****

**BEACONHOUSE NATIONAL UNIVERSITY**

**Ebryx x BNU (Ml Project)**

**PRJ-F23/330**

**PROJECT DOCUMENTATION**

**EXTERNAL SUPERVISOR**

**Arslan Kazmi (Ebryx)**

**INTERNAL SUPERVISORS  
  
Huda Sarfaraz**

**&**

**Shazia Rizwan**

**GROUP MEMBERS**

**Ammara Amin F2021-394**

**Awais Amin F2021-395**

**SCHOOL OF COMPUTER &IT**

**September 2023**

# **Introduction:**

This project is a collaboration between BNU and EBRYX, an IT company specializing in cybersecurity solutions.

Initially, we were given requirements by EBRYX, focusing on the detection of various cybersecurity threats. The use cases included Phishing, Cryptojacking, Ransomware, Remote Access Trojans (RATs), Botnets, Data Exfiltration, Generic Malicious Behavior, Malicious Domains, and Suspicious Network Traffic Patterns. We were advised to either delve deeply into one specific use case or to explore each one briefly. After careful consideration and taking advise from EBRYX, we chose to focus on Phishing, from which we specifically selected Email Phishing and URL Phishing as our primary areas of study.

The purpose of this document is to present the two main stages of our project in developing a phishing detection system. Stage one comprised of researching and identifying potential solutions. Here, we examined various approaches to find effective methods for phishing detection, our primary focus centered on identifying effective methodologies for detecting phishing attempts in emails and URLs. Stage two involves the development of a machine learning-based system, designed to function as a browser extension, complete with a development plan and a structured system architecture.

# **Stage 1: Research and Identification of Potential Solutions**

# **Objective**

The primary objective of Stage 1 was to conduct extensive research and a literature survey. This stage was dedicated to understanding the current landscape of phishing detection methodologies in emails and URLs. The aim was to identify potential approaches, solutions and strategies that could be further explored and developed.

**Phishing: A Brief Overview**

Phishing is a cybercrime where attackers deceive people into sharing sensitive information by disguising themselves as legitimate entities via email or misleading URLs. Typically, these attacks involve emails or websites that mimic reputable companies, tricking individuals into revealing personal data like passwords or financial details. The goal is often identity theft or financial fraud. Awareness and caution are essential to combat these increasingly sophisticated scams.

# **Literature Survey**

By doing a literature survey our objective was to gain a comprehensive understanding of the methodologies and strategies used for detecting phishing in emails and URLs. This review aims to dissect the various approaches to identifying phishing activities normally and through email and browsing traffic analysis.

In the paper "Phishing Detection Using Traffic Behavior Spectral Clustering and Random Forests" DeBarr et al, **[1]** presents a novel approach to phishing detection using Spectral Clustering and Random Forests. Their method focuses on analyzing traffic behavior and URL substrings in email messages. They utilize the Phishing Email Corpus and the Spam Assassin Email Corpus for evaluation, showing that their approach significantly improves performance metrics like AUC and F-measure compared to traditional content filtering techniques.

In "Fuzzy Inference for Anomaly Detection in Email Traffic Communication Behavior," Lim et al, **[2]** focus on monitoring email traffic for abnormal changes in communication patterns using fuzzy logic. They define several behavior measures and use a hierarchy of fuzzy inference systems to provide an abnormality rating for changes in communication behavior of suspect email accounts. Their methodology is demonstrated with a case study using the Enron email corpus, showing how fuzzy inference helps in summarizing and prioritizing abnormal changes in email communication behavior.

In paper "On the Recent Use of Email Through Traffic and Network Analysis", Giuseppe Aceto and Antonio Pescape's **[3]** present a comprehensive analysis of the trends in email traffic over the last decade. The study is based on heterogeneous data sources, including traffic traces, OSNs usage logs, and internet usage statistics. The authors found evidence of a small decreasing trend in the use of email, particularly SMTP traffic, and attribute this to the rise of other communication platforms like social networks. They also note an increase in the average size of exchanged emails, suggesting a shift in communication patterns influenced by the adoption of other platforms.

