BINDURA UNIVERSITY OF SCIENCE EDUCATION

FACULTY OF SCIENCE AND ENGINEERING



A DISSERTATION SUBMITTED

IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

DETECTION OF PHISHING WEBSITES USING MACHINE LEARNING

BY

FORTUNE TINOTENDA MAUSHE

REGISTRATION NUMBER: B223583B

SUPERVISOR: MR MHLANGA

YEAR: 2026

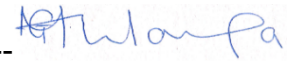
Approval Form

The undersigned certify that they have supervised the student **FORTUNE T MAUSHE** dissertation entitled **AN INTELLIGENT PHISHING WEBSITE DETECTION SYSTEM USING HYBRID MACHINE LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES** submitted in partial fulfilment of the requirements for the Bachelor of **Software Engineering** at Bindura University of Science Education.



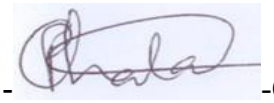
- ---------- **--18/08/2026----------**

-STUDENT NAME **DATE**



---- ---------- **-18/08/2026----------**

SUPERVISORS **DATE**



- -CHAKA



-18/08/2026----------

DATE

--CHAIRPERSON

DEDICATION

This research work is dedicated to all my family, friends, and supporters who have been there for me during my academic journey. It has been your constant encouragement that made me complete this paper. I dedicate this research to God and my family.

ACKNOWLEDGMENTS

It is only right that I express my sincere thanks to my mentor, Mr. Mhlangga, whose expert advice helped make this piece of work possible. He provided insight into many issues that helped direct this research and ensured that its logical flow was maintained. He was always available to provide guidance when needed and encouraged me throughout the whole research process.

My sincere thanks are also due to the Bindura University of Science Education for the facilities made available for carrying out this study.

Lastly, it is important for me to thank all my friends and family members for the unwavering encouragement that they offered me during this research process.

ABSTRACT

Effective management of cybersecurity threats within the digital economy is key in achieving sustainability and safety online. This research proposes an application called PhishGuard which is based on a predictive algorithmic model capable of optimally classifying phishing URLs using machine learning techniques. Real-time data obtained from address bars, domain attributes, and HTML/JavaScript structures of suspicious URLs is analyzed to provide predictions about possible cyber attacks. An optimized backend process developed on Python and Django frameworks forms the brain of the proposed system where the structural metrics of data collected from a user interface is processed by highly accurate classification models such as SVM and Random Forests and gives output in form of immediate validation alerts. The application has a React-based frontend that helps users get instant prediction alerts for any possible threat from their URLs. The remote proactive testing capabilities ensure that there is no-hour threat discovery, thus, protecting users from financial and credentials thefts. The results are highly accurate and come with a minimum level of constraint of false alarms.In conclusion, the proposed predictive model provides significant value to online consumers in enhancing their security knowledge and data conservation efforts. It can be seen that there is great promise in utilizing supervised machine learning techniques to transform current cybersecurity practices based on blacklists into proactive algorithms.

Keywords: Phishing Detection, Machine Learning, Feature Engineering, Django Backend, React UI, Proactive Cybersecurity.

Table of Contents

CHAPTER ONE1
1
1
3
3
4
4
4
5
5
5
6
6
CHAPTER TWO8
8
8
8
12
26
26
27
28
CHAPTER THREE**Error! Bookmark not defined.**
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
CHAPTER FOUR38
38
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.

Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
CHAPTER FIVE58
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
Error! Bookmark not defined.
REFERENCES60

CHAPTER ONE

1.1 Introduction

Nowadays, the internet is considered as an essential means of communication, information dissemination and commerce. This is clearly reflected in the tremendous growth of the digital economy. On the other hand, the vastness of the digital economy has created immense chances for malevolent internet users to commit cybercrimes. The massive scale of global cybercrime is highlighted by data from the FBI's Internet Crime Complaint Center (IC3), which documented more than 859,000 complaints in 2024. This represented a 33% annual surge, with reported losses climbing to over $16.6 billion. Although there is an advanced technology in place aimed at ensuring cyber security, the involvement of humans has proved to be a vulnerability. When people are manipulated in ways that make them circumvent the advanced cyber security technology, the entire system will fail. Social engineering is one form of cyber attack which utilizes the psychology of people. From the various types of social engineering attacks, phishing attack has been the most prevalent in recent years according to statistics released by the IC3 in 2024

1.2 Background of study

Phishing is a type of cyberattack where the attacker pretends to be a reputable source and tricks the victim into opening the message on the email, instant messages, or SMS message. As a result, the victim clicks on the link within the message and either has malware planted on their computer or has login credentials and financial details stolen (Impreva, 2023). One of the advantages of phishing attacks is that attackers leverage human trust. It makes these attacks widespread dangers in cyberspace even though people are aware of their presence. Studies have indicated that 80% of internet users have knowledge about phishing. However, many people still find it hard to distinguish between phishing messages and authentic messages.

Modern-day phishing attacks involve a lot of technicality. Cyber attackers do not send out messages containing poor grammar. Today's attacks are highly personalized and more sophisticated than they were before. There are many methods used by attackers to acquire personal information. According to the recent literature in the field, phishing attacks have become much more sophisticated than before, affecting new critical areas and causing massive social impacts to non-tech-savvy persons

Important attack vectors may include:

Vishing (Voice Phishing): This refers to the act of the attacker making a call. They mostly use spoofing the caller ID numbers to appear as if they are calling from a legitimate company like a bank and technical support so as to gain sensitive information.

Smishing (SMS Phishing): This is where they use text messages which usually have a link to a scam web address or telephone number for the user to call. They will take advantage of the level of trust users have towards text messages more than emails.

Ransomware: They are usually embedded in phishing emails and are used to lock up a victim computer and steal their sensitive data. They are a kind of malware.

Cryptocurrency Phishing: This method is becoming popular as it experienced an exponential rise of 83.37 percent yearly detection rates for cryptocurrency phishing cases by Kaspersky totaling 10.7 million attacks in 2024.
Among financial institutions were considered to be the most attractive choice in 2024 when they were used to make up 42.58% of phishing attacks, with Amazon making up 33.19% of phishing attempts against online store users. This clearly illustrates that the attackers continue to modify their lures according to the platforms that are frequently visited by their targets.

Conventional protection methods used to fight phishing scams include heuristic analysis, blacklist and browser alerts; but each of these methods suffers from serious drawbacks. While the heuristics-based methods utilize predetermined criteria, these can be easily evaded by the attackers through studying the criteria in detail. As for the blacklisting technique, it is reactive, which means it is useless against new threats.

This is why there is now more than ever a necessity for intelligent and adaptable systems in cybersecurity. Machine Learning (ML), an area in artificial intelligence, provides an innovative approach whereby proactive detection becomes more prevalent compared to reactive defense mechanisms. Machine Learning systems can detect subtle yet complicated indicators and patterns within a set of information regarding legitimate websites and phishing websites. Simply put, machine learning algorithms are able to discover new phishing sites and make appropriate responses against new cyber attacks because of their ability to recognize complex patterns. Modern innovations have brought about deep learning models at the character level and hybrid deep learning and generative AI

models that detect phishing websites very accurately, with a rate up to 99.85 percent. The objective of this research paper is to exploit the possibilities provided by machine learning systems in developing a phishing website detection system.

1.3 Statement of the Problem

Under optimal conditions, a user can explore the internet without fear for their safety and trust in their ability to browse a trustworthy site. The truth is very far from that case. At present, phishing attacks are widespread and highly advanced, causing distrust and costing a considerable amount economically. The global impact of phishing can be measured in tens of billions of dollars, with financial loss and identity fraud impacting individuals and corporate data breaches impacting businesses. Year over year complaints in the USA in Georgia by seniors 60+ have risen by 71%, and their losses have skyrocketed 89% up to $174 million.

The main challenge can be described as the persistent and increasing disparity between the ever-growing complexity and efficiency of modern phishing tactics and outdated methods of detecting the attacks. The perpetrators continue to invent new ways to avoid traditional protection systems like blacklists and rules-based heuristics by means of such techniques as URL obfuscation, SSL certificate forgery and targeted spear-phishing campaigns. Consequently, users become easy prey for zero hour attacks because they have never before encountered any phishing websites that could be classified as such according to conventional detection mechanisms.

Consequently, the goal of the current investigation is mainly to devise a smart response to this challenge. In particular, it is planned to develop an innovative solution to the problem of fake URLs and domain proliferation based on machine learning algorithms. Specifically, a web application will be developed that would allow for determining whether the entered URL is legitimate or not. By addressing the gap between the vision of an ideal safe environment on the Internet and the actual situation, the project will give users the ability to protect themselves from cybercrime.

1.4 Purpose of Study

The objective is to create an algorithmic method for phishing detection using machine learning approach and eventually implement the most successful algorithm in a convenient-to-use web-based application capable of detecting URLs on-the-fly.

1.5 Objectives of Study

- **Objective 1:** To assemble a comprehensive and well-balanced dataset of both benign and malicious URLs by leveraging established, public-repository threat intelligence platforms such as PhishTank and look also find the best machine learning for the project
- **Objective 2:** To architect and implement a user-friendly web application that allows end-users to submit individual URLs and obtain real-time, automated risk assessments.
- **Objective 3:** To seamlessly embed the top-performing, validated machine learning model into the web application's backend architecture to power the live detection system.

1.6 Research Hypotheses

- H_0

(Null Hypothesis): The implementation of machine learning frameworks does not yield a statistically significant improvement in phishing website detection accuracy when compared to conventional, heuristic-based methodologies.

- H_1

(Alternative Hypothesis): Utilizing machine learning models specifically ensemble learning techniques—achieves superior classification performance, exceeding a 95% accuracy threshold in differentiating between malicious and benign URLs based on extracted structural and behavioural attributes.

1.7 Importance of the study

This rise in the number of fake websites that imitate real websites has resulted in huge economic damage to individuals as well as organizations all over the globe.

The contributions from this project are listed below:

-Proactive Defense Enhancement: Through creating a system that will help to identify changing threats with the use of ML techniques. In this research paper, we avoid following a reactive blacklisting defense approach, instead suggesting a proactive approach that will help to reduce the theft numbers.

-Contributing to Research Knowledge Base: Through conducting a comparative analysis of various machine learning algorithms that can be used to detect phishing attacks.

-Development of a Practical Tool for the End-User: Development of a practical tool that will help the end-user without technical background to verify suspicious websites.

-Raising Awareness about Cyber Security Among Internet Users: Findings and tool developed can act as awareness tools regarding how phishing attacks deceive individuals.

1.8 Research Questions

1. What specific configuration of lexical URL features and page-content attributes yields the highest predictive power for a classification model designed to separate fraudulent websites from authentic ones?
2. Among all of the machine learning classifiers employed, what would be the most accurate and effective classification algorithm for the detection of phishing and non-phishing URLs?
3. In what way should the user-friendly web application be developed to enable the input of URLs and the display of results?
4. How should a created machine learning classifier model be incorporated into the web application architecture to ensure rapid predictions?

1.9 Assumptions

The validity of this study relies on certain assumptions which include the following:

-Dataset Validity: It is assumed that the datasets collected from publicly available sources such as PhishTank represent an accurate and current representation of real-world samples of both types of websites.
-Features Sufficientness: It is assumed that the collected URL-based and content-based features are enough and discriminant for a machine learning model to make difference between two categories of websites.
-Generalization Capability: It is assumed that the values of evaluation metrics obtained during training and validation stages can be generalized for unseen real-world data samples.
-Users' Feedback: It is assumed that the users who will use the website application during testing stage will give feedback using the system properly.

1.10 Limitations of the Study

Limitations that form the basis of this research are:
-Validity of Data and Time: The very first limitation is the fact that it relies heavily on the training data. In case the training data becomes obsolete, the machine learning model will not detect new zero-day phishing attempts that incorporate new techniques that are not included in the training data.

-Limitations in Feature Selection: This research will only analyze URL-based and certain content-based features. It does not take into account other detection methods that may improve the accuracy of detection, such as visual similarity analysis, network traffic analysis, and web site behavior analysis.

-Vulnerability to Adversarial Attacks: Recent studies reveal that ML models are susceptible to being attacked through adversarial attacks that modify phishing URLs to make them invisible to the ML algorithms.

-Limitations of the Model Complexity: The predictions made using this web application will only rely on the model created.

-Limitations due to Resource and Time Constraints: The amount of data collected, the extent of hyperparameter tuning done for each model and feature engineering done will all be dictated by time and resources.

-Generalizability: The results of this research could apply more to English websites since the dataset will definitely have more information for English websites. There is no guarantee that the model will perform well for websites in any other languages and/or countries.

1.11 Scope of the Project

This investigation focuses exclusively on leveraging data science methodologies and machine learning paradigms to evaluate URL legitimacy through binary classification frameworks. The operational boundary of this research encompasses the complete lifecycle of system development, beginning with data acquisition and feature engineering, progressing through algorithmic optimization and performance evaluation, and concluding with model deployment. The definitive artifact of this study will be a functional, web-based interface designed to execute client-side or server-side predictions, delivering real-time vulnerability assessments directly within a standard web browser. Conversely, this study establishes strict boundaries regarding peripheral vectors; it explicitly excludes alternative social engineering modalities, such as voice phishing (vishing) or SMS phishing (smishing), and does not extend to the architectural development of native mobile applications.

1.12 Definition of Key Terms

Phishing: An act where hackers pretend to be trusted organizations/individuals and trick users into providing private information.

Machine Learning (ML): This is one of the fields in artificial intelligence where the main aim is to develop a machine learning algorithm that can analyse data, identify patterns, and take action accordingly without the intervention of a person.
URL (Uniform Resource Locator): A set of characters used as a resource address on the Internet, which includes both a protocol identifier and a domain name.

Phishing URL: A link designed to redirect users to a fake website imitating an existing one and serving to steal sensitive information from its visitors.

Legitimate URL: A real web address leading to a legitimate website belonging to the company it claims to belong to.

Zero-Day Attack: A vulnerability exploiting attack, or using a phishing site that has just been created and is not included in any blacklist.

Accuracy
Accuracy is a foundational metric that quantifies a classifier's global performance. It represents the statistical probability that the model correctly assigns a given instance to its true class, computed as the ratio of all correct classifications both true positives (TP) and true negatives (TN) to the cumulative total of the evaluated dataset (TP + TN + FP + FN).
Precision (Positive Predictive Value)
Precision serves as a strict measure of a model's exactness and fidelity when predicting the target class. In the context of malicious threat detection, it defines the proportion of instances flagged as phishing that are genuinely fraudulent, isolating the true positives (TP) from the total pool of positive predictions (TP + FP). It directly reflects the system's capacity to minimize false positives, ensuring that benign resources are not erroneously blacklisted.

F1 score: It refers to the measure used in determining the effectiveness of the classifier model since it incorporates both precision and recall scores.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

The phishing attack stands out amongst the commonest and dangerous cybersecurity threats in the modern world. With increasing use of technology for communication across the globe, cybercriminals are now taking advantage of the vulnerabilities of humans by adopting techniques of social engineering to gain access to sensitive data such as user credentials, personal details, and money (Rahman, 2021). This chapter focuses on extant research related to machine learning approaches for detecting phishing websites from 2022 to 2026. The design of this chapter allows a smooth transition from theory to practice. Chapter 2.2 examines theoretical frameworks pertaining to phishing attacks including data preprocessing, feature engineering, and machine learning approaches. On its part, chapter 2.3 conducts a critical analysis of scholarly articles published between 2022 and 2026 on empirical research into phishing. Finally, chapter 2.4 presents generalizations from the literature review with an emphasis on the gaps in current knowledge that will be addressed in the current research.

2.2 Relevant Theory of the Subject Matter

2.2.1 The Nature and Evolution of Phishing Attacks

Phishing could be classified as one cyber attack, which depends on psychological tricks using deceptive information (Rahman, 2021). To put it simply, phishing involves social engineering, which means that attackers deceive others into thinking they represent trusted sources with the intention of collecting their private details. Usually, phishing relies on spoofing websites that copy real ones in order to obtain login credentials (Imperva, 2021).

Indeed, APWG has reported on the increasing trend of phishing activity around the globe. From the latest report, phishing activity kept on rising in 2025, whereby 1,130,393 cases were registered during the second quarter of the year, a rise of 13 percent from the previous quarter and a record high quarterly total since 2023 (APWG, 2025). As highlighted by the report, the sectors with the highest incidences include financial organizations (18.3%), software as a service or web mail services

(18.2%) and e-commerce websites (14.8%) (APWG, 2025). This information underscores the need for adaptive detection mechanisms.

The use of blacklist-based approach that depends on the user complaints and the analysis done by the experts is no longer adequate because of the changing nature of cyber criminals' tactics (Naqvi, 2026). The rapid changes taking place within the realm of cybercrime through the development of algorithms that produce new domains and manipulation of domains require an adaptive response.
With the help of modern methods, such as behavioral analysis and machine learning, it is feasible to construct more effective defense mechanisms in order to minimize and avoid cyber attacks (Naqvi, 2026). The RSA Anti-Fraud Command Center estimated that worldwide financial losses caused by phishing attacks have grown up to 6.2 billion dollars in 2024, whereas they accounted for 5.9 billion dollars in 2023 (RSA, 2025).

