DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING

PROJECT REPORT

Submitted by

DHINAKARAN V (21EUCS504)

LAKSHAN BALAJI R K (20EUCS069)

SRUTHI R (21EUCS514)

in partial fulfilment for the award of the

degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai 600025)

MARCH 2024

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution. Affiliated to Anna University, Chennai)
Kuniamuthur, Coimbatore - 641 008

BONAFIDE CERTIFICATE

Certified that this project report **"DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING"** is the bonafide work of "**SRUTHI R(21EUCS514), DHINAKARAN V(21EUCS504), LAKSHAN BALAJI R K(20EUCS069)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. K. SASI KALA RANI, M.E, Ph.D.,	**Mr. RAMESH K, M.E, (Ph.D.),**
HEAD OF THE DEPARTMENT	**SUPERVISOR**

Department of Computer Science and Engineering

Sri Krishna College of Engineering and Technology Kuniamuthur,

Coimbatore

Submitted for the Project Viva-Voce examination held on _____

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our sincere thanks to the management and **Dr. S. SOPHIA, M.E., Ph.D.,** Principal, Sri Krishna College of Engineering and Technology, Coimbatore for providing us the facilities to carry out this project work.

We are highly indebt to **Dr. K. SASI KALA RANI, M.E., Ph.D.,** Head of Computer Science and Engineering for her continuous evaluation, valuable suggestions and comments given during the course of the project work.

We are thankful to **M. VENGATESHWARAN, M.E., (Ph.D.),** Project Co-ordinator**,** Department of Computer Science and Engineering for his continuous evaluation, valuable suggestions and comments given during the course of the project work.

We express our deep sense of gratitude to our guide **Mr. RAMESH K, M.E, (Ph.D.),** Professorin the department of Computer science and Engineering for her valuable advice, guidance and support during the course of our project work.

By this, we express our heartfelt sense of gratitude and thanks to our beloved parents, family and friends who have all helped in collecting the resources and materials needed for this project and for their support during the study and implementation this project.

ABSTRACT

In recent years there has been a huge increase in the number of phishing websites. Phishing is a widespread tactic used to trick gullible people into disclosing their personal information by using bogus websites. Phishing website URLs are designed to steal personal data, including user names, passwords, and online financial activities. Phishers employ websites that resemble those genuine websites both aesthetically and linguistically. Utilizing anti-phishing methods to identify phishing is necessary to stop the rapid advancement of phishing techniques as a result of advancing technology. A strong tool for thwarting phishing assaults is machine learning. Attackers frequently use phishing because it is simpler to fool a victim into clicking a malicious link that looks authentic than to try to get past a computer's security measures. The malicious links within the message body are intended to appear to go to the spoofed company utilizing that company's logos and other genuine information. In the method that is being presented, machine learning is used to create a revolutionary approach for detecting phishing websites. Multiple models were compiled and utilized in our suggested strategy to identify phishing websites based on aspects of URL significance. By extracting and comparing different characteristics between legitimate and phishing URLs, the suggested method uses the combined model to identify phishing URLs. The studies' findings demonstrate that the suggested approach successfully identifies legitimate websites from bogus ones in real time.

(12) PATENT APPLICATION PUBLICATION
(19) INDIA
(22) Date of filing of Application :07/03/2024

(21) Application No.202441016551 A
(43) Publication Date : 22/03/2024

(54) Title of the invention : DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING MODEL

(51) International classification :G06N0020000000, G06F0021550000, G06F0040205000, H04L0051000000, H04L0009120000
(86) International Application No :NA
Filing Date :NA
(87) International Publication No : NA
(61) Patent of Addition to Application Number :NA
Filing Date :NA
(62) Divisional to Application Number :NA
Filing Date :NA

(71)**Name of Applicant :**
1)Dr. Ramesh K
Address of Applicant :Professor, Department of CSE, Sri Krishna College of Engineering and Technology ---------- ----------
2)Dr. Sasi Kala Rani K
3)Mr Vengateshwaran M
4)Lakshan Balaji R K
5)Dhinakaran V
6)Sruthi R
Name of Applicant : NA
Address of Applicant : NA
(72)**Name of Inventor :**
1)Dr. Ramesh K
Address of Applicant :Professor, Department of CSE, Sri Krishna College of Engineering and Technology ---------- ----------
2)Lakshan Balaji R K
Address of Applicant :UG students, Department of CSE, Sri Krishna College of Engineering and Technology ---------- ----------
3)Dhinakaran V
Address of Applicant :UG students, Department of CSE, Sri Krishna College of Engineering and Technology ---------- ----------
4)Sruthi R
Address of Applicant :UG students, Department of CSE, Sri Krishna College of Engineering and Technology ---------- ----------

(57) Abstract :
4. DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING MODEL 5. ABSTRACT 6. Now a days there has been an overall increase in the total number of phishing websites. It is now a widespread issue since attackers trick naive users by designing fake websites with some real elements. These websites are designed to steal personal information such as email ids, usernames, passwords, financial information, etc. Attackers deploy these fake websites that somewhat imitate their original counterparts by incorporating some realistic elements. By Using and Utilizing secure methods to proactively identify phishing websites is imperative to impede the rapid development of phishing techniques because of advancing techniques and technology. A great tool for defending against phishing assaults is machine learning algorithms. Attackers often prefer to use phishing because it is easier and simpler to fraud a victim into accessing a malicious link that looks genuine than to try to bypass a computer's security measures. The malevolent links inside the body of the message appear like links intended to direct the users to the official pages but are intended to direct the user to the spoofed company utilizing that official company's logos and other genuine information. In the methodology that is being presented here, machine learning algorithms are used to create a comprehensive approach for identifying phishing websites. Multiple models were compiled and utilized in our proposed strategy to identify phishing websites based on various parts of the URL specification. The various characteristics of the URLs of both phishing and legitimate websites can be compared by extracting the properties of them. The proposed method uses the combined model to identify the URLs if the phishing websites. This study demonstrates the performance of the suggested approach and successfully identify the legitimate websites from fake ones in real-time.

No. of Pages : 10 No. of Claims : 3

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

ABBREVATION	EXPANSION
ML	Machine Learning
URL	Uniform Resource Locator
DT	Decision Trees
SVM	Support Vector Machines
UML	Uniform Modelling Language
RF	Random Forest
HTML	Hyper Text Markup Language
CSS	Common Style Sheets
py	Python
IDE	Integrated Development Environment
EDA	Exploratory Data Analysis
ANN	Artificial Neural Networks
XGBOOST	Extreme Gradient Boosting
AI	Artificial Intelligence
BDT	Boosted Decision Tree
CART	Classification and Regression Tree
MLM	Machine Learning Model

CHAPTER 1

INTRODUCTION

1.1 DOMAIN INTRODUCTION

In the current age of the Fourth Industrial Revolution (4IR or Industry 4.0), the digital world has a wealth of data, such as Internet of Things (IoT) data, cybersecurity data, mobile data, business data, social media data, health data, etc. To intelligently analyze these data and develop the corresponding smart and automated applications, the knowledge of artificial intelligence (AI), particularly, machine learning (ML) is the key. Various types of machine learning algorithms such as supervised, unsupervised, semi-supervised, and reinforcement learning exist in the area. Besides, the deep learning, which is part of a broader family of machine learning methods, can intelligently analyze the data on a large scale. In this paper, we present a comprehensive view on these machine learning algorithms that can be applied to enhance the intelligence and the capabilities of an application. Thus, this study's key contribution is explaining the principles of different machine learning techniques and their applicability in various real-world application domains, such as cybersecurity systems, smart cities, healthcare, e-commerce, agriculture, and many more. We also highlight the challenges and potential research directions based on our study. Overall, this paper aims to serve as a reference point for both academia and industry professionals as well as for decision-makers in various real-world situations and application areas, particularly from the technical point of view.

We live in the age of data, where everything around us is connected to a data source, and everything in our lives is digitally recorded. For instance, the current electronic world has a wealth of various kinds of data, such as the Internet of Things (IoT) data,

cybersecurity data, smart city data, business data, smartphone data, social media data, health data, COVID-19 data, and many more. The data can be structured, semi-structured, or unstructured, which is increasing day-by-day. Extracting insights from these data can be used to build various intelligent applications in the relevant domains. For instance, to build a data-driven automated and intelligent cybersecurity system, the relevant cybersecurity data can be used to build personalized context-aware smart mobile applications, the relevant mobile data can be used and so on. Thus, the data management tools and techniques having the capability of extracting insights or useful knowledge from the data in a timely and intelligent way is urgently needed, on which the real-world applications are based.

Artificial intelligence (AI), particularly, machine learning (ML) have grown rapidly in recent years in the context of data analysis and computing that typically allows the applications to function in an intelligent manner. ML usually provides systems with the ability to learn and enhance from experience automatically without being specifically programmed and is generally referred to as the most popular latest technologies in the fourth industrial revolution (4IR or Industry 4.0). "Industry 4.0" is typically the ongoing automation of conventional manufacturing and industrial practices, including exploratory data processing, using new smart technologies such as machine learning automation. Thus, to intelligently analyze these data and to develop the corresponding real-world applications, machine learning algorithms is the key. The learning algorithms can be categorized into four major types, such as supervised, unsupervised, semi-supervised, and reinforcement learning in the area. The popularity of these approaches to learning is increasing day-by-day, based on data collected from Google Trends over the last five years. These statistics motivate us to study on machine learning in this paper, which can play an important role in the real-world through Industry 4.0 automation.

1.2 TYPES OF MACHINE LEARNING TECHNIQUES

Machine Learning algorithms are mainly divided into four categories: Supervised learning, Unsupervised learning, Semi-supervised learning, and Reinforcement learning, as shown in Fig. 1.1. In the following, we briefly discuss each type of learning technique with the scope of their applicability to solve real-world problems.

