PHISHING DETECTION BROWSER EXTENSION

NGONIDZAISHE KUWODZA CHIRUME

THABO INNOCENT MOHLALA

MOREBLESSING SANDI TSHUMA

A project report submitted in partial
fulfilment of the requirements for the award of
Diploma Programme

University Malaysia of Computer Science and Engineering

May 2025

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other diploma or award at University Malaysia of Computer Science and Engineering or other institutions.

Signature	**Name**	**ID No.**	**Programme**	**Date**
NKC	Ngonidzaishe Kuwodza Chirume	2022_4534_2920	Dip in IT (CYBERSECURITY)	12/05/2025
M.Thabo	Mohlala Thabo Innocent	2022_3752_2891	Dip in IT (CYBERSECURITY)	12/05/2025
MST	Moreblessing Sandi Tshuma	2022 3418 2889	Dip in IT (CYBERSECURITY) (DCY7)	12/05/2025

APPROVAL FOR SUBMISSION

I certify that this project report entitled **PHISHING DETECTION BROWSER EXTENSION** was prepared by my supervisee has met the required standard for submission in partial fulfilment of the requirements for the award of Diploma programme at University Malaysia of Computer Science and Engineering.

Approved by,

Signature	:	Zoubeir Aungnoo
Supervisor	:	Aungnoo Zoubeir
Date	:	15-May-2025

ACKNOWLEDGEMENTS

We would like to thank everyone who contributed to the successful completion of this project. Special thanks goes to our supervisor (Mr Zoubeir) for his guidance, feedback, and support throughout the development of the project.

We are also grateful to our families and friends for their encouragement and understanding during this period.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI – Artificial Intelligence

ML – Machine Learning

IDS – Intrusion Detection Systems

NLP – Natural Language Processing

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

API – Application Programming Interface

UCI – University of California, Irvine (Machine Learning Repository)

SVM – Support Vector Machine

KNN – K-Nearest Neighbors

MV3 – Manifest Version 3 (Chrome Extensions API)

SMEs – Small and Medium Enterprises

LIST OF APPENDICES

ABSTRACT

Phishing, which targets users on websites, emails, and social media, is still one of the most common and destructive types of cyberattacks in the world. It takes advantage of people's trust. In order to offer real-time protection against malicious links, this project presents the design, **development, and testing of a phishing detection browser extension driven by machine learning.** The extension is designed to actively track user browsing behavior, examine URL structures and content, and use a trained model to categorize web pages as safe or phishing.

To increase regional relevance, a filtered African subset was added to a rich dataset from the **UCI Machine Learning Repository** to create a **Random Forest classifier**. **Python, Flask, Scikit-learn, BeautifulSoup, JavaScript, HTML/CSS**, and the **Chrome Extensions API** are some of the important technologies and tools that are integrated into the system. For smooth integration and real-time prediction in browser environments, the extension uses a RESTful API to communicate with a Flask-based backend that serves the trained model.

A number of **technical difficulties** arose during implementation. Most significantly, in order to achieve accurate detection in real-world settings, the extension had to get around browser-enforced **web security policies (CORS),** which blocked it from freely interacting with the local Flask server. To solve this, we decided to temporarily turn off browser web security while testing in order to confirm and enhance the model's performance in real-world scenarios. Additionally, because of the limitations of real-time HTML parsing and URL structure anomalies, we saw mismatched or inconsistent predictions for some URLs. This suggests that future iterations should improve dataset quality and enhance content-based feature extraction.

Despite these difficulties, the finished system offers a convincing method of real-time phishing detection and has a lot of potential for wider implementation, particularly in institutional and corporate settings. This work establishes the groundwork for future developments like enterprise-level deployment, crowdsourced threat feedback, and integration of **Natural Language Processing (NLP)**, which will make this tool a scalable and intelligent defense against phishing.

CHAPTER 1

INTRODUCTION

1.1 Project Background

Phishing attacks have become one of the most significant cybersecurity threats, responsible for over 36% of global data breaches *(According to the 2022 Verizon Data Breach Investigations Report)*. These attacks manipulate human trust by hiding as legitimate entities to steal sensitive information, such as login credentials, financial data, and personal details.

Even with improvements in security solutions, conventional phishing detection techniques like heuristic-based filtering and blacklists are unable to keep up with increasingly complex and dynamic attacks. Attackers commonly alter email content, URLs, and email structures to get around established security measures.

To address this gap, our capstone project focuses on developing a machine learning, powered browser extension that provides real-time phishing detection while users browse the internet. Unlike conventional detection methods, our solution dynamically analyzes URLs, domain reputation, and content-based indicators to proactively identify and block phishing attempts.

1.2 Problem statement

Traditional phishing detection methods rely on:

- Static blacklists, which fail against new and zero-day phishing sites.
- Manual user reporting, which is slow and inefficient.
- Rule-based filters, which struggle against advanced social engineering techniques.

These approaches lack adaptability, allowing phishing attacks to bypass detection mechanisms. Given that phishing attacks evolve daily, there is a need for an intelligent, self-learning solution that can detect and mitigate phishing threats in real time.

1.3 Review on Existing Project/System

This review aims to explore current or existing techniques, their functionalities, advantages and limitations, this will help in determining why the proposed solution is a need or in demand. The following techniques are currently in the market:

Table 1.1 : Comparison of Phishing Detection Techniques

Detection Technique	**Description**	**Limitations**
Blacklist-Based Detection	Relies on databases of known phishing websites maintained by security providers (e.g., Google Safe Browsing, Microsoft SmartScreen). URLs are compared to these lists and users are warned accordingly.	- Ineffective against new or zero-day phishing sites - Attackers frequently change URLs to bypass detection - Cannot detect unknown threats
Heuristic-Based Detection	Analyzes characteristics of web pages such as URL structure, SSL certificate, login forms, and suspicious scripts (HTML/JavaScript) to identify potentially malicious behavior.	- Tends to generate false positives, flagging safe sites as malicious
Machine Learning-Based Detection	Uses trained models to identify phishing based on patterns in domain age, reputation, HTML layout, and user interaction behaviors. Continuously improves as new data is added.	- Requires a large volume of labeled data for accurate training

1.4 Objectives

The main objective of this project is to create a browser extension that can:

- Collect and analyze phishing and safe website data for training the model.
- Build and optimize a Machine Learning model to classify phishing threats.
- Develop a user-friendly browser extension that provides real-time alerts.
- Continuously improve accuracy by updating the model with new data.

1.5 Scope

The system covers:

- Supervised ML model training (Random Forest)
- Flask API for real-time inference
- Chrome extension for user alerts
- Compatibility with Chrome, Edge, Firefox

Excludes:

- Mobile support
- Detection of smishing or voice phishing

The primary users of this solution include:

- **Businesses (SMEs and Enterprises)** seeking to improve employee security awareness.
- **Educational institutions**, including schools and universities.
- **General internet users** who frequently browse the web and interact with email links.

CHAPTER 2

ANALYSIS AND DESIGN

2.1 Literature Review

Introduction to Phishing Attacks

Phishing is a cyber-attack method in which attackers impersonate legitimate entities to deceive users into providing sensitive information such as login credentials, financial details, or personal data. Phishing attacks typically exploit human psychology, leveraging urgency, fear, or curiosity to manipulate victims *(Verizon, 2023)*

Types of Phishing Attacks

- **Email Phishing** – Attackers send fake emails mimicking trusted organizations (e.g., banks, government agencies) to trick users into clicking malicious links. Example: A user receives an email claiming to be from PayPal, asking them to verify their account by clicking a link leading to a fake login *page (Google and Stanford University ,2023).*

- **Website Phishing** – Cybercriminals create fake websites that resemble legitimate ones to trick users into entering credentials. Example: A fake Facebook login page looks identical to the real one, capturing user credentials when they attempt to log in *page (Google and Stanford University ,2023).*

Impact of Phishing Attacks in Africa as a Region

Phishing attacks in Africa have increased due to fast digital growth, lack of cybersecurity awareness, and limited protection tools. According to recent reports, ***Africa*** has one of the highest rates of unprotected internet users, making phishing easier for cybercriminals *(ITU,2023; Interpol,2023)*

In 2023, ***over 60%*** of cyberattacks in Africa involved phishing tactics, mostly through fake emails and websites.

South Africa, Nigeria, and Kenya are among the ***top 10 most attacked countries*** by phishing in Africa *(Kaspersky,2023).*

Challenges in Combating Phishing Attacks in Africa

Factors that contribute to the challenges in addressing phishing in Africa:

- **Rapid Digital Adoption:** The swift integration of digital technologies without corresponding cybersecurity measures has created vulnerabilities *(ITU,2023).*
- **Lack of Awareness:** Limited cybersecurity awareness among users makes them susceptible to phishing tactics.
- **Resource Constraints:** Many organizations lack the necessary resources and expertise to implement robust cybersecurity defenses *(Africa Cybersecurity Report,2023)*

Statistics Globally

According to the Verizon 2023 Data Breach Investigations Report, 36% of data breaches involve phishing attacks. A Google and Stanford University study found that machine-learning models could detect phishing websites with an accuracy of over 90%. These statistics highlight the critical need for robust phishing detection mechanisms, such as browser-based security tools.

2.2 AI in Cybersecurity

AI in Cybersecurity and how it is used

In terms of Cybersecurity, Artificial Intelligence refers to the use of machine learning (ML), deep learning, and automation to detect, prevent, and respond to cyber threats more effectively than traditional methods. AI enables security systems to analyze large volumes of data, identify patterns, detect anomalies, and automate responses to attacks in real time

Table 2.1 : Applications of AI in Cybersecurity

Application Area	**How AI is Used**	**Example**	**Tools/Platforms**
Threat Detection & Anomaly Identification	- AI monitors network traffic, user behavior, and system activity to detect unusual patterns. - ML models distinguish between normal and suspicious activity.	AI-powered Intrusion Detection Systems (IDS) detect unauthorized access attempts.	CrowdStrike Falcon, Darktrace
Phishing Detection & Email Security	- AI scans emails, URLs, attachments, and sender behavior to identify phishing attempts. - NLP helps detect deceptive emails even with altered wording.	Google's AI filters 99.9% of phishing emails in Gmail.	Microsoft Defender for Office 365, Tessian, Avanan

Vulnerability Management & Patch Prediction	- AI scans systems and applications to uncover vulnerabilities. - Predicts which vulnerabilities are most likely to be exploited.	Microsoft Defender AI helps prioritize patching of critical threats.	Qualys, Rapid7 InsightVM

Table 2.2: Types of Machine Learning

Type of Learning	**How It Works**	**Examples**	**Use Cases**
Supervised Learning	Learns from labeled data. Maps input (X) to known output (Y) using training datasets.	- Spam Detection - Fraud Detection	Email classification, financial fraud detection
Unsupervised Learning	Learns from unlabeled data by finding hidden patterns or clusters. Detects anomalies autonomously.	- Customer Segmentation - Anomaly Detection	Marketing analytics, banking fraud monitoring
Semi-Supervised Learning	Trained on a small labeled dataset and a larger unlabeled dataset. Uses labeled data to guide learning.	Google Photos face grouping with few labeled images	Image recognition, document classification

Reinforcement Learning	Learns by interacting with the environment. Uses rewards and penalties to find optimal actions.	- Autonomous Driving - Robotics Task Training	Self-driving cars, industrial robotics

2.4 What type of ML was used for this project and why?

For this project, **supervised learning** was selected as the most suitable machine learning approach. This method was chosen because the available dataset consisted of clearly labeled examples, with URLs classified as either phishing or safe. Supervised learning enables the model to learn from these predefined categories, which helps achieve higher accuracy in identifying threats. This approach is particularly effective for real-time phishing detection, as it allows the system to make quick and informed decisions when users interact with links through email or web browsing. By using supervised learning, the browser extension can reliably classify malicious links and provide immediate warnings to users. Among the tested algorithms, the **Random Forest classifier** was selected for its strong performance and robustness, achieving an accuracy of approximately 99.9% during training.

