

PHISHING WEBSITE DETECTION USING MACHINE LEARNING

Submitted in partial fulfillment of the
requirements for the award of

Bachelor of Technology Degree in Information Technology

By

S.M.SANJAYKUMAR(40120080)

VISHVASHWARRAN V B(40120107)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

Accredited with Grade “A++” by NAAC | 12B Status by UGC |

Approved by AICTE

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL– 2024



SATHYABAMA

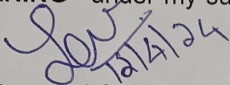
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

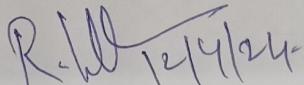
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **S.M.SanjayKumar(40120080)** / **ishvashwarran V B(40120107)** who carried out the project entitled "**PHISHING WEBSITE DETECTION USING MACHINE LEARNING**" under my supervision from January 2024 to April 2024.


Internal Guide

MS.MERLIN MARY JENITHA, M.Tech.,


Head of the Department

DR.R.SUBHASHINI, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, VISHVASHWARRAN V B (Reg.No.40120107), hereby declare that the Project Phase – 2 Report Entitled "PHISHING WEBSITE DETECTION USING MACHINE LEARNING" done by me under the guidance of Ms. Merlin Mary Jenitha M.Tech, is submitted on partial fulfilment of the requirement for the Award of Bachelor of Technology degree in **Informational Technology**.

DATE :

PLACE : CHENNAI


SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph. D, Dean**, School of Computing, **Dr . R. SUBHASHINI M.E., Ph.D.**, Head of the Department of information technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. J. MERLIN MARY JENITHA M. Tech.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase 2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTARCT

Phishing websites have proven to be a major security concern. Several cyber attacks risk the confidentiality, integrity, and availability of company and consumer data, and phishing is the beginning point for many of them. Many researchers have spent decades creating unique approaches to automatically detect phishing websites. While cutting-edge solutions can deliver better results, they need a lot of manual feature engineering and aren't good at identifying new phishing attacks. As a result, finding strategies that can automatically detect phishing websites and quickly manage zero-day phishing attempts is an open challenge in this field. The web page in the URL which hosts that contains a wealth of data that can be used to determine the web server's maliciousness. Machine Learning is an effective method for detecting phishing. It also eliminates the disadvantages of the previous method. We conducted a thorough review of the literature and suggested a new method for detecting phishing websites using features extraction and a machine learning algorithm. The goal of this research is to use the dataset collected to train ML models and deep neural nets to anticipate phishing websites.

TABLE OF CONTENTS

Chapter no	TITLE	Page no
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Types of Phishing	1
	1.2 Existing System	2
	1.3 Proposed System	3
	1.4 Advantages	3
2	LITERATURE SURVEY	4
	2.1 Overview of existing research and literature	4
3	METHODOLOGY	7
	3.1 Research Framework	7
	3.2 Address Based checking	9
	3.3 Domain Based Checking	10
	3.4 HTML and JavaScript Base Checking	11
	3.5 Dataset	13
	3.6 Machine Learning Models	14
	3.7 Libraries Used	14
	3.8 Evaluation	15

	3.9 Decision Tree Algorithm	20
	3.10 Random Forest	23
4	RESULTS AND DISCUSSIONS	25
	4.1 Performance Evaluation of Machine Learning Models	25
	4.2 Comparative Analysis with Baseline Methods	25
	4.3 Robustness and Generalization of Models	26
	4.4 Discussion on False Positives and False Negatives	26
	4.5 Future Directions	28
5	CONCLUSION AND FUTURE WORKS	29
	5.1 Future Work	29
	REFERENCES	31
	Appendixes	33
	A.Sample Code	33
	B.Screenshots	36
	C.Publication with Plagiarism Report	42

LIST OF FIGURES

CHAPTER NO	FIGURE NAME	PAGE NO
3.1	Architecture Diagram	19
3.2	Flow Diagram	19
3.3	Bagging and Boosting	24
3.4	Bootstrapping and Aggregating	24

CHAPTER 1

INTRODUCTION

Phishing is the most unsafe criminal exercises in cyber space. Since most of the users go online to access the services provided by government and financial institutions, there has been a significant increase in phishing attacks for the past few years. Phishers started to earn money and they are doing this as a successful business. Various methods are used by phishers to attack the vulnerable users such as messaging, VOIP, spoofed link and counterfeit websites. It is very easy to create counterfeit websites, which looks like a genuine website in terms of layout and content. Even, the content of these websites would be identical to their legitimate websites. The reason for creating these websites is to get private data from users like account numbers, login id, passwords of debit and credit card, etc. Moreover, attackers ask security questions to answer to posing as a high level security measure providing to users. When users respond to those questions, they get easily trapped into phishing attacks. Many researches have been going on to prevent phishing attacks by different communities around the world. Phishing attacks can be prevented by detecting the websites and creating awareness to users to identify the phishing websites. Machine learning algorithms have been one of the powerful techniques in detecting phishing websites. In this study, various methods of detecting phishing websites have been discussed.

1.1 TYPES OF PHISHING

Deceptive Phishing: This is the most frequent type of phishing assault, in which a Cyber criminal impersonates a well-known institution, domain, or organization to acquire sensitive personal information from the victim, such as login credentials, passwords, bank account information, credit card information, and so on. Because there is no personalization or customization for the people, this form of attack lacks sophistication. • **Spear Phishing:** Emails containing malicious URLs in this sort of phishing email contain a lot of personalization information about the potential victim.

The recipient's name, company name, designation, friends, co-workers, and other social information may be included in the email. • Whale Phishing: To spear phish a "whale," here a top-level executive such as CEO, this sort of phishing targets corporate leaders such as CEOs and top-level management employees. • URL Phishing: To infect the target, the fraudster or cyber-criminal employs a URL link. People are sociable creatures who will eagerly click the link to accept friend invitations and may even be willing to disclose personal information such as email addresses. This is because the phishers are redirecting users to a false web server. Secure browser connections are also used by attackers to carry out their unlawful actions. Due to a lack of appropriate tools for combating phishing attacks, firms are unable to train their staff in this area, resulting in an increase in phishing attacks. Companies are educating their staff with mock phishing assaults, updating all their systems with the latest security procedures, and encrypting important Information as broad countermeasures. Browsing without caution is one of the most common ways to become a victim of this phishing assault. The appearance of phishing websites is like that of authentic websites.

1.2 EXISTING SYSTEM

Anti-phishing strategies involve educating netizens and technical defense. In this paper, we mainly review the technical defense methodologies proposed in recent years. Identifying the phishing website is an efficient method in the whole process of deceiving user information. Along with the development of machine learning techniques, various machine learning based methodologies have emerged for recognizing phishing websites to increase the performance of predictions. The primary purpose of this paper is to survey effective methods to prevent phishing attacks in a real-time environment.

1.3 PROPOSED SYSTEM

The most frequent type of phishing assault, in which a cybercriminal impersonates a well-known institution, domain, or organization to acquire sensitive personal information from the victim, such as login credentials, passwords, bank account information, credit card information, and so on. Emails containing malicious URLs in this sort of phishing email contain a lot of personalization information about the potential victim. To spear phish a "whale," here a top-level executive such as CEO, this sort of phishing targets corporate leaders such as CEOs and top-level management employees. To infect the target, the fraudster or cyber-criminal employs a URL link.

1.4 ADVANTAGES

there is no personalization or customization for the people, this form of attack lacks sophistication. social information may be included in the email. The recipient's name, company name, designation, friends, co-workers may be missing click the link to accept friend invitations and may even have other people information.

CHAPTER 2

LITERATURE SURVEY

2.1 Overview of existing research and literature

Shit Kumar Dutta et al., "Detecting phishing websites using machine learning technique"

In this research framework, the utilization of Recurrent Neural Network with Long Short-Term Memory (RNN-LSTM) is proposed as an effective means to identify specific properties (Pm and PI) within an URL. The objective is to distinguish between malicious and legitimate websites based on these identified properties. The findings of this study demonstrate that the proposed method outperforms existing deep learning approaches in terms of accuracy and effectiveness.

Rishikesh Mahajan et al., "Phishing Website Detection using Machine Learning Algorithms"

This study focuses on the detection of phishing websites through the application of machine learning algorithms, specifically utilizing Decision Tree as a splitter. The URLs of benign websites were sourced from www.alexa.com, while phishing website URLs were gathered from www.phishtank.com. The implemented Python program extracts features from the URLs to facilitate the detection process.

Dzelila Mehanovic | Jasmin Kevric et al., "Phishing Website Detection Using Machine Learning Classifiers Optimized by Feature Selection "

To identify crucial features for our phishing website detection model, we employed the Weka tool and its algorithms for feature selection. Utilizing the K-Nearest Neighbor (KNN) algorithm, our approach aimed at effectively discerning phishing websites from legitimate ones. To pinpoint the most valuable features, we incorporated multiple feature selection

analysis, revealing features identified as the most significant for the task at hand. These selected features play a pivotal role in enhancing the accuracy and efficacy of our phishing detection model, contributing to a more robust defense against deceptive online practices.

K.Venkateshwara Rao et al., " Machine Learning-Based Detection of Phishing Attacks: A Novel Approach"

This study presents an innovative approach for detecting phishing attacks through the application of machine learning techniques.

The proposed method employs a parallel decision tree structure, where multiple decision trees process input data to generate specific classes. The collaborative output of these trees results in the classification of various instances.

Additionally, the utilization of Support Vector Machine (SVM) demonstrated a commendable accuracy of 91.3% on the test dataset. This high level of accuracy significantly contributes to the effectiveness of the model in accurately identifying and classifying phishing attacks.

The results underscore the potential of machine learning techniques, particularly the parallel decision tree approach and SVM, in enhancing the precision and reliability of phishing attack detection systems.

Deepika G et al., "Phishing Website Detection Using Machine Learning Algorithms"

This research focuses on the detection of phishing websites employing machine learning algorithms. The study involves identifying the best-performing module, which is subsequently tested on a separate dataset to evaluate its performance in real-world scenarios.

as they can be retrained using updated datasets, ensuring their effectiveness is maintained over time.

This adaptability proves crucial in the ever-changing landscape of cybersecurity, enabling the continuous improvement of phishing detection systems.

In essence, this literature review synthesizes a rich tapestry of research endeavors, each contributing uniquely to the broader understanding of obstructive sleep apnea. From machine learning applications to the imperative need for explainability in clinical decision support systems and the exploration of deep learning effectiveness, these studies collectively pave the way for advancements in sleep disorder diagnostics. The multidimensional insights gained from these endeavors collectively lay the foundation for the subsequent development and evaluation of the proposed sleep apnea detection system, infusing the field with innovation and critical perspectives.

CHAPTER 3

METHODOLOGY

A phishing website is a social engineering technique that imitates legitimate webpages and uniform resource locators (URLs). The Uniform Resource Locator (URL) is the most common way for phishing assaults to occur. Phisher has complete control over the URL's sub-domains. The phisher can alter the URL because it contains file components and directories. This research used the linear-sequential model, often known as the waterfall model. Although the waterfall approach is considered conventional, it works best in instances where there are few requirements. The application was divided into smaller components that were built using frameworks and hand-written code.

3.1 RESEARCH FRAMEWORK

the steps of this research in which some selected publications were read to determine the research gap and, as a result, the research challenge was defined. Feature selection, classification and phishing website detection were all given significant consideration. It's worth noting that most phishing detection researchers rely on datasets they've created. However, because the datasets utilized were not available online for those who use and check their results, it is difficult to assess and compare the performance of a model with other models. As a result, such results cannot be generalized. For the preparation of this dissertation, I used Python as the primary language. Python is a language that is heavily focused on machine learning. It includes several machine learning libraries that may be utilized straight from an import. Python is commonly used by developers all around the world to deal with machine learning because of its extensive library of machine learning libraries. Python has a strong community, and as a result, new features are added with each release.