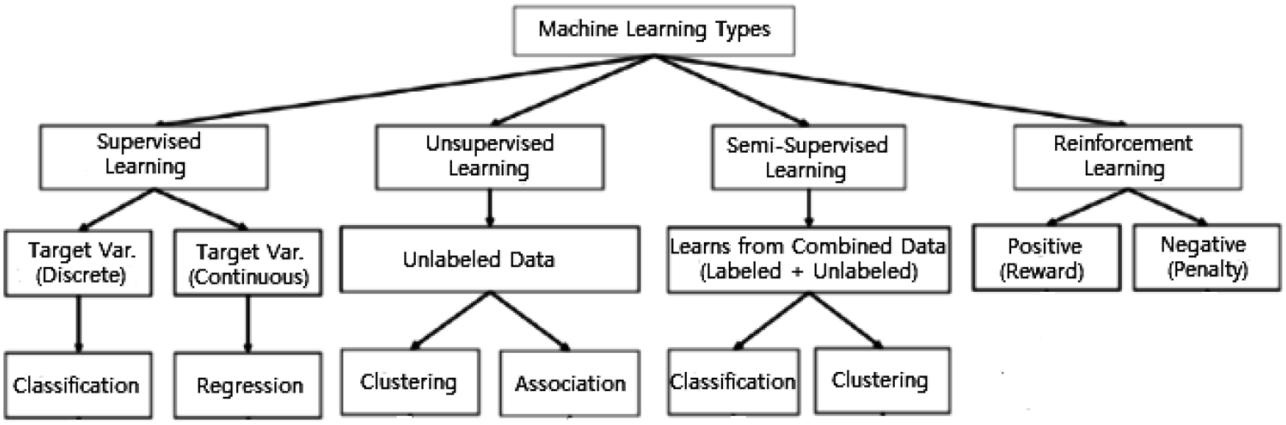

Fig 1.1 Types of Machine Learning Techniques

Supervised: Supervised learning is typically the task of machine learning to learn a function that maps an input to an output based on sample input-output pairs. It uses labeled training data and a collection of training examples to infer a function. Supervised learning is carried out when certain goals are identified to be accomplished from a certain set of inputs, i.e., a task-driven approach. The most common supervised tasks are "classification" that separates the data, and "regression" that fits the data. For instance, predicting the class label or sentiment of a piece of text, like a tweet or a product review, i.e., text classification, is an example of supervised learning.

Unsupervised: Unsupervised learning analyzes unlabeled datasets without the need for human interference, i.e., a data-driven process. This is widely used for extracting generative features, identifying meaningful trends and structures, groupings in results, and

exploratory purposes. The most common unsupervised learning tasks are clustering, density estimation, feature learning, dimensionality reduction, finding association rules, anomaly detection, etc.

Semi-supervised: Semi-supervised learning can be defined as a hybridization of the above-mentioned supervised and unsupervised methods, as it operates on both labeled and unlabeled data. Thus, it falls between learning "without supervision" and learning "with supervision". In the real world, labeled data could be rare in several contexts, and unlabeled data are numerous, where semi-supervised learning is useful. The ultimate goal of a semi-supervised learning model is to provide a better outcome for prediction than that produced using the labeled data alone from the model. Some application areas where semi-supervised learning is used include machine translation, fraud detection, labeling data and text classification.

Reinforcement: Reinforcement learning is a type of machine learning algorithm that enables software agents and machines to automatically evaluate the optimal behavior in a particular context or environment to improve its efficiency, i.e., an environment-driven approach. This type of learning is based on reward or penalty, and its ultimate goal is to use insights obtained from environmental activists to take action to increase the reward or minimize the risk. It is a powerful tool for training AI models that can help increase automation or optimize the operational efficiency of sophisticated systems such as robotics, autonomous driving tasks, manufacturing and supply chain logistics, however, not preferable to use it for solving the basic or straightforward problems.

Thus, to build effective models in various application areas different types of machine learning techniques can play a significant role according to their learning capabilities, depending on the nature of the data discussed earlier, and the target outcome.

1.3 MODELS

In the field of machine learning, a model is a mathematical formula which, after being "trained" on a given dataset, can be used to make predictions or classifications on new data. During training, a learning algorithm iteratively adjusts the model's internal parameters to minimize errors in its predictions. By extension the term model can refer to several level of specify, from a general class of models and their associated learning algorithms, to a fully trained model with all its internal parameters tuned. Various types of models have been used and researched for machine learning systems, picking the best model for a task is called model selection.

1.3 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs.

1.3.1 Artificial Neural Networks

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs.

1.3.2 Decision trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining, and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels, and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision-making.

1.3.3 Support-vector machines

Support-vector machines (SVMs), also known as support-vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

1.3.4 Regression analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel), logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher-dimensional space.

1.3.5 Bayesian networks

A Bayesian network, belief network, or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning. Bayesian networks that model sequences of variables, like speech signals or protein sequences, are called dynamic Bayesian networks. Generalizations of Bayesian networks that can represent and solve decision problems under uncertainty are called influence diagrams.

1.3.6 Genetic algorithms

A genetic algorithm (GA) is a search algorithm and heuristic technique that mimics the process of natural selection, using methods such as mutation and crossover to generate new genotypes in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms were used in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

1.3.7 Belief functions

The theory of belief functions, also referred to as evidence theory or Dempster–Shafer theory, is a general framework for reasoning with uncertainty, with understood connections to other frameworks such as probability, possibility and imprecise probability theories. These theoretical frameworks can be thought of as a kind of learner and have some

analogous properties of how evidence is combined (e.g., Dempster's rule of combination), just like how in a pmf-based Bayesian approach would combine probabilities. However, there are many caveats to these beliefs functions when compared to Bayesian approaches in order to incorporate ignorance and Uncertainty quantification. These belief function approaches that are implemented within the machine learning domain typically leverage a fusion approach of various ensemble methods to better handle the learner's decision boundary, low samples, and ambiguous class issues that standard machine learning approach tend to have difficulty resolving. However, the computational complexity of these algorithms are dependent on the number of propositions (classes), and can lead a much higher computation time when compared to other machine learning approaches.

1.3.8 Training models

Typically, machine learning models require a high quantity of reliable data in order for the models to perform accurate predictions. When training a machine learning model, machine learning engineers need to target and collect a large and representative sample of data. Data from the training set can be as varied as a corpus of text, a collection of images, sensor data, and data collected from individual users of a service. Overfitting is something to watch out for when training a machine learning model. Trained models derived from biased or non-evaluated data can result in skewed or undesired predictions. Bias models may result in detrimental outcomes thereby furthering the negative impacts on society or objectives. Algorithmic bias is a potential result of data not being fully prepared for training. Machine learning ethics is becoming a field of study and notably be integrated within machine learning engineering teams.

1.3.9 Federated learning

Federated learning is an adapted form of distributed artificial intelligence to training machine learning models that decentralizes the training process, allowing for users' privacy to be maintained by not needing to send their data to a centralized server. This also increases efficiency by decentralizing the training process to many devices. For example, Gboard uses federated machine learning to train search query prediction models on users' mobile phones without having to send individual searches back to Google.

1.4 SUMMARY

In conclusion, Machine Learning (ML) stands as a cornerstone of modern artificial intelligence, offering remarkable capabilities in learning from data and making predictions or decisions without explicit programming. Through its three primary paradigms—supervised, unsupervised, and reinforcement learning—ML has revolutionized countless industries and applications, from healthcare to finance, marketing, and beyond.

ML algorithms, ranging from classical methods like linear regression and decision trees to advanced deep learning architectures such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), enable computers to extract meaningful patterns from complex datasets and make informed decisions based on them.

The applications of ML are diverse and profound, contributing to advancements in disease diagnosis, fraud detection, recommendation systems, language translation, image recognition, and more. Its impact on society continues to expand as researchers and practitioners unlock new ways to leverage data for solving real-world problems.

However, ML is not without its challenges. Issues such as data quality and quantity, interpretability, fairness, and security pose significant hurdles that must be addressed to

ensure the responsible and ethical deployment of ML systems. Efforts to mitigate these challenges through data preprocessing techniques, explainable AI approaches, fairness-aware algorithms, and secure ML methodologies are ongoing.

Looking ahead, the future of ML holds promise and excitement. Emerging trends such as explainable AI, federated learning, quantum machine learning, and multi-modal learning offer tantalizing prospects for further innovation and impact. Moreover, the ethical considerations surrounding AI and ML are gaining increasing attention, emphasizing the need for responsible development and deployment practices.

In conclusion, ML represents not just a technological advancement but a transformative force that is reshaping how we interact with data and make decisions. As research and development in ML continue to progress, it is imperative to remain vigilant in addressing challenges, fostering ethical practices, and harnessing the full potential of this powerful technology for the betterment of society.

CHAPTER 2

RELATED WORK

2.1 OVERVIEW

This chapter offers an insight into various important studies conducted by excellent scholars from articles, books, and other sources relevant to the detection of phishing websites. It provides the project with a review to demonstrate understanding of the project.

2.2 LITERATURE REVIEW

S. Arvind Anwekar, V. Agrawal [19]: In this study, the authors focused on extracting features from URLs, in addition to other features such as the age of the SSL certificate and the universal resource locator of the anchor, IFRAME, and website rank. They collected URLs of phishing websites from PhishTank and URLs of benign websites from the Alexa website. Using a combination of the random forest (RF), decision tree (DT), and support vector machine (SVM), contributed to improving the detection mechanism for phishing websites and achieved a high noticeable detection accuracy of 97.14%, with a low rate of false positives at 3.14%. The results also showed that the classifier's performance improves with more training data.

N. Choudhary b, K. Jain, S. Jain [11]: This study emphasizes the significance of only using attributes from the URL. Both the Kaggle and Phishtank websites make it easy to get the dataset used in this study. The researchers used a hybrid approach that combined Principal Component Analysis (PCA) with Support Vector Machine (SVM) and Random Forest algorithms to reduce the dataset's dimensionality while keeping all im\portant data, and it produced a higher accuracy rate of 96.8% compared to other tech-niques investigated.

A. Lakshmanarao, P. Surya, M Bala Krishna [12]: This thesis collected a dataset of phishing websites from the UCI repository and used various Machine learning tech-niques, including decision trees, AdaBoost, support vector machines (SVM), and ran-dom forests, to analyze selected features (such as web traffic, port, URL length, IP address, and URL_of_Anchor). The most effective model for detecting phishing web-sites was chosen, and two priority-based algorithms (PA1 and PA2) were proposed. The team utilized a new fusion classifier in conjunction with these algorithms and attained an accuracy rate of 97%. when compared to previous works in phishing website detection.

L. Tang, Q. Mahmoud [13]: The proposed approach in the current study uses URLs collected from a variety of platforms, including Kaggle, Phish Storm, Phish Tank, and ISCX-UR, to identify phishing websites. The researchers made a big contribution since they created a browser plug-in that can quickly recognize phishing risks and offer warnings. Various datasets and machine learning techniques were investigated, and the pro-posed RNN-GRU model outperformed SVM, Random Forest (RF), and Logistic Regression with a maximum accuracy rate of 99.18%. On the other hand, the suggested method is not always accurate in identifying if short links are phishing risks.

