

AN ENGINEERING PROJECT REPORT

ON

“Phishing URL detection using ML”

Submitted By

Sushil Paudel – 200348

Bibek Pandeya – 200311

Bhushan Bartaula – 200310

Prakash Lamichhane – 200323

Submitted To

Department of IT and Computer Engineering

**In partial fulfillment of requirement for the degree of Bachelor of engineering in
Computer Engineering.**



Cosmos College of Management & Technology

(Affiliated with Pokhara University)

Tutepani, Lalitpur, Nepal

Date of Submission: - 2082/04/08

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who supported and guided us throughout the journey of this project. First and foremost, we extend our sincere thanks to our **supervisor Er.Ajaya Adhikari**, whose invaluable support, timely feedback, and continuous encouragement played a crucial role during the implementation and successful completion of this project.

We are also deeply thankful to the **Department of Information and Communications Technology, Cosmos College of Management and Technology, Tutepani, Lalitpur** for providing us with the necessary consultation, knowledge, and academic environment that made this project possible.

We would like to especially acknowledge **Er. Chiranjibi Pandey**, Deputy Head of Department (DHOD), for his insightful guidance, encouragement, and support throughout the project. His contributions have been instrumental in helping us maintain academic rigor and clarity throughout our work.

Our special thanks go to all the faculty members for their motivation, guidance, and encouragement. Their insightful suggestions and constructive feedback were instrumental in helping us refine our work.

Lastly, we warmly welcome any constructive criticism and suggestions for further improvement of this project.

Sushil Paudel

Bibek Pandeya

Bhushan Bartaula

Prakash Lamichhane

ABSTRACT

Phishing attacks continue to evolve as a major cybersecurity threat, resulting in significant financial and data losses for individuals and organizations worldwide. These attacks typically rely on deceptive tactics to trick users into revealing sensitive information such as usernames, passwords, and financial credentials. Common phishing vectors include emails, instant messages, pop-up alerts, and fraudulent websites.

This project presents a machine learning-based approach to detect phishing websites by classifying URLs as either legitimate or malicious. The system is trained on a comprehensive dataset containing both phishing and benign URLs, collected from publicly available threat intelligence sources and academic datasets.

To improve prediction accuracy, multiple machine learning models, including deep neural networks, are employed and evaluated. The dataset, consisting of over 5,000 raw URLs, is expanded into a total of 10,000 samples, split into 80% training and 20% testing data. The features used for classification are grouped into three categories: **address bar-based features**, **domain-based features**, and **HTML & JavaScript-based features**.

As a practical implementation, a web application is developed that allows users to input any URL and receive real-time feedback on whether the link is safe or potentially harmful. Among the models tested, **XGBoost achieved the highest accuracy of 86.6%**, making it the most reliable choice for deployment.

This system aims to contribute to safer internet browsing by providing an intelligent and accessible phishing detection tool.

Keywords: Phishing Detection, Machine Learning, URL Classification, Cybersecurity, Web Application, Deep Neural Network, Feature Extraction, XGBoost.

Table of Contents

ACKNOWLEDGEMENT	ii
ABSTRACT	iii
List of Figures:	vii
List of abbreviations: -	viii
CHAPTER 1: INTRODUCTION	1
1.1 Background Information:	1
1.2 Statements regarding the problems:	2
1.3 Scope of the study:	2
1.4 Objectives:	2
1.5 Methodology:	2
1.5.1 Data Collection:	3
1.5.2 Data Preprocessing:	3
1.5.3 Feature Extraction:	3
1.5.4 Model Implementation:	3
1.5.5 Model Evaluation:	4
1.5.6 Web Application Development:	4
1.6 Significance of the Project:	4
CHAPTER 2: LITERATURE REVIEW	5
2.1 Overview of the Study	5
2.2 Theoretical Review:	5
CHAPTER 3: SYSTEM ANALYSIS AND DESIGN	8
3.1 Overview of System Analysis	8
3.2 Analysis of Existing System	8
3.3 Proposed System	9
3.3.1 Benefits of the new system	9
3.4 Model Development Model	9
3.4.1 Data Processing	9
3.4.2 Preprocessing	10

3.4.3	Exploratory Data Analysis	10
3.4.4	Feature Extraction.....	10
3.4.5	Model Training.....	11
3.4.6	Model Testing.....	11
3.4.7	Model Evaluation	11
3.5	System Modelling.....	12
3.5.1	System Architecture.....	12
3.5.2	Use a case diagram of the system.....	12
3.5.3	Flowchart of the system	14
CHAPTER 4: SYSTEM IMPLEMENTATION AND RESULTS		17
4.1	Data collection.....	17
4.1.1	Phishing URLs:	17
4.1.2	Legitimate URLs:.....	18
4.2	Data Preprocessing	19
4.2.1	Data encoding:.....	20
4.3	Exploratory Data Analysis (EDA)	20
4.4	Feature Extraction.....	21
4.4.1	Address Bar-Based Features:	21
4.4.2	Domain-Based Features:.....	22
4.4.3	HTML & JavaScript-Based Features:	23
4.5	Model Training	26
4.5.1	Decision Tree:.....	26
4.5.2	Random Forest:.....	26
4.5.3	Support Vector Machine (SVM):.....	27
4.5.4	XGBoost:	27
4.5.5	Multilayer Perceptron (MLP):	27
4.5.6	Autoencoder Neural Network:.....	27
4.6	Model Evaluation.....	28
4.7	Deployment	28
4.8	Discussion of the Results:.....	28
4.9	Feature importance:	29

4.10 Impact of training data size:	29
4.11 Comparison with literature:	29
CHAPTER 5: CONCLUSION	31
5.1 Summary	31
5.2 Limitations	32
5.3 Future Enhancement	33
REFERENCES	34

List of Figures:

Figure 2.0.1 Worldwide financial losses (in billion) due to phishing attacks	6
Figure 2.0.2 growth of phishing attacks from 2005 to 2015	7
Figure 3.1 Machine Learning development Process	11
Figure 3.2 Architectural Design of the Proposed System	13
Figure 3.3 Use Case diagram for Proposed System	14
Figure 3.4 Flowchart of the proposed System	15
Figure 3.5 Flowchart of the web interface	16
Figure 4.1.1 Dataset of Phishing URLs	17
Figure 4.1.2 Datasets of Legitimate URLs	18
Figure 4.2.1 Summary of the dataset	19
Figure 4.2.2 Number of missing values in the dataset	19
Figure 4.3 heatmap	20
Figure 4.4.1 Code for Address bar based feature extraction	22
Figure 4.4.3 Code for Address bar based feature extraction	24
Figure 4.4.4 Code computation for all the feature extraction used dataset.	25
Figure 4.5.1 Feature importance for Decision Tree classifier	26
Figure 4.5.2 Feature importance for Random forest classifier	27
Figure 4.11.0.1 Frontend demo	29
Figure 4.11.2 Model train vs test accuracy	30
Figure 4.11.3 Django admin panel	30

List of abbreviations: -

Abbreviations	Full Form
API	Application Programming Interface
CCMT	Cosmos College of Management and Technology
CSV	Comma Separated Values
DNN	Deep Neural Network
DNS	Domain Name System
EDA	Exploratory Data Analysis
FNR	False Negative Rate
FPR	False Positive Rate
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
ML	Machine Learning
MLP	Multilayer Perceptron
SVM	Support Vector Machine
UI	User Interface
URL	Uniform Resource Locator
WHOIS	Domain WHOIS Lookup
XAI	Explainable Artificial Intelligence
XGBoost	Extreme Gradient Boosting

CHAPTER 1: INTRODUCTION

1.1 Background Information:

The rapid growth of the internet and digital communication has significantly transformed the way individuals and organizations interact, share information, and conduct business. While this digital transformation has enabled countless opportunities, it has also opened the door to various cybersecurity threats, one of the most persistent and damaging being **phishing attacks**.

Phishing is a form of cybercrime that involves tricking individuals into revealing confidential information such as usernames, passwords, credit card numbers, and other sensitive data by impersonating a trustworthy entity in digital communication. Cybercriminals exploit various communication channels including emails, websites, social media platforms, and instant messaging services to carry out phishing attempts.

As phishing techniques become increasingly sophisticated, traditional rule-based detection methods often fail to provide adequate protection. In recent years, **machine learning** has emerged as a promising solution in the field of cybersecurity, offering the ability to analyze patterns, detect anomalies, and classify malicious content more accurately.

This project is motivated by the increasing need for **automated, intelligent solutions** that can detect phishing URLs with high accuracy and minimal human intervention. The goal is to develop a machine learning-based system capable of identifying phishing websites based on a wide range of features extracted from URLs. By training and testing the model on a robust dataset consisting of both legitimate and phishing URLs and incorporating different types of feature sets (such as address bar-based, domain-based, and HTML/JavaScript-based), the system aims to provide reliable classification results.

Furthermore, a user-friendly **web application** is developed as part of the project, allowing end-users to input URLs and receive instant feedback on their legitimacy. This solution contributes to raising awareness and enhancing security among everyday internet users, ultimately reducing the risk posed by phishing threats.

1.2 Statements regarding the problems:

Phishing attacks have gotten increasingly complex; it is very difficult for an average person to determine if an email message link or website is legitimate. Cyber-attacks by criminals that employ phishing schemes are so prevalent and successful nowadays. Hence, this project seeks to address fake URLs and domain names by identifying phishing website links. Therefore, having a web application that provides the user with an interface to check if a URL is Phishing or legitimate will help decrease security risks to individuals and organizations.