2.5 Proposed Solution

We propose a **browser extension-based phishing detection tool** that utilizes supervised machine learning algorithms to classify emails, web links, and browser activity as phishing or legitimate.

Our proposed solution aims to address these challenges by leveraging machine learning models integrated into a browser extension for real-time phishing detection. This extension will analyze URLs, domain reputation, and webpage content to classify whether a site is safe or malicious, without requiring user intervention.

Key Features:

Real-Time Detection: Monitors active browsing sessions and emails to flag phishing attempts.

Key Indicators:

- URL structure and redirection patterns.
- Domain reputation analysis.
- Content inspection for phishing keywords (e.g., urgency, threats).

User Interface

Displays phishing warnings, offers insights into flagged activity, and provides user education.

Cross-Browser Compatibility

Compatible with Chrome, Edge, and Firefox.

How Fast Will It Work?

- Detection time: In seconds
- Works in real time to ensure users are protected before they click.
- Optimized to run efficiently without slowing down browsing.
- Uses a lightweight AI model to provide instant results.

2.6 System Architecture Overview

This is how the ML model, Flask API, and browser extension interact to provide real-time phishing detection

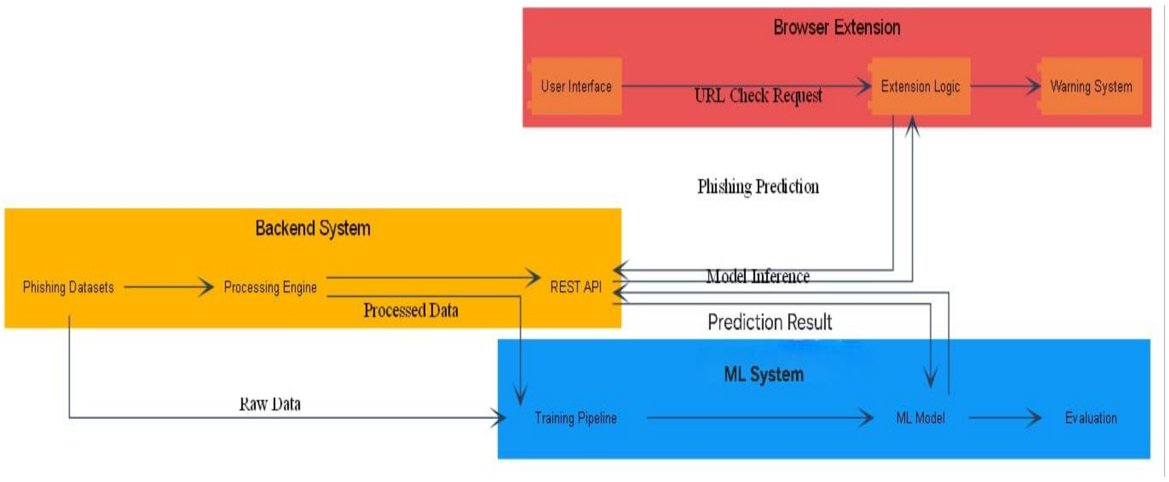
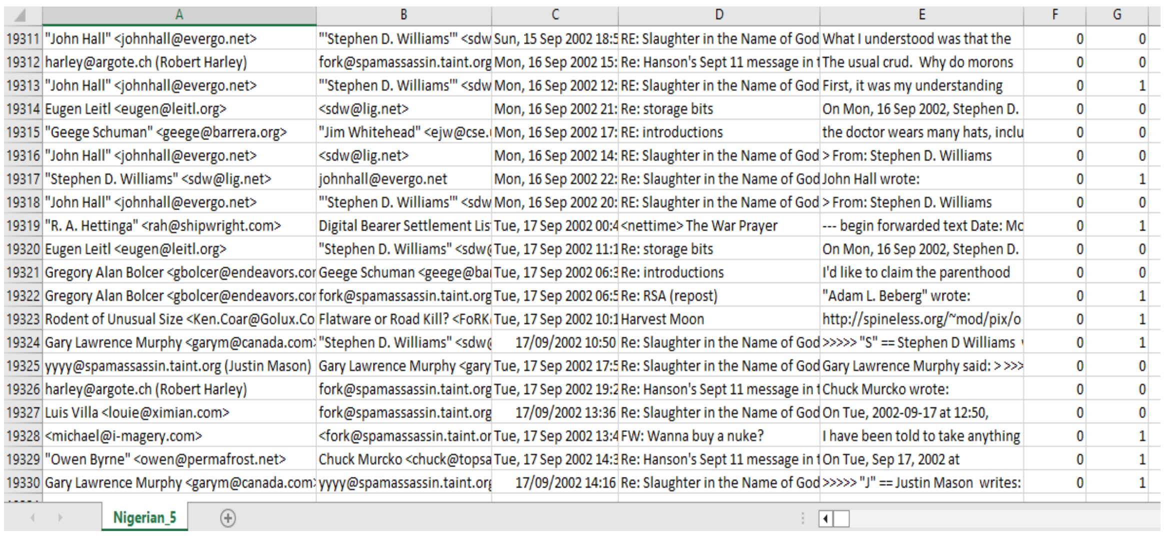


Figure 2.6.1: Architecture diagram

2.7 Raw Dataset Shape Before and After Filtering

This figure shows the initial size of the dataset and the filtered size containing African country code top-level domains (ccTLDs). (*Phishing Emails and URLS from enron database)*



	A	B	C	D	E	F	G
19311	"John Hall" <johnhall@evergo.net>	"'Stephen D. Williams'" <sdw	Sun, 15 Sep 2002 18:5	RE: Slaughter in the Name of God	What I understood was that the	0	0
19312	harley@argote.ch (Robert Harley)	fork@spamassassin.taint.org	Mon, 16 Sep 2002 15:	Re: Hanson's Sept 11 message in t	The usual crud. Why do morons	0	0
19313	"John Hall" <johnhall@evergo.net>	"'Stephen D. Williams'" <sdw	Mon, 16 Sep 2002 12:	RE: Slaughter in the Name of God	First, it was my understanding	0	1
19314	Eugen Leitl <eugen@leitl.org>	<sdw@lig.net>	Mon, 16 Sep 2002 21:	Re: storage bits	On Mon, 16 Sep 2002, Stephen D.	0	0
19315	"Geege Schuman" <geege@barrera.org>	"Jim Whitehead" <ejw@cse.	Mon, 16 Sep 2002 17:	RE: introductions	the doctor wears many hats, inclu	0	0
19316	"John Hall" <johnhall@evergo.net>	<sdw@lig.net>	Mon, 16 Sep 2002 14:	RE: Slaughter in the Name of God	> From: Stephen D. Williams	0	0
19317	"Stephen D. Williams" <sdw@lig.net>	johnhall@evergo.net	Mon, 16 Sep 2002 22:	Re: Slaughter in the Name of God	John Hall wrote:	0	1
19318	"John Hall" <johnhall@evergo.net>	"'Stephen D. Williams'" <sdw	Mon, 16 Sep 2002 20:	RE: Slaughter in the Name of God	> From: Stephen D. Williams	0	0
19319	"R. A. Hettinga" <rah@shipwright.com>	Digital Bearer Settlement Lis	Tue, 17 Sep 2002 00:4	<nettime> The War Prayer	--- begin forwarded text Date: Mc	0	1
19320	Eugen Leitl <eugen@leitl.org>	"Stephen D. Williams" <sdw(	Tue, 17 Sep 2002 11:1	Re: storage bits	On Mon, 16 Sep 2002, Stephen D.	0	0
19321	Gregory Alan Bolcer <gbolcer@endeavors.cor	Geege Schuman <geege@ba	Tue, 17 Sep 2002 06:3	Re: introductions	I'd like to claim the parenthood	0	0
19322	Gregory Alan Bolcer <gbolcer@endeavors.cor	fork@spamassassin.taint.org	Tue, 17 Sep 2002 06:5	Re: RSA (repost)	"Adam L. Beberg" wrote:	0	1
19323	Rodent of Unusual Size <Ken.Coar@Golux.Co	Flatware or Road Kill? <FoRK	Tue, 17 Sep 2002 10:1	Harvest Moon	http://spineless.org/~mod/pix/o	0	1
19324	Gary Lawrence Murphy <garym@canada.com	"Stephen D. Williams" <sdw(	17/09/2002 10:50	Re: Slaughter in the Name of God	>>>>> "S" == Stephen D Williams	0	1
19325	yyyy@spamassassin.taint.org (Justin Mason)	Gary Lawrence Murphy <gary	Tue, 17 Sep 2002 17:5	Re: Slaughter in the Name of God	Gary Lawrence Murphy said: > >>>	0	0
19326	harley@argote.ch (Robert Harley)	fork@spamassassin.taint.org	Tue, 17 Sep 2002 19:2	Re: Hanson's Sept 11 message in t	Chuck Murcko wrote:	0	0
19327	Luis Villa <louie@ximian.com>	fork@spamassassin.taint.org	17/09/2002 13:36	Re: Slaughter in the Name of God	On Tue, 2002-09-17 at 12:50,	0	0
19328	<michael@i-magery.com>	<fork@spamassassin.taint.or	Tue, 17 Sep 2002 13:4	FW: Wanna buy a nuke?	I have been told to take anything	0	1
19329	"Owen Byrne" <owen@permafrost.net>	Chuck Murcko <chuck@topsa	Tue, 17 Sep 2002 14:3	Re: Hanson's Sept 11 message in t	On Tue, Sep 17, 2002 at	0	1
19330	Gary Lawrence Murphy <garym@canada.com	yyyy@spamassassin.taint.or	17/09/2002 14:16	Re: Slaughter in the Name of God	>>>>> "J" == Justin Mason writes:	0	1

Figure 2.7.1: Dataset

CHAPTER 3

IMPLEMENTATION AND TESTING

3.1 Functional and Non-Functional Requirements

3.1.1) Functional Requirements (What It Must Do)

i. **Scans links in emails and websites to detect phishing attempts**

The system continuously scans URLs from websites and emails to detect potential phishing threats.

ii. **Detect phishing in real time**

Phishing identification occurs in real time, ensuring users are alerted instantly during active browsing.

iii. **Hover alert and click warning system**

A visual warning is shown when the user hovers over a suspicious link or attempts to click on it.

iv. **Log all predictions to ml_predictions_log.txt**

All model predictions are logged in a file named `ml_predictions_log.txt` to support traceability and performance analysis

v. **Send URLs to Flask backend for ML inference**

URLs are sent to a Flask backend where the machine learning model performs phishing classification.