A. Kulkarni & L. Brown[14]: A machine learning system was created to categorize websites based on URLs from the University of California, Irvine Machine Learning Repository. Four classifiers were used: SVM, decision tree, Naive Bayesian, and neural network. The outcome of experiments utilizing the model developed with the support of a training set of data demonstrates that the classifiers were able to successfully differentiate authentic websites from fake ones with an accuracy rate of over 90%. Limitations include a small dataset and all features being discrete, which may not be suitable for some classifiers.

M. Karabatak and T. Mustafa [16]: The objective of this research is to assess the effectiveness of classification algorithms on a condensed dataset of phishing websites obtained from the UCI Machine Learning Repository. The paper investigates how data

mining and feature selection algorithms affect reduced datasets through experiments and analysis, finally selecting the methods that perform the best in terms of classification. According to the results, some classification strategies improve performance while others have the opposite impact. Ineffective classifiers for condensed phishing datasets included Lazy, BayesNet, SGD Multilayer Perceptron, PART, JRip, J48, Random Tree, and Random Forest. However, it was discovered that KStar, LMT, ID3, and R.F. Classifier were efficient. Lazy produced the highest classification accuracy rate of 97.58% on the compressed 27-feature dataset, whereas KStar performed at its best on the same dataset.

X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng [17]: A phishing detection model that applies Bagging, AdaBoost, SMO, and Random Forest algorithms to learn and test phishing detection strategies is offered as a contribution to this work. The model is based on features from URLs and extracts multi-level statistical characteristics, seman-tic features of word embedding, and semantic features from Chinese web content. Legal URLs from DirectIndustry online instructions and phishing data from the Anti-Phishing Alliance of China (APAC) are included in the dataset used to test the algorithm. The study's findings suggest that a fusion model that primarily employed semantic data to identify phishing sites with high detection efficiency had the best performance, leading to a new contribution with an F-measure of 0.99%. Keep in mind that this approach is specific to Chinese websites and is language-dependent.

W. Fadheel, M. Abusharkh, and I. Abdel-Qader [18]: The present study utilized da-tasets from the UCI machine learning repository, including Domain, HTML, Address Bar, and URLs, the main contribution was conducting a comparative analysis of the impact of feature selection on detecting phishing websites. The KMO test was applied in the study to evaluate the dataset using (LR) and (SVM) classification algorithms. The test was conducted based on a correlation matrix to analyze the performance. Re-sults showed that LR with the KMO test achieved an accuracy of 91.68%, while SVM with the KMO test yielded an accuracy of 93.59%.

2.3 PROBLEM STATEMENT

Phishing attacks have gotten increasingly complex, it is very difficult for an average person to determine if an email message link or website is legitimate. Cyber-attacks by criminals that employ phishing schemes are so prevalent and successful nowadays.

Hence, this project seeks to address fake URLs and domain names by identifying phishing website links. Therefore, having a web application that provides the user an interface to check if a URL is Phishing or legitimate will help decrease security risks to individuals and organizations.

The problem is to develop effective algorithms for timely identification of phishing websites, given the escalating threat of sophisticated phishing attacks. Challenges include data imbalance, feature selection, real-time detection, and resilience against adversarial attacks. Objectives involve creating accurate detection models, handling class imbalance, selecting relevant features, implementing real-time mechanisms, and addressing adversarial evasion. Successful outcomes include a robust detection system, improved resilience, and enhanced user security. This research is significant for safeguarding users, preventing financial losses, preserving trust, complying with regulations, mitigating disruption, and combating evolving threats.

The need for phishing detection is paramount in today's digital landscape due to the pervasive threat posed by phishing attacks. Several factors underscore the importance of robust phishing detection mechanisms.

2.4 SUMMARY

The literature study on phishing detection reveals a growing body of research focused on developing effective strategies to combat the pervasive threat of phishing attacks. Key findings from the literature.

Overall, the literature study highlights the importance of continuous research and innovation in phishing detection to stay ahead of evolving threats. By developing robust algorithms, addressing class imbalance issues, selecting relevant features, enabling realtime detection, and enhancing resilience against adversarial attacks, researchers aim to bolster cybersecurity measures and protect users from the pervasive threat of phishing attacks.

CHAPTER 3

PROPOSED SYSTEM

3.1 INTRODUCTION

This chapter describes the various process, methods, and procedures adopted by the researcher to achieve the set aim and objectives and the conceptual structure within which the research was conducted.

The methodology of any research work refers to the research approach adopted by the researcher to tackle the stated problem. Since the efficiency and maintainability of any application are solely dependent on how designs are prepared. This chapter provides detailed descriptions of methods employed to proffer solutions to the stated objectives of the research work.

According to the Merriam-Webster dictionary (11th.Ed), system analysis is "the process of studying a procedure or business to identify its goals and purposes and create systems and procedures that will efficiently achieve them". It is also the act, process, or profession of studying an activity (such as a procedure, a business, or a physiological function) typically by mathematical means to define its goals or purposes and to discover operations and procedures for accomplishing them most efficiently. System analysis is used in every field where the development of something is done. Before planning and development, you need to understand the old systems thoroughly and use the knowledge to determine how best your new system can work.

3.2 ANALYSIS OF EXISTING SYSTEM

The existing system of phishing detection techniques suffers low detection accuracy and high false alarm especially when different phishing approaches are introduced. Above and beyond, the most common technique used is the blacklist-based method which is inefficient

in responding to emanating phishing attacks since registering a new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database for phishing detection.

3.3 OBJECTIVES

To accomplish the project's purpose, the following particular objectives have been established:

i. dataset collection and pre-processing.

ii. machine-learning model selection and development.

iii. development of a web-based application for detection.

iv. Integration of the developed model to a web application.

This project aims to detect phishing websites using machine learning and deep neural networks by developing a web application that allows users to check if a URL is phishing or legitimate and have access to resources to help tackle phishing attacks.

This study explores data science and machine learning models that use datasets gotten from open-source platforms to analyze website links and distinguish between phishing and legitimate URL links. The model will be integrated into a web application, allowing a user to predict if a URL link is legitimate or phishing. This online application is compatible with a variety of browsers.

3.4 PROPOSED SYSTEM

The proposed phishing detection system utilizes machine learning models and deep neural networks. The system comprises two major parts, which are the machine learning models and a web application. These models consist of Gradient Boosting Classifier.

This model is selected after different comparison-based performances of multiple machine learning algorithms. This model is trained and tested on a website content-based feature, extracted from both phishing and legitimate dataset.

Hence, the model with the highest accuracy is selected and integrated into a web application that will enable a user to predict if a URL link is phishing or legitimate.

3.4.1 BENEFITS OF PROPOSED SYSTEM

i. Will be able to differentiate between phishing(0) and legitimate(1) URLs

ii. It Will help reduce phishing data breaches for an organization

iii. It Will be helpful to individuals and organizations

iv. It is easy to use

3.4.2 MODEL DEVELOPMENT

The model development method takes several models, tests them, and adds them to an iterative process until a model that meets the required requirements is developed. Figure 3.1 shows the steps used in the development of machine learning models using both supervised and unsupervised learning.

i. Data Collection

The data used to generate the datasets on which the models are trained are gotten from different open-source platforms. The dataset collection consists of phishing and legitimate URL dataset.

The set of phishing URLs and legitimate URLs are collected from an open-source service called Kaggle. This service provides a set of phishing URLs in multiple formats like CSV, JSON and so on that gets updated hourly. This dataset is accessible from the kaggle.com website. From this dataset, over 10,000 random phishing URLs are collected to train the ML models.

ii. Preprocessing

Data preprocessing is the first and crucial step after data collection. The raw dataset obtained for phishing detection was prepared by removing redundant and irregular data and also encoded using the One-Hot Encoding technique into a useful and efficient format suitable for the machine learning model.

iii. Exploratory data analysis

Exploratory data analysis (EDA) technique was used on the dataset after series of data cleaning. The data visualization method was employed to analyze, explore and summarize the dataset. These visualization consist of heat-map, histograms, box plots, scatter plots, and pair plots to uncover patterns and insights within data.

iv. Feature Extraction

Feature Extraction aims to reduce the number of features in a dataset by creating new features from the existing ones. Thus, Website content-based features were extracted from phishing and legitimate datasets such as the Address bar-based feature which consists of 9 features, Domain-based feature which consists of 4 features, and HTML & JavaScript-based Feature which consists of 4 features. So, altogether 17 features were extracted for phishing detection.

v. Model Training

Model Training involves feeding Machine learning algorithms with data to help identify and learn good attributes of the dataset.

This research problem is a product of supervised learning, which falls under the classification problem. The algorithms used for phishing detection consist of supervised machine learning models (4) and deep neural network (2) which was used to train the dataset. The algorithm used for this project is Gradient Boosting Classifier. All these

models were trained on the dataset. Thus, the dataset is spitted into a training and testing set. The training model consists of 80% of the dataset to enable the machine learning models to learn more about the data and be able to distinguish between phishing and legitimate URLs.

vi. Model Testing

Model Testing involves the process where the performance of a fully trained model is evaluated on a testing set. Thus, after 80% of data has been trained, 20% of the dataset is used to evaluate the trained dataset to see the performance of the models.

vii. Model Evaluation

Model Evaluation involves estimating the generalization accuracy of models and deciding whether the model performs better or not. Thus, Scikit-learn (sklearn-matrics) module was used to implements several score and utility functions to measure the classification performance to properly evaluate the models deployed for phishing detection.

3.5 SYSTEM DESIGN

System modeling involves the process of developing an abstract model of a system, with each model presenting a different view or perspective of the system. It is the process of representing a system using various graphical notations that shows how users will interact with the system and how certain parts of the system function. The proposed system was modeled using the following diagrams:

System modeling involves the process of developing an abstract model of a system, with each model presenting a different view or perspective of the system. It is the process of representing a system using various graphical notations that shows how users will interact with the system and how certain parts of the system function.

The proposed system will be implemented using Python Programming language along with different machine learning models and libraries such as pandas, scikit-learn, python who-is, beautiful-Soup, NumPy, seaborn, and matplotlib. Etc.