1.3 Scope of the study:

This study explores data science and machine learning models that use datasets gotten from open-source platforms to analyze website links and distinguish between phishing and legitimate URL links.

The model is integrated into a web application, allowing a user to predict if a URL link is legitimate or phishing. This online application is compatible with a variety of browsers.

1.4 Objectives:

The main objective of our project is to develop a machine learning-based system capable of accurately detecting and classifying phishing URLs and deploying them as a web application for real-time usage. Besides the main objective, the specific objectives can be enlisted as:

- a) Collect and preprocess a dataset of phishing and legitimate URLs.
- b) Extract key features from URLs for classification.
- c) Implement and evaluate multiple machine learning models.
- d) Compare model performance based on accuracy and other metrics.
- e) Develop a user-friendly web application for URL detection.
- f) Promote phishing awareness through intelligent detection.

1.5 Methodology:

The development of the phishing URL detection system is carried out through a structured methodology involving data collection, preprocessing, feature extraction, model implementation, evaluation, and deployment. The entire process is designed to ensure that the model achieves high accuracy and usability.

1.5.1 Data Collection:

Datasets are gathered from publicly available sources containing both **phishing** and **legitimate** URLs. Sources include open phishing databases and academic repositories, ensuring a diverse and balanced dataset.

1.5.2 Data Preprocessing:

Duplicate and null entries are removed to maintain data quality. URLs are labeled as "**phishing**" or "**legitimate**" based on their source. The dataset is split into **training (80%)** and **testing (20%)** sets for model development and validation.

1.5.3 Feature Extraction:

Features are extracted from each URL based on three main categories:

- 1) **Address Bar-Based Features:** Length of the URL, presence of IP address, use of “” symbol, etc.
- 2) **Domain-Based Features:** Age of the domain, domain registration length, DNS record, etc.
- 3) **HTML & JavaScript-Based Features:** Use of suspicious scripts, iframe tags, and redirection behavior.

1.5.4 Model Implementation:

Multiple machine learning algorithms are implemented, including:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Naïve Bayes
- Deep Neural Networks (DNN)

Models are trained using the training dataset and validated using the testing dataset.

1.5.5 Model Evaluation:

Performance of each model is evaluated using the following metrics:

- **Accuracy**
- **False Positive Rate**
- **False Negative Rate**

Comparative analysis is performed to determine the most effective model for phishing URL detection.

1.5.6 Web Application Development:

A **web-based interface** is built using appropriate front-end and back-end technologies. Users can enter a URL to check its legitimacy in real time. The trained model is integrated into the backend to provide instant classification results.

1.6 Significance of the Project:

This project plays an important role in addressing the growing threat of phishing attacks by using machine learning to detect malicious URLs. It promotes cybersecurity awareness and provides a practical tool for users to check the legitimacy of websites in real time. Academically, it demonstrates the application of machine learning in real-world scenarios and contributes to ongoing research in intelligent threat detection systems.

CHAPTER 2: LITERATURE REVIEW

2.1 Overview of the Study

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

2.2 Theoretical Review:

According to the [2024 Verizon DBIR](#) [2], the human element is contained in 68% of breaches. Of those, the [Comcast Business Cybersecurity Threat Report says 80-95%](#) [9] are initiated by a phishing attack, and the total volume of phishing attacks has skyrocketed by 4,151% since the advent of ChatGPT in 2022, [according to SlashNext](#).

At an estimated \$4.88M per phishing breach ([IBM Cost of a Data Breach Report 2024](#)) [11], social engineers are making billions by being better at making people click than we are at understanding what makes them tick.

Top 5 phishy industries

Rank	Industry	Malicious Threats reported / user / year	Total phish / 1K org	Breaches / 1K org (Hoxhunt)	Breaches / 1000 (Hoxhunt)	% Reduction
—	Global Average	1.40	2330	466	74.56	86%
1	Media Production	2.91	4610	922	138.00	85%
2	Government	2.08	3010	602	93.31	85%
3	Manufacturing & Construction	1.65	2540	508	81.28	84%
4	Financial Services	1.41	1990	398	45.77	89%
5	Oil & Energy	1.28	1820	364	47.32	87%

The **RSA Anti-Fraud Command Center** reported a **160% increase in phishing attacks in 2012** compared to 2011, indicating a steep rise in cybercrime activities. In **2013**, the total number of phishing attacks rose to approximately **450,000**, with estimated **financial losses exceeding USD 5.9 billion**. These numbers underscore the seriousness of phishing threats as shown in Fig: 2.2.2.

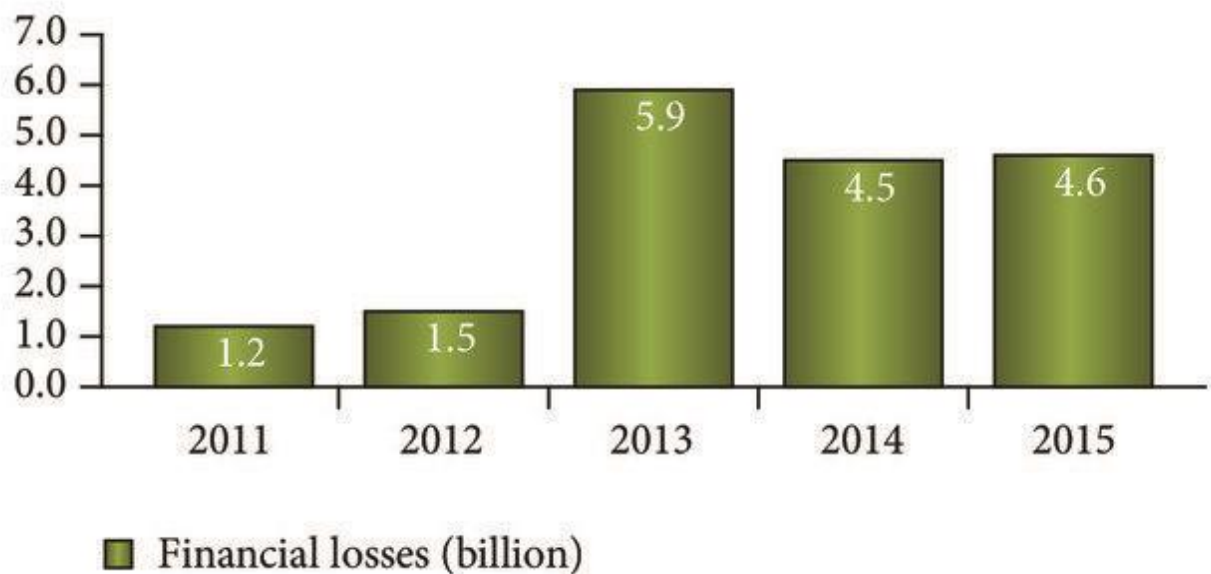


Figure 2.0.1 Worldwide financial losses (in billion) due to phishing attacks

According to the **APWG Phishing Activity Trends Report (2014)**, the **first quarter of 2014** recorded **125,215 phishing attacks**, marking a **10.7% increase** over the last quarter of 2013. Additionally, over **55% of phishing websites** included the name of the targeted organization to appear more trustworthy, and **99.4%** of phishing websites operated on **port 80**—the standard HTTP port—making them blend seamlessly with legitimate traffic as shown in Fig: 2.2.3.

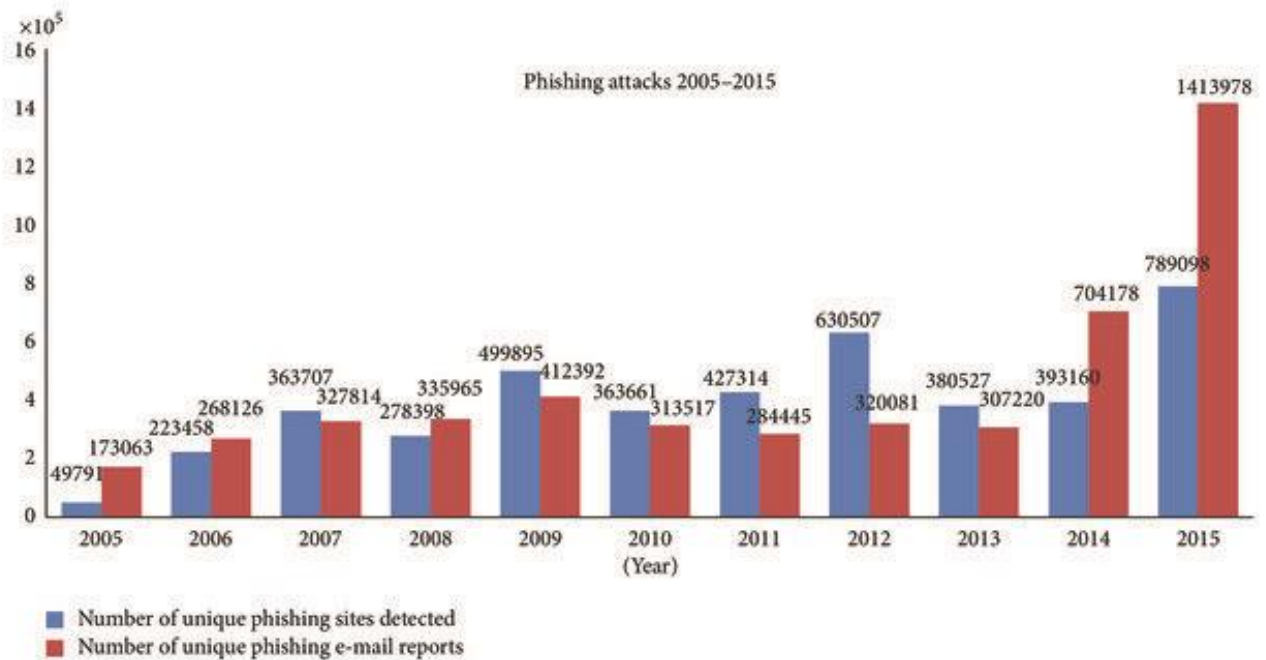


Figure 2.0.2 growth of phishing attacks from 2005 to 2015

According to **Ankit and Gupta (2017)**, as cited in *Internet World Stats (2014)*, the global number of internet users reached **2.97 billion in 2014**, representing more than **38% of the world population**. The rapid digital expansion has provided attackers with a larger pool of potential victims, especially those unaware of phishing tactics.