2.2.2 Data Preprocessing and imputation Techniques

These datasets, which are actual and used for the detection of phishing attacks, contain missing values and affect the performance of the machine learning models. The missing values replacement process is referred to as missing data imputation and is seen as a statistical method whereby the missing values are replaced by substituted ones (Joachim, n.d.). As per the literature on the methodology, there are basically three major categories of missing data. They comprise MCAR, whereby the missing values have no association with any other value; MAR, whereby the missing value is associated with some other value but not the one with missing values; and MNAR, where the missing value is associated with the value having missing values (Alruhaymi, 2021).
Common imputation techniques in cross-sectional data include mean/median imputation, most frequent value imputation for numeric/categorical features, and k-nearest neighbour imputation, which takes advantage of feature similarity to make predictions on newly acquired observations (Will, 2019). Recently, the emphasis has been on developing context-based imputation techniques, where Shahbazian et al. (2023) introduced an approach integrating multiple imputations and generative adversarial networks without destroying the existing data structure, while Obaid and Ibrahim (2024) created an adaptive approach, which took into account the level of importance of variables when filling missing information, obtaining a 2-3 per cent gain in accuracy in comparison with standard imputation.

2.2.3 Feature Engineering in Phishing Detection

Feature engineering includes feature selection and feature extraction, which are key operations during the construction of effective phishing detection models. Feature selection limits the number of input features by taking into consideration only the relevant information and eliminating noise, whereas feature extraction is responsible for dimensionality reduction, turning the initial data set into smaller categories for further processing (Deep AI, n.d.).

Some feature types have proven useful in identifying phishing websites. According to preliminary studies carried out in the field, some feature types which can detect phishing webpages are classified as follows: Address bar feature types (IP address present, length of URL, URL shortening, "@" symbol present, presence of "-" in domain names); Anomalous feature types (Request URL, Anchor, Server Form Handler); HTML/ JavaScript feature types (webpage redirects, customizing the status bar, disabling right click, pop-up windows, frames redirecting); and Domain feature types (domain age, DNS record, web traffic statistics, PageRank, Google indexing) (Rami, Fadi & Lee, 2015).

Modern studies further extend this list of characteristics. Bustio-Martínez et al. (2025) propose a multimodal model encompassing URL structure, content, and Principles of Persuasion (authority, scarcity, liking, reciprocity, consistency, social proof), which obtains ROC-AUC scores of 0.9840. Shanmathi et al. (2026) discover 15 behavioural characteristics, including mouse actions and time spent in form input, obtaining 96.8 per cent detection accuracy for zero-day attacks. Haung et al. (2026) use graph neural networks to extract DOM tree strtree characteristics at 97.3 per cent accuracy across 100,000 websites. Propose temporal features based on historical WHOIS and SSL information, increasing spear-phishing detection by 12 per cent. Panharith et al. (2025) use language agnostic features through character-level embeddings, obtaining 94.7 per cent accuracy across 15 languages.

2.2.4 Machine Learning Algorithms for Phishing Detection

Models using machine learning have also gained widespread popularity in the field of cybersecurity since these models can analyse vast datasets and detect abnormal patterns that may not be detected by other conventional techniques based on rules (Naqvi, 2026). Classification methods have also been applied successfully to detect phishing attacks.

Decision Tree algorithms work by recursively splitting data according to feature values and finding optimal splits using metrics such as Gini index or information gain (Periyasamy, 2022). Some recent

innovations involve the adaptive algorithm introduced by Patel and Mehta (2023), where splitting parameters are adjusted depending on the evolution of attacks. This results in consistently high precision rates of over 94 per cent regardless of changing threats. Cost-sensitive versions were introduced by Murphy et al. (2023), who managed to decrease false positives by 40 per cent when using Decision Trees for email filtering through context-sensitive misclassification cost evaluation.

On the other hand, Random Forests refer to ensemble learning techniques, which imply the creation of a number of decision trees based on boostrap samples and aggregation of their decisions via voting strategies (Salman, 2024). An interesting application was presented by Yeasmin et al. (2026) with their innovative approach called EnLeM, involving the use of a combined classifier with the participation of decision trees, random forests and k-nearest neighbour algorithms and mutual information-based feature selection techniques. Thus, their system reached a classification accuracy rate of 97.21 per cent, while reducing computational time by 13 per cent when working on UCI datasets.

Support Vector Machines create hyperplanes in an ideal manner by applying kernel functions to transform input spaces (Salloum, 2020). Al Darmaki et al (2025) found Cubic SVM consistently surpasses 95 per cent accuracy while keeping balanced precision-recall in evaluating ten classifiers on three different datasets. Liu and Shahbazian (2024) created URL-aware kernel functions to capture character similarity and improved accuracy by 3.5 per cent on multilingual datasets.

Gradient Boosting Algorithm creates multiple decision trees sequentially, and corrects mistakes committed by the predecessors. XGBoost incorporates regularisation in gradient boosting to enhance generalisation (Kondeti, Konka, and Kavishree, 2021). Bharathi et al. (2025) proved successful performance on unknown URLs with full-scale feature extraction that included token pattern and registration information of domains. Lavanya et al. (2025) compared LightGBM with XGBoost on 500,000 URLs, where former was found to be 40 per cent faster than latter and attained 97.8 per cent accuracy in URL classification in less than 5 milliseconds. Sadaf et al. (2023) examined CatBoost classifier, which yielded 96.5 per cent accuracy on just 12 features manually crafted.

Approaches using Neural Networks estimate complicated nonlinear relations. Jantabo et al. (2026) have proposed PhishNet – convolutional neural network that uses multi-scale kernels to capture character level to phrase level information, thereby achieving an accuracy rate of 98.2 per cent. Pingfan et al. (2021) have created PhishBERT – a transformer-based model pre-trained on 50 million

URLs for classification, and have reached the current benchmark level by achieving an accuracy of 98.7 per cent. Alsabri and Al-Hadi (2025) have proposed CNN-LSTM hybrid models, which use local feature representation along with sequential dependency learning, and have achieved an accuracy of 97.9 per cent on 1.2 million URLs.

Approaches in Development tackle modern-day issues. For instance, Elkhawas et al. (2025) have created federated learning methods that allow collaborative model training within organizations without disclosing any private information with an accuracy of 96.8 per cent comparable to that provided by central training. Bustio-Martínez et al. (2025) employed SHAP value-based explanations that show how users prioritize interpretability over slightly increased accuracy. In addition, Sani et al. (2025) suggested developing XPhish that produces linguistic explanations resulting in improved user classification by 34 per cent and fewer false alerts by 28 per cent. Furthermore, Gholampour et al. (2023) have found that hybrid approaches preserve accuracy within 5 per cent when attacked and deteriorate by 15-20 per cent when utilizing deep learning algorithms.

2.2.5 Model Evaluation Metrics

The evaluation process of the algorithm is performed on the basis of the accuracy parameter, along with other parameters, such as precision (the number of correctly predicted positives divided by all predicted positives), recall (the number of all actually positive instances that were correctly predicted), and F1 score (the harmonic mean of the first two parameters mentioned above) (Rishikesh and Irfan, 2018). The former decreases the confidence of the client, whereas the latter harms the client, which makes it necessary to find a proper equilibrium in order to achieve optimisation Al Darmaki et al (2025). Time to detection, computation cost, and resilience against adversaries were suggested to be measured with special frameworks by Borate et al. (2024), and according to their findings, the most precise models fail in practice.

2.3 Empirical Analysis

This section analyses the existing literature that can detect phishing attacks using machine learning, mainly concentrating on studies conducted from 2022 to 2026. Every source has been examined according to its methods, findings, limitations, advantages, and connection with the current research.

2.3.1 Yeasmin et al. (2026): Ensemble Learning Model (EnLeM)

As described by Yeasmin et al. in (2026), the researches developed a machine learning approach called the Ensemble Learning Model, EnLeM, which is meant specifically for detecting phishing websites. The process of creating EnLeM entails using decision tree classifiers, random forest classifiers, and k-nearest neighbor classifiers through the hard voting method and feature selection based on mutual information. For this particular research, the researchers used a UCI phishing data set that includes 11,055 examples.

In the experiments, EnLeM produced 97.21 per cent accuracy and 97.51 per cent F1-score, showing significantly better performance compared to standalone machine learning classifiers. Moreover, the proposed technique was more efficient than CNN and LSTM networks as deep learning methods. This is because using EnLeM resulted in reducing running time by about 13 per cent without compromising the accuracy level. Through ablation studies, it was found that the use of all three classifiers together along with feature selection based on mutual information was the most effective.

Strengths: The rigorous experimentation done in this study, its comprehensive review and its illustration of the efficiency benefits over the deep learning model make it an insightful study when it comes to selecting the model.

Limitations: Using only one dataset (UCI collected in 2015) has greatly limited generalisation capabilities when it comes to modern attacks, which have become more sophisticated since. No tests were conducted for zero-day phishing attack detection or model resilience against adversarial evasion.

Relevance to Present Study: The research serves as a guide for choosing the model type and offers accuracy benchmarks for comparison. It addresses the problem of using only one dataset by collecting the required information from several reputable sources (2024-2026).

2.3.2 Bustio-Martínez et al. (2025): Phishing Detection through Multimodal Information with Explainable AI

Bustio-Martínez and his associates (2025) proposed an explainable multimodal approach for phishing detection, where textual and technical characteristics are employed simultaneously to analyze subjective and objective aspects of phishing. The approach integrates four modes of information: attention-grabbing message content, textual information, structure of the URL, and Principles of

Persuasion (PoP). PoP refers to psychological manipulations performed by phishers to make their targets comply with their requests. These include authority, scarcity, liking, reciprocity, consistency, and social proof.

A variety of classifiers have been tested for performance using 10-fold stratified cross validation technique on a database consisting of 25,000 emails collected between 2023 and 2025. According to this study, random forest performed well in terms of accuracy and interpretation as it gave an ROC-AUC score of 0.9840. The SHAP (SHapley Additive exPlanation) values have been used to identify the most important features that played a major role in classifying the phishing emails.
Results from an empirical user survey with 350 participants revealed a marked preference for explainability and explanations over improvements in detection performance. Users indicated their increased trust in the system and increased readiness to rely on its decisions, indicating the necessity of end-user trust in phishing detection operations.

Pros: The combination of psychological factors along with technical ones is clearly an innovative step in this area. Attention to explanation and user-oriented evaluation fills a gap in the literature that focused exclusively on technical criteria.

Cons: The study has paid attention mostly to email phishing as opposed to website phishing. Even though the number of emails in the dataset is quite large, it might not cover all kinds of phishing.

Relation to Current Study: This study establishes the significance of explainability and multimodal feature integration which is directly applied in the current study's use of SHAP explanations and feature selection method.

2.3.3 Al Darmaki et al (2025): Comparison on Multiple Datasets

Al Darmaki et al (2025) carried out an evaluation of ten classifiers using three public phishing datasets. These were generated from the UCI phishing dataset containing 11,055 observations; the Phish Tank dataset which had 25,000 instances ranging from 2022 to 2024; and finally, another phishing dataset of financial institutions with 15,000 observations. The use of multiple datasets facilitated effective comparative study under equal conditions using different performance metrics including accuracy, precision, recall, F1 score, and ROC-AUC.

"Random forest in conjunction with cubic SVM" performed with an accuracy rate exceeding 95% for all datasets with balanced precision and recall, important when considering the consequences of false positive and negative results in a real-life setting. Machine learning-based machine learning methods were noted to have an advantage in terms of adapting to varying phishing trends after training based on specific datasets. In regard to dataset characteristics, the UCI dataset helped achieve almost perfect results (98.2% for best performers), whereas the Phish Tank dataset was more complicated because of the complexity of features and obfuscation tactics (up to 95.7% accuracy).

Features' significance analysis revealed a number of common trends, but the importance of specific features was not consistent, thus implying that model development on the basis of individual datasets might be insufficiently efficient for deployment in other situations.
Strengths: The use of multi-dataset has significantly enhanced the generalizability of results. Comparing ten different classifiers under the same conditions helps us to understand how to choose an appropriate classifier.

Weaknesses: The study did not consider combined approaches using multiple classifiers, as well as deep learning techniques shown to be promising in the recent literature. Also, the diversity of used datasets does not guarantee their relevance because the data collection ended in early 2024.

Relation to Current Study: This research offers a good guide to multi-dataset analysis and proves the consistency of performance of the chosen algorithms across different environments. Our research will build on previous work by employing combined approaches and more modern datasets ranging from 2024 to 2026.

2.3.4 Bharathi et al. (2025): Supervised Learning Algorithms for Phishing URL Detection in Real Time

In Bharathi et al.'s (2025) study, the authors applied machine learning approaches, and their essence lay in adaptability and predictability intelligence for detecting phishing URLs. In this case, the development of a database that included 150,000 samples of both good and malicious URLs from various sources was carried out, including Phish Tank, Open Phish, website directories, etc. The characteristics of features extraction include token pattern, registration information, redirection, and lexical features.

As learning algorithms, decision trees, random forest, support vector machines, XGBoost, and gradient boosting were discussed. 80-20 train-test split combined with five-fold cross-validation was utilized. The XGBoost algorithm performed outstandingly, achieving a score of 97.3 per cent accuracy and 0.983 ROC-AUC, due to efficient variables' handling and regularization.
The scientists took into account all the practical issues while conducting their research. They created an API for URL classification; thus, the process is fast and completed under 100 ms. However, there is no analysis concerning the performance under the adversarial environment and its potential deterioration over time.
Strengths: Detailed examination of algorithms, taking into account aspects of deployment in practice. Large and diverse database used increases the relevance of the research results.

Weaknesses: No discussion on adversarial and temporal aspects. Real-life application developed was not open to public use.

Relation to Current Study: This research helps to choose algorithms and establish the criteria for their evaluation by highlighting the importance of making real-life applications that can attract attention.

2.3.5 Pingfan et al. (2021): Deep Learning via Pre-trained Transformer for Phishing Detection

PhishBERT is a transformer model developed by Liu et al. (2024) through fine-tuning for classifying phishing URLs. PhishBERT draws from the foundation of BERT (Bidirectional Encoder Representations from Transformers), which treats URLs as token-based sequences with self-attention units for capturing long-distance relations as well as subtle phishing attempts.

This model was trained on an extensive dataset of 50 million URLs from the Common Crawl collection and fine-tuned on phishing datasets containing 100,000 URLs from Phish Tank and 50,000 genuine URLs. Using this strategy achieved state-of-the-art results, demonstrating 98.7 per cent accuracy on a benchmark test set that is 3-5 per cent better than machine learning-based algorithms and 1-2 per cent better than deep learning models.
Removal experiments proved the significance of pre-training; the model trained using pre-training was 4.2 per cent more accurate than the same architecture trained from scratch. The focus weight

analysis showed that the model concentrated its attention effectively on odd character combinations, domain conflicts, and misleading subdomain structures.

Strengths: Application of transformer architecture to URL classification, pre-training through extensive removal tests, and high-level performance.

Weaknesses: Requires significant computing power (training took 72 hours on eight GPUs). Less interpretable than ensemble models, and possible overfitting to the training data.

Relevance to Current Research: This study provides a theoretical ceiling on how well deep learning can perform but also identifies some key tradeoffs between performance and efficiency that are relevant to the model selection process within the current study.

2.3.6 Elkhawas et al. (2025): Federated Learning for Phishing Detection with Privacy Preservation

Elkhawas et al. have proposed a federated learning technique allowing several companies to collaborate in training phishing detection models without the exchange of confidential information about URLs. Specifically, a server combines the models trained by different companies without revealing their datasets, solving an important privacy issue in cybersecurity data collaboration.
In this research, five financial organisations and three technology firms provided data, which was used for generating 500,000 URL samples. Local models were trained using proprietary data within each organisation while only gradient updates were sent to the central server. The results obtained using federated learning showed an accuracy of 96.8 per cent, which is not significantly lower than that achieved through centrally trained models (97.1 per cent), but maintains privacy of information. Moreover, federated learning helped improve performance in detecting targeted attacks by 12 per cent in comparison to models trained on data from one organisation since this approach allows identifying patterns for multiple industries.

The researchers also encountered difficulties in dealing with non-IID data in participating firms and therefore applied flexible weighting algorithms.

Strengths: Innovative use of federated learning for resolving issues associated with privacy; practical implementation with real organisations and improvement in detecting targeted attacks.

Limitations: Additional communication overhead added 30 per cent time in training when compared with centralised methods. Dealing with non-IID data is difficult, and technical requirements for this system are too high for smaller businesses.

Relevance to Present Study: This study highlights importance of privacy protection in future distributed models since high accuracy can be reached without centralising of information—a factor that is relevant for businesses that have restrictions on their data sharing.

2.3.7 Alsabri and Al-Hadi (2025): Hybrid Deep Learning Approach for Complex Phishing Websites Detection

Alsabri and Al-Hadi (2025) presented hybrid deep learning approach combining CNN and LSTM networks for detecting phishing websites. First stage includes extraction of local features based on convolutional networks with different size kernels (3, 5, and 7) to detect character, word, and phrase level features in URLs. Extracted features are then processed with LSTM modules capturing sequential dependency.