In the paper "Phishing Detection Using Machine Learning Algorithm," Jibrilla Tanimu and Stavros Shiaeles **[4]** from the Department of Computing at the University of Portsmouth explore the development of a novel machine learning-based model to enhance the accuracy and efficacy of phishing detection. They address the significant increase in phishing attacks and the inadequacy of current detection methods by proposing a unique approach that incorporates image visualization of website code and feature extraction from malicious URLs. Their method aims to overcome the limitations of existing techniques by utilizing a large dataset of phishing and legitimate websites to create distinctive images that train the machine learning model for improved prediction accuracy. The paper outlines their ongoing work and future plans to refine their model through feature elimination techniques, such as Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA), to optimize image creation and improve the model's performance further. Tanimu and Shiaeles' research contributes to the cybersecurity field by proposing an innovative solution to mitigate phishing threats, with future work focused on enhancing the detection capabilities and efficiency of their model through comprehensive testing and comparison with existing methods.

In the study "Anomaly Detection in Emails using Machine Learning and Header Information," Beaman and Isah **[5]** explore the potential of using email header datasets for the detection of spam and phishing emails. Unlike previous studies that primarily focused on the content within the email body and subject, this research emphasizes the significance of email headers. The authors utilized a variety of supervised learning algorithms, including Random Forest,

In the paper titled "Look before you leap: Detecting phishing web pages by exploiting raw URL and HTML characteristics," Opara, Chen, and Wei **[6]** address the challenge of detecting phishing websites, which are often used for email and internet fraud. They introduce WebPhish, an end-to-end deep neural network model that detects phishing attacks using raw URLs and HTML content without relying on hand-crafted features. The model uses an embedding technique to generate homologous dense vectors from URL characters and HTML words, followed by a convolutional layer to model semantic relationships.

In the paper "Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey," Salloum et al, **[7]** focus on the use of NLP and ML for detecting phishing emails. They discuss various features for detecting phishing emails, including email body-based, subject-based, URL-based, script-based, and sender-based characteristics. The paper also highlights the use of AI in phishing email detection, discussing advanced models like THEMIS and character-level convolutional neural networks.

In the study "Phishing Email Detection Model Using Deep Learning," Samer Atawneh and Hamzah Aljehani **[8]** explores the use of deep learning techniques, including CNNs, LSTM networks, RNNs, and BERT, for detecting email phishing attacks. The authors utilized a dataset of phishing and benign emails, extracting relevant features using NLP techniques. The deep learning model achieved high accuracy, with the best performance seen when using BERT and LSTM. The study demonstrates the effectiveness of deep learning in improving email phishing detection.

In the study “Phishing URL detection using machine learning methods," Shaik and Hasane, **[9]** focus is on detecting malicious URLs, a common phishing technique, using machine learning algorithms. The study highlights the limitations of blacklisting services in identifying malicious websites and proposes a solution using algorithms like Random Forests, Decision Trees, Light GBM, Logistic Regression, and SVM. The research involves extracting features from URLs and applying machine learning models to determine if a URL is fraudulent.

In the study "Phishing Detection in E-mails using Machine Learning," Rawal et al, **[10]** explore the use of machine learning algorithms for classification of emails into phished or ham (legitimate) using machine learning algorithms. The paper describes the extraction of 9 features from a dataset of 1,605 emails, comprising both phished and ham emails. The features are fed into classifiers like SVM and Random Forest, achieving a maximum accuracy of 99.87%. The research concludes that machine learning can effectively classify emails, suggesting future improvements by increasing the dataset to better replicate real-life scenarios.

In conclusion, our literature review provided valuable insights into the diverse methods used for phishing detection. The studies reviewed offered a broad perspective on the techniques and technologies employed to identify and mitigate phishing threats. These insights are instrumental in our future strategies and building ml models.