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 Overview of System Analysis

This section presents a comprehensive overview of the analytical approach adopted throughout the project to meet its objectives. System analysis plays a critical role in understanding the problem domain, evaluating existing solutions, and designing a more efficient and robust system. It serves as the foundation upon which effective system development is built.

The methodology chosen for this research guided the process of identifying, analyzing, and solving the challenges associated with phishing URL detection. A clear understanding of the problem, combined with a structured analysis of existing systems and technologies, enabled the development of a more intelligent and user-friendly solution.

According to the **Merriam-Webster Dictionary**, system analysis is defined as the process of studying a procedure or operation to determine its goals and purposes, and to develop systems and procedures that achieve those goals efficiently. In the context of this project, system analysis involves examining how phishing detection systems operate, identifying their limitations, and exploring opportunities to enhance their accuracy and usability through machine learning.

By thoroughly analyzing both the problem and the existing methods, this project ensures that the proposed solution is not only technically sound but also practically applicable in real-world scenarios.

3.2 Analysis of Existing System

The existing phishing detection systems often suffer from low accuracy and high false alarm rates, particularly when new and evolving phishing techniques are introduced. Most rely heavily on blacklist-based methods, which are ineffective against emerging threats. As registering new domains has become increasingly easy, no blacklist can remain fully up to date, making such approaches insufficient for reliable phishing detection.

3.3 Proposed System

The proposed phishing detection system leverages machine learning models and deep neural networks, comprising two main components: the learning models and a web application. The models include Decision Tree, Support Vector Machine, XGBoost, Multilayer Perceptron, Autoencoder Neural Network, and Random Forest.

These models are chosen based on performance comparisons among various algorithms. Each model is trained and tested using content-based features extracted from both phishing and legitimate datasets.

Ultimately, the model with the highest accuracy is selected and integrated into a web application, allowing users to check whether a given URL is phishing or legitimate.

3.3.1 Benefits of the new system

- Capable of accurately distinguishing between phishing (0) and legitimate (1) URLs.
- Helps minimize phishing-related data breaches within organizations.
- Beneficial to both individuals and organizations in enhancing cybersecurity.
- Designed with a user-friendly interface, making it simple and accessible for all users.

3.4 Model Development Model

The model development method involves utilizing multiple machine learning models, evaluating their performance, and refining them through an iterative process until a model that meets the desired requirements is achieved. Figure 3.1 illustrates the systematic steps followed in developing these machine learning models, incorporating both supervised and unsupervised learning techniques.

The following are the major stages involved in developing the machine learning model for phishing detection systems:

3.4.1 Data Processing

A variety of open-source platforms provided the datasets used to train the models. The collection includes both authentic and phishing URLs. The open-source site PhishTank, which offers phishing URL lists in CSV and JSON formats and is updated hourly, is where the phishing URLs were gathered. A random selection of more than 5,000 phishing URLs were selected from this dataset.

The legitimate URLs were obtained from the open datasets provided by the University of New Brunswick. This dataset includes benign, spam, phishing, malware, and defacement URLs. For this project, only benign URLs were considered, and over 5,000 of them were randomly selected to train the machine learning models.

3.4.2 Preprocessing

Data preprocessing is a critical step following data collection. The raw dataset was cleaned by removing redundant and inconsistent data. The dataset was then encoded using the One-Hot Encoding technique to transform it into a suitable format for machine learning models.

3.4.3 Exploratory Data Analysis

After preprocessing, exploratory data analysis techniques were applied to gain insights into the dataset. Visualization tools such as heatmaps, histograms, box plots, scatter plots, and pair plots were used to explore and summarize the data. These visualizations helped in identifying patterns, correlations, and outliers.

3.4.4 Feature Extraction

The aim of feature extraction is to reduce dimensionality while preserving important information. Website content-based features were extracted from both phishing and legitimate URLs. These features include:

Address Bar-Based Features (9 features)

Domain-Based Features (4 features)

HTML & JavaScript-Based Features (4 features)

In total, 17 features were extracted and used for phishing detection.

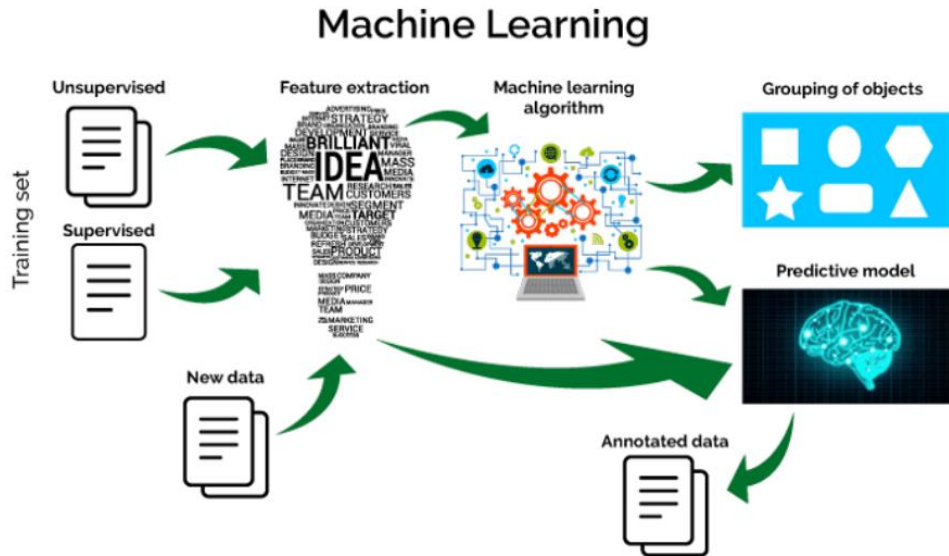


Figure 3.1 Machine Learning development Process

3.4.5 Model Training

Model training involves feeding machine learning algorithms with data to help identify and learn good attributes of the dataset. This research problem is a product of supervised learning, which falls under the classification problem. The algorithms used for phishing detection consist of supervised machine learning models (4) and deep neural networks (2), which were used to train the dataset. These algorithms include Decision Tree, Random Forest, Support Vector Machines, XGBooster, Multilayer Perceptron, and Auto-encoder Neural Network.

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

3.4.6 Model Testing

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

3.4.7 Model Evaluation

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

3.5 System Modelling

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

- i. Architecture diagram
- ii. Use case diagram
- iii. Flowcharts

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

3.5.1 System Architecture

Architectural design is concerned with understanding how a system should be organized and designing the overall structure of that system, it shows how different components of the system work together to achieve its main objectives. It is the process for identifying sub-systems making up a system and the framework for sub-system control and communication. The diagram below represents a graphical overview of the architectural design of the proposed system. Figure 3.1 shows the architecture view of the proposed phishing detection system such that a user enters a URL link and the link moves through different trained machine learning and deep neural network models and the best model with the highest accuracy is selected. Thus, the selected model is deployed as an API (Application Programming Interface) which is then integrated into a web application. Hence, a user interacts with the web application which is accessible across different display devices such as computers, tablets, and mobile devices.

3.5.2 Use case diagram of the system

The Use Case diagram describes the functionality of the system as designed from the requirements; it summarizes the details of a system and the users within the system. It is a behavior diagram and visualizes the observable interactions between actors and the system

under development. The Use case diagram consists of the system, the related use cases, and actors and relates to each other.

Figure 3.2 shows the Use case scenarios that a user can carry out on the phishing detection system.