3.1.2) Non-Functional Requirements (Performance & Quality)

i. **High Accuracy – At least 95%+ phishing detection rate**

The system aims for a phishing detection accuracy rate of at least 95% to ensure reliable protection.

ii. **Fast Response – Detect phishing links within 1-2 seconds.**

Each URL is analyzed and classified within 1 to 2 seconds to maintain a smooth user experience.

iii. **Cross-Browser Support – Works on Chrome, Firefox, and Edge.**

The extension is compatible with major browsers such as Chrome, Firefox, and Edge.

iv. **Lightweight – Designed to run smoothly without affecting browsing speed.**

The extension is built to run efficiently without consuming excessive system resources or slowing down browsing.

v. **Privacy-Focused – Does not collect or store users' personal data**

User data is not collected, stored, or transmitted, ensuring compliance with privacy standards.

3.2 Technologies Used

3.2.1) Backend & Model Development

Python

Used to handle data preprocessing, cleaning, and formatting of phishing and safe website URLs for model training. In the project, Python scripts were responsible for extracting features such as domain length, special characters in URLs, and other indicators of phishing.

Scikit-learn / TensorFlow

These frameworks were used to train the supervised machine learning model that distinguishes between phishing and legitimate websites.

Flask / FastAPI

A Flask backend was created to serve the trained model and expose it through a lightweight API. The browser extension sends each hovered or clicked link to this backend, which returns a prediction result (safe or phishing) in real time.

3.2.2) Frontend & Extension Development

JavaScript (Chrome Extensions API)

JavaScript was used to build the extension logic, enabling interaction with webpage elements like hyperlinks. The extension listens for user actions (hovering or clicking on links) and sends URLs to the backend for analysis.

HTML, CSS and JavaScript

These were used to design and style the popup interface and in-page warnings displayed to users. The user interface includes colored banners and alert messages that inform the user if a visited or hovered link is identified as a phishing attempt.

WebAssembly

WebAssembly was explored as an option to embed parts of the phishing detection model directly in the browser. This allowed faster response times by reducing dependency on server calls for predictions.

3.2.3) Data Sources

PhishTankAPI

The extension uses PhishTank to collect real-time lists of known phishing URLs.

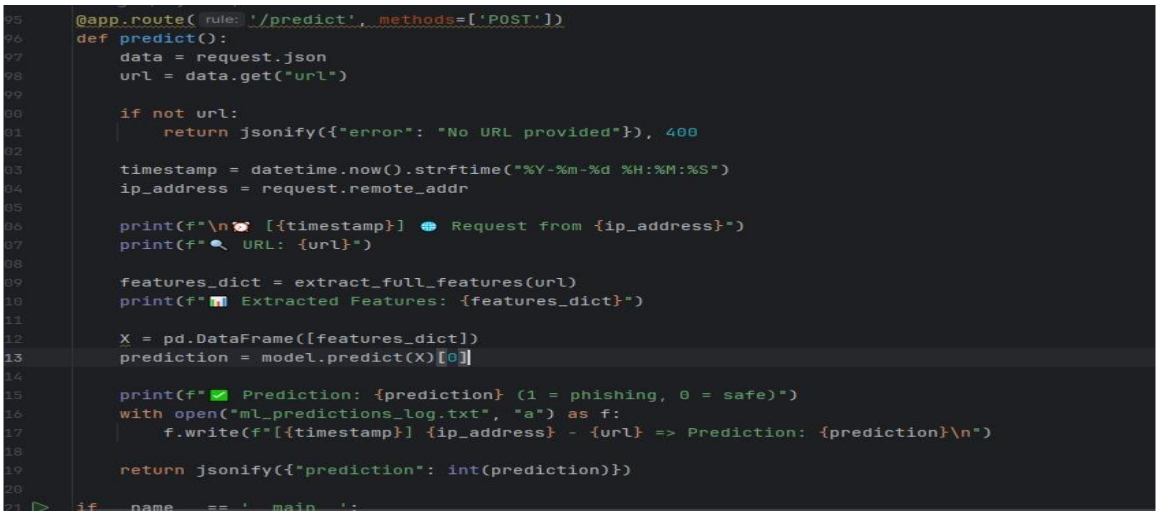
This data was used both during training and to enhance the model's performance by supplementing it with up-to-date threats

UCI ML Phishing Dataset

This dataset was used during the early stages of model training and testing. It provided labeled examples of phishing and legitimate URLs, with pre-extracted features that helped train the supervised model effectively.

3.4 Backend and Frontend Integration

This code defines the `/predict` endpoint in the Flask backend, which receives a URL, extracts features, runs inference using the trained machine learning model, logs the result, and returns the prediction as a JSON response.



```
@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    url = data.get("url")

    if not url:
        return jsonify({"error": "No URL provided"}), 400

    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    ip_address = request.remote_addr

    print(f"\n⏰ [{timestamp}] 🌐 Request from {ip_address}")
    print(f"🔍 URL: {url}")

    features_dict = extract_full_features(url)
    print(f"📊 Extracted Features: {features_dict}")

    X = pd.DataFrame([features_dict])
    prediction = model.predict(X)[0]

    print(f"✅ Prediction: {prediction} (1 = phishing, 0 = safe)")
    with open("ml_predictions_log.txt", "a") as f:
        f.write(f"[{timestamp}] {ip_address} - {url} => Prediction: {prediction}\n")

    return jsonify({"prediction": int(prediction)})

if __name__ == '__main__':
```

Figure 3.4.1: Flask API Route for Phishing Prediction

3.5 Browser Extension Process Diagrams

- Flowchart showing the browser extension's decision-making process: it collects a URL when the user clicks or hovers over a link, analyzes it using a machine learning model, alerts the user if the link is potentially phishing, and provides options to proceed, exit, or report an issue. Feedback from users can be used to improve model performance

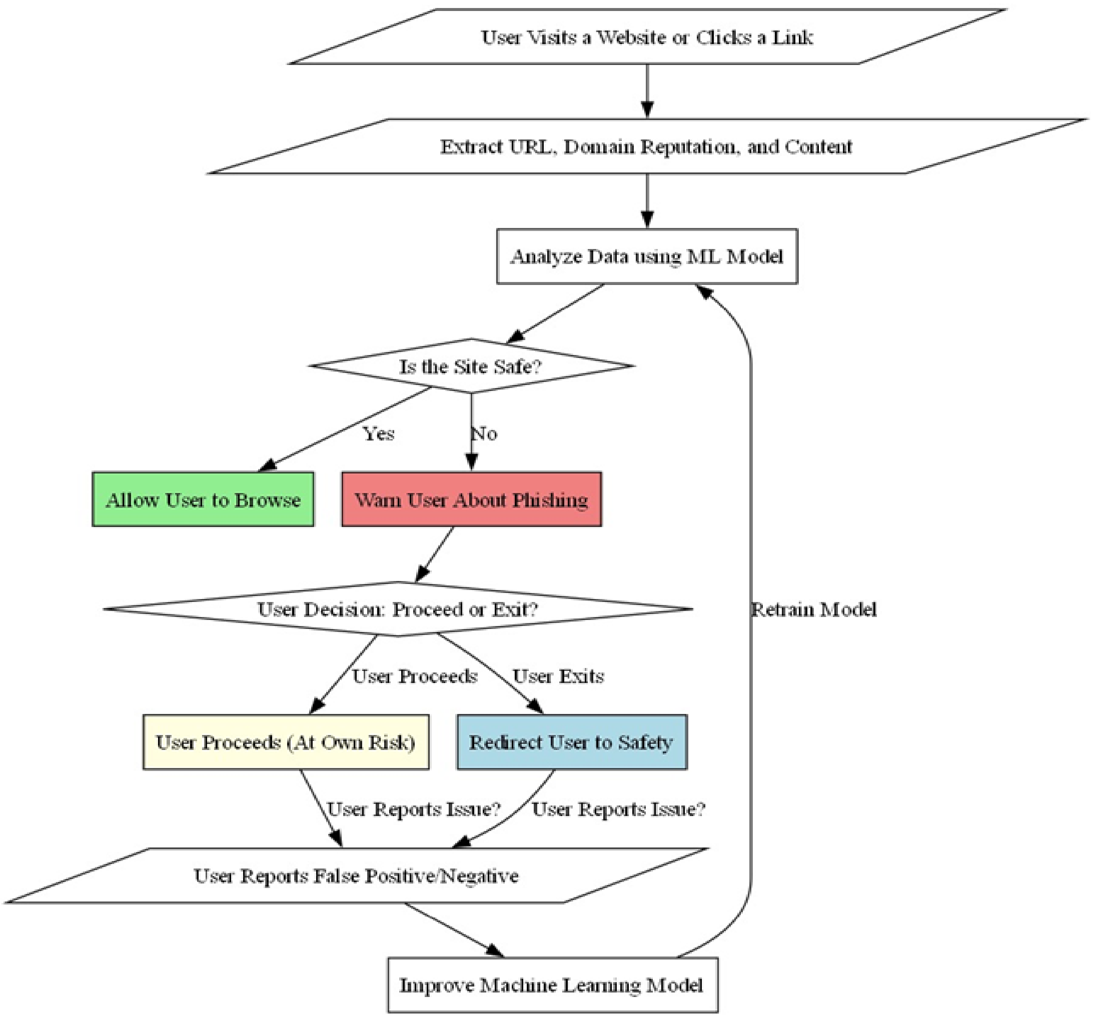

User Visits a Website or Clicks a Link
Extract URL, Domain Reputation, and Content
Analyze Data using ML Model
Is the Site Safe?
Yes
No
Allow User to Browse
Warn User About Phishing
User Decision: Proceed or Exit?
User Proceeds
User Exits
User Proceeds (At Own Risk)
Redirect User to Safety
User Reports Issue?
User Reports Issue?
User Reports False Positive/Negative
Improve Machine Learning Model
Retrain Model

Figure 3.5.1: Flowchart

- This process below shows dataset preprocessing, feature extraction, model training, and deployment.