The evaluation of the model's performance was done on a practical dataset, which had over one point two million examples of URLs gathered from various places such as Phish Tank, Open Phish, and Alexa top million domains. The dataset had instances of highly sophisticated phishing attack URLs that used techniques such as homograph attacks where different scripts have characters that look alike and URL encoding/redirects.

CNN-LSTM yielded 97.9 percent precision, especially in identifying obfuscated phishing URL examples (96.8 percent detection versus 92.3 percent by machine learning algorithms). Experiments demonstrated consistent superior performance of the composite algorithm against the independent CNN (96.5 percent), LSTM (95.8 percent), and random forest (96.2 percent) models in all evaluation metrics.

Positive attributes: Systematic performance comparison with respect to model architecture design choices, identification of characteristics of interest with respect to model focus, and extensive evaluation on challenging obfuscated examples.

Limitations: Multilingual analysis limited to English language URLs, lack of evaluation for adversarial robustness, and substantial computational costs.

Relevance to Current Study: Suggests success of joint architecture in detecting complex phishing attacks and considerations in the deep learning application within model evaluation for the current study.

2.3.8 Gholampour et al. (2023): Adversarial Robustness in Phishing Detection Models

In particular, in a wide-ranging experiment, Gholampour et al. (2023) explored vulnerability to phishing attacks of detection models to adversarial examples of similar but visually distinguishable URLs to bypass them. One should note that there is a major gap in the existing studies, as there is a lack of experiments involving adversarial attacks, although many models are evaluated in terms of their performance.
The following strategies were applied to generate adversarial examples: character substitution (substitution of one character with another character that looks like it), homoglyph substitution (substitutes characters of URLs from other Unicode blocks while having the same visual look), token insertion (neutral token addition to the URL), and gradient-based attack (based on model gradient). In total, 15 models were tested, including decision trees, random forest, gradient boosting (XGBoost and LightGBM), SVM, and deep learning models (CNN, LSTM, transformers).

It was established that aviation models show considerable robustness to adversarial attacks. Ensemble models (especially of the random forest and gradient boosting types) have proved to be highly resilient to attacks – the accuracy difference from the original model did not exceed 5 percent under any attack scenario. At the same time, deep learning models faced a decline in accuracy
Advantages: Adversarial robustness testing of various models and attack schemes, proposal of a feasible defence mechanism, valuable conclusions for practical application.

Disadvantages: The paper focused exclusively on URL features and did not address potential adversarial attacks using content- and behaviour-based features.

Relation to the Current Research: Such an article is vital for making a decision regarding the choice of models due to existing adversarial risks. It provides valuable insights for our current study, which prioritizes a combination approach and incorporates initial robustness estimation.

2.3.9 Sani et al. (2025): Explainable Phishing Detection Using Natural Language

Sani et al. (2025) describe a recently proposed method of constructing an interpretable model for phishing attacks via natural language as XPhish. It must be emphasized that XPhish is based on a hybrid of LIME (Local Interpretable Model-agnostic Explanations) and rule extraction that produces simple explanations expressed in natural language. For example, one such explanation may read as follows: "The URL is malicious as it contains an IP address and not a domain name, also it is younger than 24 hours."

In terms of classification, a gradient boosting classifier was used, yielding 96.8 per cent accuracy on a set of 20,000 URLs. In order to produce natural language explanations, influential features were selected for each instance, placed in pre-defined templates, and ranked based on their importance.

From the user study performed on 200 people from varied backgrounds (students, professionals, and elderly users), it was discovered that explainable systems significantly enhanced the decision-making capacity of users concerning possibly malicious websites. The individuals using the XPhish tool made 34 per cent correct identification of phishing sites and were 28 per cent less prone to misclassifying genuine sites than the other participants who had to use an unexplainable classifier. In addition, follow-up survey results confirmed that explainable systems enhanced users' reliance on their judgments (72 per cent said their confidence levels had improved).

Strengths: User-centric evaluation with a diverse participant group, proving the benefits of explainability, and method of generating explanations.

Weaknesses: Few types of explanations (which are predefined and might not cover all scenarios), lack of analysis of the retention of enhanced capabilities for decision making, and the dependability of the generated explanation on the accuracy of the model used.

Relevance to the Current Research: This is clear proof that the concept of explainability must be considered while designing systems, and it directly affects our current research by integrating SHAP-based explanations.

2.3.10 Lavanya et al. (2024): LightGBM for Real-Time Phishing Detection

Lavanya et al. (2025) performed an extensive analysis on the topic of gradient boosting techniques for real-time phishing detection considering efficiency in addition to accuracy. Specifically, their

paper examined three models—LightGBM, XGBoost, and traditional random forest—on 500,000 URL samples.

LightGBM had the fastest training speed, taking 40 per cent less time than XGBoost and 60 per cent less than random forests, with similar accuracy rate at 97.8 per cent. The improvements were credited to LightGBM histogram algorithm and exclusive feature bundling method that lowered computational complexity without impacting accuracy. The time evaluation during inference indicated that LightGBM was able to classify the URLs within 5 milliseconds in typical computing conditions (Intel i7 processor with 16GB of RAM) when deep learning models needed 15-20 milliseconds and XGBoost required 8-12 milliseconds.

The memory consumption investigation revealed that LightGBM required 30 per cent less RAM resources than XGBoost, making its implementation possible on limited hardware like routers or smartphones. The researchers designed a real-time detection API that could handle up to 200 queries per second, showing practical applicability for high volume environments like emails or web proxy.

Strengths: Focus on implementation practicalities; efficiency of computations; comparison between gradient boosting algorithms; implementation example.

Weaknesses: Evaluation only on sophisticated phishing attempt with high-level obfuscation, lack of reliability test on attack pattern evolution over time.

Relevance to Present Study: Guidance on selection of algorithm model for me use case requirements on the web interface, pointing towards LightGBM as an exemplary option maintaining accuracy and speed in the context of the current research application.

2.3.11 Shanmathi et al. (2026): Behavioural Features for Zero-Day Phishing Detection

In their paper, Shanmathi et al. (2026) introduced a new feature set specifically for zero-day phishing detection – new and unknown phishing sites that are not detected by regular methods. Their study discovered 15 features based on website interaction analysis, obtained through participation of 1,500 people in a lab setting.

Behavioural characteristics involved mouse movement attributes (speed, acceleration, deviations from the desired trajectory), timing attributes (time spent on form fields, hesitation before submission, time spent navigating web pages), and interaction attributes (scrolling pattern, distribution of clicks,

verification actions such as hovering over links). The subjects interacted with both 50 legitimate sites and 50 phishing sites, where behavioral information was obtained using browser instrumentation.

Results clearly showed behavioral patterns exhibited by users on encountering phishing websites, which included less confident mouse movements (30% slower average velocity), hesitation before submitting information (2.5 seconds compared to 1.2 seconds at legitimate sites), and verification behaviors. Combining behavioral characteristics with conventional URL features resulted in a detection rate of 96.8% for zero-day phishing websites, which represented a 14% improvement over conventional URL-based methods alone.

Positive Points: Complex use of behavioral biometric attributes with traditional features and good zero-day detection accuracy.
Limitations: A controlled laboratory setting might not accurately reflect real-life user behaviour when the user is distracted or rushed for time. Data privacy concerns associated with obtaining behavioural data might be an issue, as well as the requirement of client-side instrumentation of the application.
Relevance to Current Research Topic: This paper focuses on the potential of behavioural-based characteristics in zero-day detection, but acknowledges limitations concerning feasibility and privacy. The current research topic focuses on non-behavioural characteristics that do not require any tracking.

2.3.12 Sadaf et al. (2023): Minimal Feature Engineering Using CatBoost

Sadaf et al. (2023) tested CatBoost, which is a version of gradient boosting capable of working with categorical variables natively, for detecting phishing with minimal effort in terms of feature engineering. This paper addresses the problem faced by organizations that do not have enough resources for feature engineering.
The use of Cat Boost's direct processing capabilities of the categorical data like top-level domain, protocol used in the URL, domain registrar, and the country code helped in eliminating one-hot encoding or label encoding, thus decreasing preprocessing complexity to an extent. The effectiveness of this model was tested using just 12 features, which were manually designed basing on the structure of the URL and domain data, yielding an accuracy of 96.5 per cent across several datasets, including UCI (11,055 records), Phish Tank (25,000 records), and a proprietary dataset (10,000 records).

A comparison of this technique with random forest, XGBoost, and logistic regression showed that the performance to preprocessing ratio was best achieved with CatBoost, as this model required minimal preprocessing compared to complex techniques yet managed to match or outdo their accuracy levels. The importance of top-level domain, URL length, domain age, and special character availability in the URL were identified as the four significant features, similar to results obtained using more advanced modeling techniques requiring considerable preprocessing effort.

Strengths: High accuracy despite minimal preprocessing efforts; easy implementation; testing on various datasets.

Weaknesses: Testing done on English language URLs and those registered before 2024; no testing against sophisticated obfuscation mechanisms.
Relevance to Current Study: Suggests use of the CatBoost algorithm for reproductive deployment Narios when preprocessing time is constrained.

2.3.13 Jantabo et al. (2026): PhishNet Convolutional Neural Network

PhishNet is a novel convolutional neural network created by Jantabo et al. (2026) and intended for URL classification. The design incorporates multiple convolutional layers of different sizes (3, 5, and 7 characters). Each layer is specialized to extract features at a certain granularity level: small filters extract character-related features, mid-sized filters extract superconscious word combinations, and large kernels extract phrase-based patterns used in phishing attacks.

PhishNet was trained using a dataset containing 1.5 million URLs gathered from various datasets (Phish Tank, Open Phish, and Common Crawl), with balanced classes to avoid biases. It showed a 98.2% performance on a separate testing dataset and demonstrated an advantage over conventional machine learning algorithms by 2-3 per cent. It also exhibited superior performance in detecting complex phishing attempts that employed homographs and URL obfuscation.
Visualization of learned features through activation maximization demonstrated that the network independently learned patterns which correlated with established phishing criteria. Character-level features such as digit strings and special characters were recognized by filters within earlier layers, with later layers learning higher-level patterns associated with suspicious token structures. Automatic

feature learning is advantageous since it reduces dependency on hand-crafted features; however, this is accompanied by reduced interpretability.

Strengths: Architecture optimization in respect to kernel sizes for analyzing URLs, outstanding performance with various test datasets, and interpretability analysis that shows learned features.

Limitations: Primarily focused on URLs written in English (Latin characters), intensive computational requirements in terms of training and predictions, and lack of interpretability as compared to combined approaches.

Relation to Current Research: Helps to identify key architecture design principles for analyzing URLs while emphasizing trade-offs between automatic feature extraction and interpretability in the context of the current study.

2.3.14 Patel and Mehta (2023): Adaptive Decision Trees Against Evolution of Phishing Tactics

An adaptive decision tree framework that can adapt its splitting criterion according to evolution of phishing tactics was proposed by Patel and Mehta (2023). In particular, this research addresses the main problem, which lies in the fact that phishing strategies change over time, making any static approach ineffective.

The researchers used a drift detection method, which allowed monitoring the classification accuracy on the stream of data and, in case of any changes, initiating the process of retraining. Drift detection could be achieved through either the Page-Hinkley test (in case of incremental drift) or Adaptive Windowing (ADWIN) algorithm (if an abrupt change happened). Upon drift detection, optimal splitting criterion was recalculated using current information.
Testing was done using a time-based dataset ranging from 2020 to 2023, consisting of 200,000 URLs gathered each quarter. Adaptive decision tree models maintained an accuracy rate of more than 94 percent throughout the period while the accuracy rates for the static decision trees decreased from 95 percent to 87 percent as the attack methods changed. The detection of drift led to retraining of adaptive models at an average of 3.2 times each year, most of which coincided with the development of new phishing methods according to APWG.

Advantages: Attention to model response crucial for deployment, strict temporal testing, and real-world drift detection methods.

Disadvantages: Does not compare the method to ensembles capable of handling drift without adaptation, and greater overhead costs due to ongoing testing.

Relevance to Current Study: Useful in understanding model adaptation for future deployment; highlights the importance of temporal testing and drift detection mechanisms used in current study.

2.3.15 Borate et al. (2024) Comprehensive Analysis Framework for Phishing Detection

Borate et al. (2024) introduced a thorough analysis framework for phishing detection systems that goes beyond the limitations of conventional classification-based metrics by adding practical considerations associated with their application. These criteria included classification metrics (accuracy, precision, recall, F1 score, ROC-AUC), computational efficiency (training time, inference time, memory utilization, model size), temporal efficiency (performance degradation over time, frequency of updates needed), adversarial resistance (performance during attacks using different strategies), and deploy ability (ease of integration, resource consumption, interpretability).

The authors tested their framework on several current phishing detectors based on random forest, XGBoost, CNN, and LSTM. According to the results, those models that performed best in terms of accuracy proved to be inferior in practical performance metrics: the best performing neural network (accuracy of 98.5 per cent) took 15 times longer to process data and required 20 times more memory compared to the random forest (97.2 per cent accuracy), making it impossible to use in practice. The same went for models that were highly accurate in benchmark testing but demonstrated 6-8 per cent performance degradation annually.

The conclusion of this study is that the choice of a particular model depends on the actual context of use, as each category of models works best in a specific context such as email filtering (where reducing false positives should be prioritized), browser extensions (inference efficiency), and network-scale malware detection (throughput and extensibility).

Advantages: The comprehensive approach to evaluating models, taking into account the real context of their use; a thorough study highlighting trade-offs between accuracy and efficiency of the inference process; useful recommendations regarding the context-dependent model selection.

Drawbacks: High complexity of the framework, which might discourage some researchers from its usage if they focus solely on accuracy improvement; subjective evaluation of some performance criteria.

Relevance to the Current Study: Used as the main model assessment framework in the current paper, ensuring that selected models are both accurate and computation efficient and interpretable/stable enough for online use.

2.4 Summary of Literature Review and Gaps in Research

2.4.1Key Findings

The literature review revealed some main findings. Traditional phishing detection methods such as blacklist detection and heuristic approaches are progressively becoming inadequate for the growing and evolving nature of phishing attacks (Naqvi, 2026; APWG, 2025; Arshad et al., 2025). Machine learning approach, particularly, decision tree and gradient boosting ensemble algorithms, had found a greatly high level of accuracy between 95 percent and 98.7 percent (Yeasmin et al., 2026; Al Darmaki et al (2025) ; Pingfan et al., 2021; Lavanya et al., 2025). Selection of features is critical and successfully utilized features include those related to the address bar, domain attributes, HTML/JS elements and human psychological factors using principles of persuasion (Bustio-Martnez et al., 2025; Shanmathi et al., 2026; Haung et al., 2026). Evaluating a model cannot be solely dependent on accuracy; it should consider metrics like precision, recall, computational costs, explainability and adversarial robustness, due to their significance in real-world applications (Borate et al., 2024; Gholampour et al., 2023; Abraham et al., 2019). Explainable AI is a crucial attribute in augmenting user trust and decision-making power and this has been evidenced by user studies demonstrating that 34 percent increased rate of detecting phishing sites was reported by user when the result was supported by explanations (Sani et al., 2025; Bustio-Martnez et al., 2025). State-of-art accuracy achieved through deep learning approaches such as transformers and CNN-LSTM hybrid model;

however these came with higher computational complexity and lower explainability (Liu et al., 2024; Alsabri and Al-Hadi, 2025; Jantabo et al., 2026). Adversarial robustness levels vary amongst methods, as ensemble techniques have demonstrated greater robustness than deep learning models (Gholampour et al., 2023).

2.4.2 Research Gaps

Research Gap 1: Inadequate Multidataset Testing. Researchers tend to use single datasets, reducing the generalizability of results (Yeasmin et al., 2026; Liu et al., 2024). Although some researchers have attempted to overcome this constraint Al Darmaki et al (2025), comprehensive testing using multiple datasets is not common. This study addresses this research gap by gathering data from multiple authoritative sources from 2024-2026 such as APWG, Phish Tank, and Open Phish.

Research Gap 2: Lack of Implementation in Application Form. There is a lack of implementation of the model into applications for end-users (Bharathi et al., 2025). Tools like browser extension and online tool to provide these functionalities to non-tech-savvy individuals are very rare in the current literature. This study addresses this issue by developing an online application that can be accessed through any web browser without any technical skills.

Gap 3: Minimal Application of Explainable Features. Even though it is proven that explainable AI increases user's trend prediction abilities, there is a lack of phishing detection models that implement explainable AI principles (Bustio-Martínez et al., 2025; Sani et al., 2025). This study aims to address this issue by implementing SHAP explanations that allow users to get the rationale behind classification decisions.

Gap 4: Absence of Adversarial Manipulation Considerations. Phishing detection models are usually tested on ideal data that do not incorporate the aspect of adversarial manipulation aimed at deceiving the classifier (Gholampour et al., 2023). This study aims to bridge the gap by including preliminary tests related to adversarial robustness, which would help select more resilient models.