3.5.1 DATA FLOW DIAGRAM

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

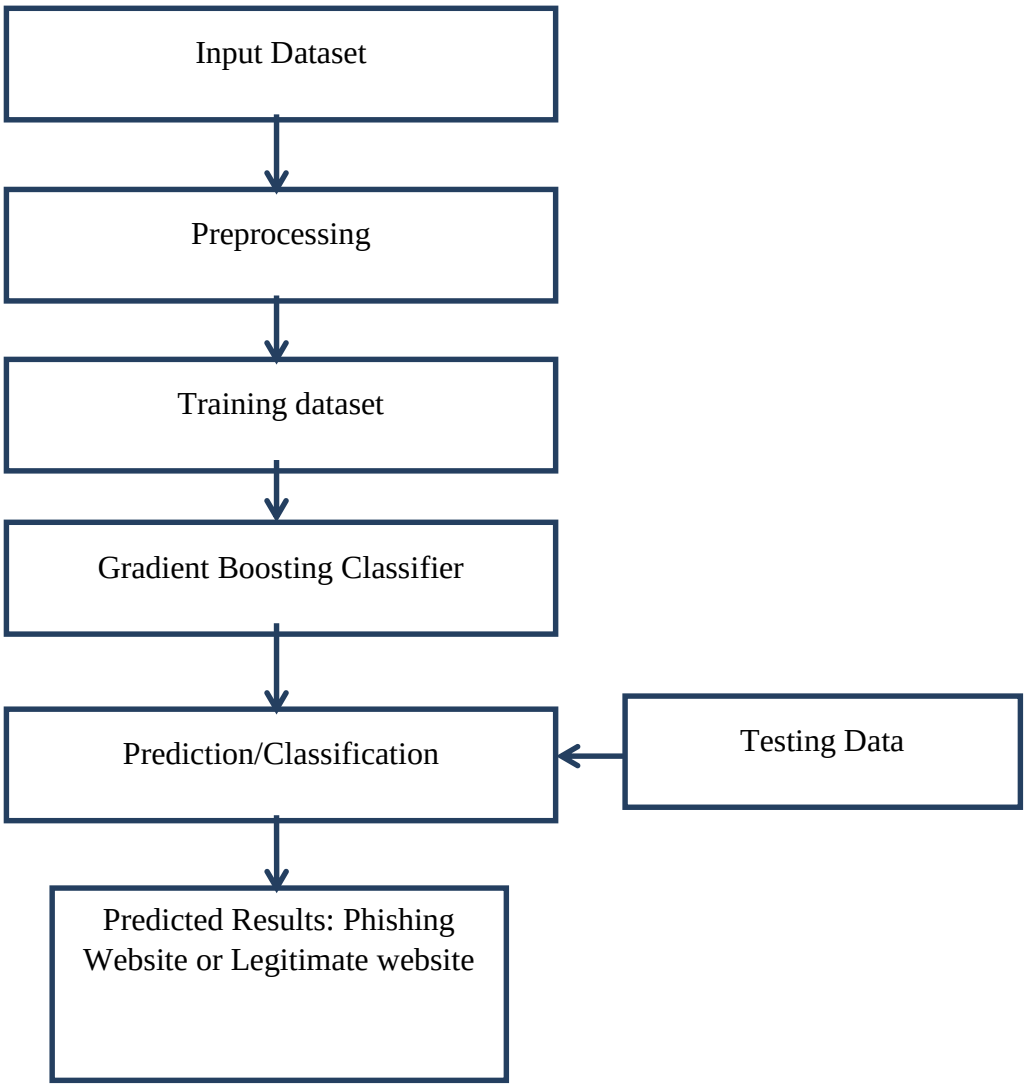

Figure 3.1 Data Flow Diagram

3.5.2 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

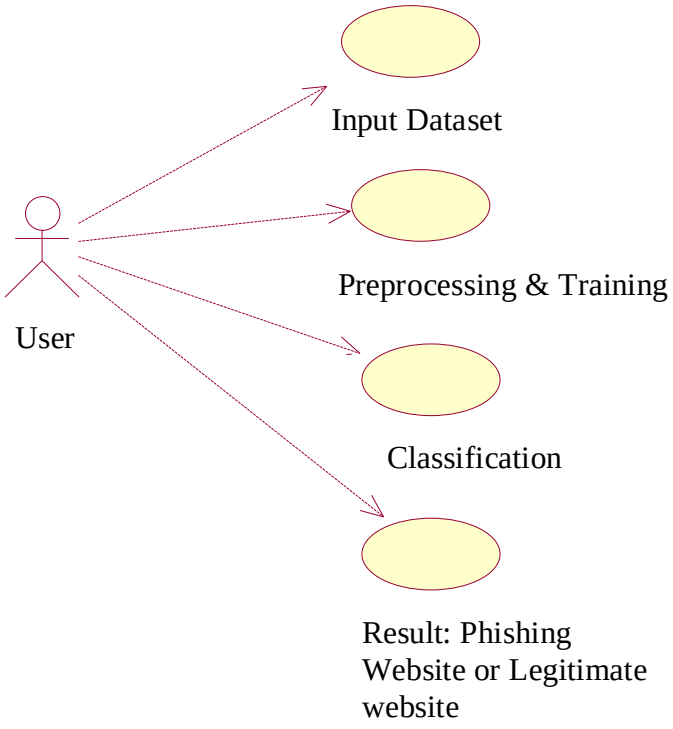

Figure 3.2 Use Case Diagram

3.5.3 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

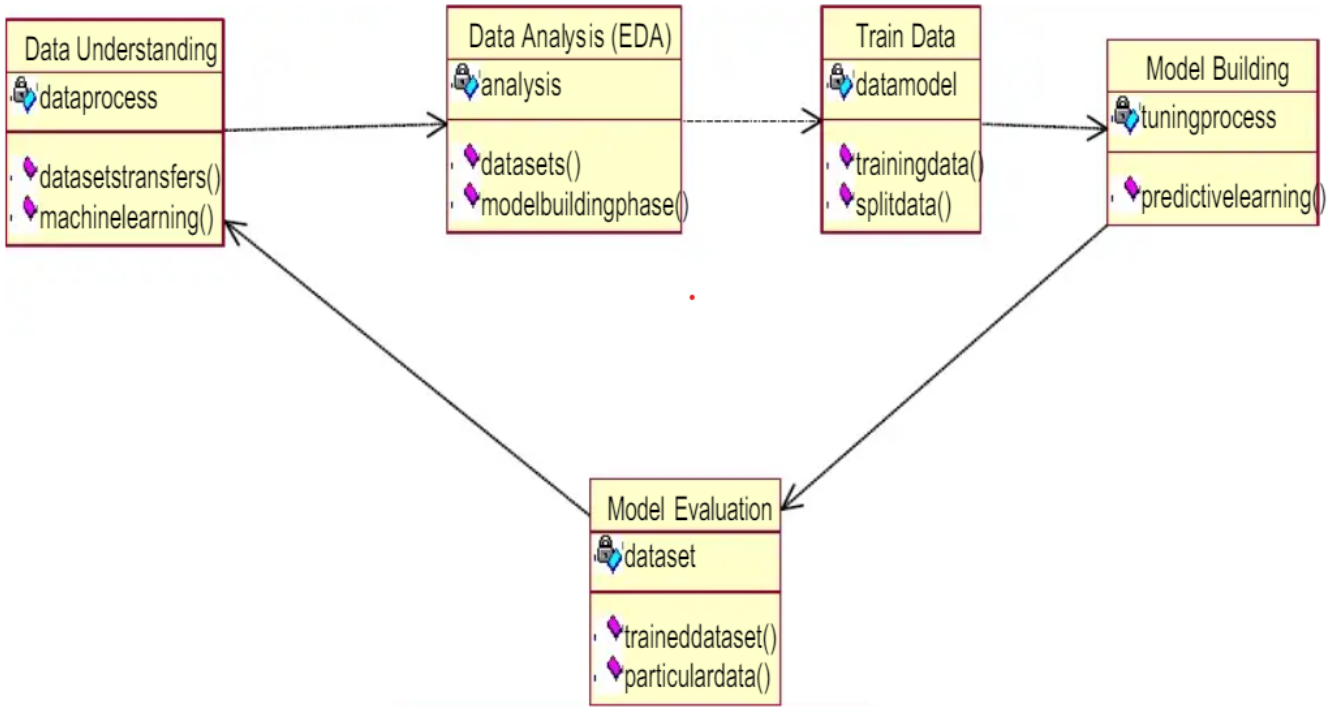

Figure 3.3 Class Diagram

3.5.4 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

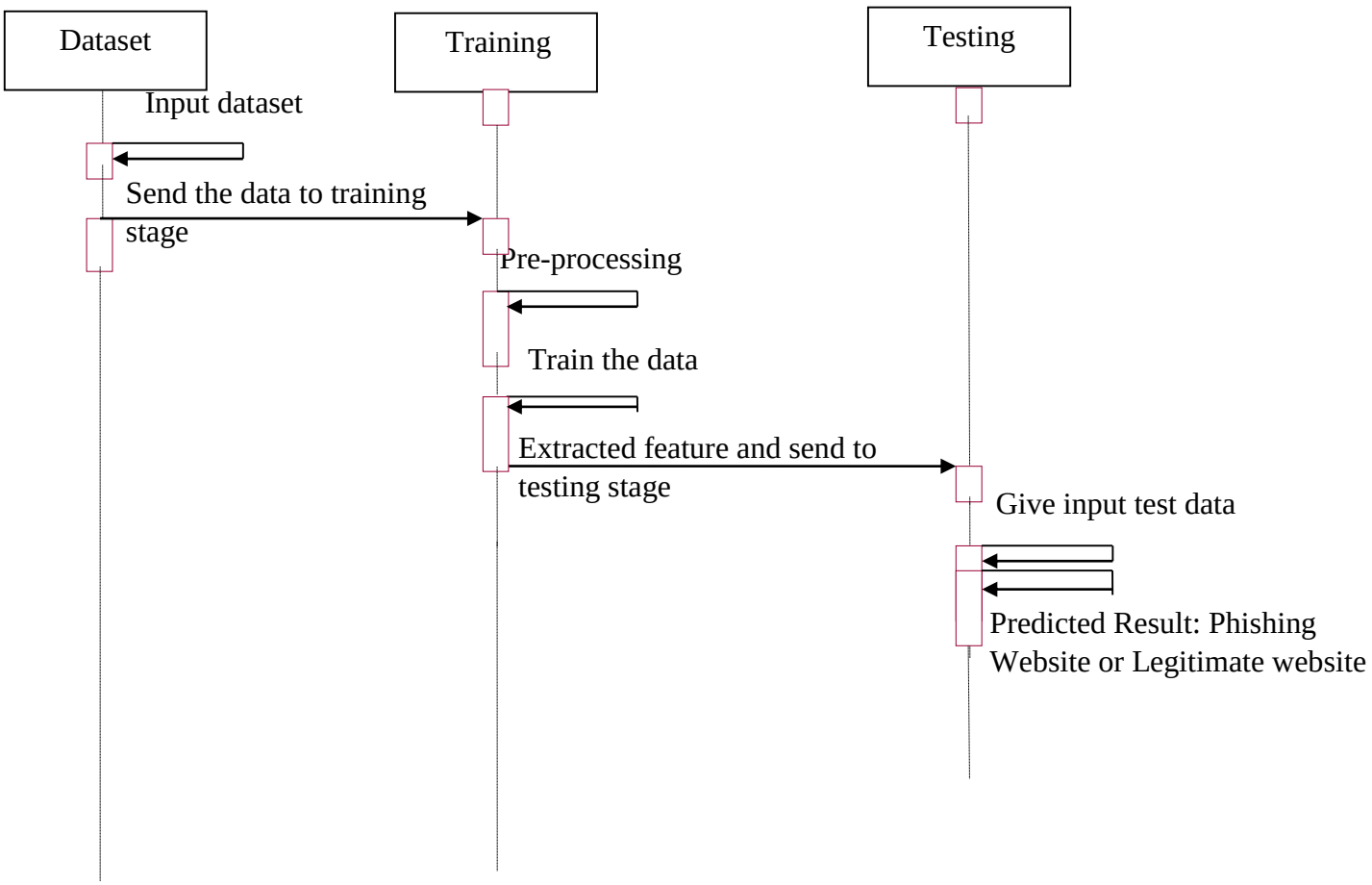

Figure 3.4 Sequence Diagram

3.5.5 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

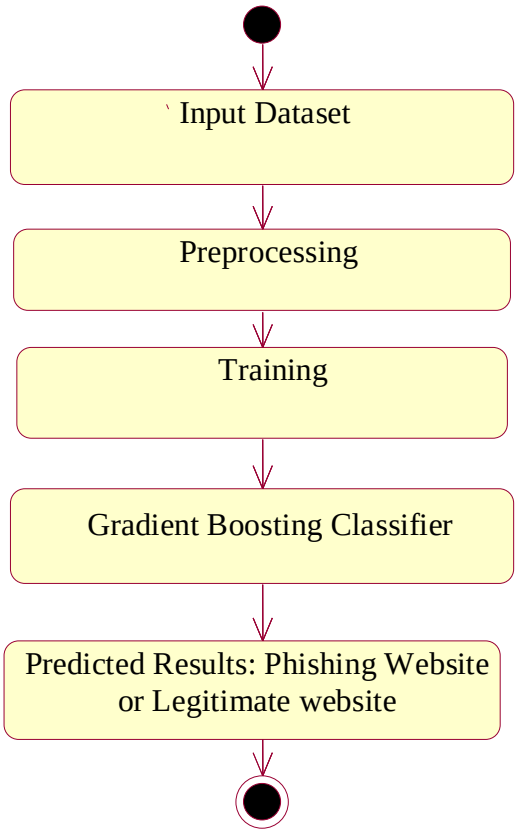

Figure 3.5 Activity Diagram

3.6 IMPLEMENTATION

This chapter deals with the implementation of multiple machine learning models for the detection of underlying diseases and illnesses as earlier designed in the preceding chapters.

The implementation is concerned with all the activities that took place to put up the newly developed system into operation (using the approach that was stated in the methodology (e.g. architectural diagram, flowchart, uses case, etc.).) to achieve the objectives of the project to convert the theoretical design into a working system. The components of the system were also tested and evaluated.

3.6.1 INSTALLAION REQUIREMENTS

The hardware (physical components of a computer system that can be seen, touched, or felt) and software (both system software and the application software installed and used in the system development) tools needed to satisfy these objectives highlighted below:

The hardware requirement includes:

i. A laptop or desktop computer (Preferably 64bit)

ii. Random Access Memory (RAM): 8 Gigabytes Minimum

iii. Processor: Intel Core i5, 2.4 GHz Minimum

The software requirements for the development of this system include:

i. Windows Operating System (8/10)

ii. Anaconda Navigator (Jupyter Notebook)

iii. PyCharm Community edition

iv. Web browser (Preferably Chrome)

v. Visual Studio Code

3.6.2 DATASET

In the first module we develop the data collection process. This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions. The dataset is referred from the popular dataset repository called kaggle. The following is the dataset link for the Detection of Phishing Websites Using Machine Learning.

The dataset consists of 11054 individual data. There are 32 columns in the dataset, which are described below.

Index: index id

UsingIP: (categorical - signed numeric) : { -1,1 }

LongURL: (categorical - signed numeric) : { 1,0,-1 }

ShortURL: (categorical - signed numeric) : { 1,-1 }

Symbol@: (categorical - signed numeric) : { 1,-1 }

Redirecting:// (categorical - signed numeric) : { -1,1 }

PrefixSuffix-: (categorical - signed numeric) : { -1,1 }

SubDomains: (categorical - signed numeric) : { -1,0,1 }

HTTPS: (categorical - signed numeric) : { -1,1,0 }

DomainRegLen: (categorical - signed numeric) : { -1,1 }

Favicon: (categorical - signed numeric) : { 1,-1 }

NonStdPort: (categorical - signed numeric) : { 1,-1 }

HTTPSDomainURL: (categorical - signed numeric) : { -1,1 }

RequestURL: (categorical - signed numeric) : { 1,-1 }

AnchorURL: (categorical - signed numeric) : { -1,0,1 }

LinksInScriptTags: (categorical - signed numeric) : { -1,0,1 }

ServerFormHandler: (categorical - signed numeric) : { -1,0,1 }

InfoEmail: (categorical - signed numeric) : { -1,1 }

AbnormalURL: (categorical - signed numeric) : { -1,1 }

WebsiteForwarding: (categorical - signed numeric) : { 0,1 }

StatusBarCust: (categorical - signed numeric) : { -1,1 }

DisableRightClick: (categorical - signed numeric) : { -1,1 }

UsingPopupWindow: (categorical - signed numeric) : { -1,1 }

IframeRedirection: (categorical - signed numeric) : { -1,1 }

AgeofDomain: (categorical - signed numeric) : { -1,1 }

DNSRecording: (categorical - signed numeric) : { -1,1 }

WebsiteTraffic: (categorical - signed numeric) : { -1,0,1 }

PageRank: (categorical - signed numeric) : { -1,1 }

GoogleIndex: (categorical - signed numeric) : { -1,1 }

LinksPointingToPage: (categorical - signed numeric) : { -1,0,1 }

StatsReport: (categorical - signed numeric) : { -1,1 }

Class: (categorical - signed numeric) : { -1,1 }