# **References**

**[1]** Debarr, Dave & Ramanathan, Venkatesh & Wechsler, Harry. (2013). Phishing detection using traffic behavior, spectral clustering, and random forests. 67-72. 10.1109/ISI.2013.6578788.

**[2]** Lim, Mark & Negnevitsky, Michael & Hartnett, Jacky. (2006). A Fuzzy Approach For Detecting Anomalous Behaviour in E-mail Traffic. Australian Digital Forensics Conference.

**[3]** Aceto, Giuseppe & Pescapè, Antonio. (2012). On the recent use of email through traffic and network analysis: the impact of OSNs, new trends, and other communication platforms. Sigmetrics Performance Evaluation Review - SIGMETRICS. 39. 61-70. 10.1145/2185395.2185442.

**[4]** Tanimu, Jibrilla & Shiaeles, Stavros. (2022). Phishing Detection Using Machine Learning Algorithm. 10.1109/CSR54599.2022.9850316..

**[5]** Beaman, C.P., & Isah, H. (2022). Anomaly Detection in Emails using Machine Learning and Header Information. ArXiv, abs/2203.10408.

**[6]** Opara, Chidimma & Chen, Yingke & wei, Bo. (2020). Look Before You Leap: Detecting Phishing Web Pages by Exploiting Raw URL And HTML Characteristics.

**[7]** Salloum, Said & Gaber, Tarek & Vadera, Sunil & Shaalan, Khaled. (2021). Phishing Email Detection Using Natural Language Processing Techniques: A Literature Survey. Procedia Computer Science. 189. 19-28. 10.1016/j.procs.2021.05.077.

**[8]** Atawneh, Samer & Aljehani, Hamzah. (2023). Phishing Email Detection Model Using Deep Learning. Electronics. 12. 4261. 10.3390/electronics12204261.

**[9]** Shaik, Hasane. (2022). Phishing URL detection using machine learning methods. 103288.

**[10]** Rawal, Srishti & Rawal, Bhuvan & Shaheen, Aakhila & Malik, Shubham. (2017). Phishing Detection in Emails using Machine Learning. International Journal of Applied Information Systems. 12. 21-24. 10.5120/ijais2017451713.

# **Identified Approaches for Email Phishing Detection**

We identified three distinct approaches for email phishing detection through our review of the literature:

1. The first approach involves analyzing email traffic to extract packet features, including packet headers, by utilizing protocols such as SMTP, IMAP, and POP3.
2. The second approach focuses on using Gmail in Chrome to investigate email headers for specific patterns that may suggest phishing activities.
3. The third approach employs Gmail in Chrome for a thorough examination of the email content using natural language processing (NLP) techniques to identify potential phishing attempts.

Note: Further research was not conducted on option no 3 because, as advised by Ebryx, it would be more of a natural language processing (NLP) task.

# **Identified Approaches for URL Phishing Detection**

We identified two distinct approaches for URL phishing detection through our review of the literature:

1. The first approach involves analyzing HTTPS browser traffic to identify phishing attempts.
2. The second approach concentrates on examining URLs to detect specific patterns indicative of phishing activities.

# **Research work on Traffic Analysis**

# **Available Traffic Datasets**

For our dataset, we needed unencrypted pcap files that were labeled as either 'phishy' or 'non-phishy', based on whether the traffic was malicious or normal. This included SMTP and IMAP protocols for emails, as well as HTTPS for browser traffic. The datasets we discovered are as follows:

* **DFIR WRCCDC PCAPs**: <https://dfir.wrccdc.org/pcaps/>
* **UNB VPN Dataset**: <https://www.unb.ca/cic/datasets/vpn.html>
* **UNB IDS 2017 Dataset**: <https://www.unb.ca/cic/datasets/ids-2017.html>

# **Problems in Dataset Collection and Solution**

During dataset collection, we encountered several issues:

* We didn't find the right type of datasets (like .pcap files) needed for our analysis.
* We were missing important SMTP and IMAP traffic data.
* The traffic datasets we found weren't labeled, so we couldn't tell which traffic was normal and which was potentially harmful.