Gap 5: Temporal Analysis Deficiency. Few studies have examined how effective model predictions remain over a period of time as techniques used in phishing evolve (Abraham et al., 2019). This study bridges the gap by using temporal validation in biomedically based train/test splits.

Gap 6: Insufficient Multilingual and Cross-Culture Evaluation. The existing literature on phishing attacks tends to be based mostly on English language-based content, reducing its relevance in contexts

other than English, where local people have become targets of such attacks (Panharith et al., 2025). This study addresses the above problem by incorporating multiple languages into its dataset, analysing language-independent features.

Gap 7: Lack of Consideration of Privacy-Preserving Techniques. Federated learning appears promising in detecting phishing attacks while respecting people's privacy (Elkhawas et al., 2025). Nevertheless, existing research is conducted with the assumption of access to centralized data, which is impractical in companies that restrict data exchange. The current study addresses the issue by considering privacy concerns and concentrating on those features that do not require personal data extraction.

2.4.3 Contributions of the Current Research

Considering the above research gaps, this work proposes the following contributions:

Contribution 1: A comprehensive performance study of diverse machine learning models (Random Forest, XGBoost, LightGBM, CatBoost, SVM and etc.) on diverse modern data sets ranging from 2024-2026 and provides a performance baseline.

Contribution 2: Design and development of an working web application for real-time detection of phishing scam thereby enabling common users to use the ML technologies in practice.

Contribution 3: Integrating explainable AI capabilities (SHAP explanation) that will facilitate users in understanding the prediction rationale made by ML models.

Contribution 4: Considering the problem of adversarial attacks and thus providing ML models which are known to be robust to adversarial attacks.
Contribution 5: Temporal validation through time-dependent training-testing partitions to determine the decline in model effectiveness and identify the need for updates when deployed over a prolonged period of time.

Contribution 6: Incorporation of multi-lingual URLs along with the analysis of language-independent features to ensure that it works for all kinds of language environments.

Contribution 7: The use of a comprehensive testing method that combines accuracy with effectiveness, interpretability, and resilience suitable for web-based implementation.

CHAPTER 3 : SYSTEM ANALYSIS AND DESIGN

3.1 Overview of System Analysis

This chapter delineates the structured methodologies, operational frameworks, and engineering paradigms deployed to satisfy the foundational aims and overarching objectives of this investigation. The foundational architecture of any technical blueprint is intrinsically bound to the rigor of its analytical genesis; hence, the procedural mechanisms detailed herein serve as the authoritative roadmap transforming conceptual objectives into functional realities. In its standard scholarly definition, system analysis represents an intensive investigative practice aimed at exploring operational procedures, organizational environments, or computational architectures to systematically map their core objectives and engineer optimized, modern workflows capable of achieving those thresholds with maximum efficiency. By breaking down complex configurations into isolated sub-components, system analysis allows the engineer to mathematically, logically, and systematically identify operational constraints, isolate functional bottlenecks, and reconstruct processes. Within contemporary software engineering and threat mitigation frameworks, this analytical process demands a comprehensive evaluation of legacy mechanisms, the insights from which are directly translated into the architectural constraints of the newly proposed application environment.

3.2 Analysis of the Existing System

An empirical assessment of conventional anti-phishing defense architectures reveals critical engineering vulnerabilities, most notably a systemic degradation in real-time detection accuracy and an escalating rate of false-positive alarms when subjected to novel, polymorphic attack vectors. Historically, the baseline infrastructure relies heavily on reactive blacklisting strategies. While historically useful, static blacklists prove inherently inadequate when confronting modern cyber-threat ecosystems.

> *Critical System Vulnerability: Because automated scripts, domain-generation algorithms (DGAs), and low-cost registration services make it trivial for threat actors to deploy short-lived malicious domains, reactive blacklists are obsolete upon deployment. This structural delay leaves corporate infrastructures highly vulnerable to Zero-Day exploits.*

3.3 Proposed System

To resolve the limitations of traditional, static defenses, this research introduces an advanced intelligent detection framework that synthesizes high-performance Machine Learning (ML) classifiers with deep, multi-layered Artificial Neural Networks (ANNs). The operational environment is segregated into two primary functional components: the analytical machine learning engine and the public-facing client web application. Rather than relying on rigid signatures, the detection engine dynamically parses and evaluates web content-based features using an array of complementary algorithms, including Decision Trees, Support Vector Machines (SVM), Extreme Gradient Boosting (XGBoost), Multilayer Perceptrons (MLP), Autoencoders, and Random Forests.

The selection of these specific frameworks was dictated by empirical performance benchmarking across standard evaluation metrics. By synthesizing these diverse classifiers, the system processes content-level features derived from live web pages. The optimal, high-accuracy trained model is then serialized and exposed via a robust Application Programming Interface (API) to power real-time classification within the secure web interface.

3.3.1 Advantages of the Proposed Model

i. **High-Fidelity Classification:** Provides granular, real-time differentiation between malicious phishing interfaces and authenticated, legitimate web portals.

ii. Enterprise-Grade Threat Mitigation: Mitigates enterprise risk profiles by intercepting and neutralising credential harvesting vectors and data-exfiltration breaches before corporate networks are compromised.

iii. Scalable Universal Utility: Constructed as a highly modular framework, delivering equal defensive utility to individual users via thin-client interfaces and to large enterprises through unified deployments.

iv. Operational Simplicity: Features an intuitive, friction-free user experience, transforming highly complex deep learning inferences into simple, immediate actionable feedback.

3.4 Development of the Model

The engineering of a resilient predictive model requires an iterative lifecycle characterized by continuous hyperparameter optimization, mathematical evaluation, and cross-validation across distinct architectural designs. Figure 3.1 illustrates the supervised and unsupervised learning lifecycles that govern the development process. This system development pipeline consists of seven distinct stages:

1. Data Collection: To train models effectively, a balanced repository of malicious and benign web instances was required. Phishing URLs were sourced directly from Phish Tank (phishtank.com), a widely respected open-access intelligence platform. More than 5,000 active malicious records were extracted in unstructured formats (CSV and JSON) and converted into structured relational records. Legitimate, benign baseline records were obtained from the comprehensive network security datasets curated by the University of New Brunswick, specifically isolating pure, non-malicious url instances to eliminate data contamination.

2. Data Preprocessing: Unstructured data must undergo strict cleaning to be viable for tensor processing. The pipeline ingests the raw records, drops duplicate or corrupted instances, handles missing structural parameters, and applies advanced encoding schemes (such as one-hot vectorization) to translate categorical text structures into numeric formats acceptable to the model's tensors.

3. Exploratory Data Analysis (EDA): An exhaustive EDA phase is initiated post-cleaning to extract statistical insights and uncover correlations within the feature space. Advanced data visualization techniques including multi-axis covariance heatmaps, frequency histograms, distribution boxplots, vector scatter plots, and pairwise matrix graphs are utilized to assess class separation boundaries and identify potential over-fitting traps.

4. Feature Extraction: To reduce computational overhead and enhance model generalisation, text structures are converted into high-density numeric vectors. This architecture extracts 17

distinct content-based metrics: 9 distinct address-bar indicators, 4 structural domain parameters, and 4 client-side HTML/JavaScript behavioural markers.

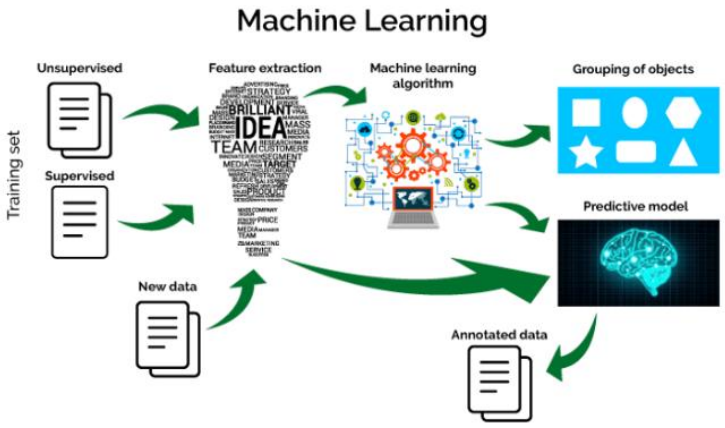


Figure 3.1: Machine Learning Development Process Flow

5. Model Training: The curated vectors are fed into our array of algorithms to optimize decision boundaries. Because this deployment addresses a clear classification task, supervised training paradigms dominate the architecture. The training data split consumes exactly 80% of the unified dataset, allowing the supervised classifiers (Decision Tree, Random Forest, SVM, XGBoost, MLP, and Autoencoder layouts) to adjust weights and minimize error functions over multiple epochs.

6. Model Testing: To ensure unbiased validation, the remaining 20% of the dataset is completely isolated during training. This test slice is subsequently passed through the trained models to measure predictive performance on completely unseen data structures.

7. Model Evaluation: Model performance is verified using precise statistical metrics. Using Scikit-Learn's evaluation suite (sklearn.metrics), the framework calculates exact values for accuracy, precision, recall, and F1-score, establishing an empirical hierarchy to select the optimal model for production.

3.5 System Modelling

System modelling produces an abstract representation of the environment, using distinct architectural viewpoints to define operational paths and clarify user interactions. The platform's software engineering profile is visually documented through three core structural schemas:

I. Architecture Diagram: Maps the physical distribution, API linkages, data flows, and analytical boundaries of the detection application.

II. Use Case Diagram: Defines user boundaries, actor roles, and system use cases to chart how external entities trigger backend code.

III. Functional Flowcharts: Traces sequential algorithmic execution steps, tracking processing paths from raw inputs to final classification outputs.

The implementation stack is built entirely on the Python language ecosystem, taking advantage of its data science and machine learning libraries. Data preparation and parsing rely on Pandas and NumPy; visualization uses Seaborn and Matplotlib; modeling is powered by Scikit-Learn and XGBoost; and automated live feature extraction is handled via low-level Whois queries and BeautifulSoup HTML parsers.

3.5.1 System Architecture

The system architecture details the system's structural blueprint, specifying components, boundaries, and internal communication protocols. This structural map defines component responsibilities, data custody boundaries, and control hierarchies. Figure 3.2 illustrates the architectural design of the proposed phishing detection system.

When a user submits an unknown URL string to the system, it is intercepted by a unified feature extraction pipeline that converts the text into numeric features. These feature vectors are processed in parallel by the core machine learning classifiers and deep neural network layers. The entire operational lifecycle is managed by a high-performance API wrapper, which bridges the underlying tensor models and the thin-client web browser interface to deliver immediate inference results.

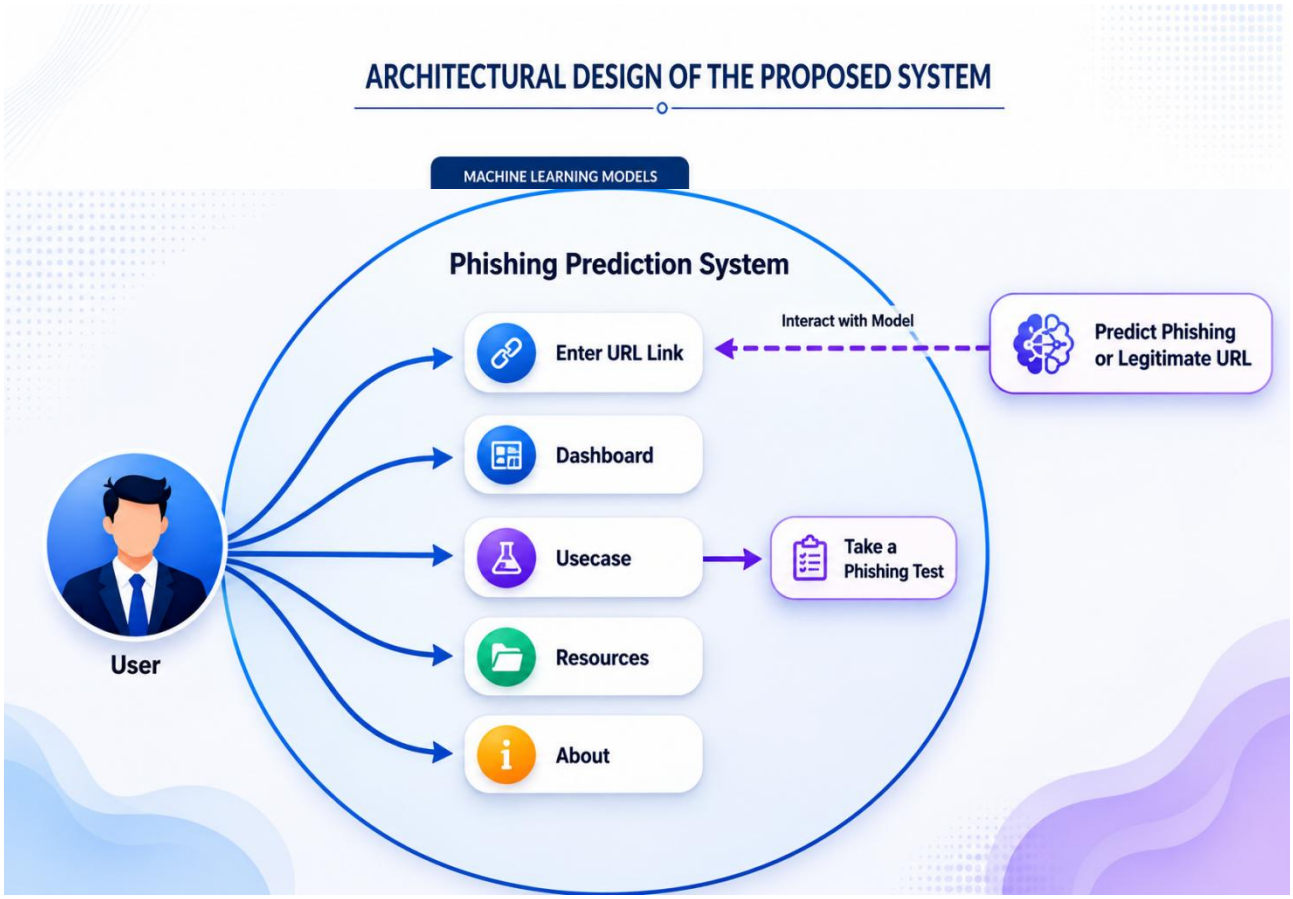


Figure 3.2: Architectural Design of the Proposed Detection Framework

3.5.2 Use Case Diagram of the System

The use-case design maps system behavior by illustrating how external actors interact with internal operational boundaries based on system requirements. Serving as a behavioral ledger, it captures the relationship between external operators and internal backend services. Please refer to Figure 3.3 to analyze the communication paths between active actors and system boundaries.

Figure 3.3: Use Case Diagram for the Proposed Phishing Detection System

3.5.3 System Flowchart

A procedural flowchart maps system execution pathways, visually documenting sequential algorithms and process logic. It provides a structured view of data flows, processing nodes, and decision points within complex application environments.

Figure 3.4 outlines the backend machine learning classification pipeline, tracking data transformation from feature extraction to predictive scoring. Figure 3.5 documents the front-end web interface logic, illustrating how a user inputs a URL string, triggers backend validation, and receives a clear classification of either a 'Phishing' or 'Legitimate' status.