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.).

Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data.

Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis.

Split into training and evaluation sets.

3.6.3 MODEL DEVELOPMENT

We used Gradient Boosting Classifier machine learning algorithm. We got an accuracy of training Accuracy 98.9% so we implemented this algorithm. Gradient Boosting Classifier Algorithm

What is boosting?

While studying machine learning you must have come across this term called Boosting. It is the most misinterpreted term in the field of Data Science. The principle behind boosting algorithms is first we built a model on the training dataset, then a second model is built to

rectify the errors present in the first model. Let me try to explain to you what exactly does this means and how does this work.

Suppose you have n data points and 2 output classes (0 and 1). You want to create a model to detect the class of the test data. Now what we do is randomly select observations from the training dataset and feed them to model 1 (M1), we also assume that initially, all the observations have an equal weight that means an equal probability of getting selected.

Remember in ensembling techniques the weak learners combine to make a strong model so here M1, M2, M3….Mn all are weak learners.

Since M1 is a weak learner, it will surely misclassify some of the observations. Now before feeding the observations to M2 what we do is update the weights of the observations which are wrongly classified. You can think of it as a bag that initially contains 10 different color balls but after some time some kid takes out his favorite color ball and put 4 red color balls instead inside the bag. Now off-course the probability of selecting a red ball is higher. This same phenomenon happens in Boosting techniques, when an observation is wrongly classified, its weight get's updated and for those which are correctly classified, their weights get decreased. The probability of selecting a wrongly classified observation gets increased hence in the next model only those observations get selected which were misclassified in model 1.

Similarly, it happens with M2, the wrongly classified weights are again updated and then fed to M3. This procedure is continued until and unless the errors are minimized, and the dataset is predicted correctly. Now when the new datapoint comes in (Test data) it passes through all the models (weak learners) and the class which gets the highest vote is the output for our test data.

What is a Gradient boosting Algorithm?

The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model. But how do we do that? How do we reduce the error? This is done by building a new model on the errors or residuals of the previous model.

When the target column is continuous, we use Gradient Boosting Regressor whereas when it is a classification problem, we use Gradient Boosting Classifier. The only difference between the two is the "Loss function". The objective here is to minimize this loss function by adding weak learners using gradient descent. Since it is based on loss function hence for regression problems, we'll have different loss functions like Mean squared error (MSE) and for classification, we will have different for e.g log-likelihood.

3.5 IMPLEMENTATION

Code implementation involves translating algorithmic solutions or programming concepts into executable code using a programming language such as Python, Java, or C++. It encompasses writing, testing, and debugging code to ensure its functionality and effectiveness in addressing a specific problem or task. The process typically involves breaking down the problem into smaller, manageable tasks, designing an appropriate solution strategy, and then implementing the solution using the syntax and constructs of the chosen programming language. Effective code implementation requires attention to detail, adherence to coding standards and best practices, as well as thorough testing to identify and resolve any errors or bugs. Additionally, documenting the code is essential to facilitate understanding, maintenance, and collaboration among developers.

Here we can see some sample code used in the project.