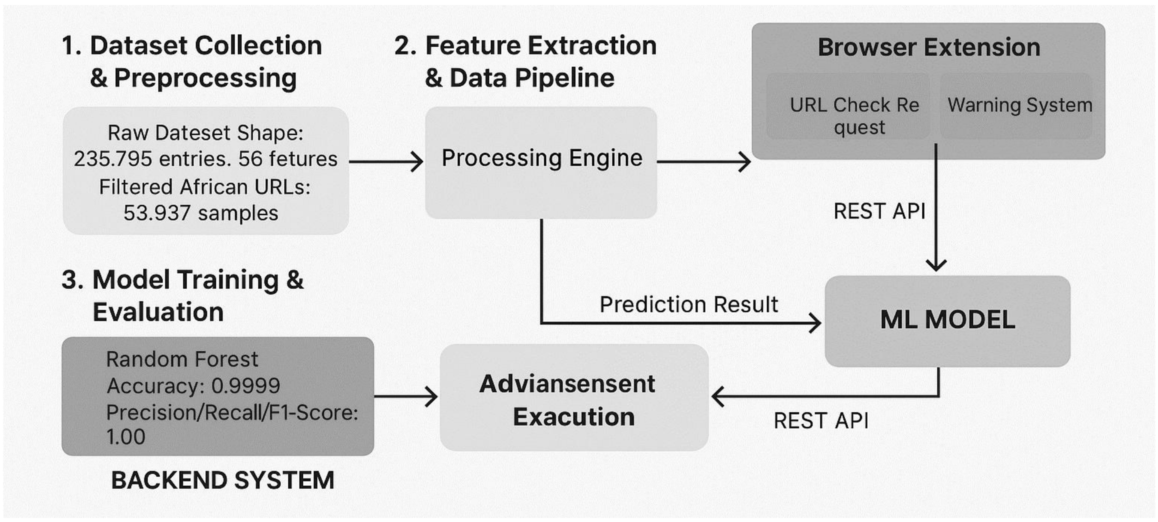

1. Dataset Collection & Preprocessing
Raw Dateset Shape: 235.795 entries. 56 fetures
Filtered African URLs: 53.937 samples
2. Feature Extraction & Data Pipeline
Processing Engine
Browser Extension
URL Check Re quest
Warning System
REST API
Prediction Result
ML MODEL
3. Model Training & Evaluation
Random Forest Accuracy: 0.9999 Precision/Recall/F1-Score: 1.00
Adviansensent Exacution
REST API
BACKEND SYSTEM

Figure 3.5.2: Workflow diagram

- This diagram illustrates the step-by-step logical flow of the phishing detection system, from capturing a user's interaction with a link to analyzing the URL using a machine learning model, delivering warnings or access based on classification results, and collecting user feedback to improve the model over time

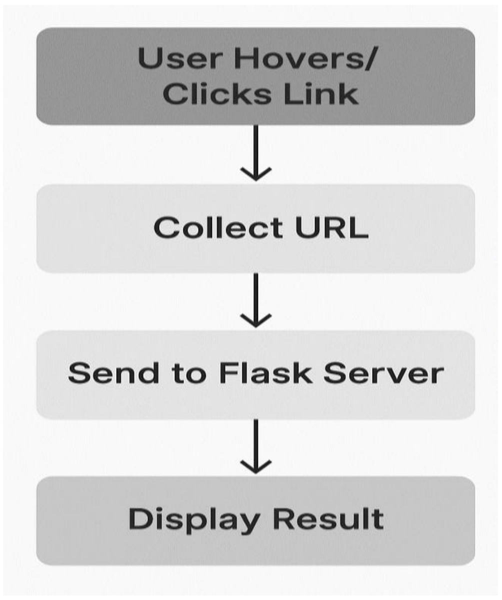


Figure 3.5.3: Logic Flow diagram

3.6 The System Interface

This is how the tool’s interface look like, it shows the initial version of the browser extension dashboard showing summarized phishing predictions for visited links. Here a user can check links, if they are safe or not by pasting the link on the search box and clicking “Check”.

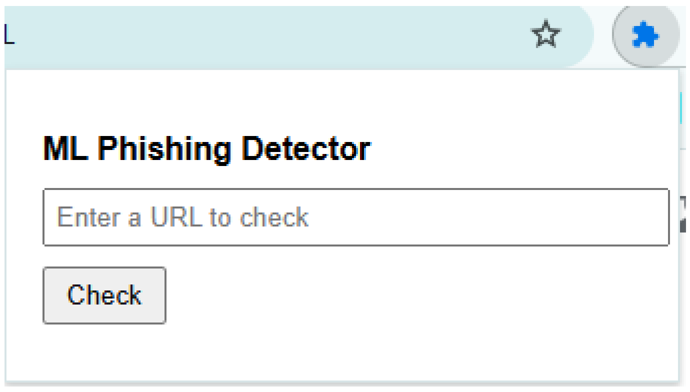
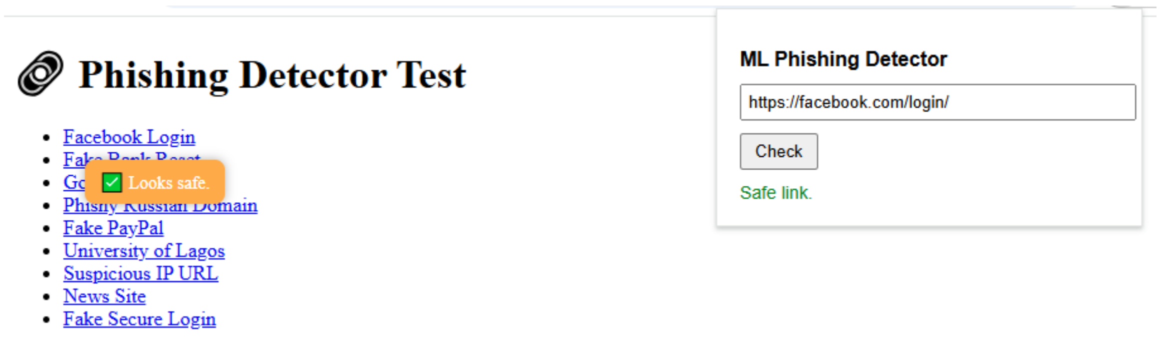


Figure 3.6.1 : Popup Dashboard Interface

Testing Dummy HTML Links with Extension

The figures below ***Figure 3.6.2(a and b)*** respectively show a simulated webpage with several test links used to verify the browser extension's real-time detection accuracy.



Phishing Detector Test

- Facebook Login
- Fake Bank Reset
- Google
- Phishy Russian Domain
- Fake PayPal
- University of Lagos
- Suspicious IP URL
- News Site
- Fake Secure Login

✅ Looks safe.

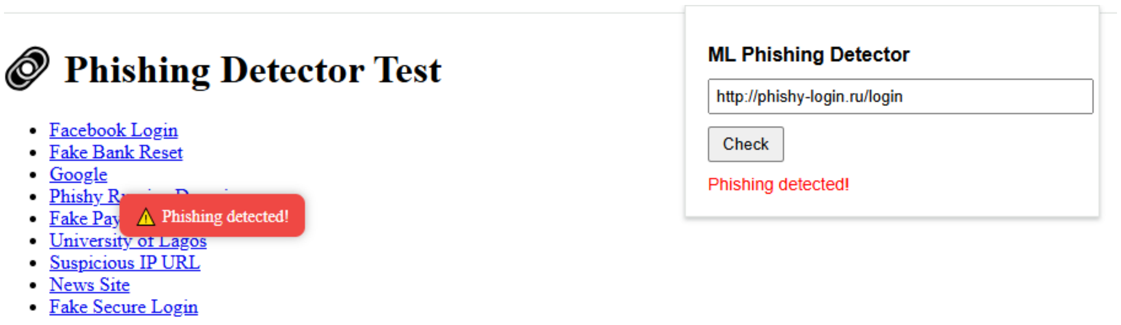
ML Phishing Detector

https://facebook.com/login/

Check

Safe link.

Figure 3.6.2(a): Safe Link



Phishing Detector Test

- Facebook Login
- Fake Bank Reset
- Google
- Phishy Russian Domain
- Fake PayPal
- University of Lagos
- Suspicious IP URL
- News Site
- Fake Secure Login

⚠ Phishing detected!

ML Phishing Detector

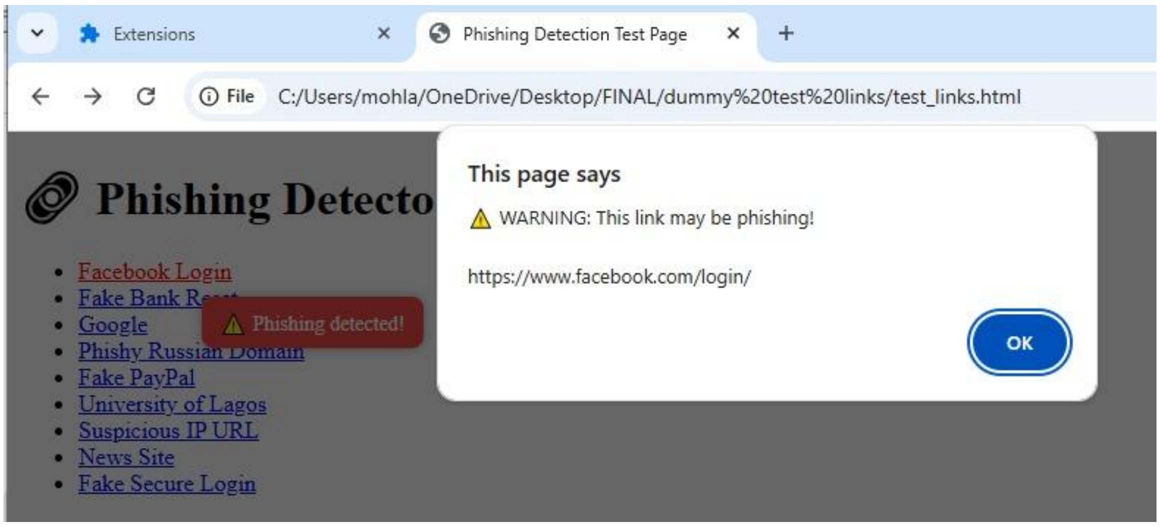
http://phishy-login.ru/login

Check

Phishing detected!