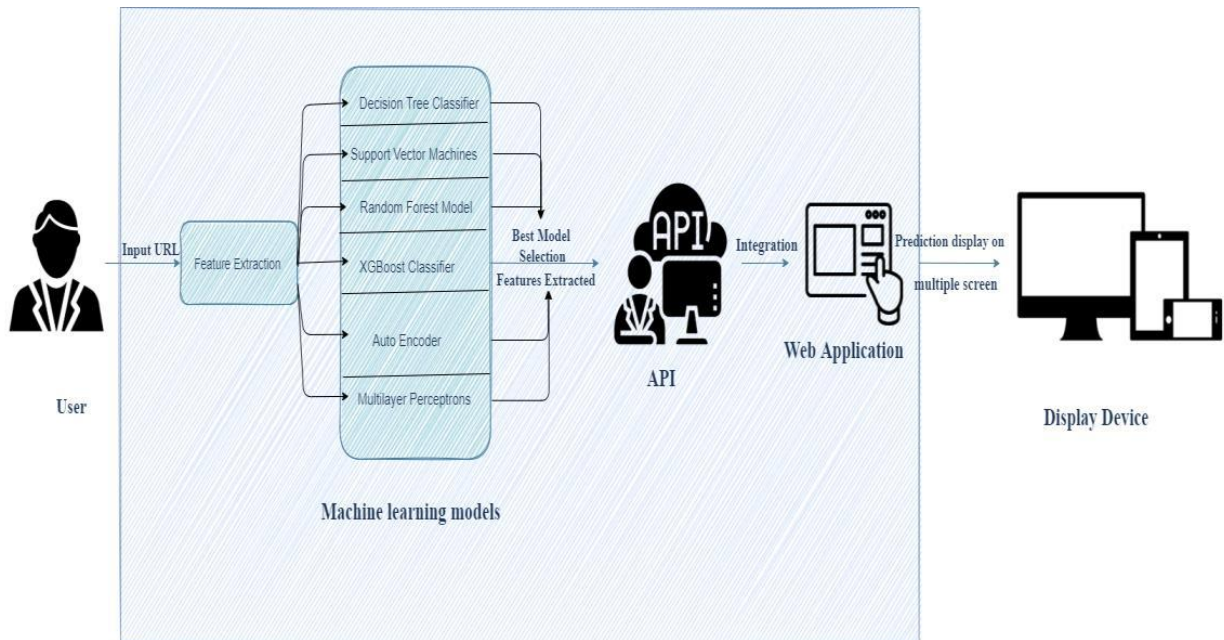


Figure 3.2 Architectural Design of the Proposed System



Figure 3.3 Use Case diagram for Proposed System

3.5.3 Flowchart of the system

A flowchart is a diagram that depicts a process, system, or computer algorithm. It is a graphical representation of the steps that are to be performed in a system, it shows the steps in sequential order. It is used in presenting the flow of algorithms and to communicate complex processes in clear, easy-to-understand diagrams.

Figure 3.4 shows the flow of phishing detection systems using the machine learning process.

Figure 3.5 shows the phishing detection web interface system. The user inputs a URL link and the website validates the format of the URL and then predicts if the link is phishing or legitimate.

