction is done for both the datasets and a final dataset is saved in a new file consisting of both URLs.
- This final Dataset is divided into 80-20 for training and testing purposes.
- Necessary packages are imported in order to perform the experiment.
- Model training and testing is done simultaneously. The accuracy for each is observed.
- Overall accuracy, precision, recall, F-score is noted for evaluation purposes.
- The results are evaluated, and conclusion is made.
- The drawbacks of this experiment are noted and the factors which can contribute to future work are proposed.

6. Evaluation and Result analysis.

In order to come to a conclusion, the model is compared with other ML models and certain characteristics like Accuracy (Acc), Precision (Pre), Recall (Rec), and F-score (F) are considered. Table 1 represents the confusion matrix on which the result is calculated.

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 1: Confusion Matrix

When the model rightly predicts a URL as phishing its True Positive (TP), when the model declares a URL as legit where its actually phishing its False negative (FN), when it predicts a legit URL as phishing its False Positive (FP), and when the model declares wrongly that a URL is Benign. Also, here accuracy is the ratio of URLs correctly predicted, Precision is the fraction of URLs predicted rightly as phishing, recall is the portion of phishing URLs recognized by the model and F-score is the harmonic mean of Precision and Recall.

$$A_{CC} = \frac{TP + TN}{TP + TN + FN + FN}$$

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F = \frac{2(P \cdot R)}{P + R}$$

	Training accuracy	**Test accuracy**
Decision Tree	0.812	0.819
Random Forest	0.819	0.822
Multilayer Perceptron	0.861	0.863
XGBoost	0.867	0.862
Support Vector Machines	0.802	0.804

Table 2: Accuracy comparison of training and test data

Note- This part was done just as a trial to see how the models perform and not for the actual evaluation.

As seen from the table above, in case of Training accuracy of XGB tops the table with relatively marginal gap with 0.867 followed by MLP with 0.861. The other models fall back of them 0.819, 0.812, 0.802 by RF, DT, SVM. In Test Accuracy the MLP leads with 0.863 whereas XGB falls a bit short (not excessively) with 0.862. the difference is not significant presently but might bring visible space when using larger dataset.

	Overall Accuracy	**Precision**	**Recall**	**F-score**
Decision Tree	0.859	0.924	0.783	0.847
Random Forest	0.859	0.920	0.786	0.848
Multilayer Perceptron	0.851	0.938	0.753	0.835
XGBoost	0.858	0.920	0.786	0.847
Support Vector Machines	0.801	0.967	0.624	0.759

Table 3: Evaluation matrix

Table 3 shows the results obtained from XGBoost with that of the other models used to compare. All the classification models were successfully implemented and by making use of same datasets. The results obtained from the implementation were satisfactory, but were not up to the expectation level. The prediction performance to detect phishing URLs using was enhanced using XGBoost, hence giving it greater accuracy and precision. The overall accuracy was separately done on all the 10000 datasets to calculate the recall, precision and F-score. The results are as shown in the table 3. The overall accuracy was led by DT and RF with 0.859, just behind them was XGBoost with 0.858 followed by MLP and SVM.   Surprisingly, the precision column is led by SVM with 0.967 followed by MLP, then DT with 0.9924, lastly XGB and RF stand at 0.920. XGB and RF lead Recall column with 0.786 followed by DT, MLP, SVM with 0.783, 0.753, 0.624. F-score is led by RF with 0.848, XGB and DT fall slightly behind by 0.847 then MLP and SVM with 0.835 and 0.759.

As seen from table 3, the results obtained by this experiment was not as planned but has still proven that XGB is one of the better classification models. Even though, XGB is built on DT and RF, and is judged to be better than the later [17]-[20], [30], it has failed to exhibit desired performance. This might be the result of dataset size, or the features used to differentiate. Hence, special attention must be given for future work.

7. Conclusion and Future Work

The undesirable results obtained from the experimentation brings out bunch of questions about the performance of XGBoost. The results depict that the base models of XGB (RF and DT) present better outgivings. XGB being superior to its subsets, must undoubtedly have given better results. The results obtained from trial round also bring out misunderstanding in the analysis. Henceforth, the future work for this research holds abundance of detailed analysis on dataset description and Feature extraction part.