Figure 3.6.2(b) : Phishing Detected

This is an example of the output where legitimate links were flagged as phishing and vice versa, highlighting model limitations.


Extensions
Phishing Detection Test Page
File
C:/Users/mohla/OneDrive/Desktop/FINAL/dummy%20test%20links/test_links.html
OK

Phishing Detecto

- Facebook Login
- Fake Bank Reset
- Google
- Phishy Russian Domain
- Fake PayPal
- University of Lagos
- Suspicious IP URL
- News Site
- Fake Secure Login

⚠ Phishing detected!

This page says

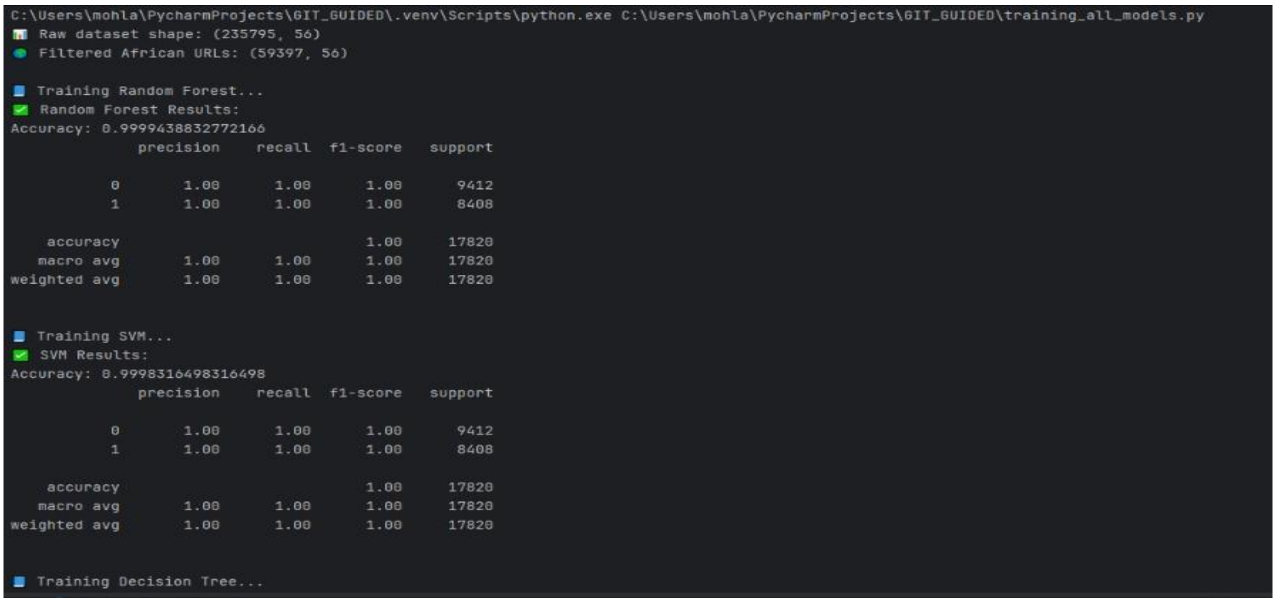
⚠ WARNING: This link may be phishing!

https://www.facebook.com/login/

Figure 3.6.3: Misclassification Log Output

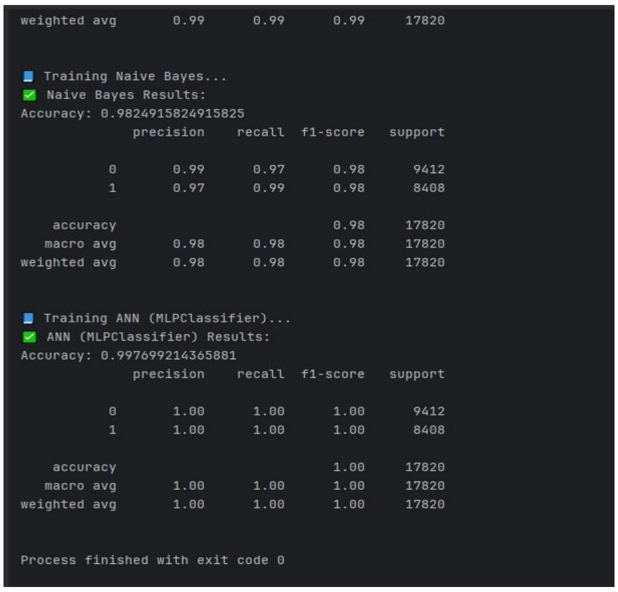
3.7 Model Training and Evaluation : Model Training Accuracy Across Multiple Classifiers

This is a console output displaying the classification results of Random Forest, SVM, Decision Tree, KNN, Naive Bayes, and MLP classifiers, with Random Forest achieving the highest accuracy.

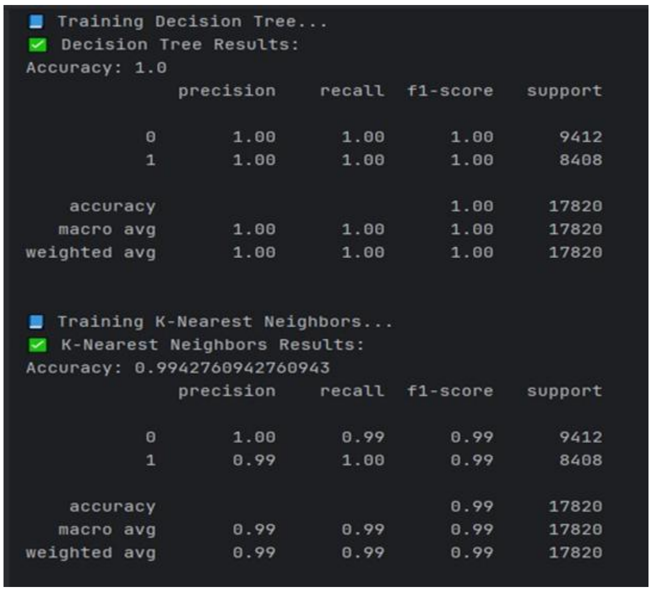


```
C:\Users\mohla\PycharmProjects\GIT_GUIDED\.venv\Scripts\python.exe C:\Users\mohla\PycharmProjects\GIT_GUIDED\training_all_models.py
📊 Raw dataset shape: (235795, 56)
🌍 Filtered African URLs: (59397, 56)

📘 Training Random Forest...
✅ Random Forest Results:
Accuracy: 0.9999438832772166
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9412
           1       1.00      1.00      1.00      8408

    accuracy                           1.00     17820
   macro avg       1.00      1.00      1.00     17820
weighted avg       1.00      1.00      1.00     17820


📘 Training SVM...
✅ SVM Results:
Accuracy: 0.9998316498316498
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9412
           1       1.00      1.00      1.00      8408

    accuracy                           1.00     17820
   macro avg       1.00      1.00      1.00     17820
weighted avg       1.00      1.00      1.00     17820


📘 Training Decision Tree...
```

Figure 3.7.1(a): Random Forest and SVM



```
weighted avg       0.99      0.99      0.99     17820


📘 Training Naive Bayes...
✅ Naive Bayes Results:
Accuracy: 0.9824915824915825
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      9412
           1       0.97      0.99      0.98      8408

    accuracy                           0.98     17820
   macro avg       0.98      0.98      0.98     17820
weighted avg       0.98      0.98      0.98     17820


📘 Training ANN (MLPClassifier)...
✅ ANN (MLPClassifier) Results:
Accuracy: 0.997699214365881
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9412
           1       1.00      1.00      1.00      8408

    accuracy                           1.00     17820
   macro avg       1.00      1.00      1.00     17820
weighted avg       1.00      1.00      1.00     17820


Process finished with exit code 0
```

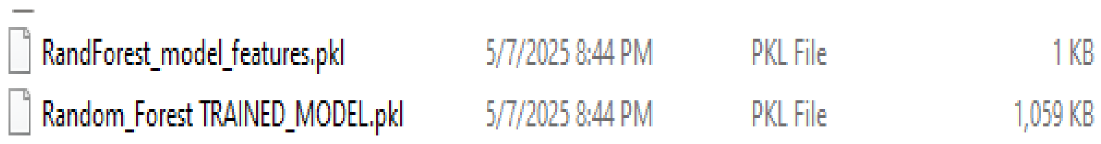


```
📘 Training Decision Tree...
✅ Decision Tree Results:
Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      9412
           1       1.00      1.00      1.00      8408

    accuracy                           1.00     17820
   macro avg       1.00      1.00      1.00     17820
weighted avg       1.00      1.00      1.00     17820


📘 Training K-Nearest Neighbors...
✅ K-Nearest Neighbors Results:
Accuracy: 0.9942760942760943
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      9412
           1       0.99      1.00      0.99      8408

    accuracy                           0.99     17820
   macro avg       0.99      0.99      0.99     17820
weighted avg       0.99      0.99      0.99     17820
```

Figure 3.7.1(b &c): Naïve Bayes, KNN, Decision Tree and K-Nearest Neighbors

Saved Model Confirmation

The output confirms that the trained Random Forest model and its corresponding features were successfully saved for later reuse.

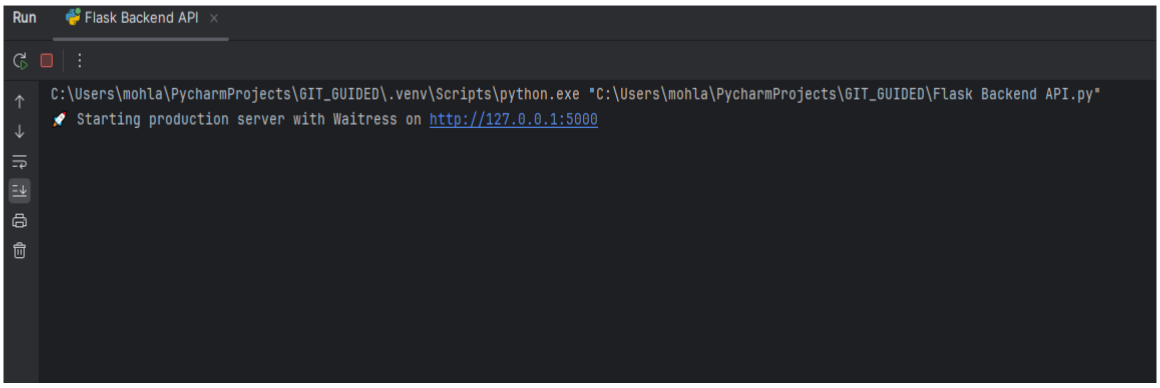


```
RandForest_model_features.pkl      5/7/2025 8:44 PM    PKL File        1 KB
Random_Forest TRAINED_MODEL.pkl    5/7/2025 8:44 PM    PKL File    1,059 KB
```

Figure 3.7.2: Trained Random Forest Model

3.8 Backend Integration or Deployment

This shows the Flask backend being deployed using the Waitress production server on port 5000.