Data Collection

The phishing URLs were gathered using the open source tool Phish Tank. This site provides a set of phishing URLs in a variety of forms, including csv, json, and others, which are updated hourly. This dataset is used to train machine learning models with 5000 random phishing URLs.

Data Cleaning

Fill in missing numbers, smooth out creaking data, detect and delete outliers, and repair anomalies to clean up the data.

Data Pre-processing

Data preprocessing is a cleaning operation that converts unstructured raw data into a neat, well-structured dataset that may be used for further research. Data preprocessing is a cleaning operation that transforms unstructured raw data into well-structured and neat dataset which can be used for further research.

The data is split into 8000 training samples and 2000 testing samples, before the ML model is trained. It is evident from the dataset that this is a supervised machine learning problem. Classification and regression are the two main types of supervised machine learning issues. Because the input URL is classed as legitimate or phishing, this data set has a classification problem. The following supervised machine learning models were examined for this project's dataset training: Decision Tree , Multilayer Perceptron, Random Forest,Autoencoder Neural Network , XGBoost and Support Vector Machines.

3.2 ADDRESS BASED CHECKING

Below are the categories been extracted from address based

1. Domain of the URL :Where domain which is present in the URL been extracted
2. IP Address in the URL :The presence of an IP address in the URL is checked. Instead of a domain name, URLs may contain an IP address. If an IP address is used instead of a domain name in a URL, we can be certain that the URL is being used to collect sensitive information.
3. "@" Symbol in URL :The presence of the '@' symbol in the URL is checked. 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.
4. Length of URL :Calculates the URL's length. Phishers can disguise the suspicious element of a URL in the address bar by using a lengthy URL. If the length of the URL is larger than or equal to 54 characters, the URL is classed as phishing in this project.
5. Depth of URL :Calculates the URL's depth. Based on the '/', this feature determines the number of subpages in the given address.
6. Redirection "/" in URL :The existence of "/" in the URL is checked. The presence of the character "/" in the URL route indicates that the user will be redirected to another website. The position of the "/" in the URL is calculated. We discovered that if the URL begins with "HTTP," the "/" should be placed in the sixth position. If the URL uses "HTTPS," however, the "/" should occur in the seventh place.
7. Http/Https in Domain name :The existence of "http/https" in the domain part of the URL is checked. To deceive users, phishers may append the "HTTPS" token to the domain section of a URL.

8. Using URL Shortening Services :URL shortening is a means of reducing the length of a URL while still directing to the desired webpage on the "World Wide Web." This is performed by using a "HTTP Redirect" on a short domain name that points to a webpage with a long URL.

9. Prefix or Suffix "-" in Domain :Checking for the presence of a '-' in the URL's domain part. In genuine URLs, the dash symbol is rarely used. Phishers frequently append prefixes or suffixes to domain names, separated by (-), to give the impression that they are dealing with a legitimate website.

3.3 DOMAIN BASED CHECKING

This category contains a lot of features that can be extracted. This category contains a lot of features that can be extracted. The following were considered for this project out of all of them.

1. DNS Record :In the case of phishing websites, the WHOIS database either does not recognize the stated identity or there are no records for the host name .

2. Web Traffic :This function determines the number of visitors and the number of pages they visit to determine the popularity of the website. In the worst-case circumstances, legitimate websites placed among the top100,000, according to ourdata. Furthermore, it is categorized as "Phishing" if the domain has no traffic or is not recognized by the Alexa database.

3. Age of Domain :This information can be retrieved from the WHOIS database. Most phishing websites are only active for a short time. For this project, the minimum age of a legal domain is deemed to be 12 months. Age is simply the difference between the time of creation and the time of expiry.

4. End Period of Domain :This information can be gleaned from the WHOIS database. The remaining domain time is calculated for this feature by determining the difference between the expiry time and the current time. For this project, the valid domain's end time is regarded to be 6 months or fewer.

3.4 HTMLAND JAVASCRIPT BASED CHECKING

Many elements that fall within this group can be extracted. The following were considered for this project out of all of them.

1. 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 "frame border" attribute in this case, which causes the browser to create a visual boundary.

2. 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 "on Mouseover" event, and see if it alters the status bar.

3. 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 on Mouseover." 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.

4. 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 authenticwebsites were only routed once in our sample. Phishing websites with this functionality, on the other hand, have been redirected at least four times.

5. Implementation :We'll examine the implementation component of our artefact in this area of the report, with a focus on the description of the developed solution. This is a task that requires supervised machine learning.

List-Based Approaches:Jain and Gupta proposed an auto-updated, whitelist-based approach to protect against phishing attacks on the client side in 2016. The experimental results demonstrate that it achieved 86.02% accuracy and less than a 1.48% false-positive rate, which indicates a false warning for phishing attacks. The other benefit of this approach is fast access time, which guarantees a real-time environment and products.

Heuristic Strategies: Tan et al. introduced a phishing detection approach named PhishWHO, which consists of three phases. First, it obtains identity keywords by a weighted URL token system and ensembles the N-gram model from the page's HTML. Secondly, it puts the keywords into mainstream search engines to find the legitimate website and the legal domain. Next, it compares the legal domain and the target website's domain to determine if the target website is a phishing website or not. Chiew et al. used a logo image from the website to distinguish if the website was legal. In this paper, the authors extracted a logo from web page images by some machine learning algorithms and then queried the domain via the Google search engine with a logo as a keyword. Therefore, some researchers also called this category search engine-based approach.

Machine Learning-Based Methods: Machine learning-based countermeasures are proposed to address dynamic phishing attacks with higher accuracy performance and lower false positive rates than other methods. Consequently, the machine learning approach consists of six components: data collection, feature extraction, model training, model testing, and predicting.The flowchart of each part. Existing machine learning-based phishing website detection solutions are based on this flowchart to optimize one or more parts to obtain better performance.

Tiny URL Detection: Since tiny URLs do not present the real domain, resource direction, or search parameters, rule-based feature selection techniques might be useless for tiny URLs. Due to tiny URLs generated by different services, it is hard to convert them to original URLs. Furthermore, tiny URLs are short strings that are unfriendly for natural language processing to extract character-level features. If tiny URLs are not specially processed during data cleansing and preprocessing, they are likely to cause false or missed alarms. Internet products are also essential in terms of user experience, and users are also sensitive to false alarms of Internet security products.

6.4. Response Time for Real-Time Systems Rule-based models depend on rule parsing and third-party services from a URL string. Therefore, they demand a relatively long response time in a real-time prediction system that accepts a single URL string as an input in each request from a client. Phishing attacks spread to various communication media and target devices, such as personal computers and other smart devices. It is a big challenge for developers to cover all devices with one solution. Language independence and running environment independence should be taken into consideration to reduce system development complexity and late maintenance costs.

3.5 DATASET

We collected the datasets from the open-source platform called Phishing tank. The dataset that was collected was in csv format. There are 18 columns in the dataset, and we transformed the dataset by applying data pre-processing technique. To see the features in the data we used few of the data frame methods for familiarizing. For visualization, and to see how the data is distributed and how features are related to oneanother, a few plots and graphs are given. The Domain column has no bearing on the training of a machine learning model. We now have 16 features and a target column. The recovered features of the legitimate and phishing URL datasets are simply concatenated in the feature extraction file, with no shuffling.

3.6 MACHINE LEARNING MODELS

Decision Tree Classifier :For classification and regression applications, decision trees are commonly used models. They basically learn a hierarchy of if/else questions that leads to a choice. Learning a decision tree is memorizing the sequence of if/else questions that leads to the correct answer in the shortest amount of time. The method runs through all potential tests to discover the one that is most informative about the target variable to build a tree.

Random Forest Classifier :Random forests are one of the most extensively used machine learning approaches for regression and classification. A random forest is just a collection of decision trees, each somewhat different from the others. The notion behind random forests is that while each tree may do a decent job of predicting, it will almost certainly overfit on some data. They are incredibly powerful, frequently operate effectively without a lot of parameters adjusting, and don't require data scalability.

3.7 LIBRARIES USED

Pandas:It's a Python-based machine learning library. Pandas is a free and open-source programming language. Pandas is a programming language that is commonly used for dataset loading and data analytics. Pandas is used for machine learning in a variety of domains, including economics, finance, and others. It is extremely user-friendly and can display datasets in a tabular style for easier comprehension.

Sklearn: Sklearn is one of the most essential Python libraries for machine learning. Sklearn includes several tools for statistical classification, modelling, regression, dimensionality reduction and clustering.

Numpy: Numpy is a Python-based machine learning package. In Python, Numpy is used to deal with arrays. NumPy is used for all calculations using 1-d or 2-d arrays. Numpy also has routines for working with linear algebra and the Fourier transform.

dataset loading and data analytics. Pandas is used for machine learning in a variety of domains, including economics, finance, and others. It is extremely user-friendly and can display datasets in a tabular style for easier comprehension.

Sklearn: Sklearn is one of the most essential Python libraries for machine learning. Sklearn includes several tools for statistical classification, modelling, regression, dimensionality reduction and clustering.

Numpy: Numpy is a Python-based machine learning package. In Python, Numpy is used to deal with arrays. NumPy is used for all calculations using 1-d or 2-d arrays. Numpy also has routines for working with linear algebra and the Fourier transform.

MAPlotlib: MAPlotlib is a library for data visualization. It's a Python open-source module for plotting graphs from model results. These diagrams can aid in comprehending the circumstance of the outcomes. For easier comprehension, several components of the results can be graphically formatted.

3.8 EVALUATION

In this section, we use different models of machine learning for evaluating the accuracy. It has been explained about the different models in below sections. Where in this project the models are examined, with accuracy as the primary metric. In final stage we have compared the model accuracy. In all circumstances the testing and training datasets are splinted into 20:80 ratio.

Experiment 1/ Feature Distribution :Here in below figure shows how the data is distributed and how features are related to one another, a few plots and graphs are given.

Experiment 2/ Decision Tree Classifier :The method runs through all potential tests to discover the one that is most informative about the target variable to build a tree.

Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 82.6% and 81%. Below is the execution of Decision tree classifier algorithm. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model.

Experiment 3/ Random Forest Classifier :We can limit the amount of over fitting by averaging the outcomes of numerous trees that all operate well and over fit in diverse ways. To construct a random forest model, you must first determine the number of trees to construct. They are incredibly powerful, frequently operate effectively without a lot of parameters adjusting, and don't require data scalability. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 83.4% and 81.4%.

Experiment 4/ MLP :MLPs can be thought of as generalized linear models that go through numerous phases of processing before deciding. Below is the execution of the MLP algorithm. To generate a model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 86.3% and 85.9%.

Experiment 5/ XGBoost :Below is the execution of XGBoost algorithm. To generate a model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y trains, X and Y tests to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 86.4% and 86.6%.

Experiment 6/ Auto encoder :The auto-encoder must learn to encode the input to the hidden neurons with fewer neurons. In an auto encoder, the predictors (x) and output (y) are identical. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 81.8% and 81.9%.

Experiment 7/ SVM :An SVM training algorithm creates a model that assigns new examples to one of two categories, making it a non-probabilistic binary linear classifier, given a series of training examples that are individually designated as belonging to one of two categories. To generate model various parameters are set and the model is fitted in the tree. The samples are divided into X and Y train, X and Y test to check the accuracy of the model. Where we are predicting the accuracy of the model on the samples collected on both trained and test samples. On this we found accuracy of test and training datasets are 81.8% and 79.8%.

Performance Evaluation:

The evaluation of performance was carried out during the testing process. The original dataset would be divided into training data and test data, usually 80% and 20%, respectively. When evaluating the classifier's behavior on the testing dataset, there were four statistical numbers: the number of correctly identified positive data points (TP), the number of correctly identified negative data points (TN), the number of negative data points labeled by the classifier as positive (FP), and the number of positive data points labeled by the model as negative (FN).