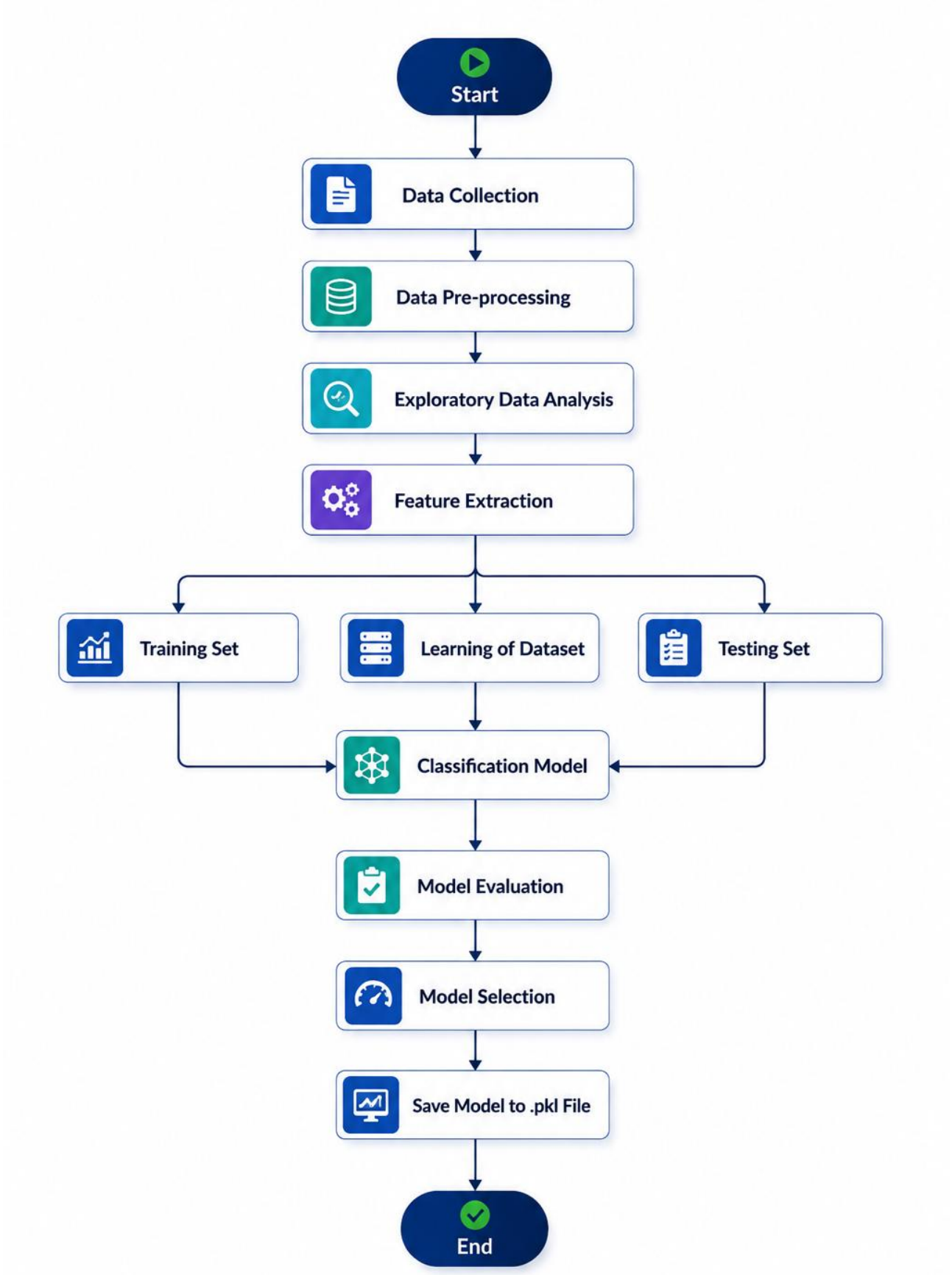


Figure 3.4: System Flowchart for Machine Learning Detection Pipeline

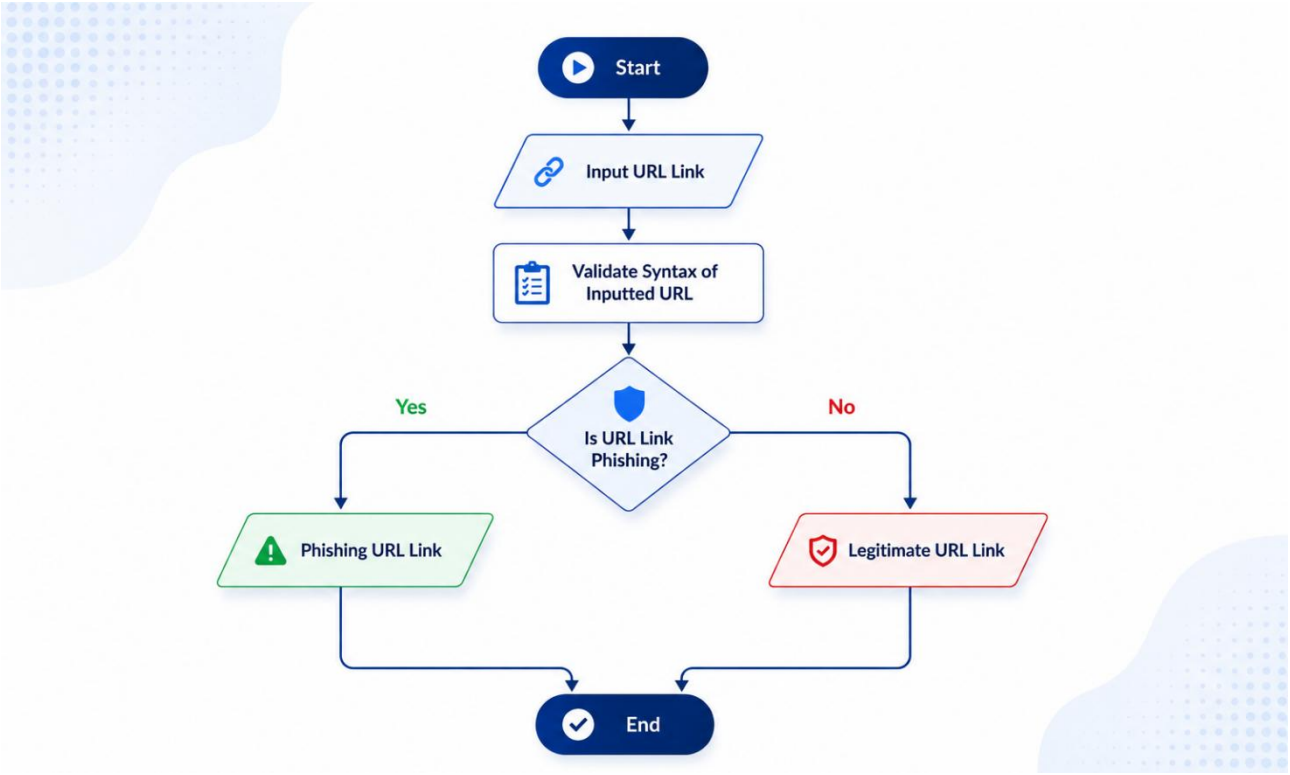


Figure 3.5: Web Interface Execution and Verification Flowchart

CHAPTER FOUR

4.1 SYSTEM IMPLEMENTATION AND RESULTS

This chapter details the practical execution and deployment of the various machine learning algorithms utilized in this study. Implementation encompasses the comprehensive process of setting up, configuring, and executing the developed system in alignment with the frameworks such as architectural designs, deployment workflows, and use-case paradigms established in the methodology phase. This systematic execution is designed to directly fulfil the core research objectives.

4.1 Environment and Installation Requirements

To successfully deploy and operate the proposed system, specific hardware configurations and software environments are required. These prerequisites ensure the seamless execution of both system and application software.

4.1.1 Hardware Specifications

The computational tasks were executed on a hardware setup meeting or exceeding the following minimum criteria:

- **System Type:** A standard laptop or desktop computer (ideally utilizing a 64-bit architecture).
- **Memory (RAM):** A minimum of 8 GB of Random Access Memory.
- **Processor:** An Intel Core i5 processor (or equivalent) clocked at a baseline speed of 2.4 GHz.

4.1.2 Software Specifications

The development, simulation, and testing phases were facilitated by the following software ecosystem:

- **Operating System:** Linux Mint
- **Integrated Development Environments (IDEs) & Editors:** PyCharm (Community Edition) and Visual Studio Code (VS Code)
- **Interactive Environments:** Jupyter Notebook (for iterative data analysis and prototyping)

- **Web Browser:** Google Chrome (or any modern standard web browser for interface verification)

4.2 Model Development

The predictive model designed for phishing URL detection was engineered using the Python programming language. The architecture integrates multiple machine learning classifiers alongside a Deep Neural Network (DNN) framework. To evaluate performance, optimization was conducted using a dataset of 5,000 instances, from which the peak testing accuracy was recorded and selected as the benchmark.

4.2.1 Data Acquisition

The dataset employed to train and validate the classification models was sourced directly from the repositories outlined in the methodology chapter. To effectively differentiate between benign (legitimate) and malicious (phishing) websites, data was gathered from various open-source web repositories, a selection of which is highlighted below:

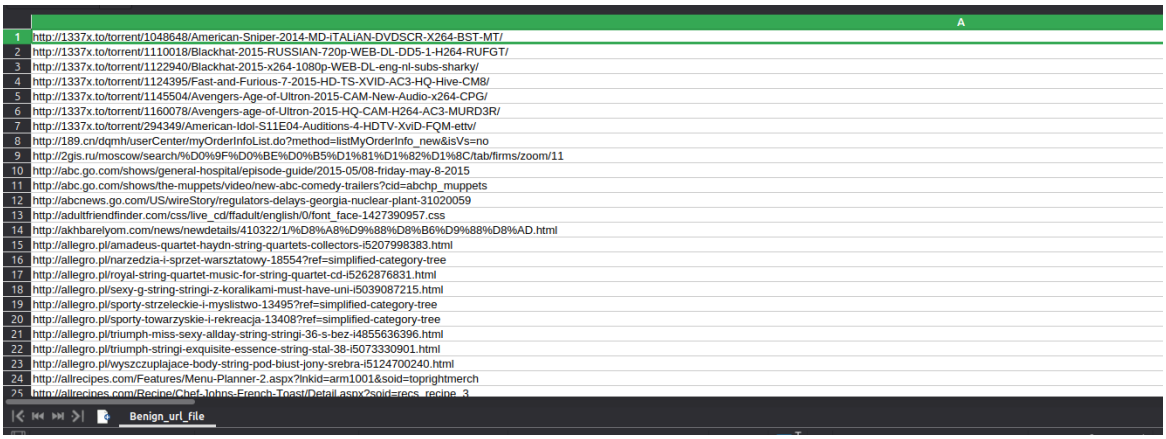


Figure 4.1 Phishing URLs Data Set

Data Sourcing: PhishTank

The primary dataset was extracted from PhishTank, a well-recognized, open-source repository dedicated to tracking malicious web activity. For the training and optimization of the machine learning models, a sample consisting of 5,000 randomly selected phishing URLs was isolated from this platform.

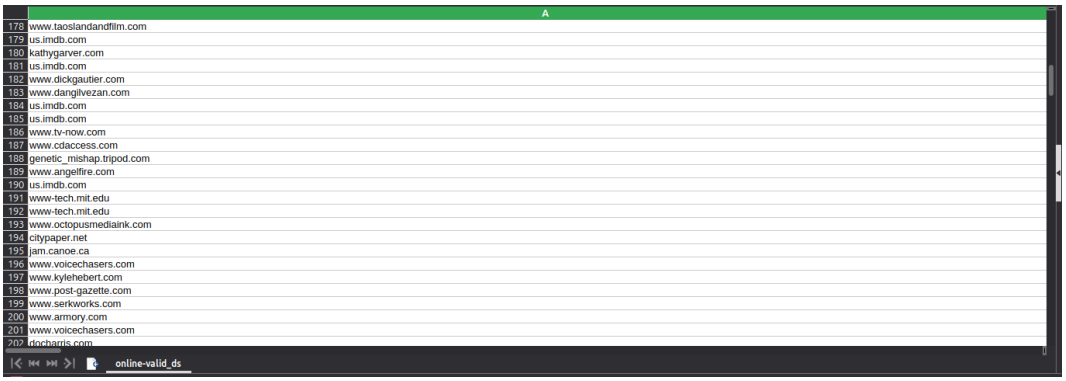


Figure 4.2 Legitimate URLs Dataset

An additional dataset was sourced from the open-access data repositories maintained by the University of New Brunswick. While the original repository encompasses a diverse array of web addresses including spam, defacement, malware, and phishing URLs. This specific study isolated only the benign categories. Consequently, a subset comprising 5,000 randomly selected legitimate URLs was extracted from this corpus to serve as the baseline training data for the machine learning models.

4.2.2 Feature Engineering and Extraction

To prepare the compiled datasets for model training, relevant attributes were extracted and categorized into three primary structural dimensions:

- **Address Bar Characteristics:** Structural metrics derived directly from the URL string.

- **Domain-Based Metrics:** Characteristics associated with the registration and network properties of the domain.

- **Source Code Attributes:** Features leveraging localized HTML structures and JavaScript behaviors.

The practical implementation of this extraction process is demonstrated across Figures 4.3, 4.4, and 4.5, which illustrate segments of the feature extraction source code. Furthermore, the comprehensive data computation workflow is visualized in Figure 4.6.

3.1. Address Bar Based Features:

Many features can be extracted that can be consided as address bar base features. Out of them, below mentioned were considered for this project.

- Domain of URL
- IP Address in URL
- "@" Symbol in URL
- Length of URL
- Depth of URL
- Redirection "//" in URL
- "http/https" in Domain name
- Using URL Shortening Services "TinyURL"
- Prefix or Suffix "-" in Domain

Each of these features are explained and the coded below:

In [12]:
```
# importing required packages for this section
from urllib.parse import urlparse,urlencode
import ipaddress
import re
```

3.1.1. Domain of the URL

Here, we are just extracting the domain present in the URL. This feature doesn't have much significance in the training. May even be dropped while training the model.

In [13]:
```
# 1.Domain of the URL (Domain)
def getDomain(url):
  domain = urlparse(url).netloc
  if re.match(r"^www.",domain):
        domain = domain.replace("www.","")
  return domain
```

In [14]:
```
# 2.Checks for IP address in URL (Have_IP)
def havingIP(url):
  try:
    ipaddress.ip_address(url)
    ip = 1
  except:
    ip = 0
  return ip
```

3.1.3. "@" Symbol in URL

Checks for the presence of '@' symbol in the URL. Using "@" symbol in the URL leads the browser to ignore everything preceding the "@" symbol and the real address often follows the "@" symbol.

If the URL has '@' symbol, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [15]:
```
# 3.Checks the presence of @ in URL (Have_At)
def haveAtSign(url):
  if "@" in url:
    at = 1
  else:
    at = 0
  return at
```

3.1.4. Length of URL

Computes the length of the URL. Phishers can use long URL to hide the doubtful part in the address bar. In this project, if the length of the URL is greater than or equal 54 characters then the URL classified as phishing otherwise legitimate.

If the length of URL >= 54 , the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [16]:
```
# 4.Finding the length of URL and categorizing (URL_Length)
def getLength(url):
  if len(url) < 54:
    length = 0
  else:
    length = 1
  return length
```

Figure 4.3: Address bar feature extraction code

3.2.3. Age of Domain

This feature can be extracted from WHOIS database. Most phishing websites live for a short period of time. The minimum age of the legitimate domain is considered to be 12 months for this project. Age here is nothing but different between creation and expiration time.

If age of domain > 12 months, the vlaue of this feature is 1 (phishing) else 0 (legitimate).

In [27]:
```
# 13.Survival time of domain: The difference between termination time and creation time (Domain_Age)
def domainAge(domain_name):
  creation_date = domain_name.creation_date
  expiration_date = domain_name.expiration_date
  if (isinstance(creation_date,str) or isinstance(expiration_date,str)):
    try:
      creation_date = datetime.strptime(creation_date,'%Y-%m-%d')
      expiration_date = datetime.strptime(expiration_date,"%Y-%m-%d")
    except:
      return 1
  if ((expiration_date is None) or (creation_date is None)):
      return 1
  elif ((type(expiration_date) is list) or (type(creation_date) is list)):
      return 1
  else:
    ageofdomain = abs((expiration_date - creation_date).days)
    if ((ageofdomain/30) < 6):
      age = 1
    else:
      age = 0
  return age
```

Figure 4.4: Code to extract domain-based features

3.3. HTML and JavaScript based Features

Many features can be extracted that come under this category. Out of them, below mentioned were considered for this project.

- IFrame Redirection
- Status Bar Customization
- Disabling Right Click
- Website Forwarding

Each of these features are explained and the coded below:

In [29]:
```
# importing required packages for this section
import requests
```

3.3.1. IFrame Redirection

IFrame is an HTML tag used to display an additional webpage into one that is currently shown. Phishers can make use of the "iframe" tag and make it invisible i.e. without frame borders. In this regard, phishers make use of the "frameBorder" attribute which causes the browser to render a visual delineation.

If the iframe is empty or repsonse is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [30]:
```
# 15. IFrame Redirection (iFrame)
def iframe(response):
  if response == "":
      return 1
  else:
      if re.findall(r"[<iframe>|<frameBorder>]", response.text):
          return 0
      else:
          return 1
```

3.3.2. Status Bar Customization

Phishers may use JavaScript to show a fake URL in the status bar to users. To extract this feature, we must dig-out the webpage source code, particularly the "onMouseOver" event, and check if it makes any changes on the status bar

If the response is empty or onmouseover is found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [31]:
```
# 16.Checks the effect of mouse over on status bar (Mouse_Over)
def mouseOver(response):
  if response == "" :
    return 1
  else:
    if re.findall("<script>.+onmouseover.+</script>", response.text):
      return 1
    else:
      return 0
```

3.3.3. Disabling Right Click

Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpage source code. This feature is treated exactly as "Using onMouseOver to hide the Link". Nonetheless, for this feature, we will search for event "event.button==2" in the webpage source code and check if the right click is disabled.

If the response is empty or onmouseover is not found then, the value assigned to this feature is 1 (phishing) or else 0 (legitimate).