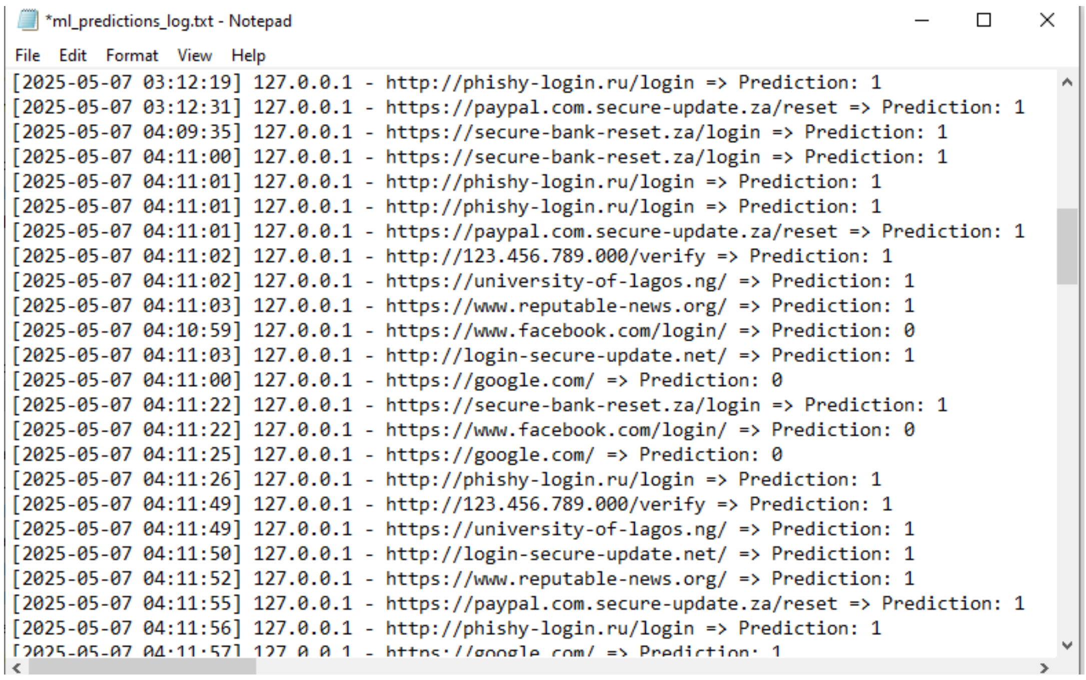


```
C:\Users\mohla\PycharmProjects\GIT_GUIDED\.venv\Scripts\python.exe "C:\Users\mohla\PycharmProjects\GIT_GUIDED\Flask Backend API.py"
🚀 Starting production server with Waitress on http://127.0.0.1:5000
```

3.8.1: Flask Backend Server Running with Waitress

3.9 Monitoring and Logging

Sample log entries recording URL predictions made through the API, including timestamps and client IPs.



```
[2025-05-07 03:12:19] 127.0.0.1 - http://phishy-login.ru/login => Prediction: 1
[2025-05-07 03:12:31] 127.0.0.1 - https://paypal.com.secure-update.za/reset => Prediction: 1
[2025-05-07 04:09:35] 127.0.0.1 - https://secure-bank-reset.za/login => Prediction: 1
[2025-05-07 04:11:00] 127.0.0.1 - https://secure-bank-reset.za/login => Prediction: 1
[2025-05-07 04:11:01] 127.0.0.1 - http://phishy-login.ru/login => Prediction: 1
[2025-05-07 04:11:01] 127.0.0.1 - http://phishy-login.ru/login => Prediction: 1
[2025-05-07 04:11:01] 127.0.0.1 - https://paypal.com.secure-update.za/reset => Prediction: 1
[2025-05-07 04:11:02] 127.0.0.1 - http://123.456.789.000/verify => Prediction: 1
[2025-05-07 04:11:02] 127.0.0.1 - https://university-of-lagos.ng/ => Prediction: 1
[2025-05-07 04:11:03] 127.0.0.1 - https://www.reputable-news.org/ => Prediction: 1
[2025-05-07 04:10:59] 127.0.0.1 - https://www.facebook.com/login/ => Prediction: 0
[2025-05-07 04:11:03] 127.0.0.1 - http://login-secure-update.net/ => Prediction: 1
[2025-05-07 04:11:00] 127.0.0.1 - https://google.com/ => Prediction: 0
[2025-05-07 04:11:22] 127.0.0.1 - https://secure-bank-reset.za/login => Prediction: 1
[2025-05-07 04:11:22] 127.0.0.1 - https://www.facebook.com/login/ => Prediction: 0
[2025-05-07 04:11:25] 127.0.0.1 - https://google.com/ => Prediction: 0
[2025-05-07 04:11:26] 127.0.0.1 - http://phishy-login.ru/login => Prediction: 1
[2025-05-07 04:11:49] 127.0.0.1 - http://123.456.789.000/verify => Prediction: 1
[2025-05-07 04:11:49] 127.0.0.1 - https://university-of-lagos.ng/ => Prediction: 1
[2025-05-07 04:11:50] 127.0.0.1 - http://login-secure-update.net/ => Prediction: 1
[2025-05-07 04:11:52] 127.0.0.1 - https://www.reputable-news.org/ => Prediction: 1
[2025-05-07 04:11:55] 127.0.0.1 - https://paypal.com.secure-update.za/reset => Prediction: 1
[2025-05-07 04:11:56] 127.0.0.1 - http://phishy-login.ru/login => Prediction: 1
[2025-05-07 04:11:57] 127.0.0.1 - https://google.com/ => Prediction: 1
```

Figure 3.9.1: Logged Predictions from Flask API Requests

CHAPTER 4

DISCUSSION AND CONCLUSION

4.1 Challenges and Limitations

During the development of the phishing detection browser extension, several challenges were encountered namely:

Difficulty in Selecting the Appropriate Machine Learning Approach

One of the initial challenges we faced was determining the most suitable type of machine learning for our phishing detection system. While options like unsupervised and semi-supervised learning were considered, they posed difficulties in terms of implementation and effectiveness due to the nature of the data we had access to.

Limited Dataset Size

High-quality labeled phishing and safe website data was limited, which constrained the accuracy of the machine learning model.

Data Imbalance

There were significantly more legitimate URLs than phishing ones, leading to potential bias in model training.

False Positives/Negatives

Some legitimate websites were flagged as phishing, while a few phishing sites passed undetected, especially when attackers used sophisticated obfuscation.

Browser Extension Permissions

Implementing real-time detection requires specific browser permissions that can raise privacy or security concerns.

Model Size and Efficiency

The model had to be lightweight to ensure fast detection without affecting browser performance.

Testing Limitations

Testing was performed on a limited set of browsers and devices, which may not reflect broader user environments.

Mismatch in features

Real-time extractor initially lacked all 52 trained features

Overfitting

Model performed perfectly on training data but misclassified real-world examples

Flask dev server

Switched to Waitress for production readiness

Browser integration bugs

Required careful debugging across MV3 manifest and content scripts

Label reversal issue

Model sometimes classifies phishing as legitimate (current bug to fix)

4.2 Recommendations and Future Work

Based on the current development and testing, several improvements and future enhancements can be considered:

Natural Language Processing (NLP)

Use NLP to analyze suspicious page content and phishing email text for context-aware detection.

Mobile Browser Support

Extend compatibility to mobile browsers to protect users on smartphones and tablets and support to Android and iOS browsers

User Feedback Mechanism

Allow users to report false positives or undetected phishing links to continuously improve the model.

Crowdsourced Threat Intelligence

Integrate real-time threat intelligence from external sources like Google Safe Browsing, VirusTotal, etc.

Phishing Email Detection

Expand the extension’s scope to directly analyze incoming emails in webmail clients.

Enterprise Deployment

Package the extension for centralized deployment and monitoring in business environments.

Use larger, cleaner, and more up-to-date datasets

By incorporating more up-to-date and representative data, the model will be better equipped to identify evolving phishing techniques and reduce bias or overfitting. Improved data quality will also aid in minimizing false positives and false negatives during real-world deployment.

Introduce feedback loop from users (flag false positives)

Allow users to report false positives or undetected phishing links to continuously improve the model

4.3 Conclusion

The development of a phishing detection browser extension represents a significant advancement in real-time cybersecurity tools tailored for everyday users. By integrating a supervised machine learning model—specifically a Random Forest classifier—with a lightweight, browser-compatible interface, this project offers an intelligent and accessible solution to one of the most persistent cybersecurity threats globally. Challenges such as CORS restrictions, dataset imbalance, and model generalization were addressed through methodical engineering solutions and iterative testing. The resulting tool not only demonstrates technical feasibility but also underscores the value of applying AI to enhance proactive digital safety mechanisms in both personal and organizational environments.

Looking ahead, the project lays a strong foundation for continued innovation in the field of browser-based cybersecurity. Future enhancements such as Natural Language Processing for phishing email detection, crowdsourced feedback integration, and mobile browser compatibility will elevate the system's adaptability and effectiveness. The scalability of this solution for enterprise use, coupled with its cross-platform design, opens promising avenues for commercialization and real-world impact. Ultimately, this work contributes to the growing body of intelligent, self-learning cybersecurity tools and reflects the transformative potential of machine learning in shaping a safer digital future.

REFERENCES

Egress. (2021) Insider Data Breach Survey. [online] Available at: https://www.egress.com/newsroom/phishing-insider-survey [Accessed 13 February 2025].
Google. (2023) Protecting Gmail from phishing with AI. [online] Available at: https://blog.google/technology/safety-security/protecting-gmail-phishing-ai/ [Accessed 09 March 2025].

Microsoft. (2022) Microsoft Defender for Office 365. [online] Available at: https://www.microsoft.com/en-us/microsoft-365/security/office-365-defender [Accessed 01 May 2025].

PhishTank. (n.d.) Phishing Website Data. [online] Available at: https://www.phishtank.com/ [Accessed 03 March 2025].

Scikit-learn Developers. (2024) scikit-learn: Machine Learning in Python. [online] Available at: https://scikit-learn.org/ [Accessed 16 March 2025].