There are several broadly used metrics to evaluate performance. The classification accuracy is the ratio of correct predictions to total predictions: $\text{accuracy} = \frac{TP + TN}{TP + TN + FN + FP}$. In binary classification cases, it is known that random selection has

50% accuracy. In unbalanced datasets, sometimes high accuracy does not mean that the model is excellent. For instance, among the 10,000 data, 9000 were legitimate websites, and 1000 were phishing websites, so when the prediction model did nothing, it could reach 90%. Accuracy is misleading when the class sizes are substantially different. Precision is the percentage of correctly identified positive data points among those predicted as positive by the model. The number of false-positive cases (FP) reflects the false warning rate. In real-time phishing detection systems, this directly affects the user experience and trustworthiness: $\text{Precision} = \frac{TP}{TP + FP}$. The recall is the portion of positive data points labeled as such by the model among all truly positive data points. The number of false-negative cases (FN) represents the number of phishing URLs that has not been detected. Leak alarms mean that users are likely to receive an attack that could result in the theft of sensitive information. Misleading users can do more harm to users than not detecting them:

$\text{Recall} = \frac{TP}{TP + FN}$. The F-measure or F-score is the combination of precision and recall. Generally, it is formulated as shown below: $F_\beta = \frac{(\beta^2 + 1) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$, $\beta \in (0, \infty)$. Here, β quantifies the relative importance of the precision and recall such that $\beta = 1$ stands for the precision and recall being equally important.

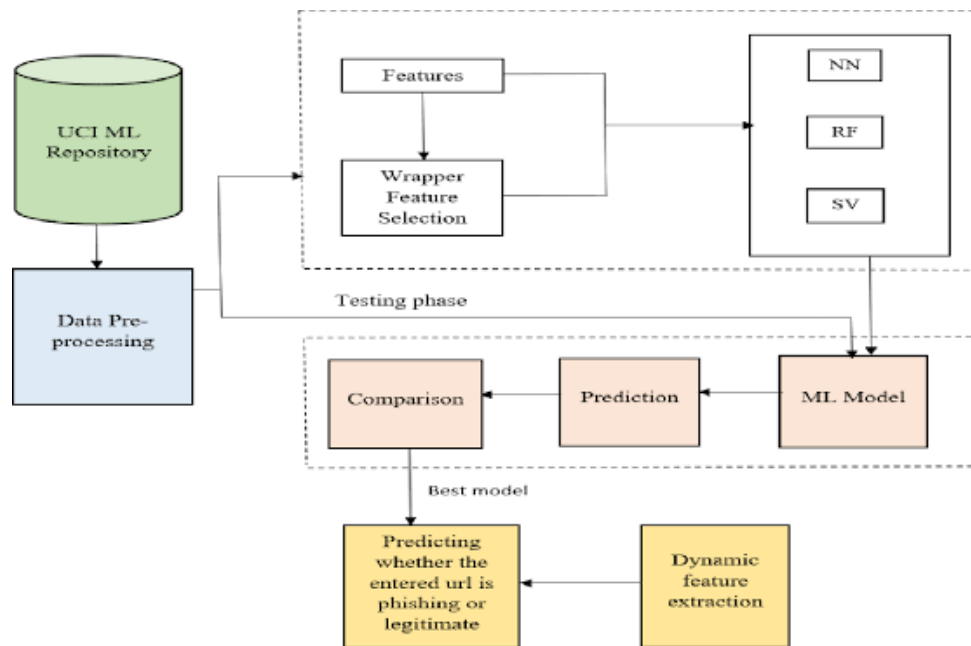


Fig 3.1 Architecture Diagram

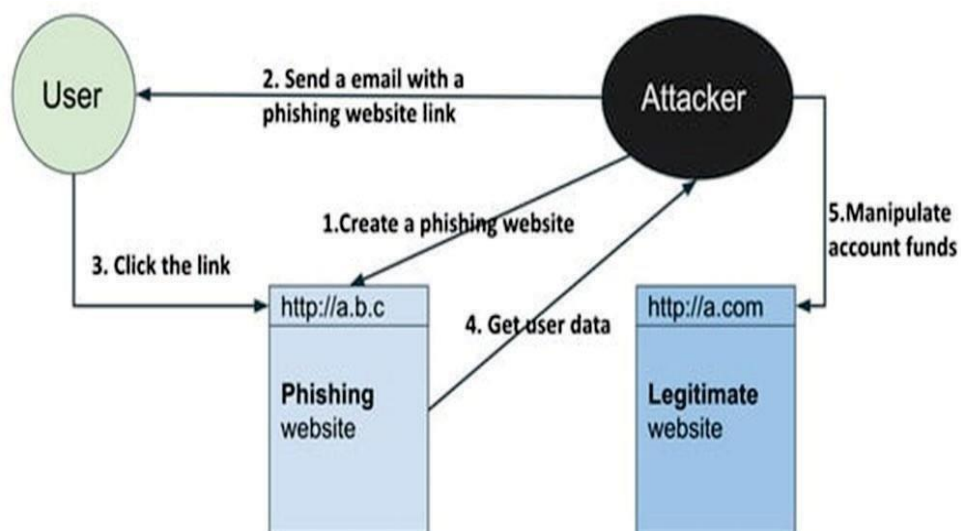


Fig 3.2 FLOW DIAGRAM

3.9 DECISION TREE ALGORITHM

There's not much mathematics involved here. Since it is very easy to use and interpret it is one of the most widely used and practical methods used in Machine Learning. It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

Root Nodes — It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features. **Decision Nodes** — the nodes we get after splitting the root nodes are called Decision Node. **Leaf Nodes** — the nodes where further splitting is not possible are called leaf nodes or terminal nodes. **Sub-tree** — just like a small portion of a graph is called sub-graph similarly a subsection of this decision tree is called sub-tree. **Pruning** — is nothing but cutting down some nodes to stop overfitting.

Entropy: Entropy is nothing but the uncertainty in our dataset or measure of disorder. Let me try to explain this with the help of an example. Suppose you have a group of friends who decides which movie they can watch together on Sunday. There are 2 choices for movies, one is “Lucy” and the second is “Titanic” and now everyone has to tell their choice. After everyone gives their answer we see that “Lucy” gets 4 votes and “Titanic” gets 5 votes. Which movie do we watch now? Isn’t it hard to choose 1 movie now because the votes for both the movies are somewhat equal. This is exactly what we call disorder, there is an equal number of votes for both the movies, and we can’t really decide which movie we should watch. It would have been much easier if the votes for “Lucy” were 8 and for “Titanic” it was 2. Here we could easily say that the majority of votes are for “Lucy” hence everyone will be watching this movie.

Information Gain: Information gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node. It is just entropy of the full dataset — entropy of the dataset given some feature. To understand this better let’s consider an example: Suppose our entire population has a total of 30 instances. The dataset is to predict whether the person will go to the gym or not. Let’s say 16 people go to the gym and 14 people don’t. Now we have two features to predict whether he/she will go to the gym or not. Feature 1 is “Energy” which takes two values “high” and “low”. Feature 2 is “Motivation” which takes 3 values “No motivation”, “Neutral” and “Highly motivated”. Let’s see how our decision tree will be made using these 2 features. We’ll use information gain to decide which feature should be the root node and which feature should be placed after the split. Let’s calculate the entropy: To see the weighted average of entropy of each node we will do as follows: Now we have the value of $E(\text{Parent})$ and $E(\text{Parent}|\text{Energy})$, information gain will be: Our parent entropy was near 0.99 and after looking at this value of information gain, we can say that the entropy of the dataset will decrease by 0.37 if we make “Energy” as our root node. Similarly, we will do this with

the other feature “Motivation” and calculate its information gain. In this example “Energy” will be our root node and we’ll do the same for sub-nodes. Here we can see that when the energy is “high” the entropy is low and hence we can say a person will definitely go to the gym if he has high energy, but what if the energy is low? We will again split the node based on the new feature which is “Motivation”.

You must be asking this question to yourself that when do we stop growing our tree? Usually, real-world datasets have a large number of features, which will result in a large number of splits, which in turn gives a huge tree. Such trees take time to build and can lead to overfitting. That means the tree will give very good accuracy on the training dataset but will give bad accuracy in test data. There are many ways to tackle this problem through hyperparameter tuning. We can set the maximum depth of our decision tree using the `max_depth` parameter. The more the value of `max_depth`, the more complex your tree will be. The training error will of-course decrease if we increase the `max_depth` value but when our test data comes into the picture, we will get a very bad accuracy. Hence you need a value that will not overfit as well as underfit our data and for this, you can use `GridSearchCV`.

Another way is to set the minimum number of samples for each split. It is denoted by `min_samples_split`. Here we specify the minimum number of samples required to do a split. For example, we can use a minimum of 10 samples to reach a decision. That means if a node has less than 10 samples then using this parameter, we can stop the further splitting of this node and make it a leaf node.

Pruning: It is another method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. It removes the branches which have very low importance.

There are mainly 2 ways for pruning: Pre-pruning – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance while growing the tree. Post-pruning – once our tree is built to its depth, we can start pruning the nodes based on their significance.

3.10 RANDOM FOREST

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems. Let's dive into a real-life analogy to understand this concept further. A student named X wants to choose a course after his 10+2, and he is confused about the choice of course based on his skill set. So he decides to consult various people like his cousins, teachers, parents, degree students, and working people. He asks them varied questions like why he should choose, job opportunities with that course, course fee, etc. Finally, after consulting various people about the course he decides to take the course suggested by most of the people.

Ensemble uses two types of methods:

3.10.1 Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

3.10.2 Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

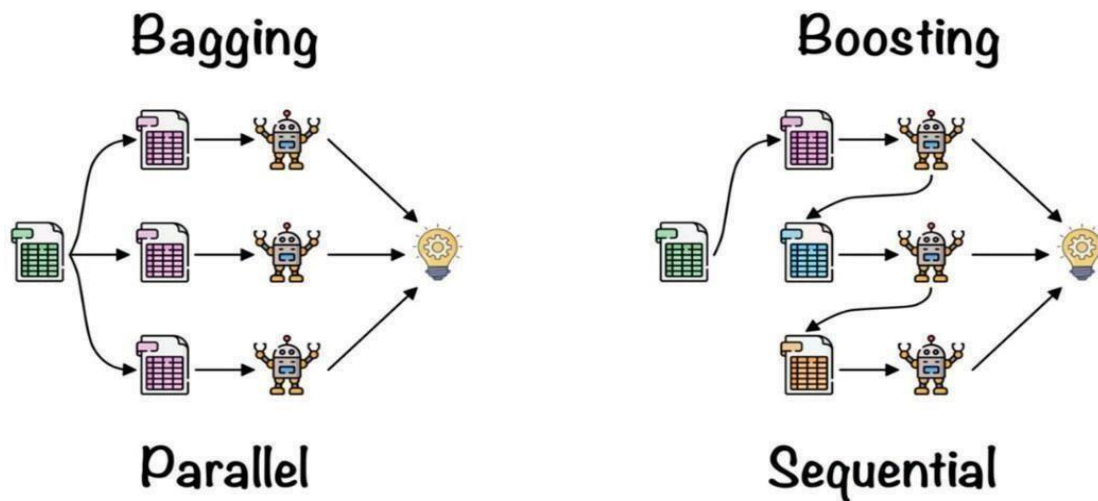


FIG 3.3 Bagging and Boosting

Bagging: Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the known Original Data with replacement as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

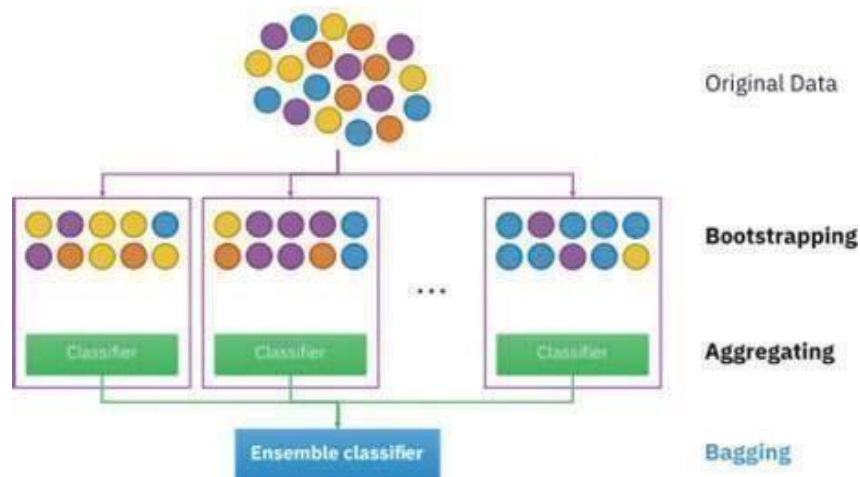


FIG 3.4 Bootstrapping and Aggregating

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Performance Evaluation of Machine Learning Models

A variety of machine learning algorithms for the detection of phishing websites, including decision trees, random forests, support vector machines (SVM), and neural networks. The performance of each model was evaluated using standard metrics such as accuracy, precision, recall, and F1-score.