[30] provides in-depth knowledge on how XGB utilizes hardware optimization and algorithmic enhancements for better classification. In addition, the design of this model exhibits that each new tree formed minimizes the error formed by the previous tree. On that basis we can attempt sorting on bigger set of data to notice the difference. Feature Extraction being one of the crucial aspects of the development, it is noted that 18 main features were used for exercising the model. Adding more prominent features for educating the model can bring significantly more accuracy. Parameter tuning [33] stayed out in this experiment. Attaching that to the model can be a deciding factor for further development. Developing a Browser extension or GUI for this model can make the phishing detecting

feasible where in the user can put in the URL and get real-time result within seconds. Lastly, comparing XGB with its near competitor ANN can be considered in forthcoming.

The links for the demonstration are as follows.
PPT- https://youtu.be/bB5wDmWa6V0
Project demonstration/walkthrough- https://youtu.be/bB5wDmWa6V0 (Part1)
https://youtu.be/QDH1khuW2Pk (Part 2)

8. References

[1] H. Berghel, "The Dumbing Down of SportPhishing," in Computer, vol. 45, no. 9, pp. 92-94, Sept. 2012, doi: 10.1109/MC.2012.322.
[2] "Phishing, BEC Scams Netting $80,000 On Average in 2020", *Digital Guardian*, 2021. [Online]. Available: https://digitalguardian.com/blog/phishing-bec-scams-netting-80000-average-2020.
[3] "Phishing | KnowBe4", *Knowbe4.com*, 2021. [Online]. Available: https://www.knowbe4.com/phishing.
[4]"Phishing | Wikiwand", *Wikiwand*, 2021. [Online]. Available: https://www.wikiwand.com/en/Phishing#/History
[5] K. D. Tandale and S. N. Pawar, "Different Types of Phishing Attacks and Detection Techniques: A Review," 2020 International Conference on Smart Innovations in Design, Environment, Management, Planning and Computing (ICSIDEMPC), 2020, pp. 295-299, doi: 10.1109/ICSIDEMPC49020.2020.9299624.
[6] G. J. W. Kathrine, P. M. Praise, A. A. Rose and E. C. Kalaivani, "Variants of phishing attacks and their detection techniques," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), 2019, pp. 255-259, doi: 10.1109/ICOEI.2019.8862697.
[7] A. A.A. and P. K., "Towards the Detection of Phishing Attacks," 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), 2020, pp. 337-343, doi: 10.1109/ICOEI48184.2020.9142967.
[8] M. Khonji, Y. Iraqi and A. Jones, "Phishing Detection: A Literature Survey," in IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2091-2121, Fourth Quarter 2013, doi: 10.1109/SURV.2013.032213.00009.
[9] M. N. Feroz and S. Mengel, "Examination of data, rule generation and detection of phishing URLs using online logistic regression," 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 2014, pp. 241-250, doi: 10.1109/BigData.2014.7004239.
[10] A. Subasi, E. Molah, F. Almkallawi and T. J. Chaudhery, "Intelligent phishing website detection using random forest classifier," 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 2017, pp. 1-5, doi: 10.1109/ICECTA.2017.8252051.
[11] S. Sindhu, S. P. Patil, A. Sreevalsan, F. Rahman and M. S. A. N., "Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation," 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), Bengaluru, India, 2020, pp. 391-394, doi: 10.1109/ICSTCEE49637.2020.9277256.
[12] Waleed Ali" Phishing Website Detection based on Supervised Machine Learning with Wrappers Features Selection", IJACSA (International Journal of Advanced Computer Science and Applications, Vol. 8 No. 9, Issue:2017
[13] K. Gajera, M. Jangid, P. Mehta and J. Mittal, "A Novel Approach to Detect Phishing Attack Using Artificial Neural Networks Combined with Pharming Detection," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 196-200, doi: 10.1109/ICECA.2019.8822053.