Due to the challenges faced above we were recommended to learn how to use Wireshark to capture email and browsing traffic. After capturing the traffic, we could then use the ID2T tool to inject malicious behavior into it for analysis.

# **Learning Wireshark**

We started by getting familiar with Wireshark, a network analysis tool used for packet capturing and traffic simulation. Our goal was to determine whether analyzing entire network packets or just their headers could help in identifying phishing attempts.

## **Monitoring with Wireshark**

We used Wireshark to track network activity, with a focus on:

* Email traffic (inbound and outbound)
* Web browsing patterns

## **Network Traffic Simulation and Capture**

We generated and monitored our own network traffic across two devices.

We specifically recorded:

* Outgoing and incoming email traffic using SMTP and IMAP protocols
* Web browsing activity using HTTPS protocol

## **Alternative Email Traffic Source**

We used Outlook for SMTP traffic because we faced difficulties capturing email traffic directly from Gmail via Chrome.

# **Limitations & Findings**

We were unable to capture unencrypted traffic due to the following limitations:

**Encryption of emails in Email Traffic**

Services like Gmail use robust HTTPS encryption which hides important details like email headers. Google further reinforces security by adding TLS (Transport Layer Security).

**Encryption of URLs in Browsing Traffic**

The encryption challenges persist in browsing traffic, especially with browsers like Chrome, which employs its proprietary TLS (Transport Layer Security) encryption, limiting the extraction of meaningful data.

## **Limited Information Availability**

## In the current encrypted environment, only basic information like time, source, destination, packet lengths, and sizes are accessible. This limits the depth of analysis possible for security purposes.

## **Difficulty in Detecting Phishing on Standard Ports**

## Phishing often uses standard communication ports, making it hard to differentiate between malicious and legitimate traffic with just packet header data.

**Ineffectiveness of Packet Headers for Phishing Detection**

Headers mainly reveal limited data such as Packet Length, Flags, TTL (Time-to-Live), and Checksums, which are often not directly connected to phishing activities.

## **Encrypted SMTP Traffic from Outlook**

Despite successfully sourcing SMTP traffic through Outlook, we encountered a limitation: the traffic was encrypted, hindering our ability to fully analyze the data for phishing detection.

## **ID2T Tool**

The ID2T tool is designed to create test traffic in cybersecurity systems by injecting malicious patterns that mimic real threats. We were unable to utilize the tool for injecting malicious behavior into traffic because modern encryption techniques, such as HTTPS in Gmail and TLS in Chrome, have rendered traditional traffic capture methods ineffective. This limitation hindered our ability to simulate realistic cyber threats.

# **Conclusion:**

In conclusion, due to the challenges in capturing encrypted traffic with Wireshark, the lack of available labeled unencrypted traffic data on the internet, and the insufficiency of packet header information, we decided to abandon the approach of phishing detection through traffic analysis.

# **Research Work for Raw Email Headers & URL**

# **Available Related Datasets**

## For our dataset, we required labeled data to distinguish between "phishy" and non-phishy elements. This involved collecting data from various sources. Specifically, for emails, we needed raw email headers, while for URLs, we required the actual raw URLs. This approach enabled us to effectively categorize and analyze the elements based on their phishing potential.