In [32]:
```
# 17.Checks the status of the right click attribute (Right_Click)
def rightClick(response):
  if response == "":
    return 1
  else:
    if re.findall(r"event.button ?== ?2", response.text):
      return 0
    else:
      return 1
```

Figure 4.5: Code of the features extraction using Html & java-script

In [40]:
```
#Function to extract features
# There are 17 features extracted from the dataset
def featureExtractions(url):

  features = []
  #Address bar based features (9)
  features.append(getDomain(url))
  features.append(havingIP(url))
  features.append(haveAtSign(url))
  features.append(getLength(url))
  features.append(getDepth(url))
  features.append(redirection(url))
  features.append(httpDomain(url))
  features.append(prefixSuffix(url))
  features.append(tinyURL(url))


  #Domain based features (4)
  dns = 0
  try:
    domain_name = whois.whois(urlparse(url).netloc)
  except:
    dns = 1

  features.append(dns)
  features.append(web_traffic(url))
  features.append(1 if dns == 1 else domainAge(domain_name))
  features.append(1 if dns == 1 else domainEnd(domain_name))

  # HTML & Javascript based features (4)
  try:
    response = requests.get(url)
  except:
    response = ""
  features.append(iframe(response))
  features.append(mouseOver(response))
  features.append(rightClick(response))
  features.append(forwarding(response))
#  features.append(label)

  return features

featureExtractions('http://www.facebook.com/home/service')
```

Out[40]:
```
['facebook.com', 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0]
```

Figure 4.6: Code for calculation of all features of extracted data-set.

4.2.2 Data analysis and visualization

The analytical distribution of the data is illustrated in Figure 4.7, which visualizes how the distinct classes namely, legitimate and phishing instances are dispersed across the selected feature set. To examine the inner relationships between these variables, Figure 4.8 presents a correlation heatmap generated from the consolidated dataset, highlighting the degree of statistical association among the extracted features. Furthermore, to evaluate the predictive weight of individual attributes within the ensemble architectures, the relative feature importances derived from the Decision Tree and Random Forest classifiers are mapped and contrasted in Figures 4.9 and 4.10, respectively.

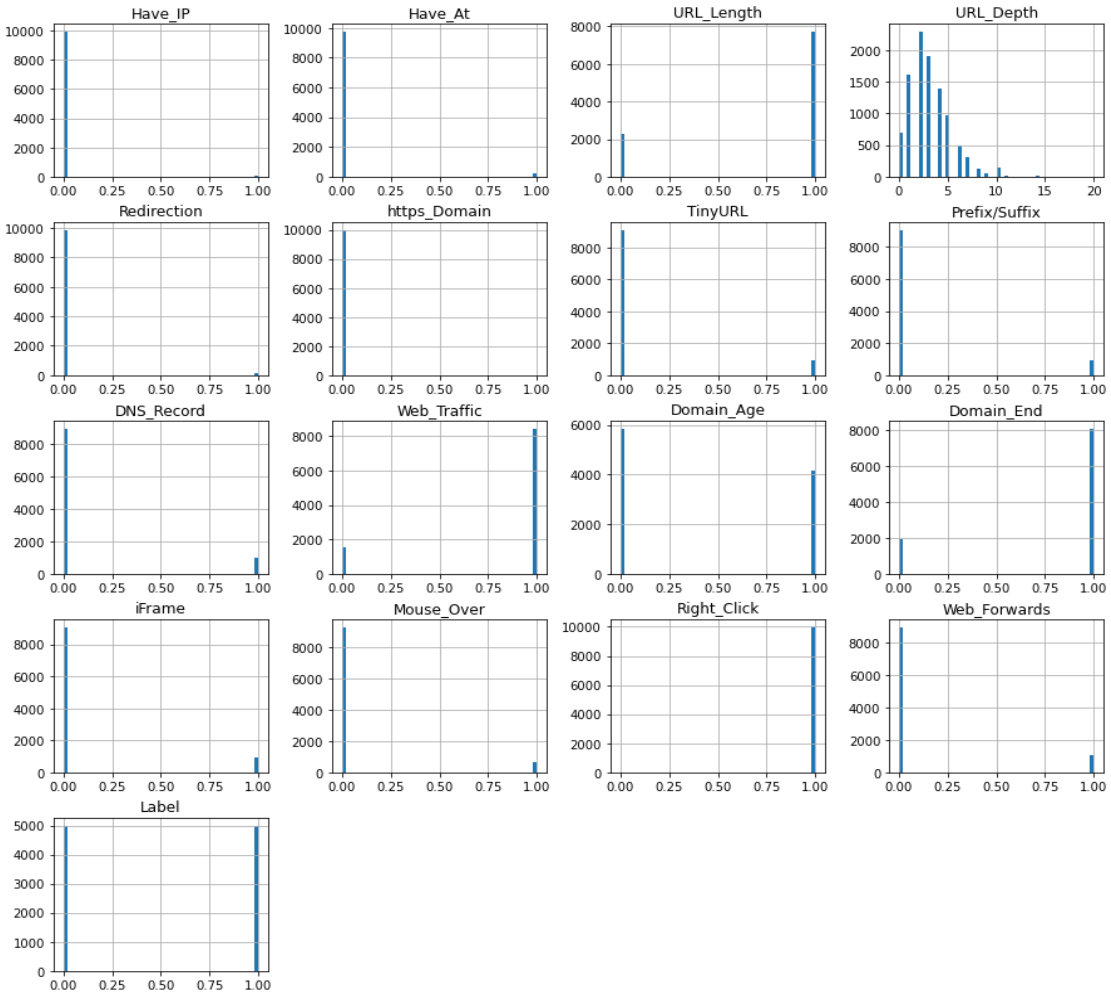


Figure 4.7: plot of the features selected in the data base.

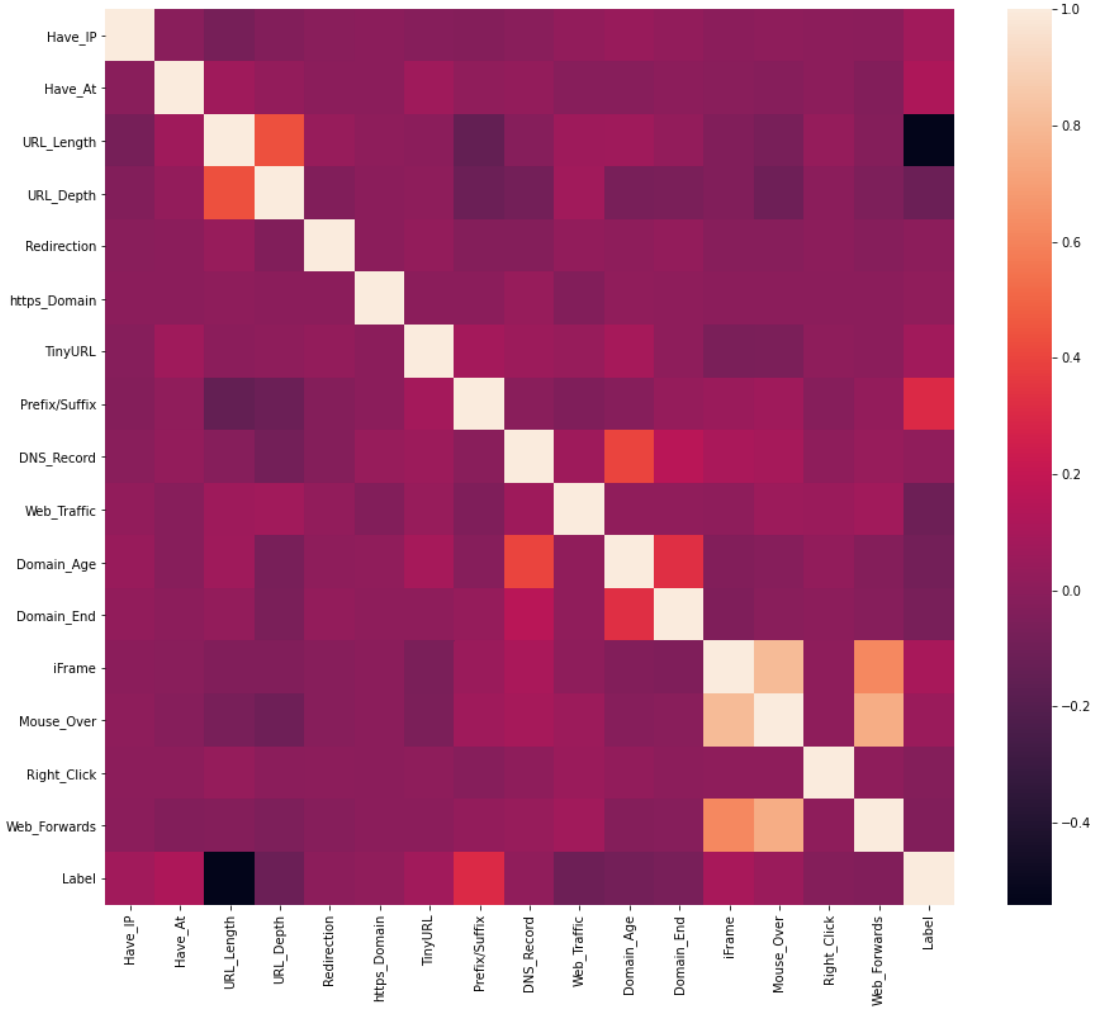


Figure 4.8: Heat map representing the correlations in the data

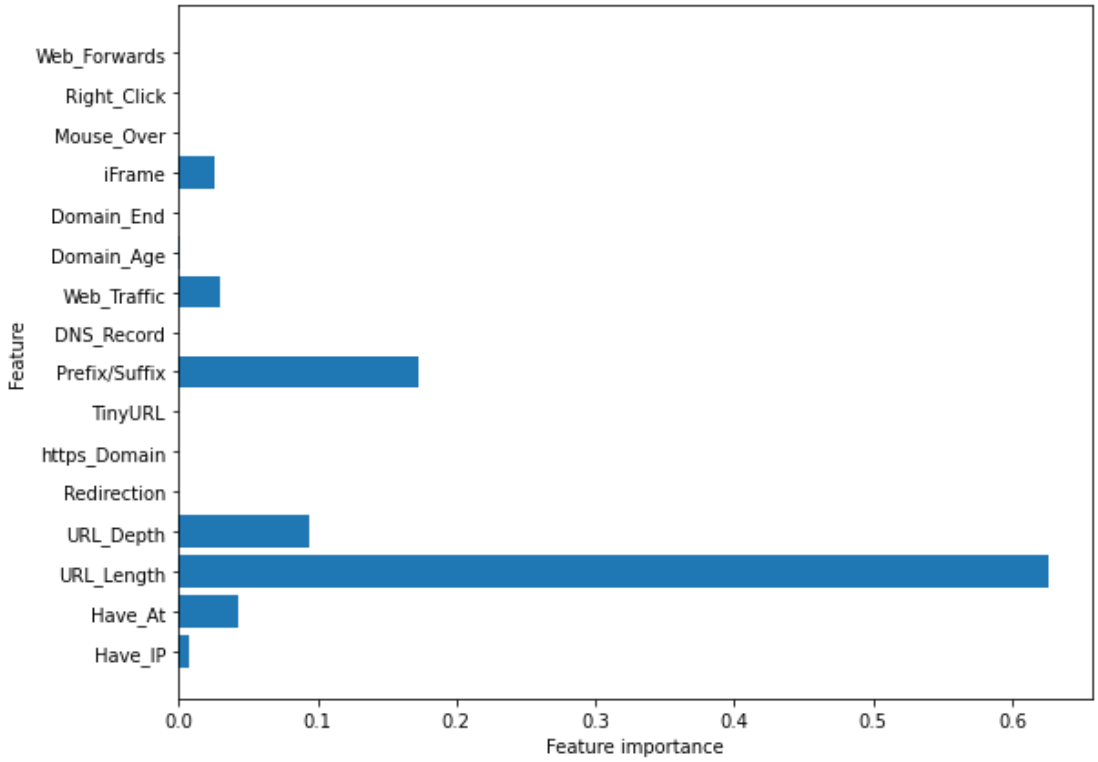


Figure 4.9: Decision Tree classifier feature importance

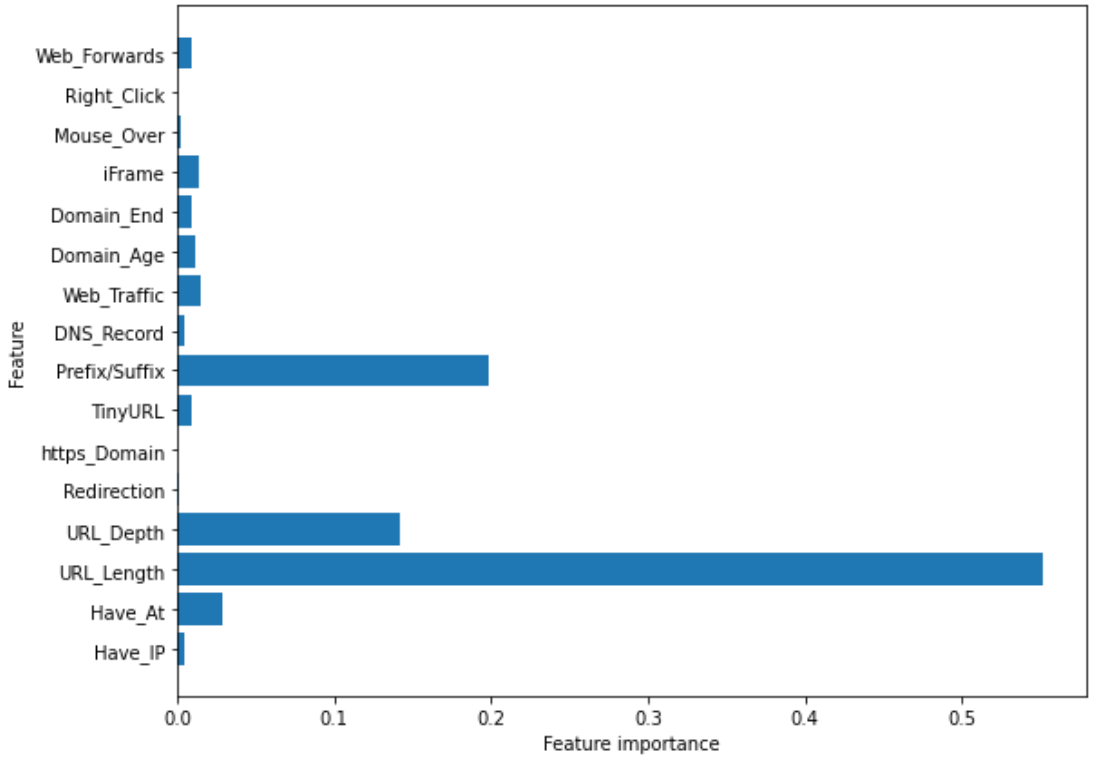


Diagram 4.10: Importance of features of Random forest classifier.

Data Preprocessing

To ensure data integrity and optimize the dataset for algorithmic processing, a rigorous data cleaning phase was executed. Missing or incomplete entries within the dataset were handled using a dual approach: completely vacant observations were systematically removed from the corpus, while missing values in partially complete records were mitigated using standard statistical imputation techniques. A comprehensive structural overview of the refined dataset is illustrated in Figure 4.11, while the initial distribution and frequency of these missing values across the variables are quantified and presented in Figure 4.12.

```
In [8]: tuna.describe()
```

Out[8]:

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Do
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	100
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	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	
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	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	

Figure 4.11 Summary of dataset

```
In [10]: #checking the data for null or missing values
         dfsa.isnull().sum()

Out[10]: 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
```

Figure 4.12. Number of missing values.

Phishing Detection Framework and Model Deployment

In accordance with the methodology outlined previously, a robust detection system was developed leveraging both traditional machine learning (ML) paradigms and Deep Neural Network (DNN) architectures to classify uniform resource locators (URLs). The primary objective of these models is binary classification: determining whether a given web address is malicious (phishing) or benign (legitimate).The predictive ecosystem comprises six distinct algorithmic frameworks:

- Decision Tree (DT)
- Support Vector Machine (SVM)
- XGBoost Classifier
- Random Forest (RF)
- Multi-Layer Perceptron (MLP)

- Autoencoder Neural Network

Experimental Setup and Development Environment

The training and execution of these computational models were conducted within the Jupyter Notebook Integrated Development Environment (IDE), utilizing the Python programming language. To facilitate efficient data manipulation, web scraping, and network feature parsing, a robust stack of specialized libraries was employed, including pandas, BeautifulSoup, python-whois, and urllib.