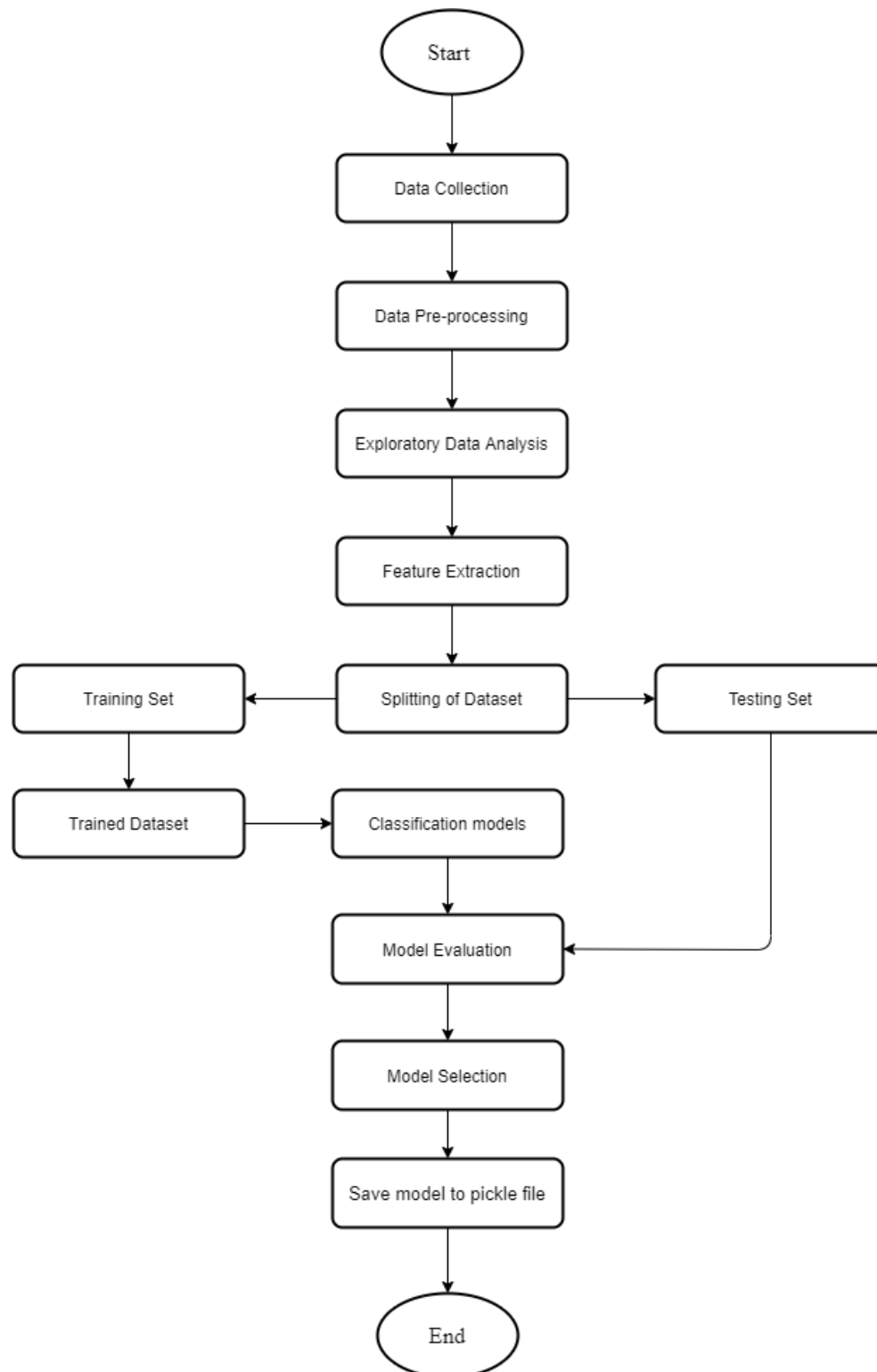


Figure 3.4 Flowchart of the proposed System

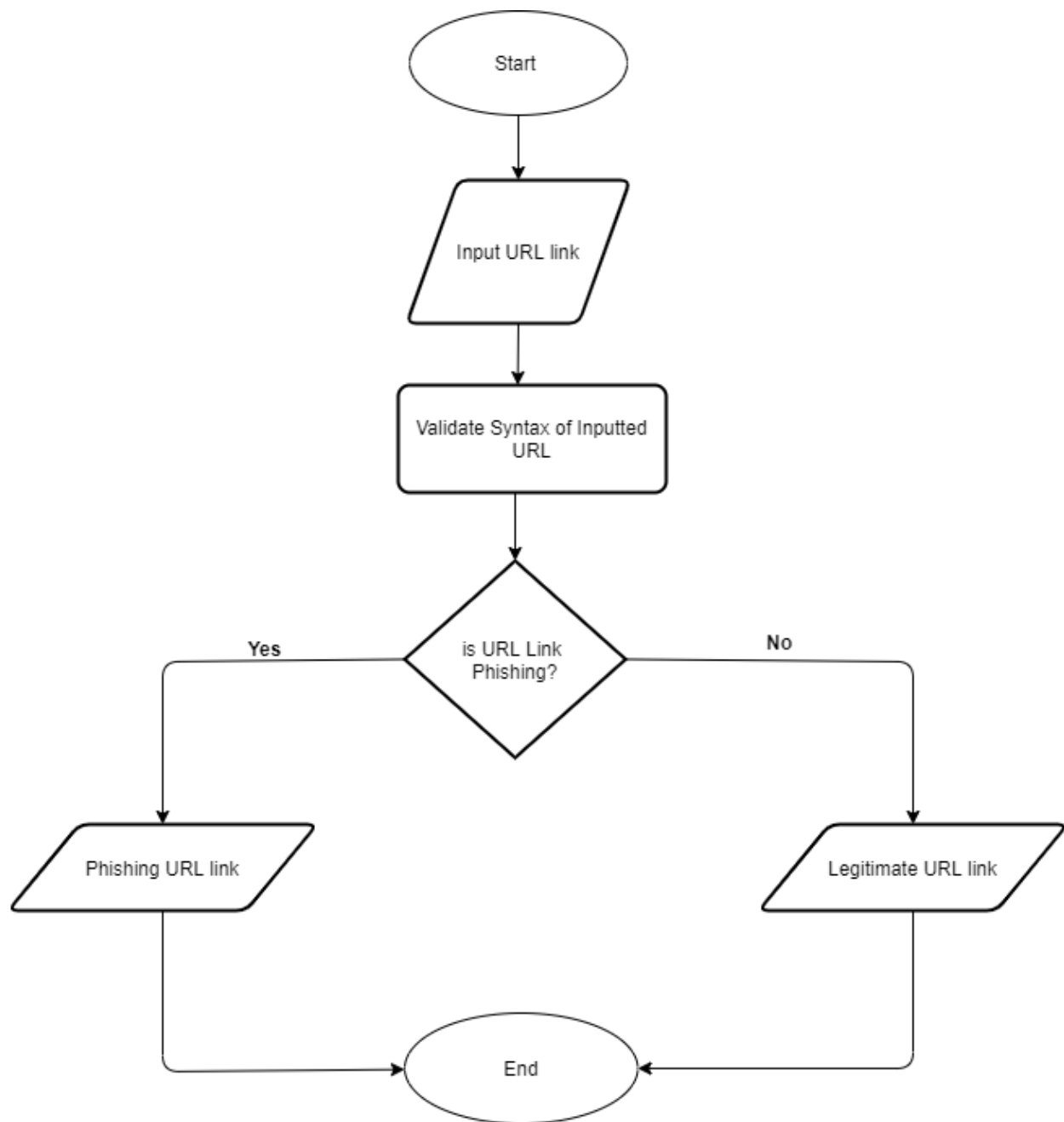


Figure 3.5 Flowchart of the web interface

CHAPTER 4: SYSTEM IMPLEMENTATION AND RESULTS

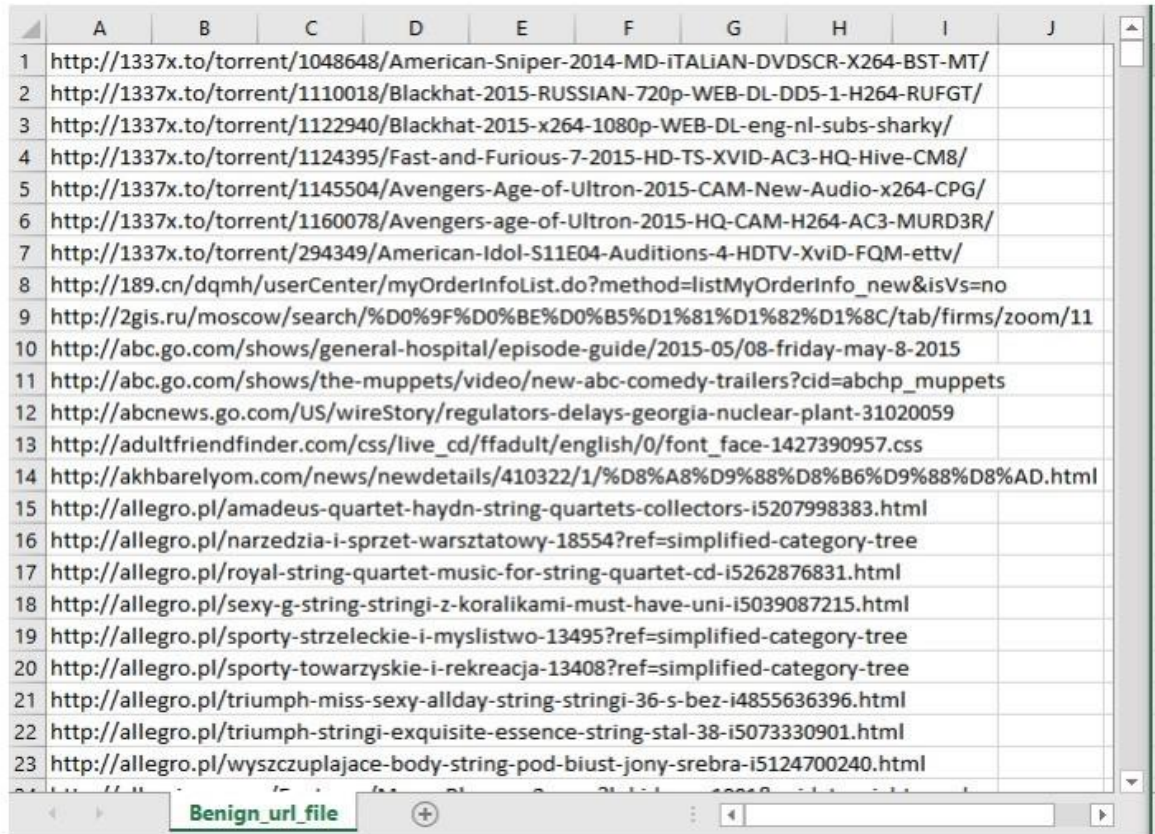
The project “Detection of Phishing Websites Using Machine Learning” was systematically implemented in several important phases to ensure accuracy, reliability, and practical usability.

4.1 Data collection

The first step was to gather relevant data, which is the backbone of any machine learning (ML) project.

4.1.1 Phishing URLs:

Collected from Phish Tank, an open-source platform known for regularly updated lists of phishing websites.

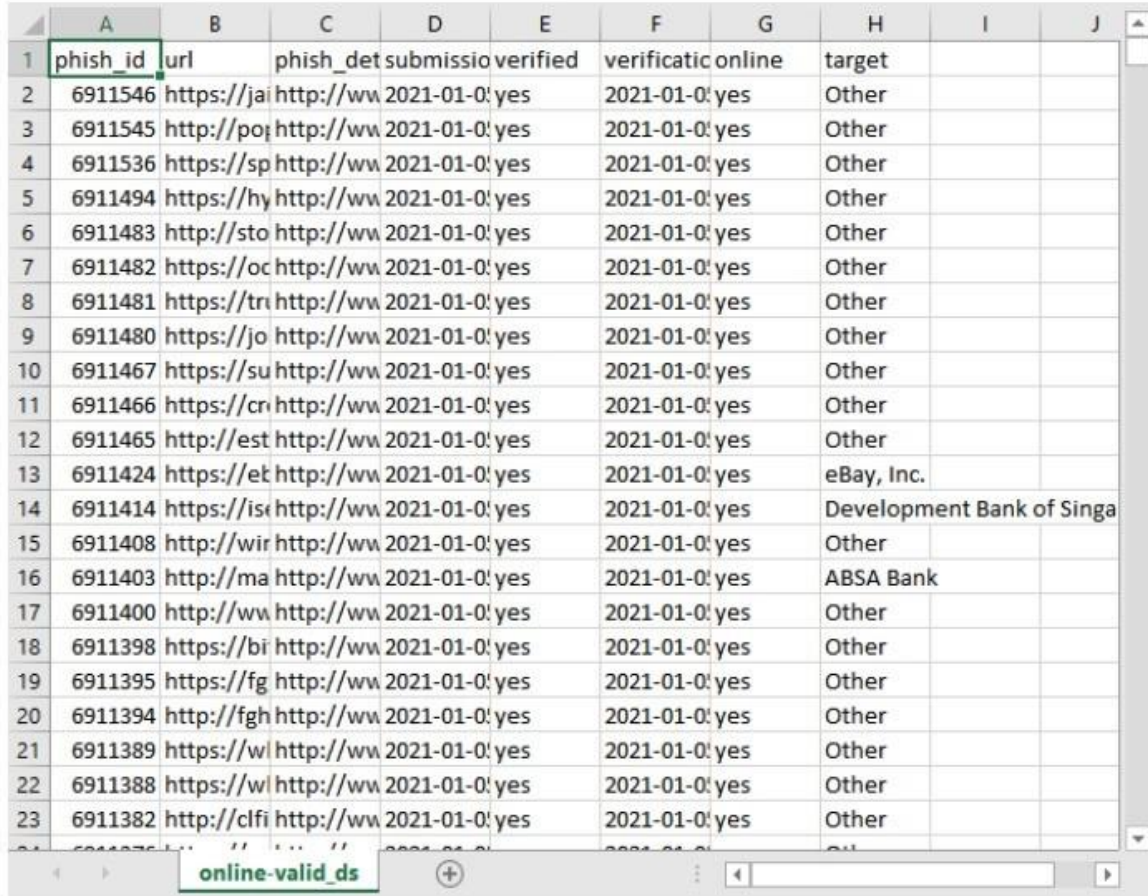


	A	B	C	D	E	F	G	H	I	J
1		http://1337x.to/torrent/1048648/American-Sniper-2014-MD-iTALiAN-DVDSCR-X264-BST-MT/								
2		http://1337x.to/torrent/1110018/Blackhat-2015-RUSSIAN-720p-WEB-DL-DD5-1-H264-RUFGT/								
3		http://1337x.to/torrent/1122940/Blackhat-2015-x264-1080p-WEB-DL-eng-nl-sub-s-sharky/								
4		http://1337x.to/torrent/1124395/Fast-and-Furious-7-2015-HD-TS-XVID-AC3-HQ-Hive-CM8/								
5		http://1337x.to/torrent/1145504/Avengers-Age-of-Ultron-2015-CAM-New-Audio-x264-CPG/								
6		http://1337x.to/torrent/1160078/Avengers-age-of-Ultron-2015-HQ-CAM-H264-AC3-MURD3R/								
7		http://1337x.to/torrent/294349/American-Idol-S11E04-Auditions-4-HDTV-XviD-FQM-ettv/								
8		http://189.cn/dqmh/userCenter/myOrderInfoList.do?method=listMyOrderInfo_new&isVs=no								
9		http://2gis.ru/moscow/search/%D0%9F%D0%BE%D0%B5%D1%81%D1%82%D1%8C/tab/firms/zoom/11								
10		http://abc.go.com/shows/general-hospital/episode-guide/2015-05/08-friday-may-8-2015								
11		http://abc.go.com/shows/the-muppets/video/new-abc-comedy-trailers?cid=abchp_muppets								
12		http://abcnews.go.com/US/wireStory/regulators-delays-georgia-nuclear-plant-31020059								
13		http://adultfriendfinder.com/css/live_cd/ffadult/english/0/font_face-1427390957.css								
14		http://akhbarelyom.com/news/newdetails/410322/1/%D8%A8%D9%88%D8%B6%D9%88%D8%AD.html								
15		http://allegro.pl/amadeus-quartet-haydn-string-quartets-collectors-i5207998383.html								
16		http://allegro.pl/narzedzia-i-sprzet-warsztatowy-18554?ref=simplified-category-tree								
17		http://allegro.pl/royal-string-quartet-music-for-string-quartet-cd-i5262876831.html								
18		http://allegro.pl/sexy-g-string-stringi-z-koralikami-must-have-uni-i5039087215.html								
19		http://allegro.pl/sporty-strzeleckie-i-myslistwo-13495?ref=simplified-category-tree								
20		http://allegro.pl/sporty-towarzyskie-i-rekreacja-13408?ref=simplified-category-tree								
21		http://allegro.pl/triumph-miss-sexy-allday-string-stringi-36-s-bez-i4855636396.html								
22		http://allegro.pl/triumph-stringi-exquisite-essence-string-stal-38-i5073330901.html								
23		http://allegro.pl/wyszczuplajace-body-string-pod-biust-jony-srebra-i5124700240.html								

Figure 4.1.1 Dataset of Phishing URLs

4.1.2 Legitimate URLs:

Collected from the University of New Brunswick dataset bank, which includes collections of benign (legitimate), spam, phishing, malware, and defacement URLs.