3.5.1 Sample Code

app.py

```
#importing required libraries

from flask import Flask, request, render_template

import numpy as np

import pandas as pd

from sklearn import metrics

import warnings

import pickle

warnings.filterwarnings('ignore')

from feature import FeatureExtraction

file = open("model.pkl","rb")

gbc = pickle.load(file)

app = Flask(__name__)

@app.route('/')

@app.route('/first')

def first():

    return render_template('first.html')
```

```
@app.route('/performance')
def performance():
    return render_template('performance.html')
@app.route('/chart')
def chart():
    return render_template('chart.html')
@app.route('/login')
def login():
    return render_template('login.html')
@app.route('/upload')
def upload():
    return render_template('upload.html')
@app.route('/preview',methods=["POST"])
def preview():
    if request.method == 'POST':
        dataset = request.files['datasetfile']
        df = pd.read_csv(dataset,encoding = 'unicode_escape')
        df.set_index('Id', inplace=True)
```

```
        return render_template("preview.html",df_view = df)
@app.route('/index')
def index():
    return render_template('index.html')
@app.route("/posts", methods=["GET", "POST"])
def posts():
    if request.method == "POST":


        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)
        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        print(y_pred)
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        print(y_pro_phishing)
```

```
        print(y_pro_non_phishing)
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
        return render_template('result.html',xx =round(y_pro_non_phishing,2),url=url )
    return render_template("result.html", xx =-1)
if __name__ == "__main__":
    app.run(debug=True)
```

CHAPTER 4

TESTING AND RESULTS

4.1 System Testing

System testing is a critical phase in the software development lifecycle where the entire integrated system is tested to ensure that it meets specified requirements and functions correctly as a whole. This testing phase evaluates the system's compliance with functional, performance, security, and usability requirements, among others. It involves executing tests on the complete system, including its interfaces with external systems or components, to validate its behavior under various scenarios and conditions. System testing aims to uncover defects such as functional errors, performance bottlenecks, compatibility issues, and security vulnerabilities before the software is deployed to end-users. By thoroughly assessing the system's functionality and performance, system testing helps ensure the reliability, quality, and readiness of the software for production release.

From actual dataset, we chose only 30 features to test and train the system:

UsingIP: (categorical - signed numeric) : { -1,1 }

LongURL: (categorical - signed numeric) : { 1,0,-1 }

ShortURL: (categorical - signed numeric) : { 1,-1 }

Symbol@: (categorical - signed numeric) : { 1,-1 }

Redirecting:// (categorical - signed numeric) : { -1,1 }

PrefixSuffix-: (categorical - signed numeric) : { -1,1 }

SubDomains: (categorical - signed numeric) : { -1,0,1 }

HTTPS: (categorical - signed numeric) : { -1,1,0 }

DomainRegLen: (categorical - signed numeric) : { -1,1 }

Favicon: (categorical - signed numeric) : { 1,-1 }

NonStdPort: (categorical - signed numeric) : { 1,-1 }

HTTPSDomainURL: (categorical - signed numeric) : { -1,1 }

RequestURL: (categorical - signed numeric) : { 1,-1 }

AnchorURL: (categorical - signed numeric) : { -1,0,1 }

LinksInScriptTags: (categorical - signed numeric) : { -1,0,1 }

ServerFormHandler: (categorical - signed numeric) : { -1,0,1 }

InfoEmail: (categorical - signed numeric) : { -1,1 }

AbnormalURL: (categorical - signed numeric) : { -1,1 }

WebsiteForwarding: (categorical - signed numeric) : { 0,1 }

StatusBarCust: (categorical - signed numeric) : { -1,1 }

DisableRightClick: (categorical - signed numeric) : { -1,1 }

UsingPopupWindow: (categorical - signed numeric) : { -1,1 }

IframeRedirection: (categorical - signed numeric) : { -1,1 }

AgeofDomain: (categorical - signed numeric) : { -1,1 }

DNSRecording: (categorical - signed numeric) : { -1,1 }

WebsiteTraffic: (categorical - signed numeric) : { -1,0,1 }

PageRank: (categorical - signed numeric) : { -1,1 }

GoogleIndex: (categorical - signed numeric) : { -1,1 }

LinksPointingToPage: (categorical - signed numeric) : { -1,0,1 }

StatsReport: (categorical - signed numeric) : { -1,1 }

Class: (categorical - signed numeric) : { -1,1 }

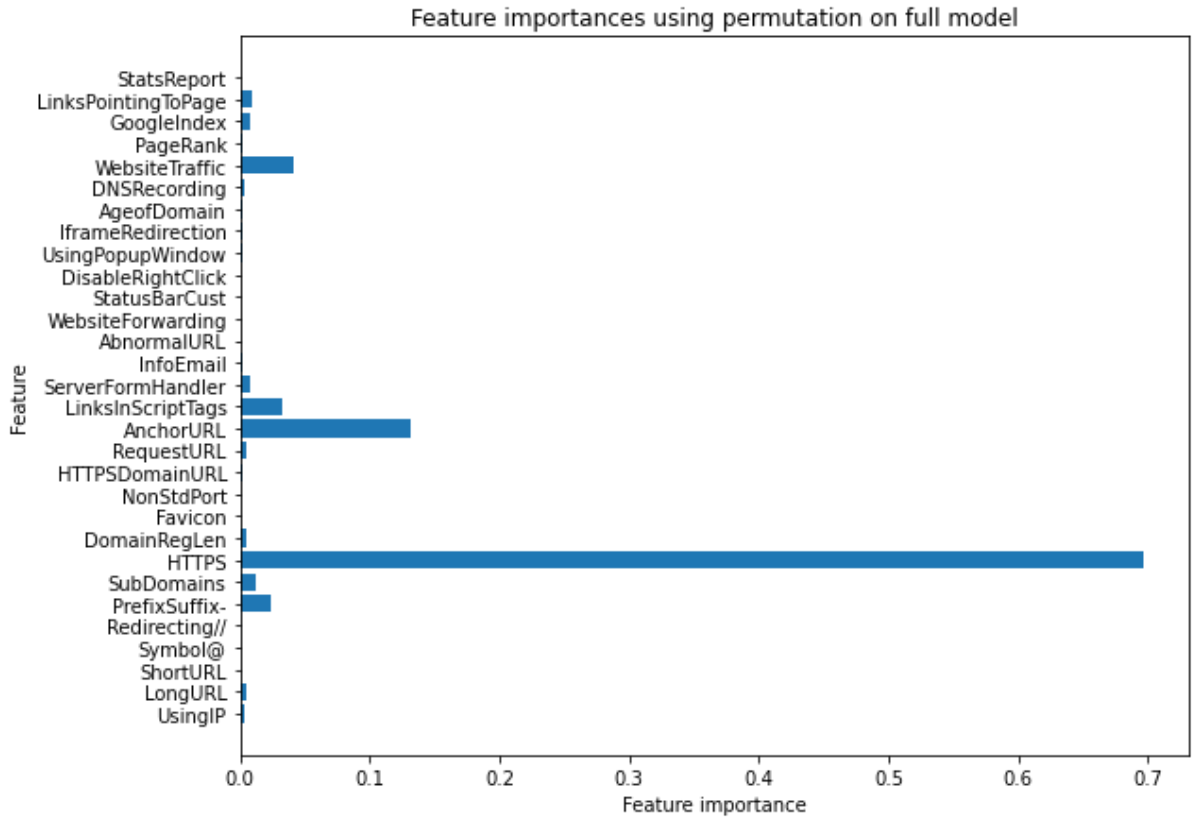

Figure 4.1 Feature Importance using Permutation on Full Model

4.2 SYSTEM ACCURACY

We got an accuracy of 97.6% on test set.

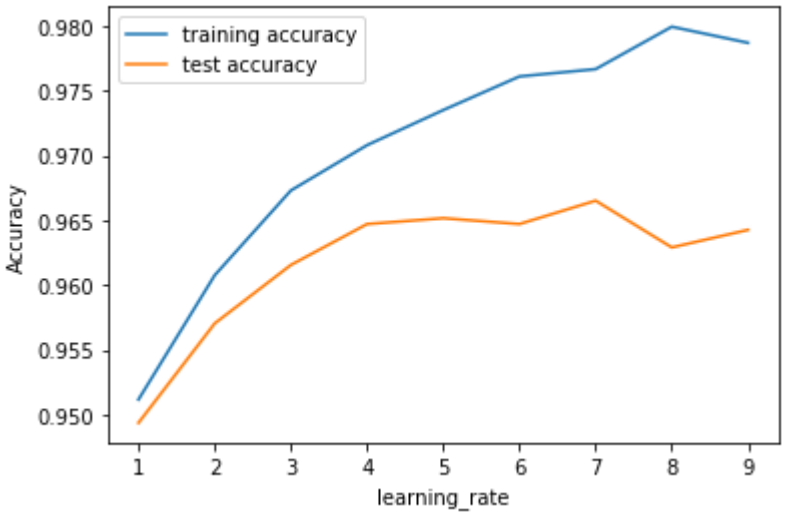

Figure 4.2 Model Training Data v1

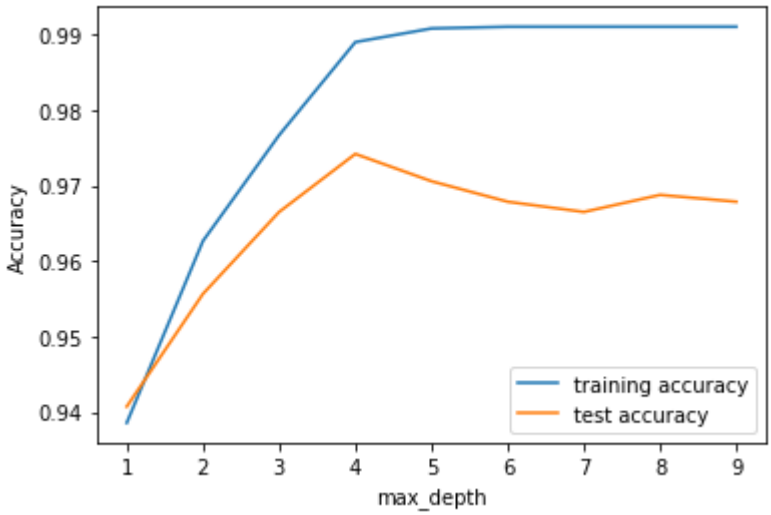

Figure 4.3 Model Training Data v2

4.3 SAVING THE TRAINED MODEL

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle. Make sure you have pickle installed in your environment. Next, let's import the module and dump the model into. pkl file.