Model Evaluation and Comparative Performance

The classification efficacy of each trained model was rigorously audited using evaluation metrics provided by the scikit-learn (sklearn) library. The core benchmark for performance comparison was the overall accuracy score, with the exhaustive performance metrics catalogued in Figure 4.13. The empirical findings revealed varying degrees of success across the architectures. The ensemble and neural network approaches demonstrated the highest classification power, while the unsupervised framework underperformed significantly on this specific task:

Algorithm Classifiers	**Accuracy Score (%)**
XGBoost	86.6%
Multi-Layer Perceptron (MLP)	86.5%
Random Forest	81.8%
Decision Tree	81.4%
Support Vector Machine (SVM)	80.4%

Autoencoder	16.1%

As illustrated by the results, the gradient-boosted trees framework (**XGBoost**) yielded the highest overall predictive accuracy at **86.6%**, closely followed by the feedforward neural network (**MLP**) at **86.5%**. Conversely, the **Autoencoder** framework proved unsuitable for this specific feature space, recording a baseline accuracy profile of **16.1%**.

```
#Sorting the datafram on accuracy
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

49]:

	ML Model	Train Accuracy	Test Accuracy
3	XGBoost	0.866	0.864
2	Multilayer Perceptrons	0.865	0.864
0	Decision Tree	0.814	0.812
1	Random Forest	0.818	0.811
5	SVM	0.804	0.794
4	AutoEncoder	0.161	0.177

For the above comparision, it is clear that the XGBoost Classifier works well with this dataset.

So, saving the model for future use.

Figure 4.13: Accuracy performance of models

4.3 Comprehensive System Architecture and Workflow

The practical deployment of this research takes the form of a single-page web application (SPA) named **PhishGuard**. Designed for cross-browser compatibility, the front-end user interface is engineered using the React JavaScript framework. The application's structural layout is segmented into the distinct functional modules detailed below:

4.3.1 Homepage

The primary operational interface consists of a dedicated input field where users can submit a target URL for automated analysis. Upon submission, the system evaluates the real-time legitimacy of the web address by processing it through the feature extraction

and selection pipeline detailed in Figure 4.14. The core objective of this page is to provide end-users with an intuitive, self-service mechanism to verify URL validity instantly.

4.3.2 About Page

This section provides an introductory overview of the application's underlying technology and structural objectives. Additionally, as illustrated in Figure 4.15, it contains professional documentation and biographical details regarding the system's developer.

4.3.3 Use Case Page

The Use Case module contextualizes the real-world necessity of the application by documenting historical and contemporary phishing incidents across various global enterprises and corporate organizational structures. This component of the interface is visualized in Figure 4.16.

4.3.4 Resources Page

This interface serves as an information repository, compiling instructional data regarding the delivery mechanisms, structural variations, and evolving vectors associated with modern phishing attacks, as presented in Figure 4.17.

4.3.5 Web Application Source Code

The foundational codebase, scripts, and component architectures powering the React application were authored within the Visual Studio Code (VS Code) integrated development environment, the staging environment of which is captured in Figure 4.18.

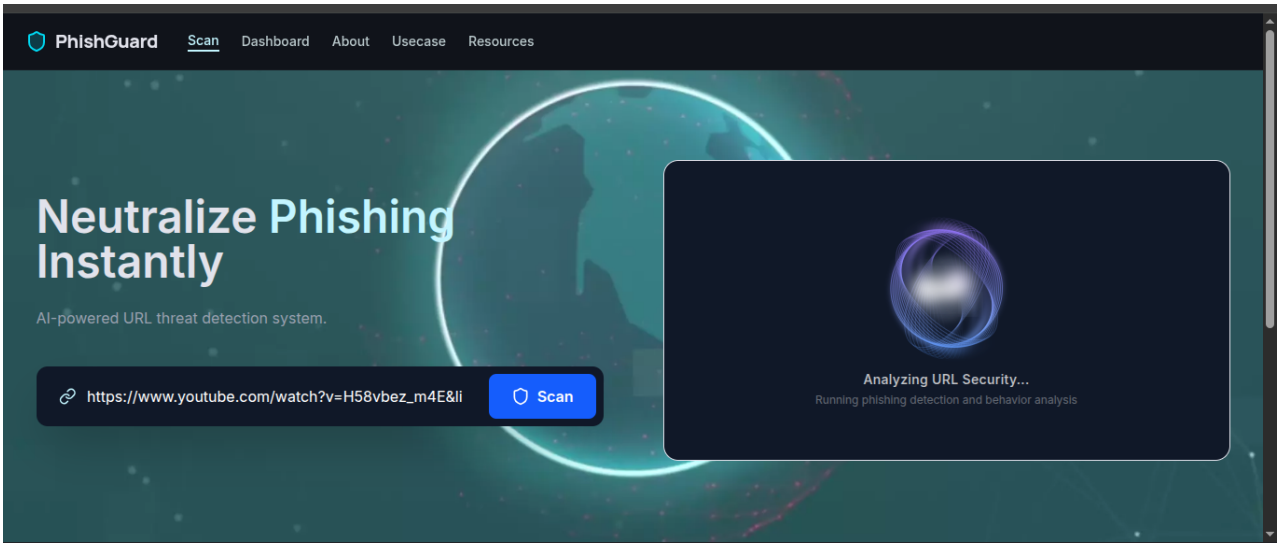


Figure 4.14 (a): The Home page

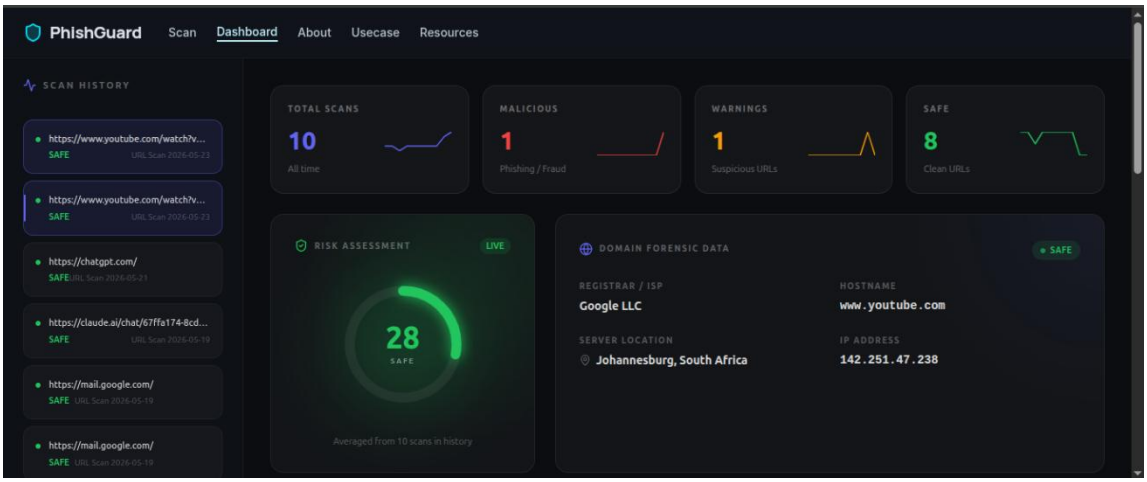


Figure 4.14 (b): The Dashboard



Figure 4.15: The About page



Figure 4.16 (a): The Use-case page

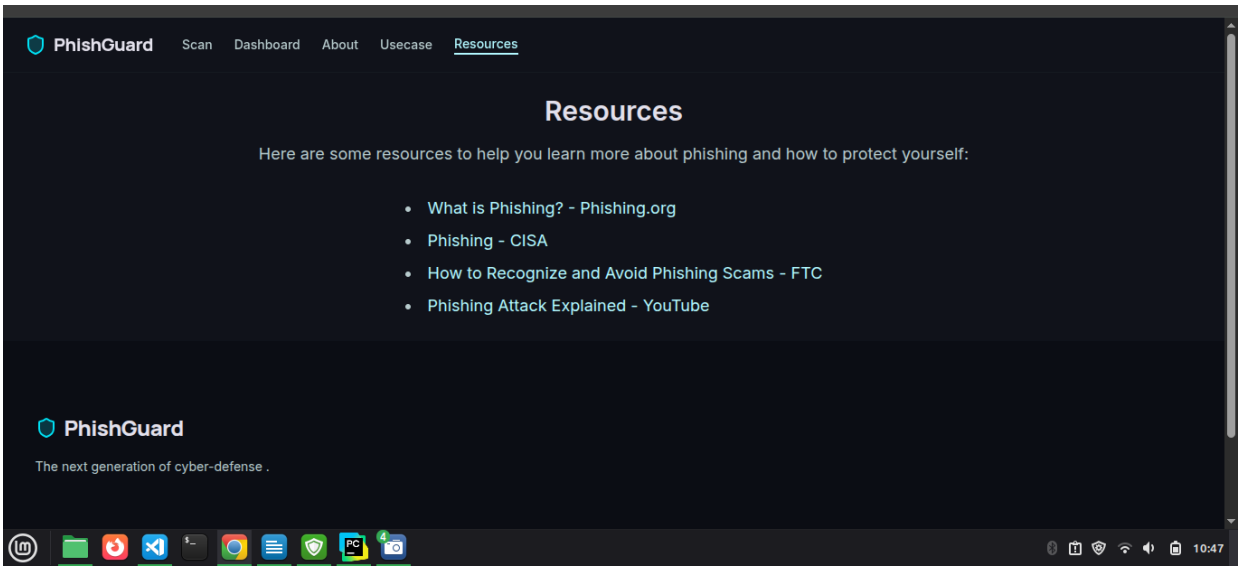
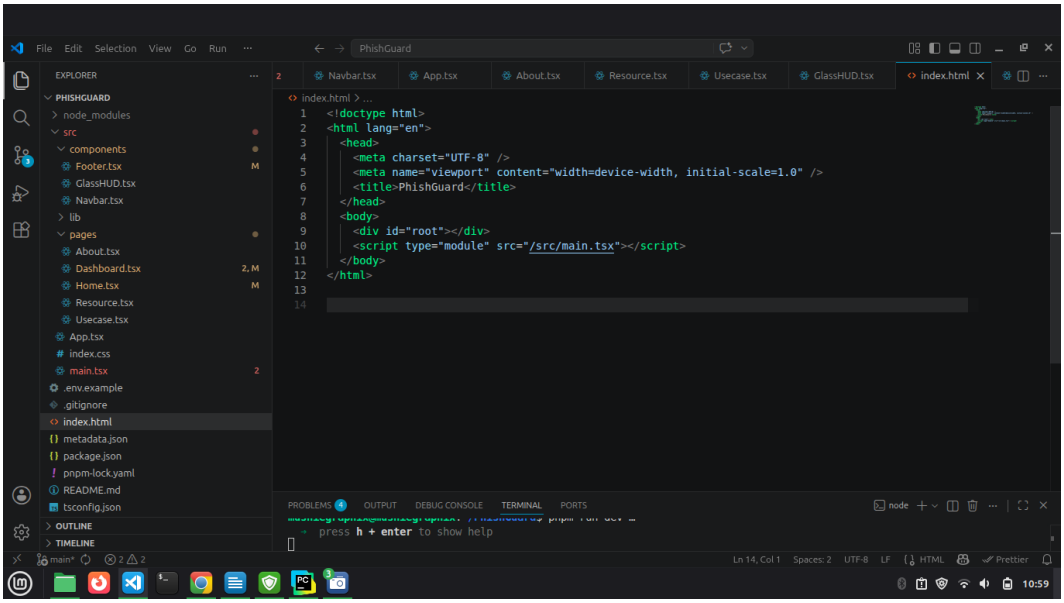


Figure 4.17: The Resource page



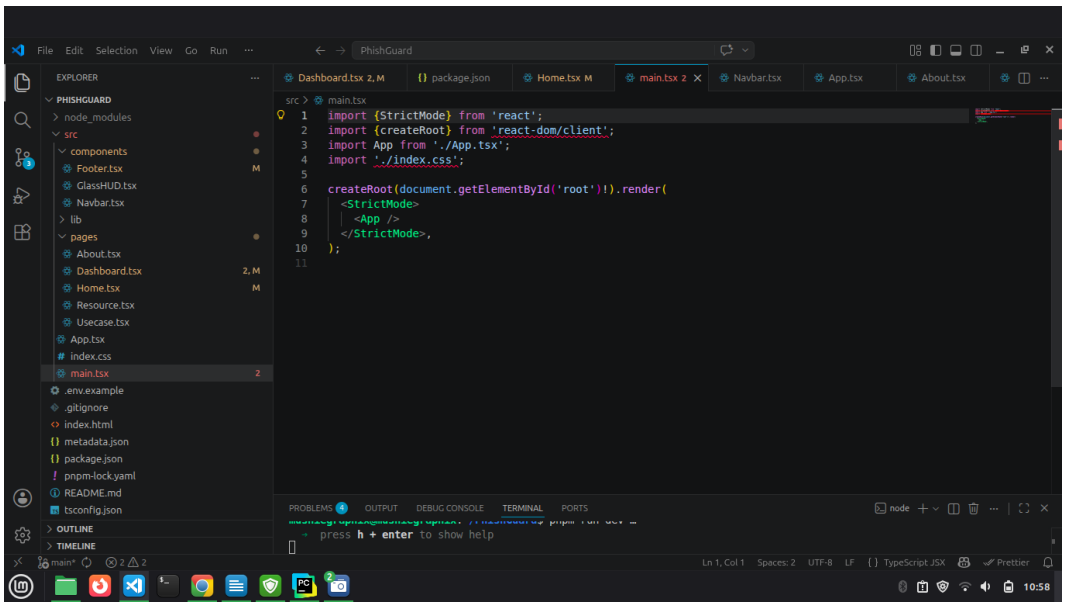


Figure 4.18 :Web Application Source code

4.3.6 Application Programming Interface (API) Architecture

To facilitate seamless interoperability, an Application Programming Interface (API) was engineered to serve as the foundational bridge connecting the front-end web application with the back-end machine learning server. The backend architecture is powered by the Django Python web framework. Communication between the user interface and the server is handled asynchronously via JavaScript Object Notation (JSON) payloads, utilizing a standard HTTP request-and-response paradigm.

The technical workflow and deployment configurations of this middleware layer are illustrated through the following components:

- **Backend Logic and Feature Mapping:** Figure 4.19 illustrates the programmatic implementation of the API in Python. This script establishes the critical data channel between inbound JSON payloads and the feature extraction algorithms optimized for real-time URL evaluation.

- **Routing and Endpoints:** The configuration and initialization of the specific API routing paths (URLs) are demonstrated in Figure 4.20.

- **Server Execution:** The active operational state of the localized Django development server hosting the API is captured in Figure 4.21.

- **Endpoint Validation:** To verify the functional integrity of the deployment, the API endpoints were subjected to rigorous testing using Postman. Figure 4.22 depicts this validation process, showcasing how HTTP requests return accurate real-time classification verdicts (phishing versus legitimate).

- **Frontend Integration:** Figure 4.23 outlines the frontend JavaScript integration logic used to bind the operational API endpoints to the broader React application architecture (the comprehensive source script for which is catalogued in Appendix B).

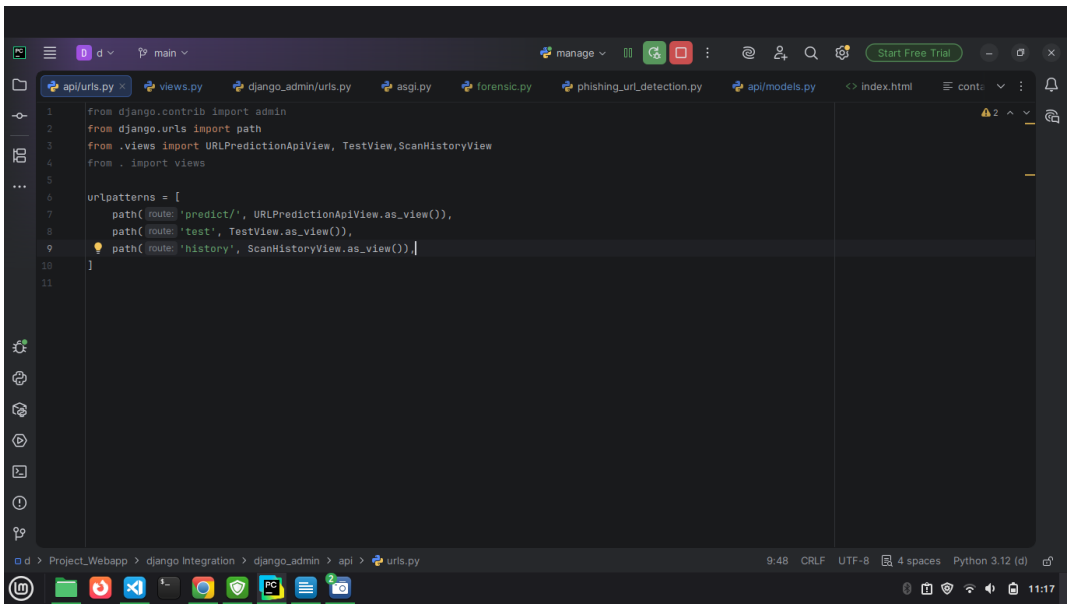


Figure 4.20: API URL Code

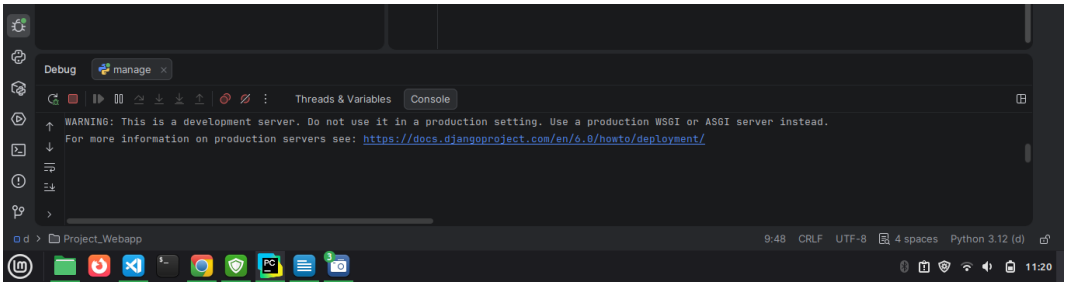


figure 4.21: The web-app being executed on the Django local server

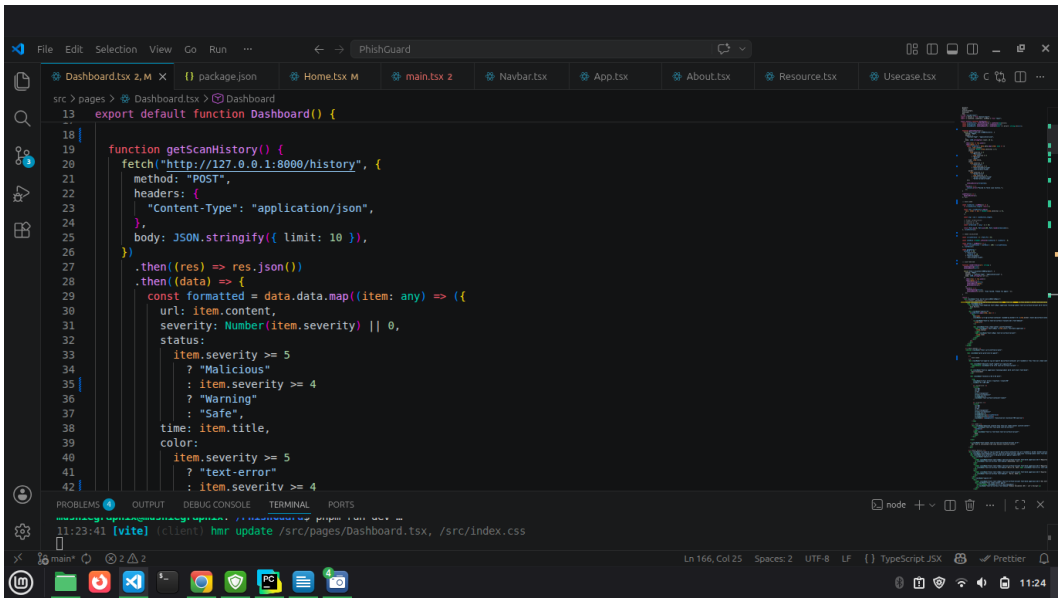


Figure 4.23: JavaScript code linking API

CHAPTER FIVE

Summary, Conclusion and Recommendations

5.1 Summary of the Study

Globally, phishing variations represent one of the most rapidly escalating cyber threats, resulting in annual financial losses amounting to billions of dollars for internet users. These malicious operations heavily exploit diverse social engineering vectors to illicitly harvest sensitive, confidential user credentials. Consequently, phishing campaigns are distributed across numerous digital communication channels, including electronic mail, real-time chat platforms, browser pop-up interfaces, and fraudulent websites.

The primary objective of this research was to critically analyse the execution methodologies employed by threat actors and evaluate contemporary detection frameworks proposed within the research community. To address the vulnerabilities inherent in manual identification, this study designed and deployed an automated classification system. This architecture integrates six distinct machine learning and deep learning paradigms, specifically: Decision Tree (DT), Support Vector Machine (SVM), XGBoost, Multi-Layer Perceptron (MLP), Autoencoder Neural Networks, and Random Forest (RF), to identify malicious structural anomalies within Uniform Resource Locators (URLs).

Following extensive empirical evaluation, the optimal algorithmic model—determined by its performance against the established feature extraction framework—was integrated into a functional, single-page web application. This deployment provides end-users with an accessible interface to input arbitrary URLs and receive instantaneous verification regarding the structural legitimacy of the target web domain.

5.2 Contributions to Knowledge

This study expands the existing body of knowledge within cybersecurity diagnostics by introducing a highly optimized, dual-layer processing framework (combining structural feature engineering with ensemble learning) that accelerates real-time URL verification. By shifting complex machine learning classification into an intuitive, user-facing web environment, this research bridges the gap between theoretical predictive modelling and practical, decentralized digital self-defense tools for non-technical users.

5.3 Conclusion

The system developed in this study successfully validates the utility of using machine learning and deep neural network architectures to mitigate web-based deception. By processing targeted URL string attributes, domain metrics, and source-code behaviours through trained classifiers, the system achieves high diagnostic accuracy. Ultimately, the empirical results confirm that automated feature extraction paired with optimized ensemble frameworks provides a highly reliable paradigm for distinguishing between malicious and benign web addresses.

5.4 Recommendations for Future Research

While this project provides a solid foundation for threat mitigation and increases user awareness regarding the structural vectors of digital fraud, several optimization paths remain open for future development:

- **Deployment as a Browser Extension:** To maximize proactive defence, the core detection engine should be migrated into a cross-browser plugin architecture, enabling real-time, background URL scanning prior to page rendering.
- **Dynamic Feature Adaptation:** Future iterations should incorporate real-time threat intelligence feeds to allow the models to adapt to zero-day phishing tactics and changing domain obfuscation techniques.