[14] S. Gupta and A. Singhal, "Phishing URL detection by using artificial neural network with PSO," 2017 2nd International Conference on Telecommunication and Networks (TEL-NET), Noida, India, 2017, pp. 1-6, doi: 10.1109/TEL-NET.2017.8343553.
[15] Al-Ahmadi, S., 2020. PDMLP: Phishing Detection Using Multilayer Perceptron. *International Journal of Network Security & Its Applications (IJNSA) Vol*, *12*.
[16] Odeh, A., Keshta, I. & Abdelfattah, E. (2020). *Efficient Detection of Phishing Websites Using Multilayer Perceptron*. International Association of Online Engineering https://www.learntechlib.org/p/217754/.
[17] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. 2007. A comparison of machine learning techniques for phishing detection. In <i>Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit</i> (<i>eCrime '07</i>). Association for Computing Machinery, New York, NY, USA, 60–69. DOI:https://doi.org/10.1145/1299015.1299021
[18] L. M. da Silva, M. R. Silveira, A. M. Cansian and H. K. Kobayashi, "Multiclass Classification of Malicious Domains Using Passive DNS with XGBoost: (Work in Progress)," 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 2020, pp. 1-3, doi: 10.1109/NCA51143.2020.9306705.
[19] M. R. Silveira, A. M. Cansian and H. K. Kobayashi, "Detection of Malicious Domains Using Passive DNS with XGBoost," 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), Arlington, VA, USA, 2020, pp. 1-3, doi: 10.1109/ISI49825.2020.9280552.
[20] E. S. Gualberto, R. T. De Sousa, T. P. De B. Vieira, J. P. C. L. Da Costa and C. G. Duque, "From Feature Engineering and Topics Models to Enhanced Prediction Rates in Phishing Detection," in IEEE Access, vol. 8, pp. 76368-76385, 2020, doi: 10.1109/ACCESS.2020.2989126.
[21] [4]"Secure Software Development - TEKenable", *TEKenable*, 2021. [Online]. Available: https://www.tekenable.ie/cyber-security/secure-software-development
[22] Kaur, Daljit & Singh, Hardeep. (2012). Secure Spiral: A Secure Software Development Model. Journal of Software Engineering. 6. 10-15. 10.3923/jse.2012.10.15.
[23] "Benefits", *Amazon Web Services, Inc.*, 2021. [Online]. Available: https://aws.amazon.com/application-hosting/benefits/.
[24] A. Das, "11 Reasons Why Linux Is Better Than Windows - It's FOSS", *It's FOSS*, 2021. [Online]. Available: https://itsfoss.com/linux-better-than-windows/.
[25] "8 Reasons Why Python is Good for Artificial Intelligence and Machine Learning", *Medium*, 2021. [Online]. Available: https://towardsdatascience.com/8-reasons-why-python-is-good-for-artificial-intelligence-and-machine-learning-4a23f6bed2e6.
[26] *Youtube.com*, 2021. [Online]. Available: https://www.youtube.com/watch?v=ulprqHHWlng&t=52s. [Accessed: 12- Aug- 2021]
[27] 2021. [Online]. Available: https://www.phishtank.com/developer_info.php
[28] "URL 2016 | Datasets | Research | Canadian Institute for Cybersecurity | UNB", *Unb.ca*, 2021. [Online]. Available: https://www.unb.ca/cic/datasets/url-2016.html.
[29] *Eprints.hud.ac.uk*, 2021. [Online]. Available: http://eprints.hud.ac.uk/id/eprint/24330/6/MohammadPhishing14July2015.pdf.
[30] "XGBoost Algorithm: Long May She Reign!", *Medium*, 2021. [Online]. Available: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
[31] T. Chen and C. Guestrin. XGBOOST: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2016, pp. 785-794.
[32] A. Gómez-Ríos, J. Luengo, and F. Herrera. "A Study on the Noise Label Influence in Boosting Algorithms: AdaBoost, GBM and XGBOOST". In International Conference on Hybrid Artificial Intelligence Systems, Springer, Cham, June 2017, pp. 268-280.
[33] Jain A. (2016). Complete Guide to Parameter Tuning in XGBOOST (with code in python) Retrieved from https:complete guide to parameter tuning in XGBOOST (with code in Python). 2017/06/13.