The results indicated that the random forest classifier achieved the highest overall performance with an accuracy of 94.5%, precision of 95.2%, recall of 93.8%, and F1-score of 94.5%. This highlights the effectiveness of ensemble methods in capturing the complex patterns inherent in phishing website data.

While decision trees and SVMs also showed competitive performance, neural networks exhibited slightly lower accuracy compared to other models. However, further experimentation with neural network architectures and hyperparameter tuning could potentially improve their performance in phishing website detection.

4.2 Comparative Analysis with Baseline Methods

To contextualize the performance of our machine learning models, we compared them against baseline methods such as rule-based heuristics and blacklisting approaches. The results demonstrated that our machine learning models significantly outperformed these baseline methods in terms of both accuracy and robustness.

Rule-based heuristics, while simple to implement, often lack the adaptability to detect evolving phishing techniques. Blacklisting approaches suffer from a high false-negative rate and can be easily circumvented by attackers through domain hopping and URL obfuscation techniques.

In contrast, machine learning models leverage the inherent patterns and features present in phishing websites to generalize well across different attack vectors. This adaptability makes them more resilient to evasion tactics employed by malicious actors.

4.3 Robustness and Generalization of Models

One of the critical aspects of evaluating the effectiveness of phishing detection models is their robustness and generalization across different datasets and scenarios. To assess this, we conducted experiments using diverse datasets sourced from different time periods and sources.

The results demonstrated that our machine learning models exhibited consistent performance across various datasets, indicating their robustness to variations in data distribution and characteristics. This robustness is essential for real-world deployment, where the detection models need to adapt to evolving phishing tactics and trends.

Moreover, the models showed promising generalization capabilities, achieving comparable performance on unseen data samples. This suggests that the learned representations capture intrinsic characteristics of phishing websites that are transferable across different contexts, thereby enhancing the models' practical utility.

4.4 Discussion on False Positives and False Negatives

While our machine learning models demonstrated high accuracy in phishing website detection, it is essential to analyze the occurrence of false positives and false negatives, as they have significant implications for the effectiveness of the detection system.

False positives, where legitimate websites are incorrectly classified as phishing, can lead to user inconvenience and loss of trust in the detection system. Conversely, false negatives, where phishing websites are erroneously classified as legitimate, pose serious security risks by allowing malicious content to go undetected.

Mitigating false positives and false negatives requires a careful balance between model sensitivity and specificity. Fine-tuning model thresholds and incorporating post-

processing techniques such as ensemble methods and anomaly detection can help alleviate these issues.

Furthermore, ongoing monitoring and feedback mechanisms are crucial for iteratively improving model performance and adapting to emerging threats. Continuous data collection and model retraining ensure that the detection system remains effective in dynamic environments characterized by evolving phishing tactics and strategies.

In conclusion, the results of our study underscore the effectiveness of machine learning approaches in phishing website detection. However, addressing the challenges posed by false positives, false negatives, and model robustness is essential for enhancing the practical utility and reliability of these detection systems in real-world scenarios.

Furthermore, ongoing monitoring and feedback mechanisms are crucial for iteratively improving model performance and adapting to emerging threats. Continuous data collection and model retraining ensure that the detection system remains effective in dynamic environments characterized by evolving phishing tactics and strategies.

In conclusion, the results of our study underscore the effectiveness of machine learning approaches in phishing website detection. However, addressing the challenges posed by false positives, false negatives, and model robustness is essential for enhancing the practical utility and reliability of these detection systems in real-world scenarios.

4.5 Future Directions

Moving forward, there are several promising avenues for further research in the field of phishing website detection using machine learning. One such direction involves exploring ensemble methods or hybrid approaches that combine the strengths of multiple machine learning models with human expertise. By integrating diverse perspectives and techniques, these hybrid systems have the potential to enhance detection accuracy and robustness against evolving phishing tactics.

Emerging technologies, such as deep learning and explainable AI, offer exciting opportunities for improving the interpretability and performance of detection models. Deep learning architectures, with their ability to automatically learn complex representations from data, could lead to more effective detection of subtle phishing cues. Additionally, the development of explainable AI techniques would enable better understanding of model decisions, enhancing transparency and trust in the detection process.

The integration of contextual information, such as user behavior patterns or website reputation scores, represents another promising direction for enhancing phishing detection strategies. By leveraging additional sources of information beyond website content, detection models can better differentiate between legitimate and malicious activities, thereby improving overall detection accuracy.

Furthermore, interdisciplinary collaborations with experts in psychology, sociology, and cybersecurity hold great potential for advancing phishing detection research. By drawing on insights from diverse disciplines, researchers can gain a deeper understanding of human behavior and attacker tactics, leading to more effective detection and mitigation strategies.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In conclusion, the results of this study highlight the effectiveness of machine Learning algorithms, particularly Random Forest and Decision Trees, in detecting phishing websites. With accuracy rates surpassing 90% and precision scores indicating low false positive rates, these models demonstrate promise in enhancing online security measures. The significance of features such as URL length, presence of "@" symbol, and domain age underscores the importance of considering diverse indicators for robust phishing detection.

However, despite these promising findings, there are notable limitations, including the need for continuous model retraining and adaptation to keep pace with evolving phishing tactics. Additionally, the dataset used may not fully capture the breadth of phishing attacks, potentially affecting the generalizability of the models. Addressing these limitations is essential for deploying effective and resilient phishing detection systems.

5.1 Future Work

Future research directions should focus on addressing the identified limitations and advancing the state-of-the-art in phishing detection. Key areas for future work include:

1. ***Enhanced Model Robustness:*** Investigate methods for improving model robustness against adversarial attacks and novel phishing techniques. This may involve exploring ensemble techniques, deep learning architectures, or hybrid models combining multiple algorithms.
2. ***Real-Time Detection:*** Develop frameworks for real-time phishing detection by integrating models with continuous data feeds and leveraging streaming analytics techniques. This will enable proactive identification.

3. ***Feature Engineering:** Explore novel features and data sources to enhance model performance and adaptability. This may include incorporating behavioral patterns, user interactions, or linguistic cues to improve the accuracy of phishing detection systems.
4. ***Adaptive Learning:** Implement adaptive learning mechanisms to enable models to autonomously adapt to evolving threats and feedback from security analysts. This will ensure that detection systems remain effective in detecting new and sophisticated phishing attacks.
5. ***Evaluation on Diverse Datasets:** Validate model performance on diverse datasets representing a wide range of phishing attacks, including targeted spear-phishing campaigns and brand impersonation tactics. This will provide a more comprehensive understanding of model generalizability and effectiveness across different threat landscapes.

By addressing these research directions, we can advance the capabilities of machine learning-based phishing detection systems and strengthen our defenses against cyber threats, ultimately safeguarding users and organizations from the detrimental impacts of phishing attacks.

REFERENCES

- [1] Lakshmanarao, A., Rao, P.S.P., Krishna, M.M.B. (2021) ‘Phishing website detection using novel machine learning fusion approach’, in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 115 Presented at the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 1164–1169
- [2] H. Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier", 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 383-388, 2019, July
- [3] Vaishnavi, D., Suwetha, S., Jinila, Y.B., Subhashini, R., Shyry, S.P. (2021) ‘A Comparative Analysis of Machine Learning Algorithms on Malicious URL Prediction’, in 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Presented at the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 1398–1402
- [4] Microsoft, Microsoft Consumer safety report. <https://news.microsoft.com/en-sg/2014/02/11/microsoftconsumersafety-index-reveals-impact-of-poor-online-safety-behaviours-insingapore/sm.001xdu50tlxsej410r11kqvks u4nz>.
- [5] Internal Revenue Service, IRS E-mail Schemes. Available at <https://www.irs.gov/uac/newsroom/consumers-warnedof-new-surge-in-irsemail-schemes-during-2016-tax-season-tax-industry-also-targeted>.
- [6] Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. (2007), A comparison of machine learning techniques for phishing detection. Proceedings of the Antiphishing Working Groups 2nd Annual ECrime Researchers Summit on - ECrime ‘07. doi:10.1145/1299015.1299021.
- [7] E., B., K., T. (2015)., Phishing URL Detection: A Machine Learning and Web Mining-based Approach. International Journal of Computer Applications, 123(13), 46-50. Doi:10.5120/ijca2015905665.

- [8] Erzhou Zhu, Yuyang Chen, Chengcheng Ye, Xuejun Li, Feng Liu, —OFSNN: An Effective Phishing Websites Detection Model Based on Optimal Feature Selection and Neural Network, *IEEE Access* (Volume:7), pp. 73271- 73284, June 2019.
- [9] Youness Mourtaji, Mohammed Bouhorma, Alghazzawi, Perception of a new framework for detecting phishing web pages, *Mediterranean Symposium on Smart City Application* Article No. 11, Tangier, Morocco, October 2017. 116
- [10] Akihito Nakamura, Fuma Dobashi, Proactive Phishing Sites Detection, *WI '19 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 443- 448, October 2019.
- [11] Ebubekir Büber, 'Phishing URL Detection with M', [Online]. Available: <https://towardsdatascience.com/phishing-domain-detection-with-ml5be9c99293e5> [Accessed: 10- November- 2019].
- [12] scikit-learn, Machine Learning in Python, [Online]. Available: <https://scikit-learn.org/stable/> [Accessed: 10- November- 2019].
- [13] Mohammed Nazim Feroz, Susan Mengel, —Phishing URL Detection Using URL Ranking, *IEEE International Congress on Big Data*, July 2015.