	A	B	C	D	E	F	G	H	I	J
1	phish_id	url	phish_det	submitted	verified	verification	online	target		
2	6911546	https://ja	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
3	6911545	http://po	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
4	6911536	https://sp	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
5	6911494	https://hy	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
6	6911483	http://sto	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
7	6911482	https://oc	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
8	6911481	https://tr	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
9	6911480	https://jo	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
10	6911467	https://su	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
11	6911466	https://cr	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
12	6911465	http://est	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
13	6911424	https://ek	http://ww	2021-01-01	yes	2021-01-01	yes	eBay, Inc.		
14	6911414	https://is	http://ww	2021-01-01	yes	2021-01-01	yes	Development Bank of Singa		
15	6911408	http://wir	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
16	6911403	http://ma	http://ww	2021-01-01	yes	2021-01-01	yes	ABSA Bank		
17	6911400	http://ww	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
18	6911398	https://bi	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
19	6911395	https://fg	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
20	6911394	http://fgh	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
21	6911389	https://w	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
22	6911388	https://w	http://ww	2021-01-01	yes	2021-01-01	yes	Other		
23	6911382	http://clfi	http://ww	2021-01-01	yes	2021-01-01	yes	Other		

Figure 4.1.2 Datasets of Legitimate URLs

The combined dataset included over 5,000 phishing URLs and 5,000 legitimate URLs, ensuring a balanced dataset which is essential to avoid bias.

4.2 Data Preprocessing

Real-world data often contains inconsistencies or missing values, which can reduce model performance if not handled properly. The project conducted cleaning of the dataset to remove anomalies and missing data.

```
[25]: tuna.describe()
```

	Have_IP	Have_At	URL_Length	URL_Depth	Redirection	https_Domain	TinyURL	Prefix/Suffix	DNS_Record	Web_Traffic	Domain_Age	Do
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10
mean	0.005500	0.022600	0.773400	3.072000	0.013500	0.000200	0.090300	0.093200	0.100800	0.845700	0.413700	
std	0.073961	0.148632	0.418653	2.128631	0.115408	0.014141	0.286625	0.290727	0.301079	0.361254	0.492521	
min	0.000000	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	0.000000	
50%	0.000000	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	
75%	0.000000	0.000000	1.000000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	
max	1.000000	1.000000	1.000000	20.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Figure 4.2.1 Summary of the dataset

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

[29]: 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 4.2.2 Number of missing values in the dataset

4.2.1 Data encoding:

Used One-Hot Encoding to convert categorical data (like some extracted features) into numerical format. This step was critical for compatibility with ML algorithms that can only handle numerical inputs. Ensured there was no data imbalance between phishing and legitimate samples, keeping the dataset robust for binary classification.

4.3 Exploratory Data Analysis (EDA)

EDA was performed to better understand the data distribution and identify hidden patterns or correlations:

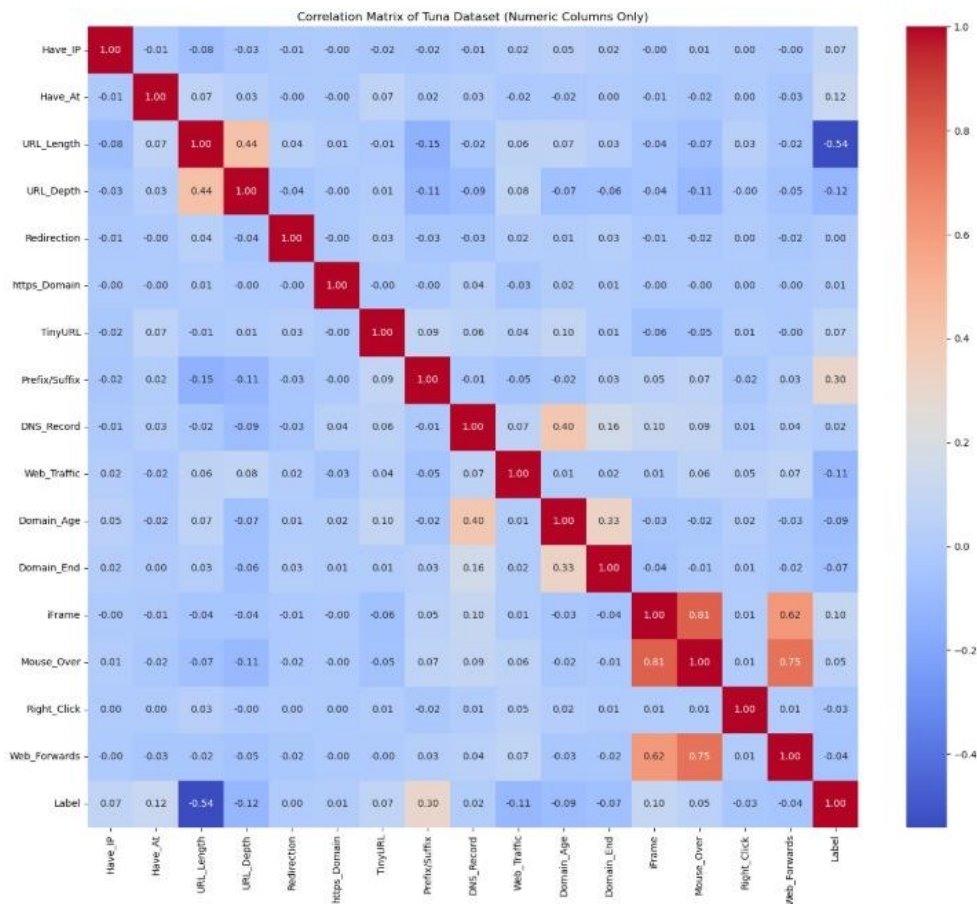


Figure 4.3 heatmap

Generated distribution plots to visualize how phishing and legitimate URLs differ based on extracted features. Created correlation heatmaps to observe relationships between different features and their contribution to the target variable. Analyzed missing data and

confirmed it was properly addressed. This step ensured that features contributing most to predictions were identified, and redundant or noisy features were considered for removal.

4.4 Feature Extraction

The quality of features significantly impacts model accuracy. The study extracted 17 features grouped into three major categories:

4.4.1 Address Bar-Based Features:

- a. Use of IP addresses in URLs
- b. Length of the URL (long URLs often hide malicious parts)
- c. Use of URL shortening services
- d. Presence of “@” symbols
- e. Presence of “//” redirection
- f. Prefix/suffix separated by ‘-’ in domain
- g. Subdomain count
- h. HTTPS usage and certificate verification
- i. Domain registration length


```

[20]: # 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

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

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

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

[29]: # 5.Gives number of '/' in URL (URL_Depth)
def getDepth(url):
    s = urlparse(url).path.split('/')
    depth = 0
    for j in range(len(s)):
        if len(s[j]) != 0:
            depth = depth+1
    return depth

[32]: # 6.Checking for redirection '//' in the url (Redirection)
def redirection(url):
    pos = url.rfind('//')
    if pos > 6:
        if pos > 7:
            return 1
        else:
            return 0
    else:
        return 0

```

Figure 4.4.1 Code for Address bar based feature extraction

4.4.2 Domain-Based Features:

- a. Age of domain (younger domains tend to be malicious)
- b. DNS records presence
- c. Website traffic rank (using Alexa database)
- d. PageRank score

```

42]: !pip install python-whois

Requirement already satisfied: python-whois in c:\users\nitro\anaconda3\lib\site-packages (0.9.5)
Requirement already satisfied: python-dateutil in c:\users\nitro\anaconda3\lib\site-packages (from python-whois) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\nitro\anaconda3\lib\site-packages (from python-dateutil->python-whois) (1.16.0)

43]: # importing required packages for this section
import re
from bs4 import BeautifulSoup
import whois
import urllib
import urllib.request
from datetime import datetime

44]: # 11.DNS Record availability (DNS_Record)
# obtained in the featureExtraction function itself

45]: # 12.Web traffic (Web_Traffic)
def web_traffic(url):
    try:
        #Filling the whitespaces in the URL if any
        url = urllib.parse.quote(url)
        rank = BeautifulSoup(urllib.request.urlopen("http://data.alex.com/data?cli=10&dat=s&url=" + url).read(), "xml").find(
            "REACH")['RANK']
        rank = int(rank)
    except TypeError:
        return 1
    if rank < 100000:
        return 1
    else:
        return 0

46]: # 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