## **For Email Headers:**

* **TREC 2007 Corpus**: <https://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html>.
* **Fraudulent Email Corpus from Kaggle:** <https://www.kaggle.com/datasets/rtatman/fraudulent-email-corpus/code>
* **Monkey org:** <https://monkey.org/~jose/phishing/>

## **For URLs:**

* **PhishTank**: [PhishTank](https://www.phishtank.com/developer_info.php).
* The University of New Brunswick's open datasets: [UNB - URL Dataset](https://www.unb.ca/cic/datasets/url-2016.html).
* **Kaggle Phishing Dataset for Machine Learning**: https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning
* **Phishing Websites Dataset from FCSIT, UNIMAS**: https://www.fcsit.unimas.my/phishing-dataset#:~:text=The%20Phishing%20Websites%20Dataset%20contains,different%20folder%20for%20each%20sample
* **Kaggle Website Phishing Dataset**: <https://www.kaggle.com/datasets/ahmednour/website-phishing-data-set>

# **Conclusion:**

For our email phishing detection, we got insights from paper [4]. Similarly, for URL analysis, we referred to papers [8] and [4]. We had access to a labeled dataset for both raw email headers and URLs. This data was instrumental in developing our methodology. We utilized this approach to formulate two distinct phishing detection methodologies, each with a dedicated focus on either email or URL-based threats. These methodologies are outlined below:

# **Proposed Methodologies**

# **Email Phishing Detection Options**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Option** | **Dataset Requirement & gathering** | **System Type** | **User Actions** | **Workflow Steps** | **System Input** | **Expected Output** | **Deliverables** |
| 1 | Combining available internet datasets of email headers | Browser Extension | 1. Open mail,  2.select 'Show Original' from email options,  3.select the header information | 1.Data Collection,  2.Feature Extraction,  3.Preprocessing,  4.Model Training,  5.Model Testing,  6.Browser Extension Development. | Email Header | Phishing or not | Ai Model, Browser Extension |
| 2 | “” | Website | 1.Open mail,  2.select 'Show Original’,  3.copy header information and paste into website | “”  6. Website development | “” | “” | Ai Model, Website |

# **URL Phishing Detection Options**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Option** | **Dataset Requirement & gathering** | **System Type** | **User Actions** | **Workflow Steps** | **System Input** | **Expected Output** | **Deliverables** |
| 1 | Combining available internet datasets of URLs | Browser Extension | Select URL | 1.Data Collection,  2.Feature Extraction,  3.Preprocessing,  4.Model Training,  5.Model Testing,  6.Browser Extension Development. | URL | Phishing or not | Ai Model, Browser Extension |
| 2 | “” | Website | Paste URL into website | “”  6. Website Development. | “” | “” | Ai Model, Website |

Note: "In the tables above, the symbol “” is used to denote that the entry is identical to the one directly above it."

# **Stage 2: Development Plan of ML-Based Phishing Detection Solution (Browser Extensions)**

Following discussions with EBRYX, we decided to focus on developing browser extensions for phishing detection models.

# **Objective**

The objective of Stage 2 is to do the following: construct a system architecture, create two specialized machine learning models tailored for phishing detection—one for emails and another for URLs—and finally, develop two browser extensions based on these models, integrating them into the browser extensions.

# **Technologies:**

The following technologies are utilized (references provided at the end of the document):