TensorFlow Developers. (2024) TensorFlow: An end-to-end open source machine learning platform. [online] Available at: https://www.tensorflow.org/ [Accessed 17 April 2025].

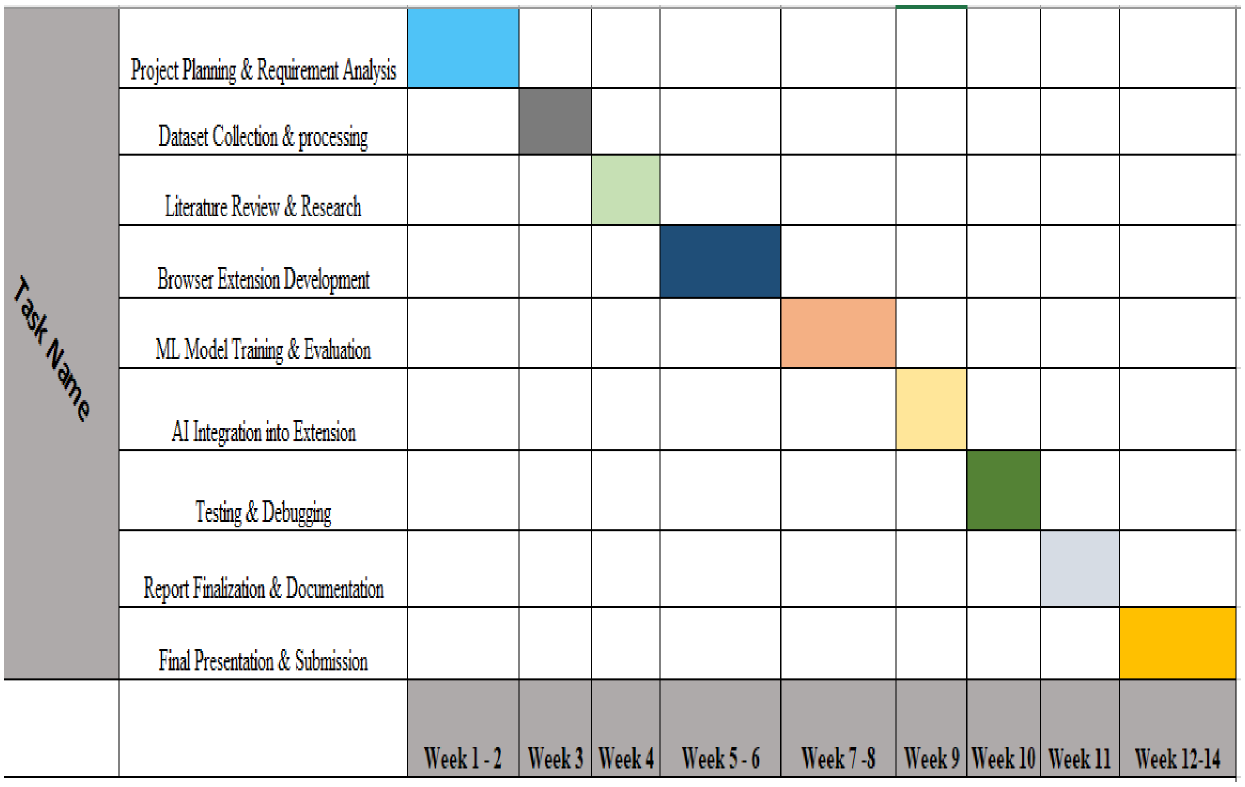
UCI Machine Learning Repository. (n.d.) Phishing Websites Dataset. [online] Available at: https://archive.ics.uci.edu/ml/datasets/phishing+websites [Accessed 27 February 2025].

Verizon. (2023) Data Breach Investigations Report (DBIR). [online] Available at: https://www.verizon.com/business/resources/reports/dbir/ [Accessed 09 April 2025].

Waitress. (n.d.) Production-Ready WSGI Server for Python. [online] Available at: https://docs.pylonsproject.org/projects/waitress/en/stable/ [Accessed 03 April 2025].

APPENDICES

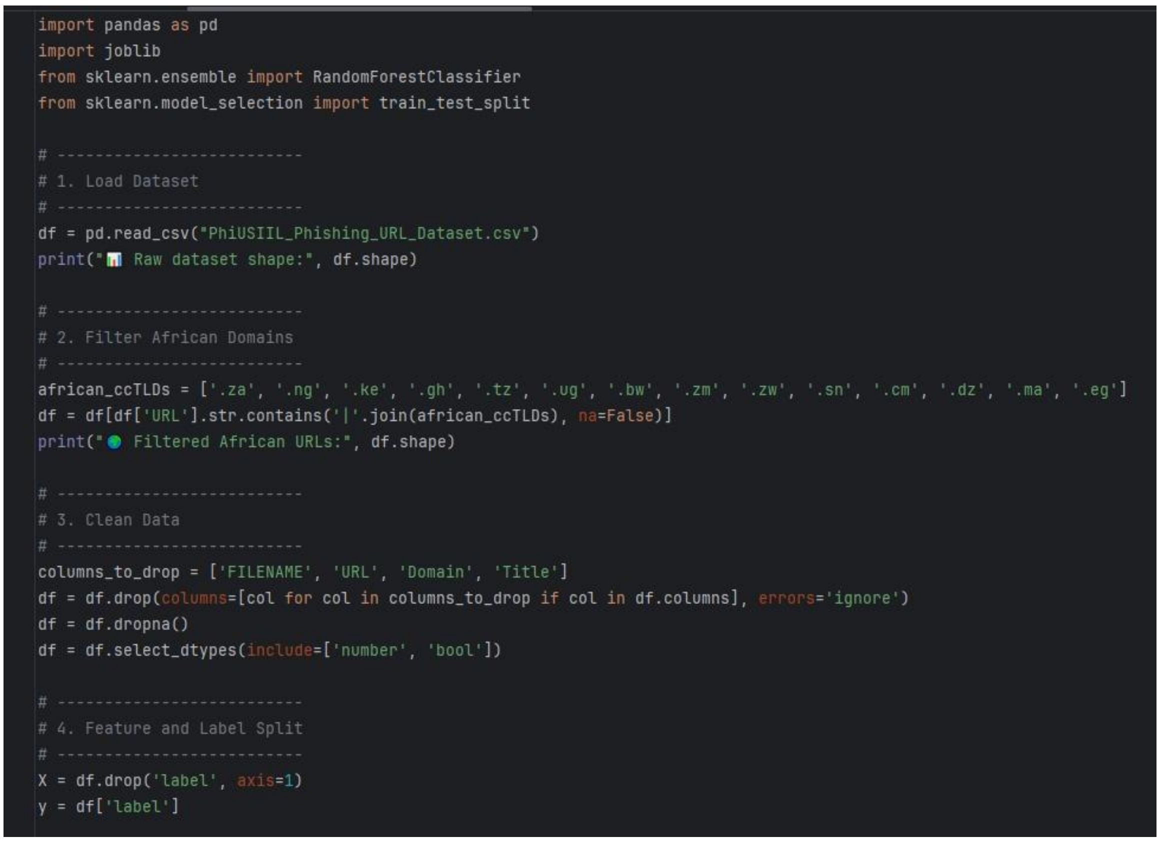
APPENDIX A: Gantt Chart



Task Name	Week 1 - 2	Week 3	Week 4	Week 5 - 6	Week 7 -8	Week 9	Week 10	Week 11	Week 12-14
Project Planning & Requirement Analysis	🟦								
Dataset Collection & processing		⬛							
Literature Review & Research			🟩						
Browser Extension Development				🟦					
ML Model Training & Evaluation					🟧				
AI Integration into Extension						🟨			
Testing & Debugging							🟩		
Report Finalization & Documentation								⬜	
Final Presentation & Submission									🟨

APPENDIX B: Source Code Listing

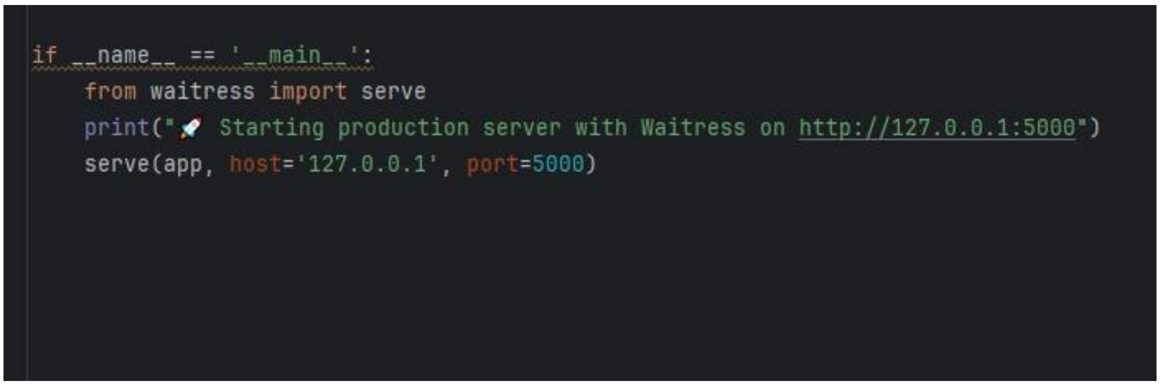
This script is responsible for training the Random Forest model using a labeled dataset of phishing and legitimate URLs. It also saves the trained model and the feature list for later use in real-time prediction



```python
import pandas as pd
import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# -------------------------
# 1. Load Dataset
# -------------------------
df = pd.read_csv("PhiUSIIL_Phishing_URL_Dataset.csv")
print("📊 Raw dataset shape:", df.shape)

# -------------------------
# 2. Filter African Domains
# -------------------------
african_ccTLDs = ['.za', '.ng', '.ke', '.gh', '.tz', '.ug', '.bw', '.zm', '.zw', '.sn', '.cm', '.dz', '.ma', '.eg']
df = df[df['URL'].str.contains('|'.join(african_ccTLDs), na=False)]
print("🌍 Filtered African URLs:", df.shape)

# -------------------------
# 3. Clean Data
# -------------------------
columns_to_drop = ['FILENAME', 'URL', 'Domain', 'Title']
df = df.drop(columns=[col for col in columns_to_drop if col in df.columns], errors='ignore')
df = df.dropna()
df = df.select_dtypes(include=['number', 'bool'])

# -------------------------
# 4. Feature and Label Split
# -------------------------
X = df.drop('label', axis=1)
y = df['label']
```