```

4.4.3 HTML & JavaScript-Based Features:

- a. Presence of iFrames
- b. Use of pop-up windows
- c. Disabling right-click via JavaScript
- d. Status bar customization to mislead users

The extracted features are based on previously established research (e.g., Mohammad et al., Rishikesh & Irfan) and were chosen because they have shown high effectiveness in distinguishing phishing sites from legitimate ones.

```
[48]: # importing required packages for this section for html and java extraction
import requests
```

```
[49]: # 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
```

```
[50]: # 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
```

```
[51]: # 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
```

```
[52]: # 18.Checks the number of forwardings (Web_Forwards)
def forwarding(response):
    if response == "":
        return 1
    else:
        if len(response.history) <= 2:
            return 0
        else:
            return 1
```

Figure 4.4.3 Code for Address bar based feature extraction


```

•[53]: #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)
    # Fix for web_traffic function that was causing the error
    try:
        traffic_rank = web_traffic(url)
        features.append(traffic_rank)
    except Exception as e:
        print(f"Error in web_traffic: {str(e)}")
        features.append(-1) # Use -1 as a default value when web_traffic fails

    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, timeout=10) # Added timeout parameter
    except Exception as e:
        print(f"Error fetching URL {url}: {str(e)}")
        response = ""

    features.append(iframe(response))
    features.append(mouseOver(response))
    features.append(rightClick(response))
    features.append(forwarding(response))
    return features

# Test the function with try-except to see any errors
try:
    result = featureExtractions('http://www.facebook.com/home/service')
    print(result)
except Exception as e:
    print(f"Error in featureExtractions: {str(e)}")

```

Error in web_traffic: <urlopen error [Errno 11001] getaddrinfo failed>
['facebook.com', 0, 0, 0, 2, 0, 0, 0, 0, -1, 0, 1, 0, 0, 1, 0]

Figure 4.4.4 Code computation for all the feature extraction used dataset.

4.5 Model Training

To predict if a URL is phishing or legitimate, the study used a range of supervised ML algorithms and deep learning models:

4.5.1 Decision Tree:

Simple, interpretable, but prone to overfitting.

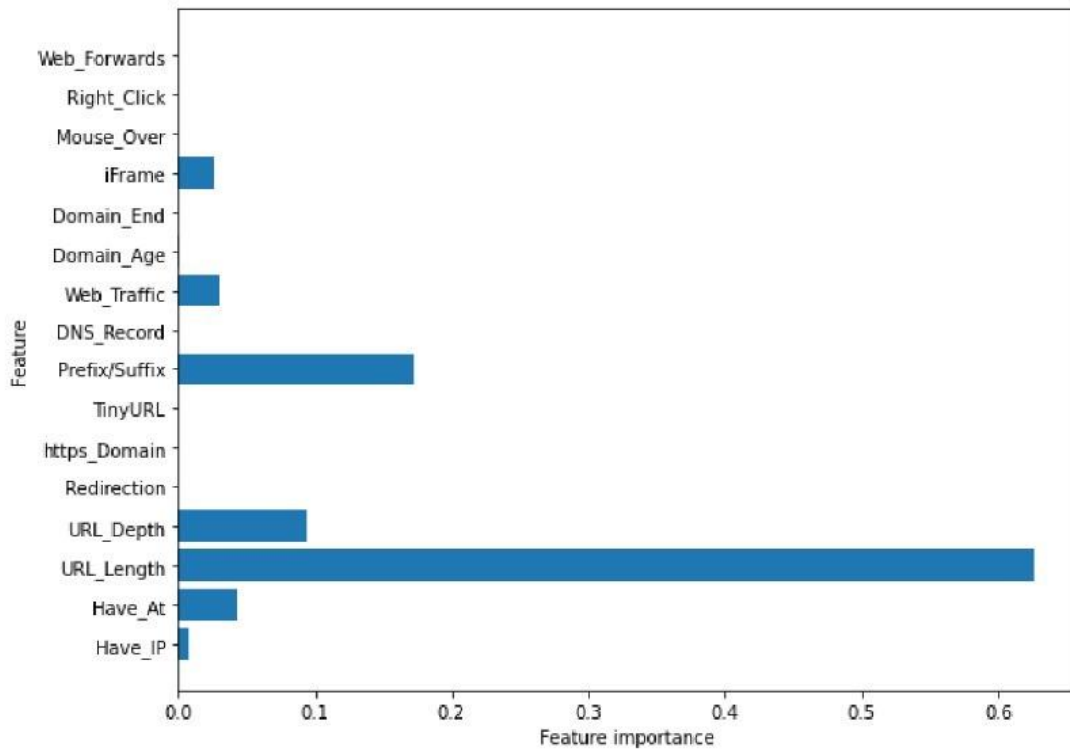


Figure 4.5.1 Feature importance for Decision Tree classifier

4.5.2 Random Forest:

An ensemble of decision trees, offering better generalization.

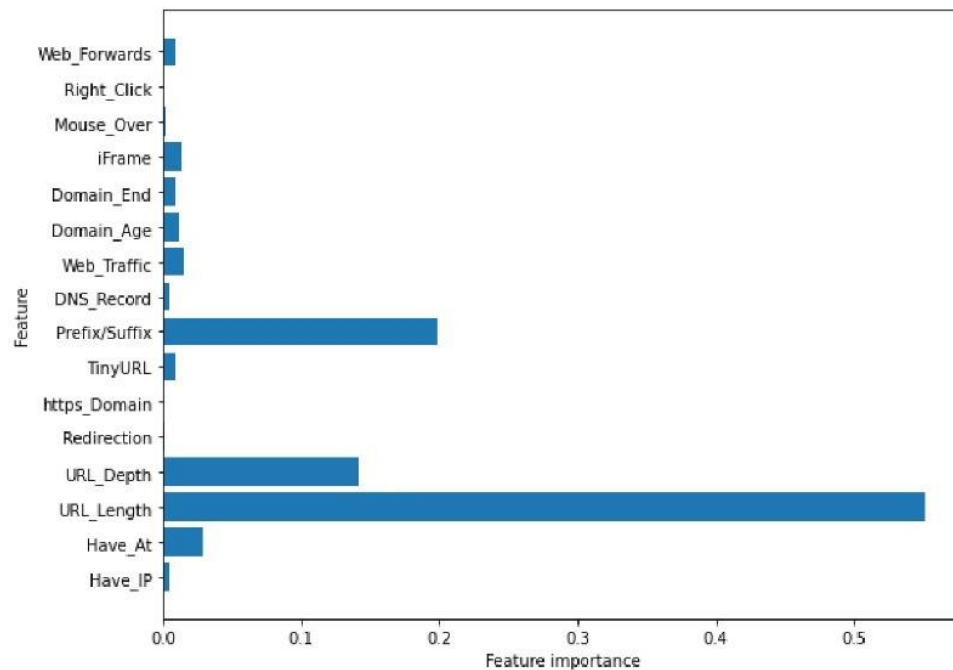


Figure 4.5.2 Feature importance for Random forest classifier

4.5.3 Support Vector Machine (SVM):

Good for high-dimensional data and handling non-linear relationships.

4.5.4 XGBoost:

A gradient boosting framework known for speed and accuracy.

4.5.5 Multilayer Perceptron (MLP):

A type of feedforward neural network.

4.5.6 Autoencoder Neural Network:

Especially useful for capturing non-linear patterns.

The dataset was split into:

- I. 80% for training
- II. 20% for testing

This split allowed the models to learn from a large subset and then evaluate their performance on unseen data.

4.6 Model Evaluation

After training, each model was evaluated based on:

- a. Accuracy: Percentage of correct predictions.
- b. False Positive Rate (FPR): Legitimate URLs incorrectly classified as phishing.
- c. False Negative Rate (FNR): Phishing URLs incorrectly classified as legitimate.

These metrics help understand not just overall performance but also how models behave under security-sensitive scenarios (where false negatives can be especially dangerous). The evaluation used tools from the scikit-learn library in Python.

4.7 Deployment

To ensure the model was usable by non-technical users:

- a. Developed a web application that allows users to input a URL.
- b. The chosen best-performing model was wrapped as an API (Application Programming Interface).
- c. The API was integrated into the web app, enabling it to:
- d. Accept user input (URL)
- e. Process the input with the model
- f. Return predictions (phishing or legitimate)

The deployment made the system accessible across multiple platforms, including mobile, tablets, and desktop browsers. The project also tested the API using Postman, confirming it could successfully process requests and return predictions.

4.8 Discussion of the Results:

After implementing and evaluating all models, the study observed the following:

- a. XGBoost achieved the highest detection accuracy:
- b. Reported accuracy: around 86.6%, making it the top performer among all tested models.
- c. Other models like Random Forest, SVM, and MLP also performed well but slightly below XGBoost.

4.9 Feature importance:

Plots for Decision Tree and Random Forest showed which features contributed most:

- Presence of IP address
- Length of the URL
- Use of HTTPS and trusted certificate issuer
- Domain age
- Presence of iFrames and pop-ups

4.10 Impact of training data size:

As more data was used for training (e.g., 90% instead of 70%), accuracy improved. This observation aligns with previous studies (e.g., Rishikesh & Irfan, 2018) that found Random Forest performed best with ~97% accuracy when trained on larger datasets.