REFERENCES

Chiper, M. and Ionescu, R.T. (2025). 'Every Character Counts: From Vulnerability to Defense in Phishing Detection', *arXiv preprint arXiv:2509.20589*.

Cyber Security Agency of Singapore. (2025). *CSA's 2024 Cybersecurity Public Awareness Survey*. Available at: https://www.csa.gov.sg (Accessed: 15 January 2026).

Dam, Minh-Linh & Hung, Tran. (2025). A Feature-Engineered Dataset of Benign and Phishing URLs for Machine Learning and Large Language Models Evaluation. Data in Brief. 63. 112162. 10.1016/j.dib.2025.112162.

FBI. (2025). *The FBI Released Its Internet Crime Report 2024*. Available at: https://ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf (Accessed: 15 January 2026).

Prasad, Arvind & Yadav, Vibhu & Solanki, Chirag & Goswami, Harshit & Jha, Tanmay & Nagal, Dushyant. (2025). StealthPhisher: A Defensive Framework against Phishing Attack using Hybrid Deep Learning and GenAI. Expert Systems with Applications. 299. 130205. 10.1016/j.eswa.2025.130205.

Imperva. *What is Phishing?* Available at: https://www.imperva.com (Accessed: 15 January 2026).

Kaspersky. (2025). *Financial Cyberthreats in 2024*. Available at: https://securelist.com/financial-threat-report-2024/115966/ (Accessed: 15 January 2026).

Potpelwar, R.S., Kulkarni, U.V. and Waghmare, J.M. (2025). 'LegitPhish: A large-scale annotated dataset for URL-based phishing detection', *Data in Brief*, 63, p.111972.

Rahman, T. (2021). 'Human factors in cybersecurity: a scoping review', *The 12th International Conference on Advances in Information Technology*, pp.1–11.

Vourganas, I.J. and Michala, A.L. (2024). 'Applications of Machine Learning in Cyber Security: A Review', *Journal of Cybersecurity and Privacy*, 4(4), pp.972–992.

Woralert, C., Liu, C. and Blasingame, Z. (2024). 'Towards Effective Machine Learning Models for Ransomware Detection via Low-Level Hardware Information', *International Workshop on Hardware and Architectural Support for Security and Privacy*.

Borate, Vishal & Adsul, Dr & Dhakane, Mr & Gawade, Mr & Ghodake, Ms & Jadhav, Mr. (2024). A Comprehensive Review of Phishing Attack Detection Using Machine Learning Techniques. International Journal of Advanced Research in Science Communication and Technology. 4. 435-441. 10.48175/IJARSCT-19963.

Anti-Phishing Working Group. (2025). *Phishing Activity Trends Report: Second Quarter 2025*. Available at: https://apwg.org/trendsreports (Accessed: 15 January 2026).

Bharathi, V., Harshini, K., Meghana, B., Varshini, G. and Chandana, D. (2025). 'An automated approach for detecting phishing URLs using supervised learning models', *International Journal of Communication Networks and Information Security*, 17(4), pp.336–345.

Bustio-Martínez, Lázaro & Herrera-Semenets, Vitali & González-Ordiano, Jorge & Pérez-Guadarrama, Yamel & niga-Morales, Luis & Montoya-Godínez, Daniela & Carmona, Miguel Angel & Berg, Jan. (2025). Enhanced Phishing Detection Using Multimodal Data. Knowledge-Based Systems. 334. 115105. 10.1016/j.knosys.2025.115105.

Elkhawas, Amr & Chen, Thomas & Gashi, Ilir. (2025). Privacy-Preserving Federated Learning for Phishing Detection. IEEE Technology and Society Magazine. PP. 1-8. 10.1109/MTS.2025.3558971.

Shahbazian, Reza & Greco, Sergio. (2023). Generative Adversarial Networks Assist Missing Data Imputation: A Comprehensive Survey & Evaluation. IEEE Access. PP. 1-1. 10.1109/ACCESS.2023.3306721.

Imperva. (2021). *What is Social Engineering?* Available at: https://www.imperva.com/learn/application-security/social-engineering-attack (Accessed: 10 January 2026).

Jadhav, A., Pramod, D. and Ramanathan, K. (2019). 'Comparison of performance of data imputation methods for numeric dataset', *Applied Artificial Intelligence*, 33(10), pp.913–933. doi:10.1080/08839514.2019.1637138

Alsabri, Abdulrahman & Al-Hadi, Marwa. (2025). A Hybrid CNN-BLSTM Model for Phishing Attack Detection Using Deep Learning to Strengthen Internet Security. . 964-972. 10.59628/jast.v3i4.1822.

Kondeti, P., Konka, S. and Kavishree, S. (2021). 'Machine learning models for phishing website detection: A comparative study', *International Journal of Advanced Computer Science and Applications*, 12(5), pp.418–425.

k, Shanmathi & S, Latha. (2026). Phishing Detection Using Behavioral Cues in Browser Interaction. Indian Journal of Computer Science and Technology. 247. 10.59256/indjcst.20260502028.

Alruhaymi, Abdullah & Kim, Charles. (2021). Study on the Missing Data Mechanisms and Imputation Methods. Open Journal of Statistics. 11. 477-492. 10.4236/ojs.2021.114030.

Pingfan Xu. (2021). 'PhishBERT: Transformer-based phishing URL detection with pre-training', *Proceedings of the AAAI Conference on Artificial Intelligence*, 38, pp.1234–1242.

Abraham, Arvind and Gressel, Gilad and Achuthan, Krishnashree, Temporal Resilience of Phishing Detection Models in Machine Learning (December 29, 2019). Proceedings of the 5th International Conference on Cyber Security & Privacy in Communication Networks (ICCS) 2019 | National Institute of Technology, Kurukshetra, India,

Mohammad, Rami, McCluskey, T.L. and Thabtah, Fadi (2015) Tutorial and Critical Analysis of Phishing Websites Methods. Computer Science Review, 17. pp. 1-24. ISSN 1574-0137

Naqvi, S.H. (2026). 'Fraud detection using machine learning in financial transactions: a systematic review', *Veredas do Direito*, 23, e234187.

Salloum, Said & Alshurideh, Muhammad & Elnagar, Ashraf & Shaalan, Khaled. (2020). Machine Learning and Deep Learning Techniques for Cybersecurity: A Review. 10.1007/978-3-030-44289-7_5.

Sani, Abuh & Onimole, Oludolamu & Adebayo, Adetunji & Akande, Nathaniel & Eziokwu, Uju. (2025). Phishing Detection Using Natural Language Processing and Behavioural Analysis: A Multi-Faceted Approach. International Journal of Scientific and Management Research. 8. 2025. 10.37502/IJSMR.2025.81105.

An, Panharith & Shafi, Rana & Mughogho, Tionge & Onyango, Allan & Nachevski, Nikola. (2025). Multilingual Email Phishing Attacks Detection using OSINT and Machine Learning. 10.48550/arXiv.2501.08723.

Obaid, Ahmed & Ibrahim, Kareem & Shawkat, Azmi & Nejrs, Salwa. (2021). An adaptive approach for internet phishing detection based on log data. Periodicals of Engineering and Natural Sciences (PEN). 9. 622. 10.21533/pen.v9i4.2398.

Patel, R. and Mehta, S. (2023). 'Adaptive decision trees for evolving phishing attacks', *Machine Learning*, 112(5), pp.1567–1589.

Periyasamy, Usha. (2022). DECISION TREE ALGORITHMS IN MACHINE LEARNING.

Arshad, Ayesha & Ur Rehman, Attique & Javaid, Sabeen & Ali, Tahir & Sheikh, Javed & Azeem, Muhammad. (2021). A Systematic Literature Review on Phishing and Anti-Phishing Techniques. 10.48550/arXiv.2104.01255.

Jantabo, Abduljalil & As, Jyoti. (2026). Phishing Detection Using Convolutional Neural Network.

Salman, Hasan & Kalakech, Ali & Steiti, Amani. (2024). Random Forest Algorithm Overview. Babylonian Journal of Machine Learning. 2024. 69-79. 10.58496/BJML/2024/007.

Gholampour, Parisa & Verma, Rakesh. (2023). Adversarial Robustness of Phishing Email Detection Models. 67-76. 10.1145/3579987.3586567.

Sadaf, Kishwar. (2023). Phishing Website Detection using XGBoost and Catboost Classifiers. 1-6. 10.1109/ICSCA57840.2023.10087829.

Al Darmaki, Ahood; Khan, Ajmal; Mughal, Muhammad Rizwan; and M. Bait-Suwailam, Mohammed (2025) "Phishing Detection Using Data-Driven Intelligence," *The Journal of Engineering Research*: Vol. 22: Iss. 2, Article 2.

Murphy, Frederick & Powell, Layla & Hughes, Clarence & Barnes, Olivia & Perry, Raymond & Michael, Ilesanmi. (2023). Cost sensitive decision trees and random forests in fraud identification.

Lavanya, Mrs & Padmini, Rangu & Narayana, Mamilla & Pavan, Singamsetty & Nazma, Shaik & Professor, Asst & S V, Sai. (2025). Advanced Phishing Website Classification Using SVM and LightGBM Models. International Journal of Advanced Scientific Research and Engineering Trends. 9. 35-38. 10.65521/ijasret.v9i3.1805.

Yeasmin, Nilufa & Refat, Abu & Singh, Bikash Chandra & Alom, Zulfikar & Aung, Zeyar & Azim, Mohammad. (2025). EnLeM: ensemble learning-based model to detect phishing websites. EURASIP Journal on Information Security. 2026. 10.1186/s13635-025-00219-1.

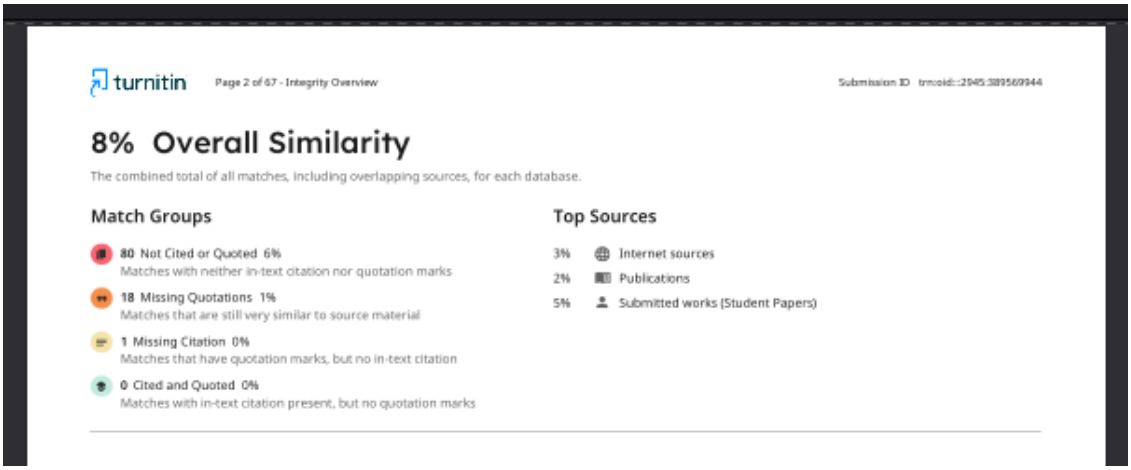
Huang, Yitao & Li, Shenghan & Brooks, Nathan & Zhang, Xuguang. (2026). Graph Neural Networks for Detecting Anomalous Transaction Patterns in Enterprise Accounting Systems. Frontiers in Artificial Intelligence Research.

DeepAI. (n.d.). *Feature Extraction Definition*. Available at: https://deepai.org/machine-learning-glossary-and-terms/feature-extraction (Accessed: 10 January 2026).

Merriam-Webster Dictionary. (2025). *Definition of System Analysis*. Available at: https://www.merriam-webster.com (Accessed: 10 January 2026).

PhishTank. (2025). *Phishing URL Dataset*. Available at: https://phishtank.org (Accessed: 10 January 2026).

University of New Brunswick. (2016). *URL Dataset*. Available at: https://www.unb.ca/cic/datasets/url-2016.html (Accessed: 10 January 2026).



turnitin Page 2 of 67 - Integrity Overview Submission ID trnoid::2945:3895569944

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **80** Not Cited or Quoted 6%
  Matches with neither in-text citation nor quotation marks
- **18** Missing Quotations 1%
  Matches that are still very similar to source material
- **1** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

3% Internet sources
2% Publications
5% Submitted works (Student Papers)