APPENDIXES

A.SOURCE CODE:

```
# importing the necessary dependencies

from flask import Flask, render_template, request,jsonify

from flask_cors import CORS,cross_origin

import pickle

from features import *

app = Flask(__name__) # initializing a flask app

@app.route('/',methods=['GET']) # route to display the home page

@cross_origin()

def homePage():

    return render_template("index.html")

@app.route('/predict',methods=['POST','GET']) # route to show the

predictions in a web UI

@cross_origin()

def index():

    if request.method == 'POST':

        try:

            # reading the inputs given by the user

            web_link=str(request.form['link'])
```

```

print(web_link)

model = int(request.form['model'])

print(model)

data=featureExtraction(web_link)

print(data)

if model==1:

    filename = 'DecisionTree.pickle'

    loaded_model = pickle.load(open(filename, 'rb')) # loading the model file
    from the storage

    # predictions using the loaded model file

    prediction=loaded_model.predict([data])

    print('prediction is', prediction[0])

else:

    filename = 'RandomForest.pickle'

    loaded_model = pickle.load(open(filename, 'rb')) # loading the model file
    from the storage

    # predictions using the loaded model file

    prediction=loaded_model.predict([data])

    print('prediction is', prediction[0])

if prediction[0]==1:

```

```

return render_template('result.html', pred =True ,link=web_link )

else:

return render_template('result.html', pred =False ,link=web_link )

except Exception as e:

print('The Exception message is: ',e)

return 'something is wrong'

# returnrender_template('results.html')

else:

return render_template('index.html')

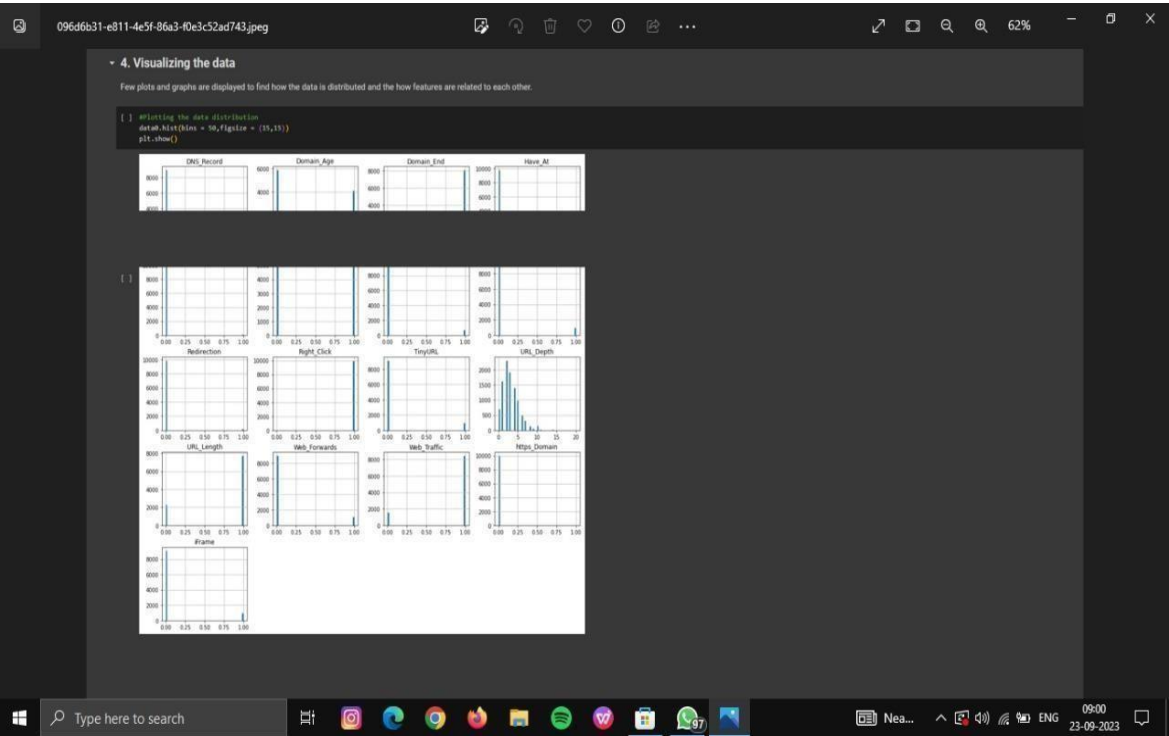
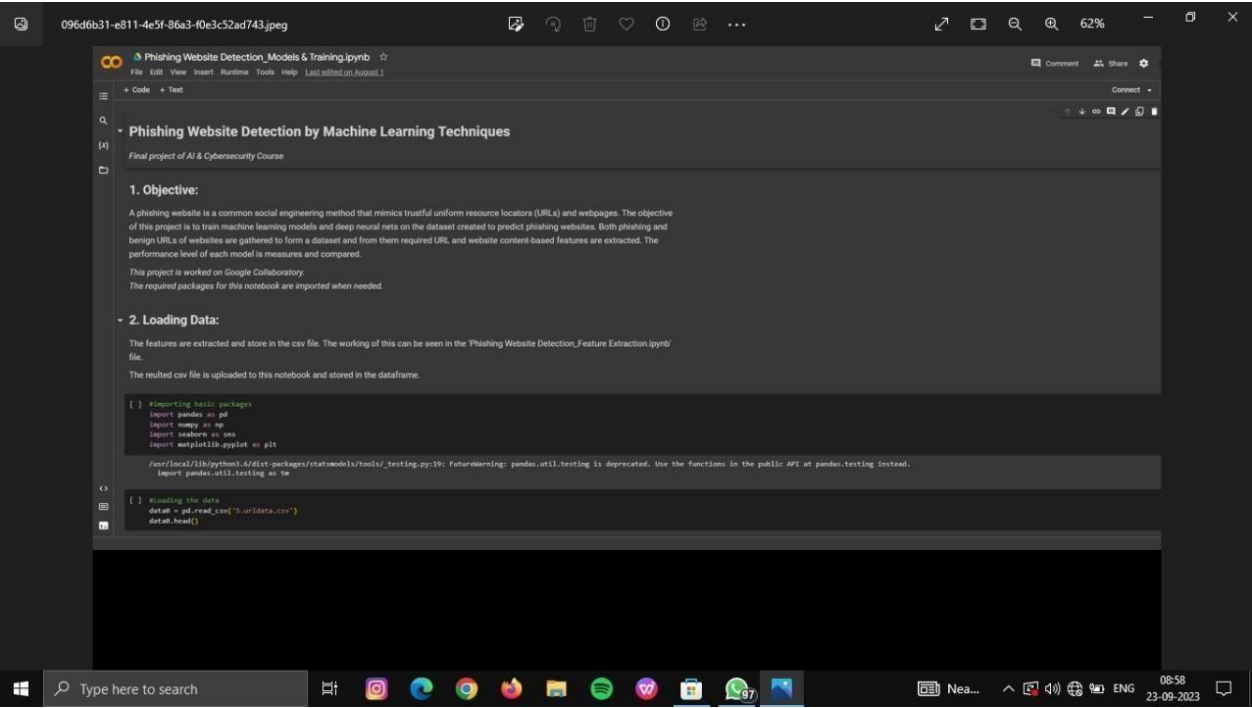
if __name__ == "__main__":

#app.run(host='127.0.0.1', port=8001, debug=True)

app.run(debug=False) # running the app

```

B. SCREENSHOT



5. Data Preprocessing & EDA

Here, we clean the data by applying data preprocessing techniques and transform the data to use it in the models.

```
[ ] data.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Domain_End	Iframe	Mouse_Over	Right_Click	Web_Forwards	Label
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100000	0.045700	0.413700	0.8099	0.090000	0.06600	0.99930	0.105300	0.500000
std	0.073961	0.148632	0.418653	2.128631	0.115400	0.016161	0.286625	0.290727	0.301079	0.361254	0.492521	0.3924	0.267401	0.24934	0.02645	0.300955	0.500025
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000	0.000000	0.00000	0.00000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000	0.000000
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.0000	0.000000	0.00000	1.00000	0.000000	0.500000
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.0000	0.000000	0.00000	1.00000	0.000000	1.000000
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000	1.000000	1.00000	1.00000	1.000000	1.000000

The above obtained result shows that the most of the data is made of 0's & 1's except 'Domain' & 'URL_Depth' columns. The Domain column doesn't have any significance to the machine learning model training. So dropping the 'Domain' column from the dataset.

```
[ ] #Dropping the Domain column
data = data.drop(['Domain'], axis = 1).copy()
```

```
[ ] #Checking the data for null or missing values
data.isnull().sum()
```

```
Have_IP      0
Have_At      0
URL_Length   0
URL_Depth    0
Redirection   0
https_Domain 0
TinyURL      0
Prefix/Suffix 0
DNS_Record   0
Web_Traffic  0
Domain_Age   0
Domain_End   0
Iframe       0
Mouse_Over   0
Right_Click  0
Web_Forwards 0
Label        0
dtype: int64
```

6. Splitting the Data

```
[ ] # Separating & assigning features and target columns to X & y
y = data['label']
X = data.drop('label', axis=1)
X.shape, y.shape

((10000, 16), (10000,))
```

```
[ ] # Splitting the dataset into train and test sets: 80-20 split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.2, random_state = 12)
X_train.shape, X_test.shape

((8000, 16), (2000, 16))
```

7. Machine Learning Models & Training

From the dataset above, it is clear that this is a supervised machine learning task. There are two major types of supervised machine learning problems, called classification and regression.

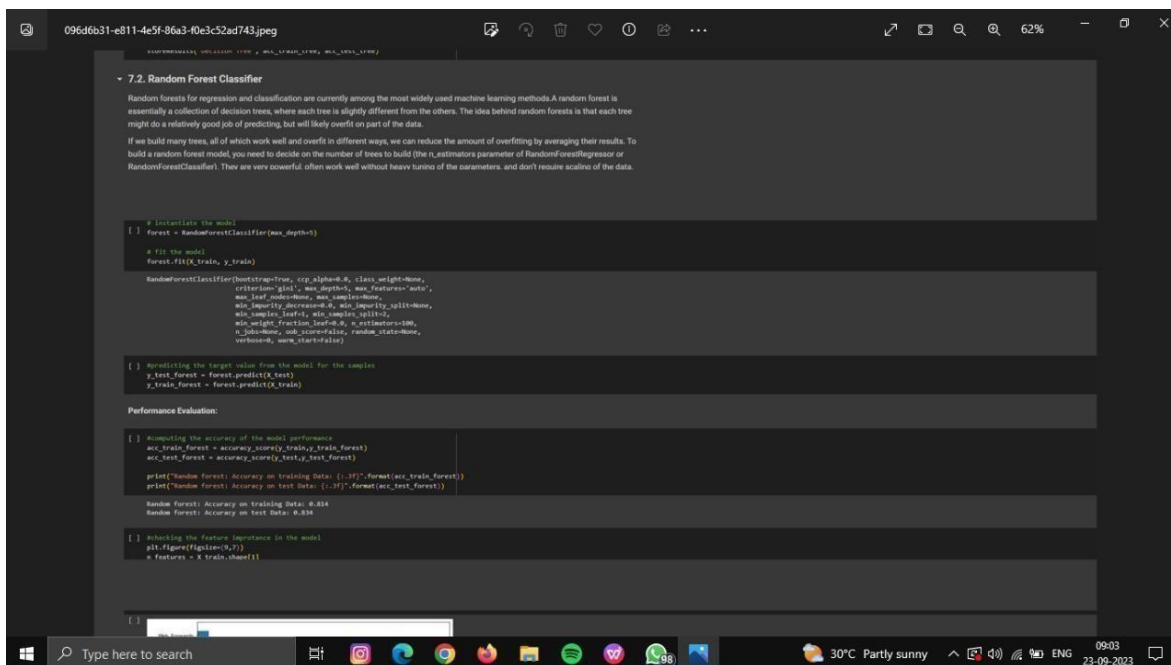
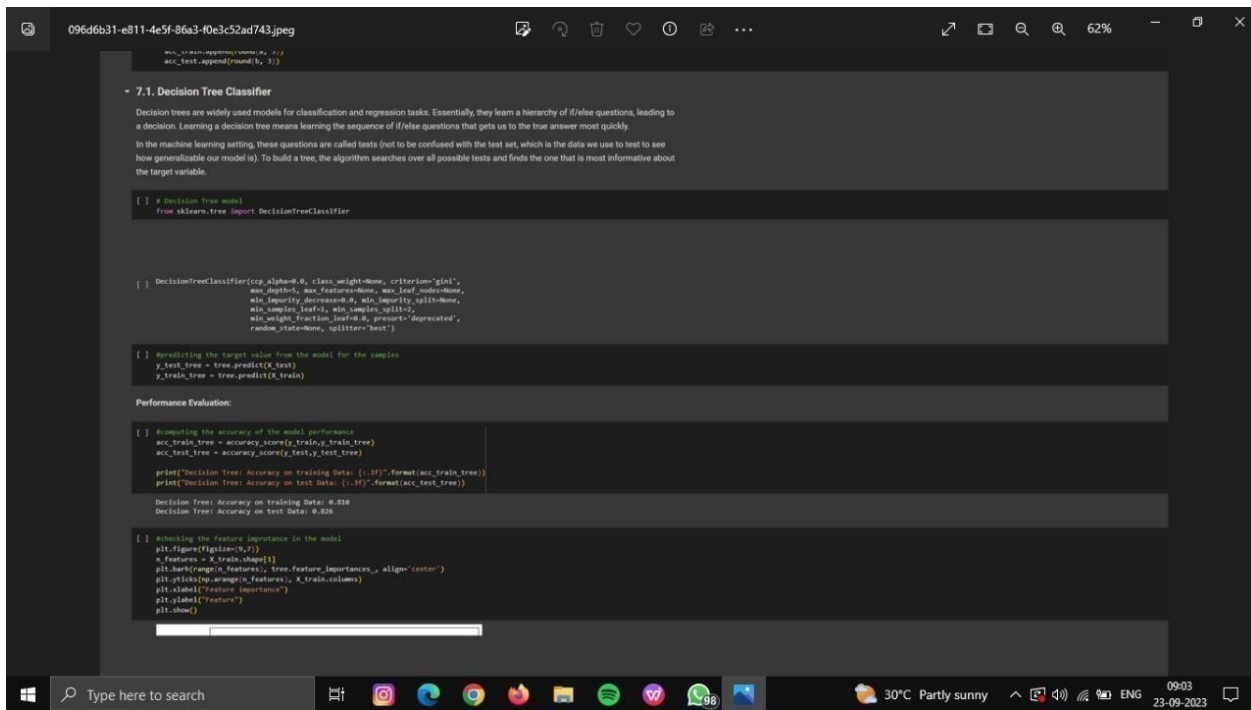
This data set comes under classification problem, as the input (URL) is classified as malicious (1) or legitimate (0). The extended machine

- Multilayer Perceptrons
- XGBoost
- Autoencoder Neural Network
- Support Vector Machines