4.11 Comparison with literature:

The Random Forest model in this study had competitive results. The XGBoost model, known for superior handling of imbalanced data and non-linear relationships, performed slightly better in this dataset.

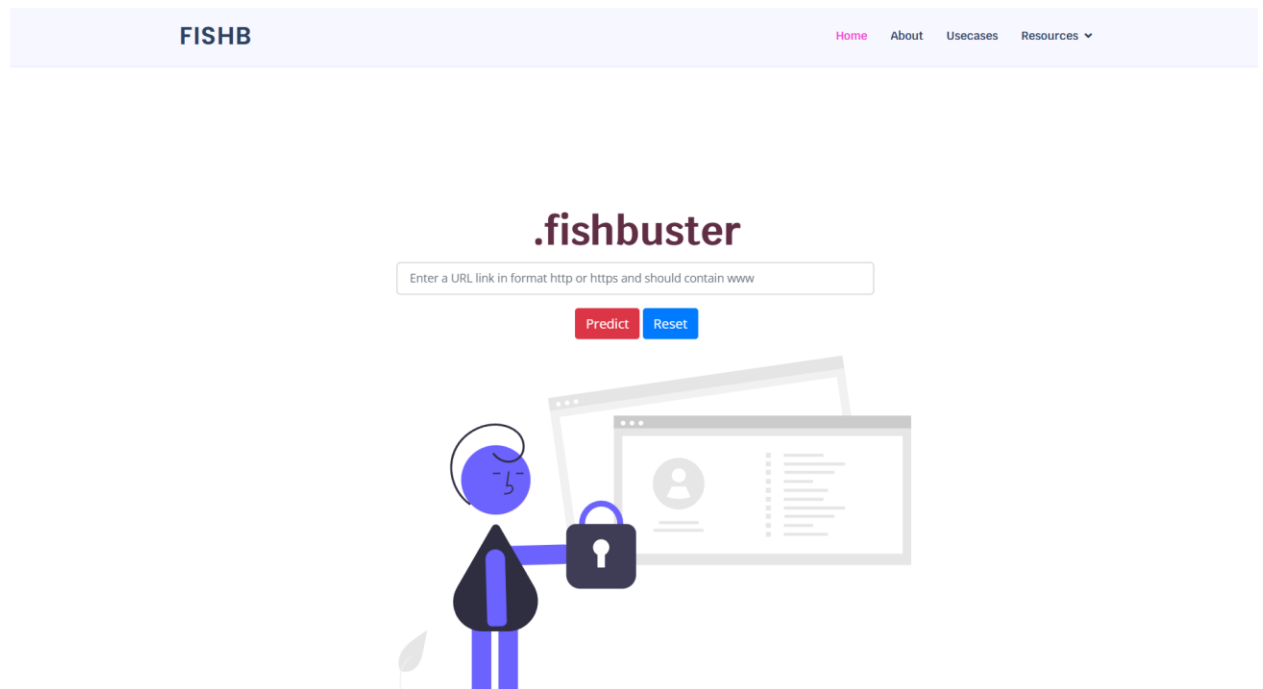


Figure 4.11.0.1 Frontend demo

	ML Model	Train Accuracy	Test Accuracy
4	XGBoost	0.868	0.861
3	Multilayer Perceptrons	0.863	0.857
5	AutoEncoder	0.847	0.834
1	Random Forest	0.819	0.826
2	Random Forest	0.819	0.826
0	Decision Tree	0.813	0.816
6	SVM	0.801	0.806

Figure 4.11.2 Model train vs test accuracy

Django administration
WELCOME, SUSHIL [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups
+ Add
Change

Users
+ Add
Change

Recent actions
My actions
None available

Figure 4.11.3 Django admin panel

CHAPTER 5: CONCLUSION

5.1 Summary

In this project, we successfully developed a web application that detects phishing URLs using machine learning techniques. The application analyzes various features extracted from the URLs such as domain properties, URL structure, and other heuristic-based indicators to predict whether a given URL is legitimate or malicious. By training and evaluating multiple classification algorithms, we identified the best-performing model and integrated it into a user-friendly web interface. Our results demonstrate that machine learning can be an effective tool in detecting phishing attacks with a high degree of accuracy. This contributes to improving user safety and mitigating the risks associated with cyber threats.

The **Phishing Detection of URLs using Machine Learning** project presents an intelligent, ML-powered web application that addresses the growing need for real-time, automated cybersecurity solutions in an increasingly digital world. Leveraging classification algorithms and heuristic-based URL feature extraction, the system enables end-users to assess the legitimacy of web links with a single input. This tool simplifies threat detection, especially for non-technical users, by offering a browser-accessible interface that flags malicious URLs before any personal or financial data is compromised. This system converts raw URL input into structured, model-ready features which then used to train and deploy a classifier that identifies phishing threats with high accuracy.

The success of the project lies in its achievements:

- Extraction of relevant URL features
- Training and evaluation of ML Models
- Development of simple and understandable web interface
- Deployment of best performing ML model into real time web system

5.2 Limitations

Despite its functional completeness and promising accuracy, our project faces several limitations that impact its performance and scalability. These constraints provide valuable insights for future improvements and align with challenges noted in related research.

1. **Static feature dependency:** The system only considers static features of URLs (e.g., length, presence of suspicious words, domain age). It does not assess the actual content or behavior of the website.
2. **Adaptive Attacks:** Sophisticated phishing methods that mimic legitimate domains or dynamically alter their behavior can evade detection, especially if such patterns are not present in the training dataset.
3. **Dataset Generalizability:** The ML model's performance is dependent on the quality and diversity of the training data. Publicly available phishing datasets may not reflect current or regional attack trends.
4. **Lack of Threat Intelligence Integration:** The model operates in isolation without utilizing external threat databases or real-time blacklist APIs, limiting its adaptability to emerging threats.
5. **Scalability and Performance:** While functional in a demo or limited deployment, the current architecture may face challenges when scaling to enterprise-level traffic or multiple concurrent users.

5.3 Future Enhancement

To address the identified limitations and build on the gaps noted in the literature review, several future enhancements are proposed to improve project's functionality, scalability, and user experience. These enhancements align with the project's objectives and leverage insights from prior research.

1. **Dynamic Content Analysis:** Integrate tools like headless browsers or sandboxed environments to render pages and analyze on-page scripts, visual elements, and login forms
2. **Browser Extension Development:** Create a Chrome/Firefox extension that evaluates URLs in real time as users navigate, providing passive protection without needing to visit the web app.
3. **Incorporate Threat Intelligence:** Incorporate APIs from known threat intelligence services (e.g., VirusTotal, PhishTank) for cross-verification process and improve accuracy with external validation.
4. **Mobile App Integration:** Design and launch a lightweight mobile app for phishing detection on Android/iOS, with offline analysis capabilities for SMS or email links.
5. **User Feedback and Explainability:** Incorporating a user feedback mechanism would allow the system to learn from incorrect predictions and adjust its model behavior. Coupling this with Explainable AI (XAI) methods such as SHAP or LIME could help users understand why a URL was classified as phishing, increasing user trust and system transparency.

These enhancements not only address the current limitations of the system but also position it for broader, real-world adoption. By incorporating dynamic analysis, real-time intelligence and scalable infrastructure, the system can evolve into a comprehensive solution for phishing detection

REFERENCES

- [1] P. Ayush, “Workflow of a Machine Learning project,” *Towards Data Science*, 2019. [Online]. Available: <https://towardsdatascience.com/>
- [2] DeepAI, “Feature Extraction Definition,” *DeepAI*. [Online]. Available: <https://deepai.org/machine-learning-glossary-and-terms/feature-extraction>
- [3] Imperva, “Phishing attacks: Attack techniques & scam examples,” *Imperva*, 2021. [Online]. Available: <https://www.imperva.com/learn/application-security/phishing-attack-scam/>
- [4] B. Noel, “Support Vector Machines: A Simple Explanation,” *KDnuggets*, 2016. [Online]. Available: <https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html>
- [5] M. Rishikesh and S. Irfan, “Phishing Website Detection using Machine Learning Algorithms,” *Int. J. Comput. Appl.*, vol. 23, pp. 45–47, 2018. doi: 10.5120/ijca2018918026
- [6] HoxHunt, “Phishing Trends Report 2025,” [Online]. Available: <https://hoxhunt.com/guide/phishing-trends-report>
- [7] ResearchGate, “Worldwide financial losses in billion due to phishing attack,” [Online]. Available: https://www.researchgate.net/figure/Worldwide-financial-losses-in-billion-due-to-phishing-attack_fig1_312205924
- [8] Verizon, “2024 Data Breach Investigations Report,” *Verizon Business*, 2024. [Online]. Available: <https://www.verizon.com/business/resources/reports/dbir/>
- [9] Comcast, “Global Cybersecurity Technology Advancements Accelerate,” *Comcast Business Cybersecurity Threat Report*. [Online]. Available: <https://corporate.comcast.com/press/releases/comcast-business-report-global-cybersecurity-technology-advancements-accelerate>
- [10] SlashNext, “Mid-Year State of Phishing Report,” *SlashNext*, 2024. [Online]. Available: <https://slashnext.com/press-release/slashnext-mid-year-state-of-phishing-report-shows-341-increase-in-bec-and-advanced-phishing-attacks/>

[11] IBM, “Cost of a Data Breach Report 2024,” *IBM Security*. [Online]. Available: <https://www.ibm.com/reports/data-breach>