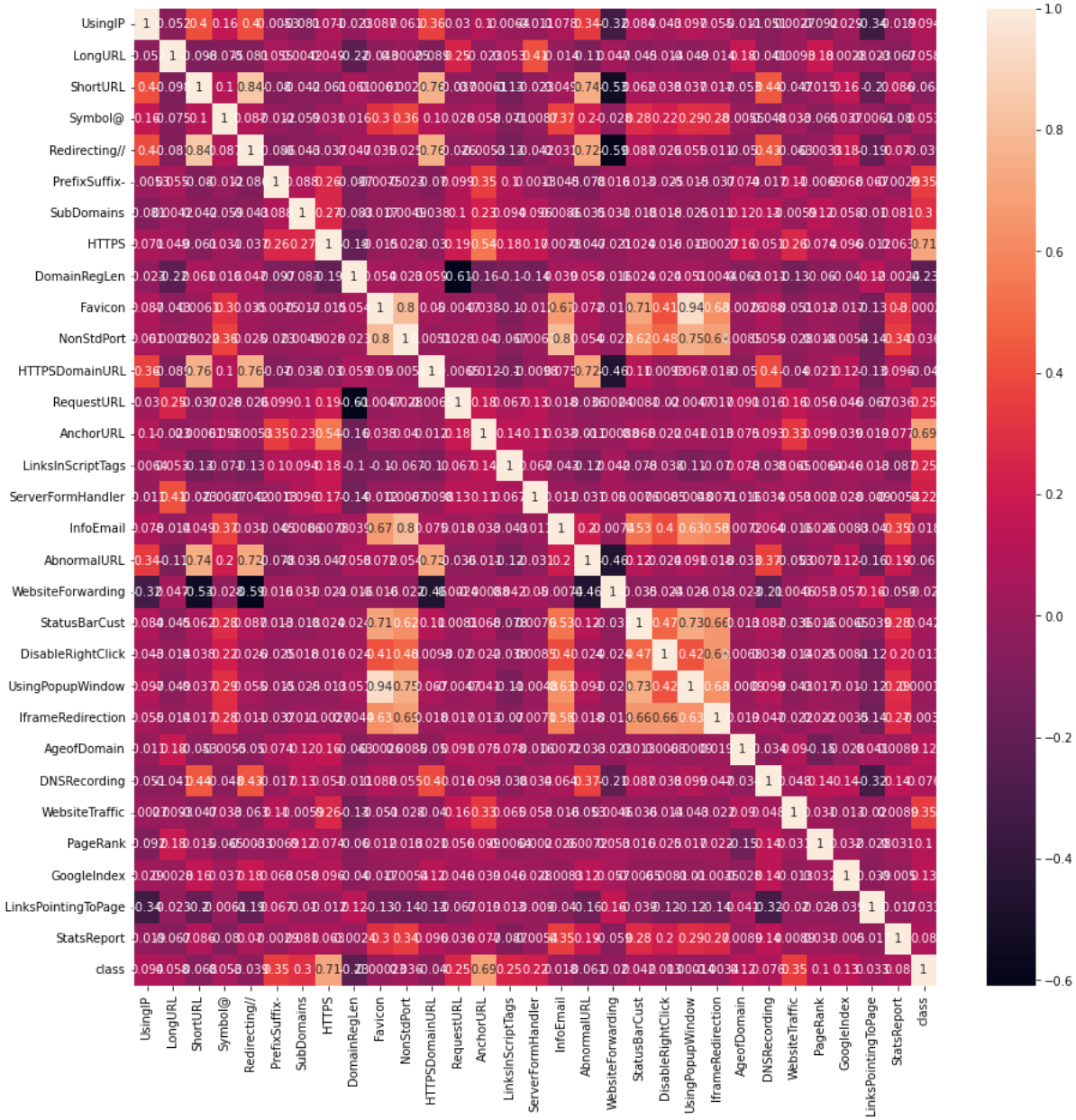

Figure 4.4 Visualizing the Data v1

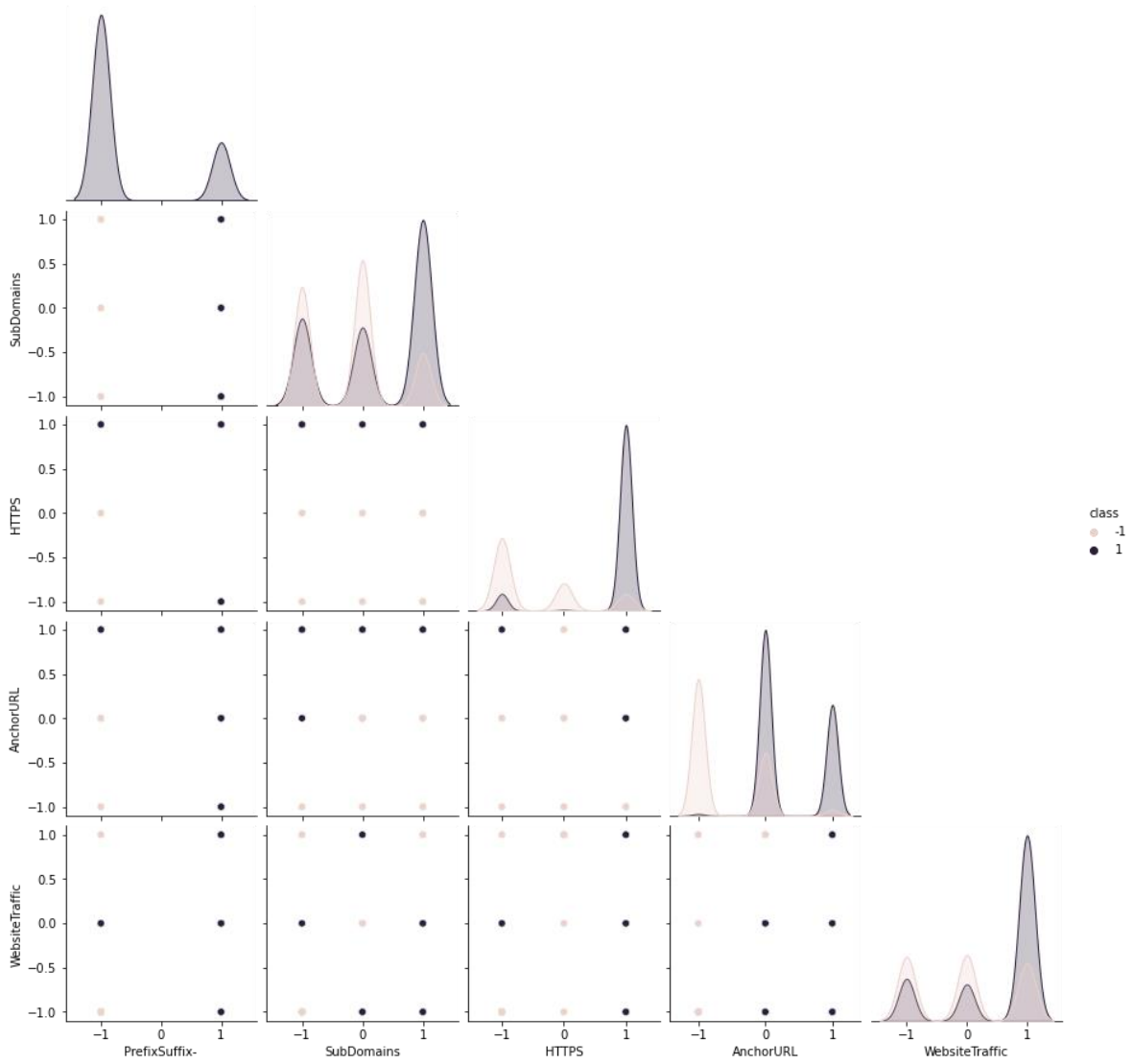

Figure 4.5 Visualizing the Data v2

4.4 OUTPUT

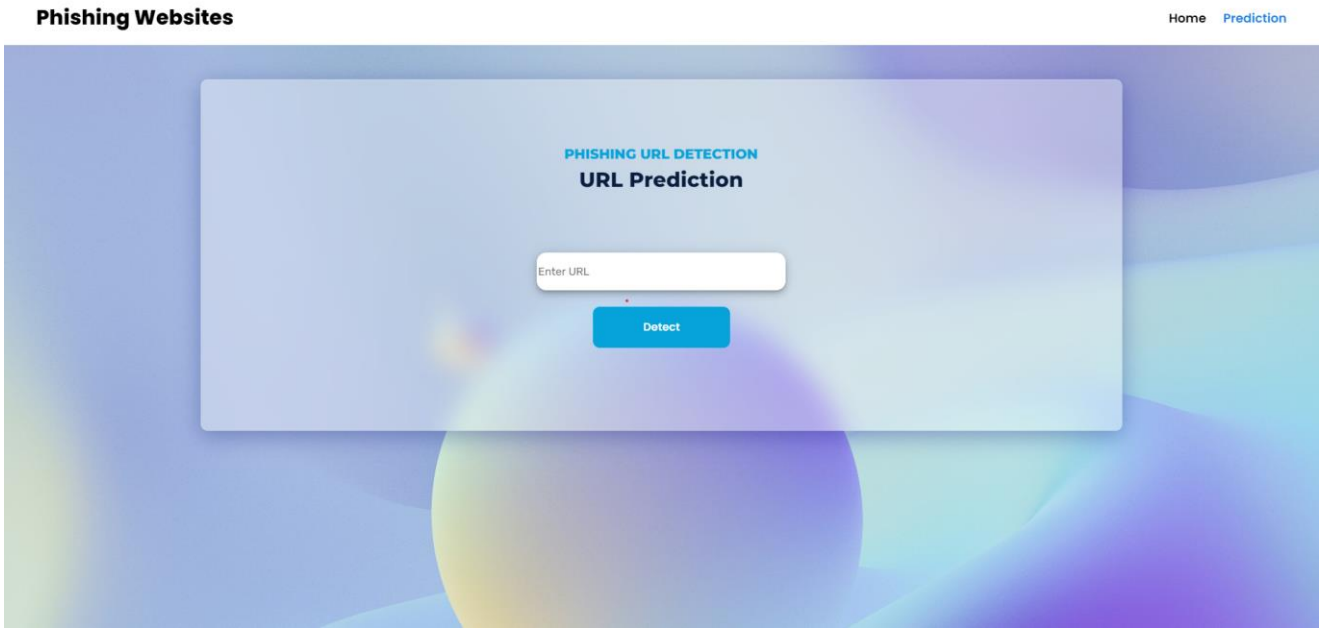

Figure 4.6 URL Prediction Page

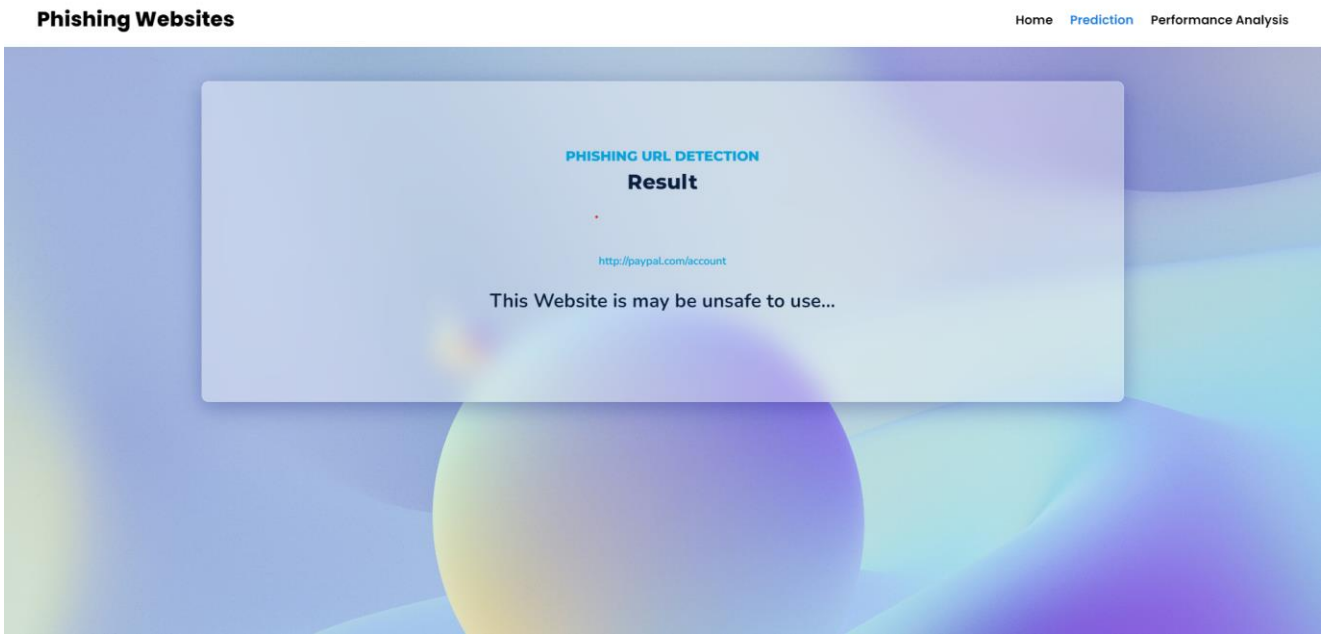

Figure 4.7 Prediction Result Page

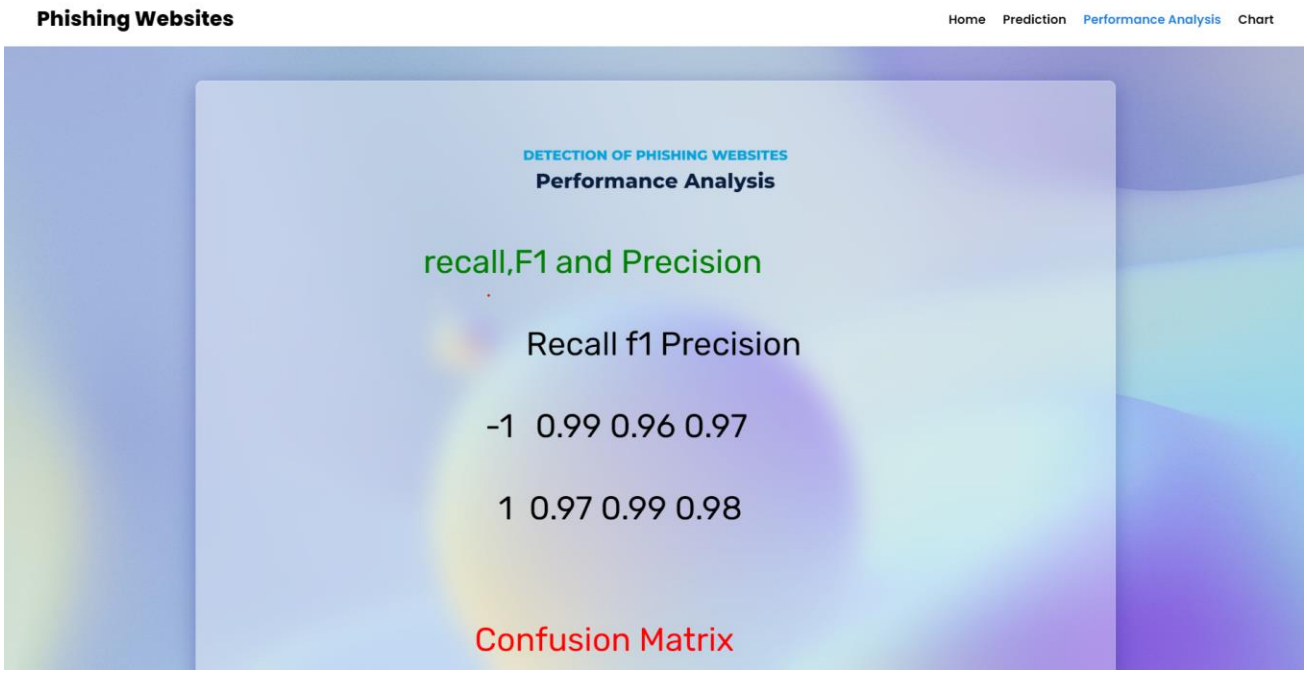

Figure 4.8 Performance Analysis Page

CHAPTER 5
CONCLUSION

5.1 SUMMARY

Phishing attacks are a rapidly expanding threat in the cyber world, costing internet users billions of dollars each year. It involves the use of a variety of social engineering tactics to obtain sensitive information from users. Hence, Phishing techniques can be detected using a variety of types of communication, including email, instant chats, pop-up messages, and web pages.

This project was able to categorize and recognize how phishers carry out phishing attacks and the different ways in which researchers have helped to solve phishing detection. Hence, the proposed system of this project worked with different feature selection and machine learning and deep neural networks such as Decision Tree,

Support Vector Machine, XGBooster, Multilayer Perceptions, Auto Encoder Neural Network, and Random Forest to identify patterns in which URL links can be detected easily.

The Model with the highest accuracy based on the feature extraction algorithm used to identify phishing URL from legitimate URL links was integrated to a web application where users can input website URL links to detect if it is legitimate or phishing.

5.2 Contribution to Knowledge

This project provides a new and faster way to help users detect if a URL link is phishing or legitimate and also provide them access to educational resources about phishing attacks.

5.3 CONCLUSION

The system developed detects if a URL link is phishing or legitimate by using machine learning models and deep neural network algorithms. The feature extraction and the models used on the dataset helped to uniquely identify phishing URLs and also the performance accuracy of the models used. It is also surprisingly accurate at detecting the genuineness of a URL link.

Through this project, one can know a lot about phishing attacks and how to prevent them. This project can be taken further by creating a browser extension that can be installed on any web browser to detect phishing URL Links.

CHAPTER 6
REFERENCES

[1] A. J. Ashutosh Kumar Singh, and Keshav Singh, "A Survey on Cyber Security Awareness and Perception among University Students in India," Journal of Advances in Mathematics and Computer Science, November 2021.

[2] S. Shams Hussein, W. Hashim Abdulsalam, and W. Abed Shukur, "Covid-19 Prediction using Machine Learning Methods: An Article Review," Wasit Journal of Pure Sciences, vol. 2, no. 1, pp. 217-230, 03/26 2023, doi 10.31185/wjps.124.