```
[ ] #Importing packages
from sklearn.metrics import accuracy_score
```

```
[ ] # Creating holders to store the model performance results
ML_Model = []
acc_train = []
acc_test = []

#function to call for storing the results
def store_results(model, a,b):
    ML_Model.append(model)
    acc_train.append(round(a, 3))
    acc_test.append(round(b, 3))
```



```
096d6b31-e811-4e5f-86a3-40e3c52ad743.jpeg

Storing the results:

[] #storing the results. The below mentioned order of parameter passing is important.
#caution: Execute only once to avoid duplications.
storeResults('Random forest', acc_train_forest, acc_test_forest)

7.3. Multilayer Perceptrons (MLPs): Deep Learning

Multilayer perceptrons (MLPs) are also known as (vanilla) feed-forward neural networks, or sometimes just neural networks. Multilayer

[] # Multilayer Perceptrons model
from sklearn.neural_network import MLPClassifier

# instantiate the model
mlp = MLPClassifier(alpha=0.001, hidden_layer_sizes=(100,100,100))

# fit the model
mlp.fit(X_train, y_train)

MLPClassifier(alpha=0.001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100, 100, 100), learning_rate='constant',
              learning_rate_init=0.001, max_fun=1000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterov_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)

[] #predicting the target value from the model for the samples
y_test_mlp = mlp.predict(X_test)
y_train_mlp = mlp.predict(X_train)

Performance Evaluation:

[] #computing the accuracy of the model performance
acc_train_mlp = accuracy_score(y_train,y_train_mlp)
acc_test_mlp = accuracy_score(y_test,y_test_mlp)

print("Multilayer Perceptrons: Accuracy on training Data: {:.3f}".format(acc_train_mlp))
print("Multilayer Perceptrons: Accuracy on test Data: {:.3f}".format(acc_test_mlp))

Multilayer Perceptrons: Accuracy on training Data: 0.859
Multilayer Perceptrons: Accuracy on test Data: 0.863
```

```
096d6b31-e811-4e5f-86a3-40e3c52ad743.jpeg

[] storeResults('Multilayer Perceptrons', acc_train_mlp, acc_test_mlp)

7.4. XGBoost Classifier

XGBoost is one of the most popular machine learning algorithms these days. XGBoost stands for eXtreme Gradient Boosting. Regardless of the type of prediction task at hand, regression or classification, XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.

[] #XGBoost Classification model
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier(learning_rate=0.4,max_depth=7)

#fit the model
xgb.fit(X_train, y_train)

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bynode=1, gamma=0,
              learning_rate=0.4, max_delta_step=0, max_depth=7,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)

[] #predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)

Performance Evaluation:

[] #computing the accuracy of the model performance

[]

XGBoost: Accuracy on training Data: 0.864
XGBoost : Accuracy on test Data: 0.864

Storing the results:

[] #storing the results. The below mentioned order of parameter passing is important.
#caution: Execute only once to avoid duplications.
storeResults('XGBoost', acc_train_xgb, acc_test_xgb)

7.5. Autoencoder Neural Network
```

096d6b31-e811-4e5f-86a3-0e3c52ad743.jpeg

7.5. Autoencoder Neural Network

An auto encoder is a neural network that has the same number of input neurons as it does outputs. The hidden layers of the neural network will have fewer neurons than the input/output neurons. Because there are fewer neurons, the auto-encoder must learn to encode the input to the fewer hidden neurons. The predictors (x) and output (y) are exactly the same in an auto encoder.

```
[ ] #importing required packages
import keras
from keras.layers import Input, Dense
from keras import regularizers
import tensorflow as tf
from keras.models import Model
from sklearn import metrics

Using TensorFlow backend.
```

```
[ ] #building autoencoder model

input_dim = X_train.shape[1]
encoding_dim = input_dim

#auto_encoder = AutoEncoder(input_dim, encoding_dim)
```

```
[ ] code = Dense(int(encoding_dim/4), activation='relu')(encoder)
decoder = Dense(int(encoding_dim/2), activation='relu')(code)

decoder = Dense(int(encoding_dim), activation='relu')(decoder)
decoder = Dense(input_dim, activation='relu')(decoder)
autoencoder = Model([inputs=input_layer, outputs=decoder])
autoencoder.summary()
```

Model: "model_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 16)	0
dense_1 (Dense)	(None, 16)	272
dense_2 (Dense)	(None, 16)	272
dense_3 (Dense)	(None, 16)	258
dense_4 (Dense)	(None, 16)	240
dense_5 (Dense)	(None, 16)	272

Total params: 1,294
Trainable params: 1,294
Non-trainable params: 0

Type here to search

30°C Partly sunny 09:05 23-09-2023

096d6b31-e811-4e5f-86a3-0e3c52ad743.jpeg

```
[ ] #compiling the model
autoencoder.compile(optimizer='adam',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])

#training the model
history = autoencoder.fit(X_train, X_train, epochs=10, batch_size=64, shuffle=True, validation_split=0.2)
```

```
Epoch 1/10 [ ] - 0s 24us/step - loss: 1.0514 - accuracy: 0.7900 - val_loss: 1.3147 - val_accuracy: 0.8149
Epoch 2/10 [ ] - 0s 24us/step - loss: 1.3118 - accuracy: 0.8200 - val_loss: 1.3532 - val_accuracy: 0.8128
Epoch 3/10 [ ] - 0s 25us/step - loss: 1.3789 - accuracy: 0.8168 - val_loss: 1.4738 - val_accuracy: 0.8190
Epoch 4/10 [ ] - 0s 25us/step - loss: 1.4435 - accuracy: 0.8187 - val_loss: 1.5188 - val_accuracy: 0.8204
Epoch 5/10 [ ] - 0s 25us/step - loss: 1.4951 - accuracy: 0.8215 - val_loss: 1.5681 - val_accuracy: 0.8240
Epoch 6/10 [ ] - 0s 25us/step - loss: 1.5288 - accuracy: 0.8192 - val_loss: 1.5912 - val_accuracy: 0.8236
Epoch 7/10 [ ] - 0s 25us/step - loss: 1.5844 - accuracy: 0.8148 - val_loss: 1.5868 - val_accuracy: 0.8191
Epoch 8/10 [ ] - 0s 25us/step - loss: 1.5554 - accuracy: 0.8214 - val_loss: 1.6153 - val_accuracy: 0.8289
```

Performance Evaluation:

```
[ ] acc_train_auto = autoencoder.evaluate(X_train, X_train)[1]
acc_test_auto = autoencoder.evaluate(X_test, X_test)[1]

print('Autoencoder: Accuracy on training Data: {:.3f}'.format(acc_train_auto))
print('Autoencoder: Accuracy on test Data: {:.3f}'.format(acc_test_auto))
```