B1: Random_Forest_Trained_Model.py

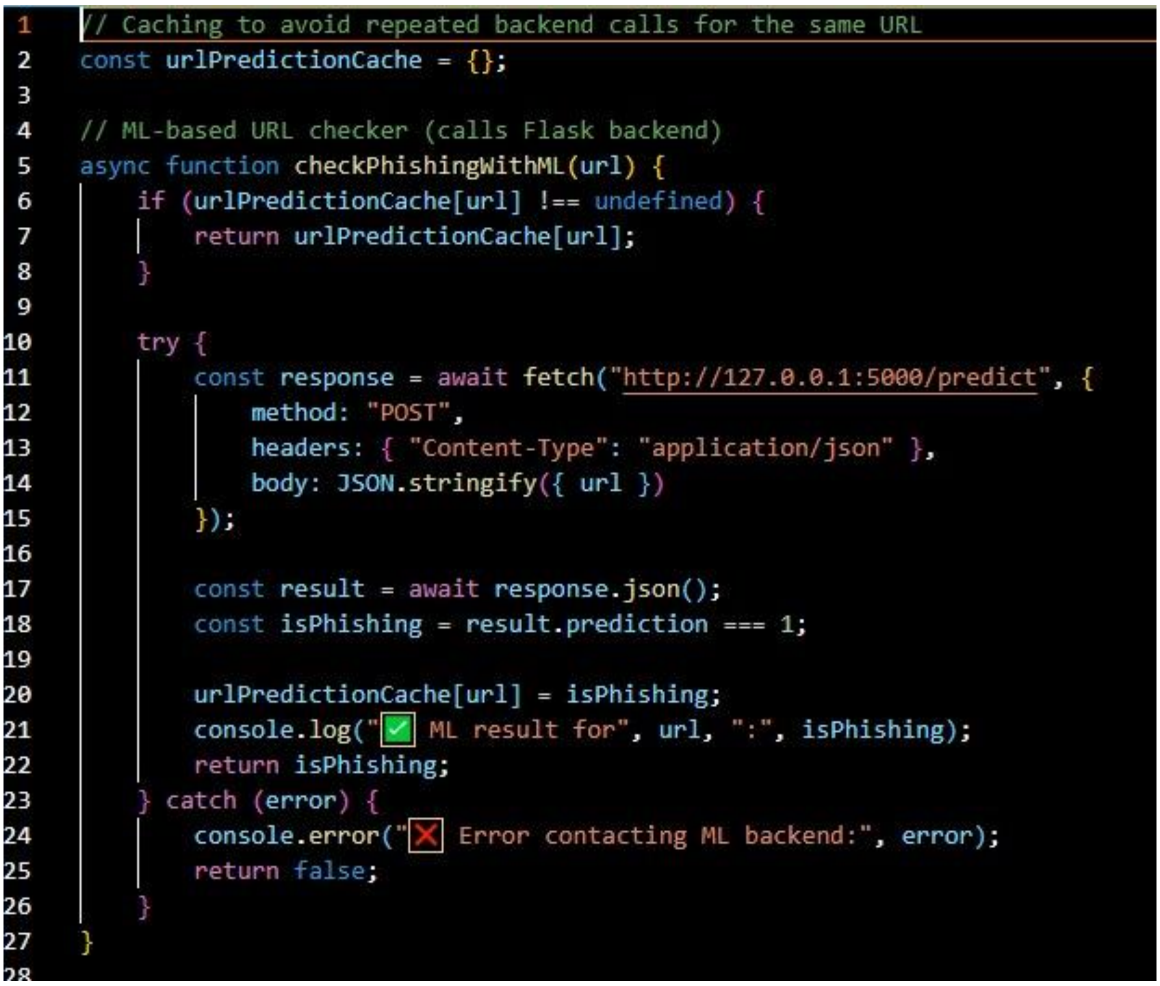
This file hosts the Flask API that serves as a real-time interface between the browser extension and the trained machine learning model. It receives URL input, extracts features, runs predictions, and returns phishing detection results.



```python
if __name__ == '__main__':
    from waitress import serve
    print("🚀 Starting production server with Waitress on http://127.0.0.1:5000")
    serve(app, host='127.0.0.1', port=5000)
```

B2: Flask Backend API

This JavaScript file runs inside the browser and communicates with the Flask backend. It monitors link interactions (hover and click), sends the URL for prediction, and displays warnings to the user if a link is flagged as suspicious.



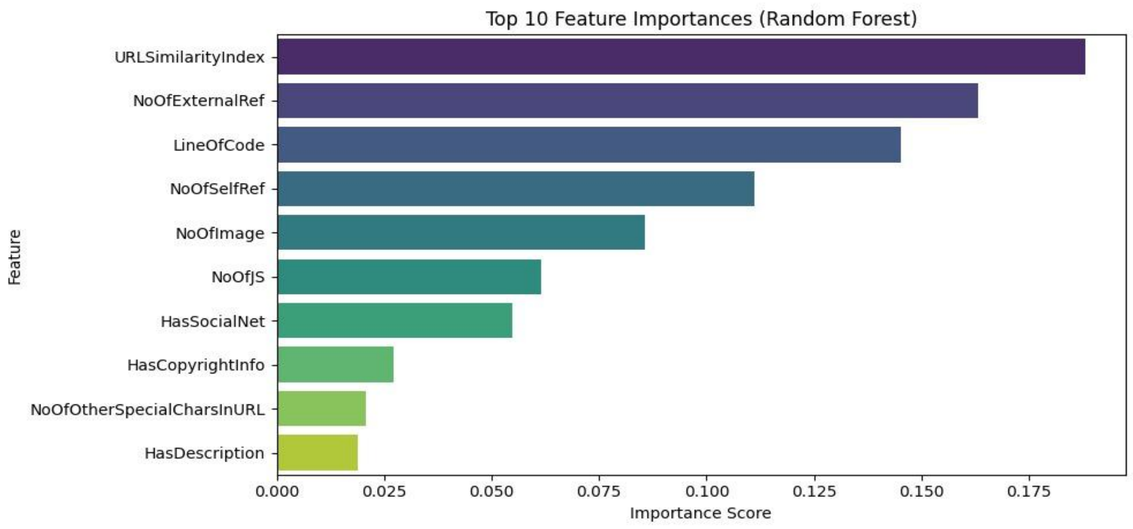
```javascript
// Caching to avoid repeated backend calls for the same URL
const urlPredictionCache = {};

// ML-based URL checker (calls Flask backend)
async function checkPhishingWithML(url) {
    if (urlPredictionCache[url] !== undefined) {
        return urlPredictionCache[url];
    }

    try {
        const response = await fetch("http://127.0.0.1:5000/predict", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ url })
        });

        const result = await response.json();
        const isPhishing = result.prediction === 1;

        urlPredictionCache[url] = isPhishing;
        console.log("✅ ML result for", url, ":", isPhishing);
        return isPhishing;
    } catch (error) {
        console.error("❌ Error contacting ML backend:", error);
        return false;
    }
}
```

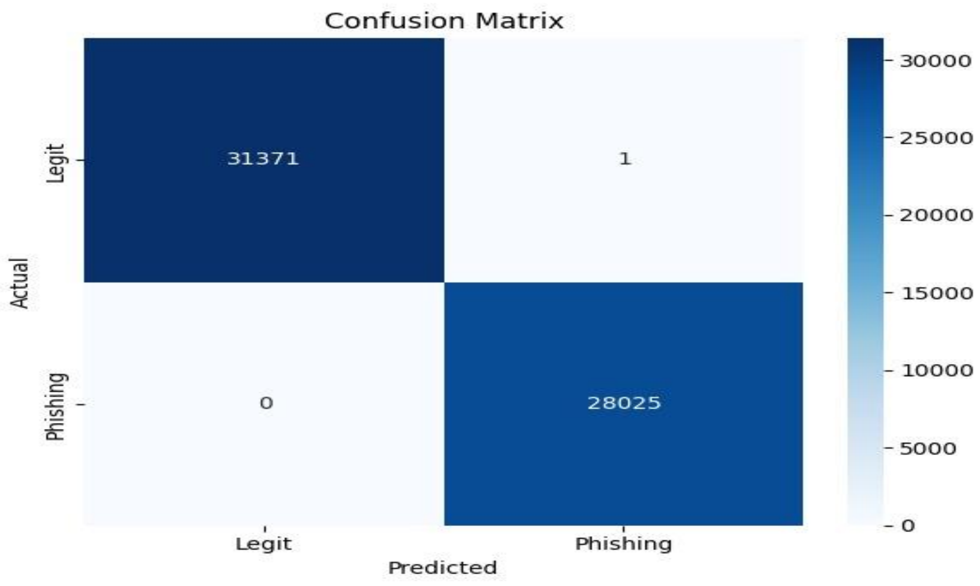
B3: Content.js

APPENDIX C: Graphs

Confusion Matrix and Feature Importance Plot

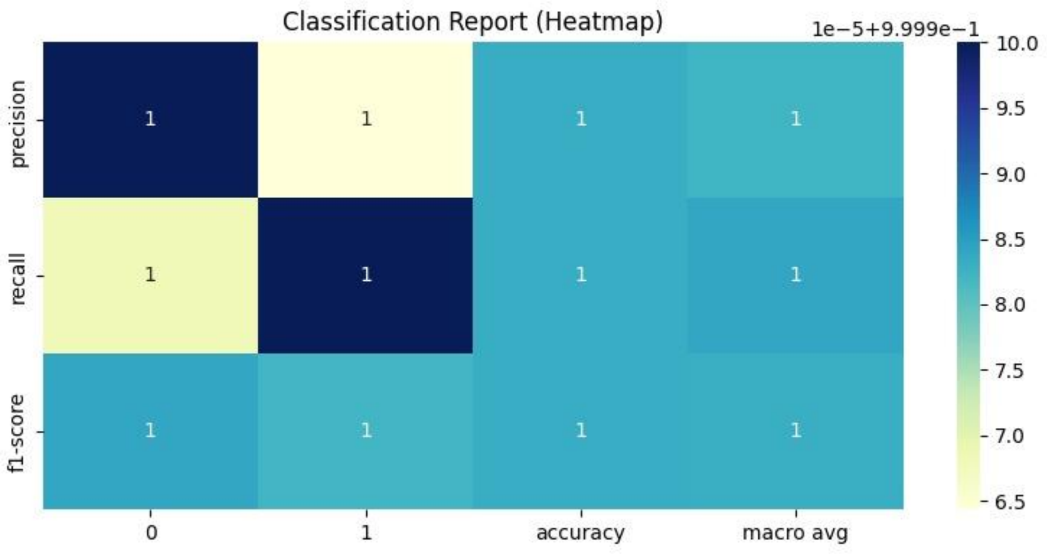
Visual representation of the classifier’s performance and key contributing features





.

Batch Prediction Accuracy Results



Batch evaluation of the model against the filtered dataset, showing near-perfect prediction accuracy