[3] S. Mahdi Muhammed, G. Abdul-Majeed, and M. Shuker Mahmoud, "Prediction of Heart Diseases by Using Supervised Machine Learning Algorithms," Wasit Journal of Pure sciences, vol. 2, no. 1, pp. 231-243, 03/26 2023, doi: 10.31185/wjps.125.

[4] N. Kareem, "Afaster Training Algorithm and Genetic Algorithm to Recognize Some of Arabic Phonemes."

[5] A. S. Hashim, W. A. Awadh, and A. K. Hamoud, "Student performance prediction model based on supervised machine learning algorithms," in IOP Conference Series: Materials Science and Engineering, 2020, vol. 928, no. 3: IOP Publishing, p. 032019.

[6] H. H. Chinaza Uchechukwu, and Jianguo Ding, "A Survey of Machine Learning Techniques for Phishing Detection," IEEE Access, August 2020.

[7] P. Kalaharsha and B. M. Mehtre, "Detecting Phishing Sites--An Overview," arXiv preprint arXiv:2103.12739, 2021.

[8] B. Sabir, M. A. Babar, R. Gaire, and A. Abuadbba, "Reliability and Robustness analysis of Machine Learning based Phishing URL Detectors," IEEE Transactions on Dependable and Secure Computing, 2022.

[9] M. Almousa, T. Zhang, A. Sarrafzadeh, and M. Anwar, "Phishing website detection: How effective are deep learning-based models and hyperparameter optimization?," Security and Privacy, vol. 5, no. 6, p. e256, 2022.

[10] H. Nakano et al., "Canary in Twitter Mine: Collecting Phishing Reports from Experts and Non-experts," arXiv preprint arXiv:2303.15847, 2023.

[11] S. Jain, "Phishing Websites Detection Using Machine Learning," Available at SSRN 4121102.

[12] A. Lakshmanarao, P. S. P. Rao, and M. B. Krishna, "Phishing website detection using novel machine learning fusion approach," in 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021: IEEE, pp. 1164-1169.

[13] L. Tang and Q. H. Mahmoud, "A Deep Learning-Based Framework for Phishing Web-site Detection," IEEE Access, vol. 10, pp. 1509-1521, 2021.

[14] A. D. Kulkarni and L. L. Brown III, "Phishing websites detection using machine learn-ing," 2019.

[15] I. Tyagi, J. Shad, S. Sharma, S. Gaur, and G. Kaur, "A novel machine learning ap-proach to detect phishing websites," in 2018 5th International conference on signal pro-cessing and integrated networks (SPIN), 2018: IEEE, pp. 425-430.

[16] M. Karabatak and T. Mustafa, "Performance comparison of classifiers on reduced phishing website dataset," in 2018 6th International Symposium on Digital Forensic and Security (ISDFS), 2018: IEEE, pp. 1-5.

[17] X. Zhang, Y. Zeng, X.-B. Jin, Z.-W. Yan, and G.-G. Geng, "Boosting the phishing detection performance by semantic analysis," in 2017 IEEE international conference on big data (big data), 2017: IEEE, pp. 1063-1070. [18] W. Fadheel, M. Abusharkh, and I. Abdel-Qader, "On Feature selection for the predic-tion of phishing websites," in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2017: IEEE, pp. 871-876.

[19] S. A. Anwekar and V. Agrawal, "PHISHING WEBSITE DETECTION USING MACHINE LEARNING ALGORITHMS."