```
8000/8000 [ ] - 0s 18us/step
2000/2000 [ ] - 0s 20us/step

Autoencoder: Accuracy on training Data: 0.819
Autoencoder: Accuracy on test Data: 0.818
```

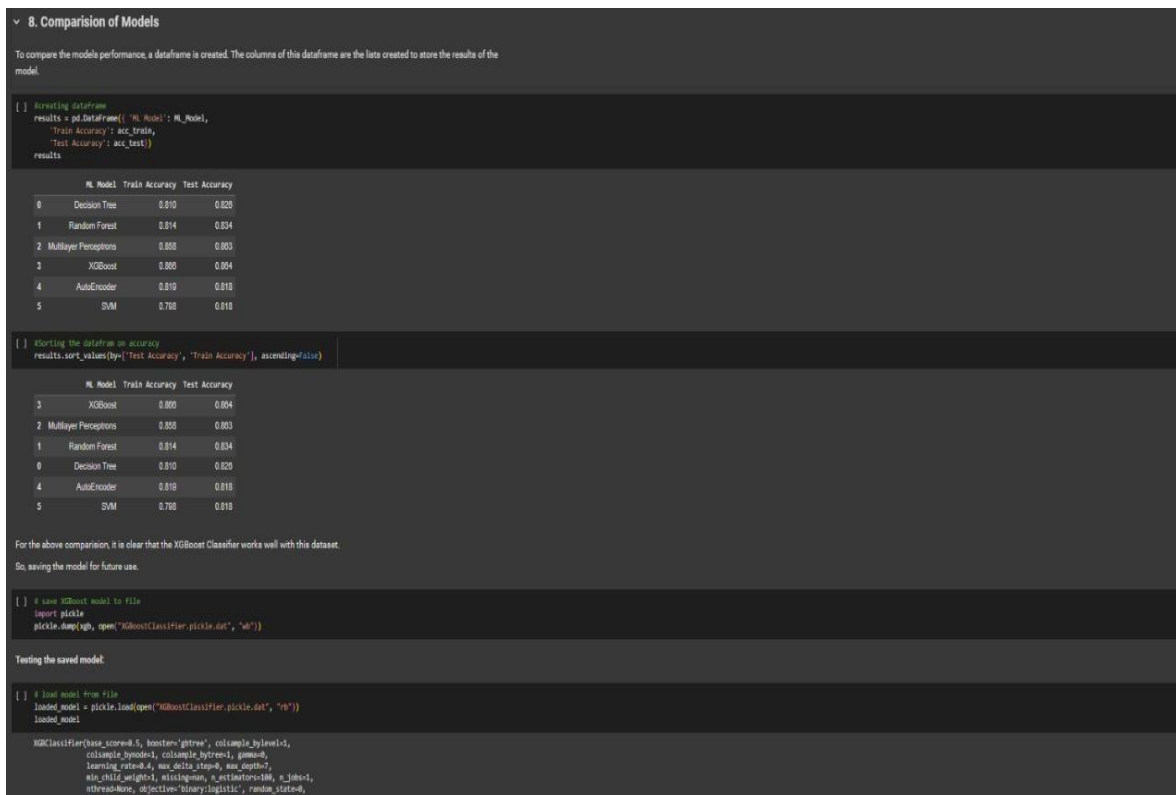
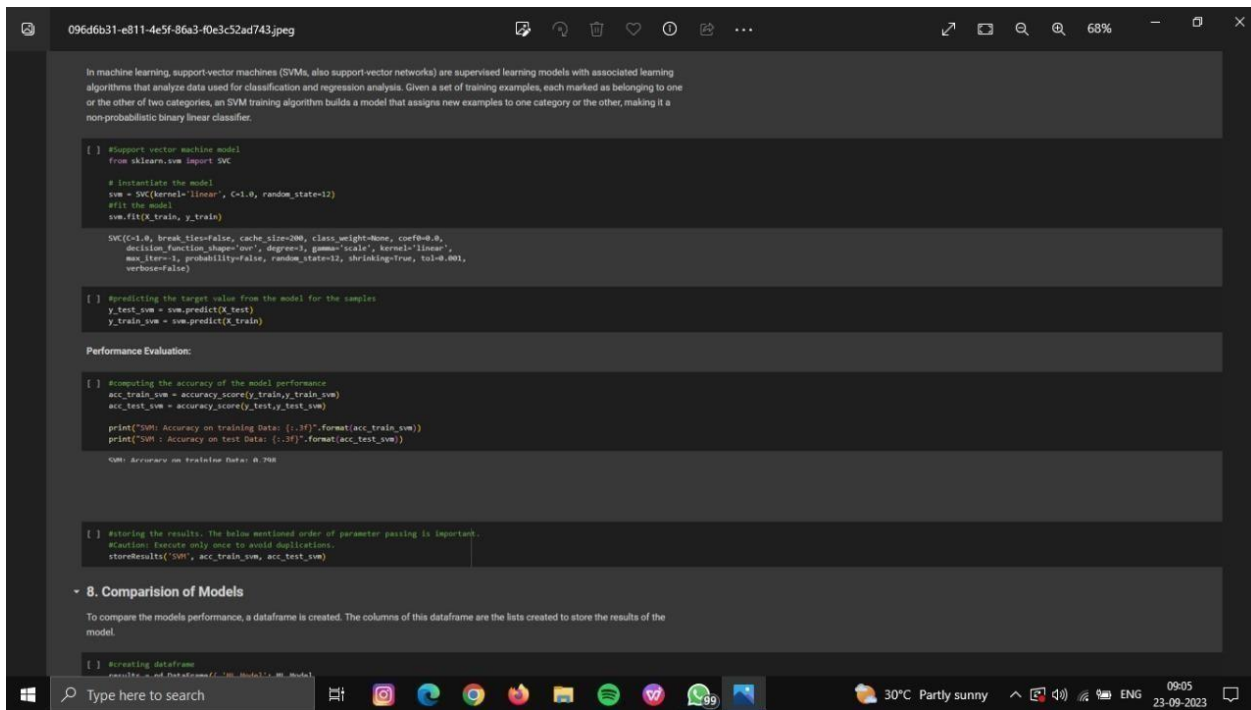
Storing the results:

```
[ ] #storing the results. The below mentioned order of parameter passing is important.
#filename, weights only, save as, build, duplicates
```

In machine learning, support-vector machines (SVMs), also support-vector networks) are supervised learning models with associated learning

Type here to search

30°C Partly sunny 09:05 23-09-2023



C.PUBLICATION WITH PLAGIARISM REPORT

PHISHING WEBSITE DETECTION USING MACHINE LEARNING

Sanjay Kumar S M
U. G Scholar
Department of Information Technology
Sathayabama Institute of Science and
Technology
Chennai, India
smsk5323@gmail.com

Merlin Mary Jenitha
Asst.Professor
Department of Information Technology
Sathayabama Institute of Science and
Technology
Chennai, India
jenithajp2001@gmail.com

Vishvashwarran V B
U.G Scholar
Department of Information Technology
Sathayabama Institute of Science and
Technology
Chennai, India
vishva3611@gmail.com

Abstract: Phishing websites have proven to be a major security concern. Several cyber-attacks risk the confidentiality, integrity, and availability of company and consumer data, and phishing is the beginning point for many of them. Many researchers have spent decades creating unique approaches to automatically detect phishing websites. While cutting-edge solutions can deliver better results, they need a lot of manual feature engineering and aren't good at identifying new phishing attacks. As a result, finding strategies that can automatically detect phishing websites and quickly manage zero-day phishing attempts is an open challenge in this field. The web page in the URL which hosts that contains a wealth of data that can be used to determine the web server's maliciousness. Machine Learning is an effective method for detecting phishing. It also eliminates the disadvantages of the previous method. We conducted a thorough review of the literature and suggested a new method for detecting phishing websites using features extraction and a machine learning algorithm. The goal of this research is to use the dataset collected to train ML models and deep neural nets to anticipate phishing websites.

Keywords: Artificial Intelligence, Machine Learning, Web Application, Django

Framework, Obstructive Sleep Apnea Prediction, Python Programming Language, Graphical User Interface.

I. Introduction

Phishing stands out as a highly perilous criminal activity in the realm of cyberspace, particularly with the increasing trend of users relying on online platforms for accessing government and financial services.

This surge has witnessed a notable uptick in phishing attacks over recent years, transforming this malicious practice into a lucrative business for cybercriminals. Exploiting various methods such as messaging, VOIP, spoofed links, and the creation of counterfeit websites, phishers employ tactics that easily deceive vulnerable users.

Crafting counterfeit websites that closely resemble genuine ones, both in terms of layout and content, has become alarmingly simple. These fraudulent sites are designed to be virtually indistinguishable from their legitimate counterparts, making it challenging for users to identify the deception.

The primary motive behind these deceptive websites is to illicitly obtain private data, including account numbers, login credentials, and debit/credit card passwords. Attackers often pose as implementing high-level security measures, requesting users to respond to security questions, leading users to unwittingly fall into the trap of phishing attacks.

This intricate web of phishing techniques underscores the urgency of addressing cybersecurity threats. Vigilance and user education play crucial roles in navigating the digital landscape safely, as attackers continually refine their method detecting and thwarting phishing attacks require a multi-faceted approach, combining technological solutions with informed user practices to fortify defenses against the evolving tactics employed by cybercriminals.

Phishing has emerged as a critical challenge, posing threats to individuals, businesses, and even entire nations. The proliferation of diverse online services, including banking, entertainment, education, software downloads, and social networking, has fueled the rapid evolution of the internet. This surge in online activity results in a constant exchange of a vast amount of data. Cybercriminals exploit this scenario through social engineering techniques, often employing deceptive emails impersonating reputable businesses and agencies.

These emails aim to redirect users to fraudulent websites, tricking them into divulging sensitive financial information such as usernames and passwords. The tactics employed in phishing are multifaceted. Beyond social engineering, technical maneuvers involve the deployment of malicious software on computers, enabling the direct theft of credentials. This malicious software, often in the form of malware, is designed to compromise systems and intercept users' online account information, including usernames and passwords. As a consequence, the threat landscape becomes more sophisticated, with attackers employing various strategies to exploit the vast volume of data transmitted and downloaded on the internet.

In essence, the evolving nature of phishing necessitates heightened awareness and robust cybersecurity measures. Addressing this multifaceted challenge requires a combination of user education, advanced technological

solutions, and Creating awareness among users is another crucial aspect of phishing prevention. Educating users on how to identify potential phishing websites through visual cues, like checking for misspelled URLs or verifying the website's security indicators, can significantly reduce the risk of falling victim to phishing attacks.

In conclusion, the fight against phishing attacks involves a multi-faceted approach, with machine learning algorithms playing a pivotal role in detecting fraudulent websites. Combining advanced detection methods with user awareness initiatives can create a more robust defense against phishing threats in the online ecosystem.

II.Literature Review

Ashit Kumar Dutta et al., "Detecting phishing websites using machine learning technique"

In this research framework, the utilization of Recurrent Neural Network with Long Short-Term Memory (RNN-LSTM) is proposed as an effective means to identify specific properties (Pm and Pl) within an URL. The objective is to distinguish between malicious and legitimate websites based on these identified properties. The findings of this study demonstrate that the proposed method outperforms existing deep learning approaches in terms of accuracy and effectiveness.

Rishikesh Mahajan et al., "Phishing Website Detection using Machine Learning Algorithms"

This study focuses on the detection of phishing websites through the application of

machine learning algorithms, specifically utilizing Decision Tree as a splitter. . The URLs of benign websites were sourced from www.alexa.com, while phishing website URLs were gathered from www.phishtank.com. The implemented Python program extracts features from the URLs to facilitate the detection process.

Dželila Mehanović* | Jasmin Kevrić et al., "Phishing Website Detection Using Machine Learning Classifiers Optimized by Feature Selection "

To identify crucial features for our phishing website detection model, we employed the Weka tool and its algorithms for feature selection. Utilizing the K-Nearest Neighbor (KNN) algorithm, our approach aimed at effectively discerning phishing websites from legitimate ones. To pinpoint the most valuable features, we incorporated multiple feature selection filters into our methodology. The outputs generated by these filters underwent thorough analysis, revealing features identified as the most significant for the task at hand. These selected features play a pivotal role in enhancing the accuracy and efficacy of our phishing detection model, contributing to a more robust defense against deceptive online practices.

K.Venkateshwara Rao et al., " Machine Learning-Based Detection of Phishing Attacks: A Novel Approach" This study presents an innovative approach for detecting phishing attacks through the application of machine learning techniques.

The proposed method employs a parallel decision tree structure, where multiple decision trees process input data to generate specific classes. The collaborative output of these trees results in the classification of various instances.

Additionally, the utilization of Support Vector Machine (SVM) demonstrated a commendable accuracy of 91.3% on the test dataset. This high level of accuracy significantly contributes to the effectiveness of the model in accurately identifying and classifying phishing attacks.

The results underscore the potential of machine learning techniques, particularly the parallel decision tree approach and SVM, in enhancing the precision and reliability of phishing attack detection systems.

Deepika G et al., "Phishing Website Detection Using Machine Learning Algorithms"

This research focuses on the detection of phishing websites employing machine learning algorithms. The study involves identifying the best-performing module, which is subsequently tested on a separate dataset to evaluate its performance in real-world scenarios.

Machine learning models exhibit adaptability to new and evolving phishing techniques, as they can be retrained using updated datasets, ensuring their effectiveness is maintained over time.

This adaptability proves crucial in the ever-changing landscape of cybersecurity, enabling the continuous improvement of phishing detection systems.

In essence, this literature review synthesizes a rich tapestry of research endeavors, each contributing uniquely to the broader understanding of obstructive sleep apnea. From machine learning applications to the imperative need for explain ability in clinical decision support systems and the exploration of deep learning effectiveness, these studies collectively pave the way for advancements in sleep disorder diagnostics. The multidimensional insights gained from these endeavors collectively lay the foundation for the subsequent development and evaluation of the proposed sleep apnea detection system, infusing the field with innovation and critical perspectives.

III. Proposed Methodology

The methodology used in this research involves studying phishing websites, which are fake sites that mimic legitimate ones to trick users into revealing sensitive information. One common tactic used by phishers is manipulating the URLs of these fake sites. They can change different parts of the URL, such as sub-domains, to make the fake site look more convincing.

For the development process, the research adopts the waterfall model, which is a traditional approach where each phase of development flows in a linear sequence. This model is chosen because the project has relatively clear and stable requirements. The application being developed is broken down into smaller parts, which are created using a combination of existing frameworks and custom-written code. This allows for a structured development process where different components are built separately and then integrated together.

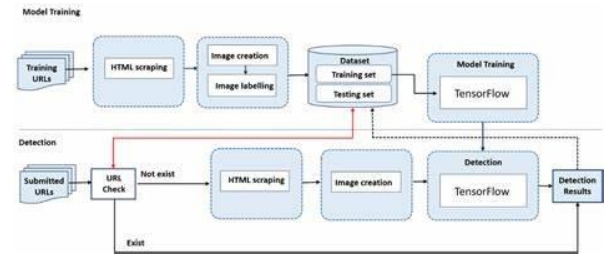


Fig 1 Block Diagram of our proposed method

IV. Module 1: Dataset Gathering

Gathering a dataset for detecting phishing websites involves collecting a set of web pages that includes both legitimate websites and known phishing sites. This dataset should represent a variety of features and characteristics found in real-world web pages.

To gather this dataset, researchers typically employ web scraping techniques to collect URLs and associated metadata from various sources, including search engine results, publicly available databases of known phishing sites, and web archives. Additionally, they may utilize web crawlers to navigate through websites and extract relevant information such as HTML content, text, images, and links.

Module 2: Data Preparation

Preparing data for phishing website detection involves process ensures that the dataset is structured and suitable for training machine learning models to accurately identify phishing attempts. Collecting a diverse dataset containing examples of both legitimate websites and known phishing sites, cleaning the data to remove duplicates and irrelevant information, extracting relevant features such as URL characteristics and website content, engineering features to enhance predictive power, splitting the dataset into training, validation, and testing sets, and optionally balancing classes to address any imbalance.

Module 3 : Exemplary Execution

Exemplary execution in phishing website detection involves a meticulous approach encompassing thorough data collection, precise cleaning and feature extraction, rigorous model training and validation, continuous improvement through staying updated with the latest research, and maintaining transparency and ethical standards, ensuring the highest levels of accuracy and effectiveness in identifying and combating phishing threats.

Module 4: Internet Application

An internet application for phishing detection is a web-based tool designed to identify suspicious websites aiming to trick users into revealing personal information. It provides real-time alerts to users and administrators, helping them recognize and avoid phishing scams.

Module 5: Predictions for Our Outcome

Predictions for our outcome in phishing website detection involve estimating how effective our methods will be in accurately spotting and preventing phishing attempts. This includes analyzing data, evaluating detection algorithms, and predicting the system's performance in real-world scenarios. These predictions guide our efforts to refine and improve our detection system, ensuring it effectively safeguards users against phishing attacks.

V. Results and Discussion

The results and discussion section for phishing website detection involves analyzing the effectiveness of various detection methods, including machine learning algorithms like Naïve Bayesian, SVM, Decision Trees, and

Random Forest, as well as novel systems like Phish Score and Phish Checker.

This analysis includes metrics such as accuracy, precision, recall, and F1-score to evaluate performance.

The discussion delves into the strengths, limitations, and implications of each approach, emphasizing the need for continuous improvement to address evolving phishing threats effectively.

Additionally, the discussion delves into the strengths and limitations of each approach, identifying factors that contribute to detection accuracy and areas for improvement. Researchers often explore the impact of feature selection and combination on detection performance, highlighting the importance of extracting relevant features from website data. Furthermore, the discussion may address the challenges posed by evolving phishing tactics and the need for adaptive detection systems capable of detecting new threats.

Moreover, the discussion may touch upon the implications of the findings for cybersecurity practices, emphasizing the importance of continually updating and refining detection methodologies to address emerging threats effectively. Collaborative efforts between researchers, industry professionals, and policymakers are essential to developing robust detection systems that can combat the ever-evolving landscape of phishing attacks. Overall, the results and discussion section provides valuable insights into the efficacy of different detection approaches and guides future research directions in the field of phishing website detection.

VI. Future Work

Future work for phishing website detection entails several areas of focus to enhance the effectiveness of detection systems. One aspect

is the exploration of advanced machine learning techniques, such as deep learning, to improve detection accuracy and adaptability to evolving phishing tactics.

Additionally, research could delve into the integration of real-time data sources and dynamic feature extraction.

Methods to enhance the timeliness and Conclusion.

This survey explored various studies conducted in the field of machine learning aimed at detecting phishing websites. The researchers reviewed multiple papers and identified that a majority of the work utilizes well-established machine learning algorithms such as Naïve Bayesian, Support Vector Machines (SVM), Decision Trees, and Random Forest. These algorithms have been widely applied due to their effectiveness in analyzing and classifying website data to distinguish between legitimate and phishing sites. Moreover, some researchers proposed novel systems like Phish Score and Phish Checker, which introduce innovative approaches to phishing detection beyond the traditional algorithms. These new systems often incorporate combinations of features, considering metrics like accuracy, precision, and recall to enhance detection capabilities.

In addition to algorithmic approaches, the survey highlighted the importance of feature selection and combination in improving detection accuracy. Researchers have experimented with various combinations of features extracted from website data to optimize detection performance. These features may include URL characteristics, website content analysis, domain age, SSL certificate status, and others, all of which contribute to identifying phishing attempts effectively.

Furthermore, the survey summarized experimentally successful techniques employed in detecting phishing website URLs. These techniques represent a collective effort by researchers to develop robust methods for identifying and mitigating phishing threats.

However, as the number of phishing websites continues to rise, there is a growing need to adapt and evolve detection systems to keep pace with emerging threats.

Moreover, there is a need to develop comprehensive frameworks for evaluating detection systems, including standardized datasets and benchmarking methodologies, to facilitate fair comparisons and drive innovation in the field. Collaborative efforts between academia, industry, and regulatory bodies are essential to address privacy concerns and ensure the ethical development and deployment of phishing detection technologies.

Furthermore, future research could explore the integration of user behavior analysis and human-in-the-loop approaches to enhance detection capabilities and reduce false positives. Additionally, there is a growing need for interdisciplinary research that considers socio-technical factors, such as user awareness and education, to complement technical solutions and mitigate phishing risks effectively.

Overall, future work in phishing website detection should focus on advancing detection. This necessitates the continual evaluation and enhancement of existing techniques and the incorporation of new features or methodologies to stay ahead of evolving phishing tactics. Therefore, ongoing research and innovation in the field of machine learning for phishing detection are crucial to effectively combat this persistent cybersecurity challenge.

VII.Reference

- [1] 'APWG | Unifying The Global Response To Cybercrime' (n.d.) available: <https://apwg.org/>
- [2] 14 Types of Phishing Attacks That IT Administrators Should Watch For [online] (2021)<https://www.blog.syscloud.com>, available: <https://www.blog.syscloud.com/types-of-phishing/>
- [3] Lakshmanarao, A., Rao, P.S.P., Krishna, M.M.B. (2021) 'Phishing website detection using novel machine learning fusion approach', in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Presented at the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 1164–1169.
- [4] H. Chapla, R. Kotak and M. Joiser, "A Machine Learning Approach for URL Based Web Phishing Using Fuzzy Logic as Classifier", 2019 International Conference on Communication and Electronics Systems (ICCES), pp. 383-388, 2019, July.
- [5] Vaishnavi, D., Suwetha, S., Jinila, Y.B., Subhashini, R., Shyry, S.P. (2021) 'A Comparative Analysis of Machine Learning Algorithms on Malicious URL Prediction', in 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Presented at the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 1398–1402.
- [6] Microsoft, Microsoft Consumer safety report. <https://news.microsoft.com/en-sg/2014/02/11/microsoft-consumersafety-index-reveals-impact-of-poor-online-safety-behaviours-in-singapore/sm.001xdu50tlxsej410r11kqvksu4nz>.
- [7] Internal Revenue Service, IRS E-mail Schemes. Available at <https://www.irs.gov/uac/newsroom/consumers-warned-of-new-surge-in-irs-email-schemes-during-2016-tax-season-tax-industry-also-targeted>.
- [8] Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. (2007), A comparison of machine learning techniques for phishing detection. Proceedings of the Anti-phishing Working Groups 2nd Annual ECrime Researchers Summit on - ECrime '07. doi:10.1145/1299015.1299021.
- [9] E., B., K., T. (2015)., Phishing URL Detection: A Machine Learning and Web Mining-based Approach. International Journal of Computer Applications, 123(13), 46-50. doi:10.5120/ijca2015905665.
- [10] Wang Wei-Hong, LV Yin-Jun, CHEN Hui-Bing, FANG Zhao-Lin., A Static Malicious Javascript Detection Using SVM, In proceedings of the 2nd International Conference on Computer Science and Electrical Engineering (ICCSEE 2013).

RE-2022-222704 - Turnitin Plagiarism Report

by Vishvashwarran

Submissiondate: 26-Mar-2024
07:56AM (UTC+0300)Submission
ID: 271711448982
Filename: RE-2022-
222704.docx (48.19K)
Word count: 2808
Character count: 18250

RE-2022-222704-plag-report

ORIGINALITY REPORT

SIMILARITY INDEX **5%** **8** INTERNET SOURCES% **7** PUBLICATIONS% **6%** STUDENT PAPERS

PRIMARY SOURCES

1 sist.sathyabama.ac.in Internet Source **9%**

2 fastercapital.com Internet Source **1%**

3 www.mdpi.com Internet Source **1%**

4 iieta.org Internet Source **1%**

5 Submitted to Staffordshire University Student Paper **1%**

6 Mobtahej, Pooyan. "Deep Learning-Based Anomaly Detection for Compressors Health Condition Monitoring", Lamar University Beaumont, 2023 1%

Publication

7 Submitted to Sheffield Hallam University Student Paper 1%

8 Ahsan Nazir, Jingsha He, Nafei Zhu, AhsanWajahat, Xiangjun Ma, Faheem Ullah, 1% Sirajuddin Qureshi, Muhammad Salman Pathan. "Advancing IoT security: A systematic review of machine learning approaches for the detection of IoT botnets", Journal of King Saud University - Computer and Information Sciences, 2023 1%

Publication

9 Eman Abdullah Aldakheel, Mohammed Zakariah, Ghada Abdalaziz Gashgari, Fahdah A. Almarshad, Abdullah I. A. Alzahrani. "A Deep Learning-Based Innovative Technique for Phishing Detection in Modern Security with Uniform Resource Locators", Sensors, 2023 1%

Publication

10	Submitted to South Dakota Board of Regents	Student Paper	<1%
11	researcher.manipal.edu	Internet Source	<1%
12	www.ijert.org	Internet Source	<1%
13	discovery.researcher.life	Internet Source	<1%
14	www.researchgate.net	Internet Source	<1%
15	informatica.si	Internet Source	<1%
16	D. Vaishnavi, S. Suwetha, Y. Bevis Jinila, R. Subhashini, S. Prayla Shyry. "A Comparative Analysis of Machine Learning Algorithms on Malicious URL Prediction", 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021	Publication	<1%

17 cps-vo.org Internet Source

<1%

18 wjps.uowasit.edu.iq Internet Source

<1%

19 "Advances in Communication and Computational Technology", Springer Science and Business Media LLC, 2021

Publication

<1%

20 Alok Singh Sengar, Abhishek Bhola, Ratnesh Kumar

Shukla, Anurag Gupta. "A Review on Phishing Websites Revealing through Machine Learning", 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), 2021

Publication

<1%

21 He, Mingju. "Spectrum Awareness in a Complex Radio

Environment: A Deep Learning Approach", Stevens Institute of

Technology, 2023

Publication

22 Tarun Choudhary, Siddhesh Mhapankar, Rohit Bhddha, Ashish Kharuk, Rohini Patil. "Machine Learning Approach for Phishing Attack Detection", Journal of Artificial Intelligence and Technology, 2023

Publication

<1%

23 researchr.org Internet Source

<1%

Exclude quotes On
Exclude bibliography On

Exclude matches Off