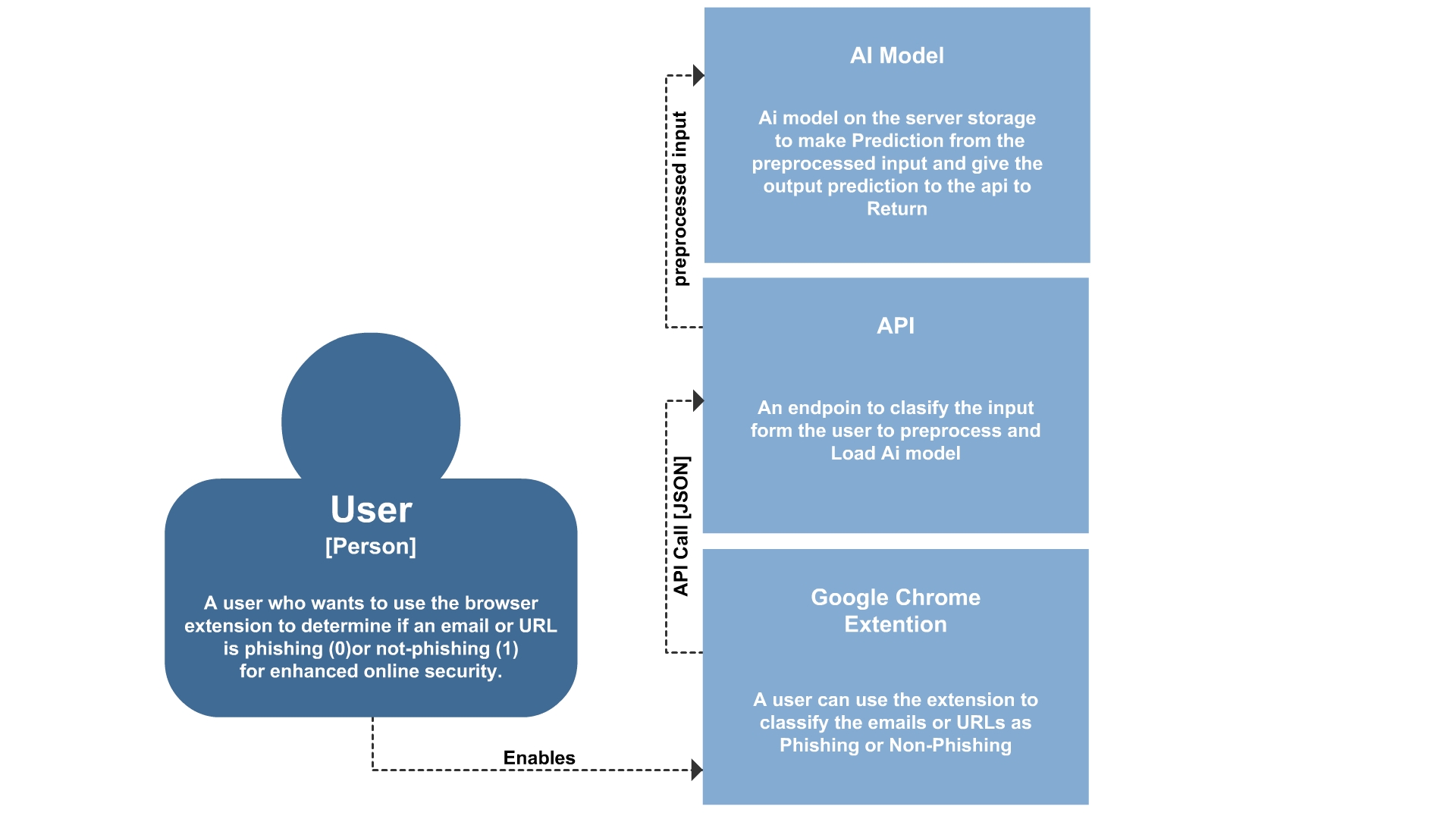
1. **HTML/CSS/JavaScript:** Essential for the browser extension's interface; they provide structure, style, and interactivity. Retrieved from <https://www.w3.org/Style/CSS/> , <https://www.w3.org/html/>
2. **React:** Chosen for its efficient, component-based UI development, enhancing user experience in the extension. Retrieved from <https://react.dev/>
3. **Flask:** Selected for its simplicity and flexibility, ideal for quickly creating the extension's backend API. Retrieved from <https://flask.palletsprojects.com/en/3.0.x/>
4. **Python:** Used for its readability and extensive libraries, particularly suitable for backend development and Machine learning. Retrieved from <https://www.python.org/>
5. **Scikit-learn:** Picked for its comprehensive machine learning tools, crucial for the extension's predictive functionalities. Retrieved from <https://scikit-learn.org/stable/>

**Note:** We will be utilizing the same architecture and flow of interaction for both email phishing detection and URL phishing detection systems, but they will operate as two distinct extensions, however, we are not repeating it twice to avoid repetition.

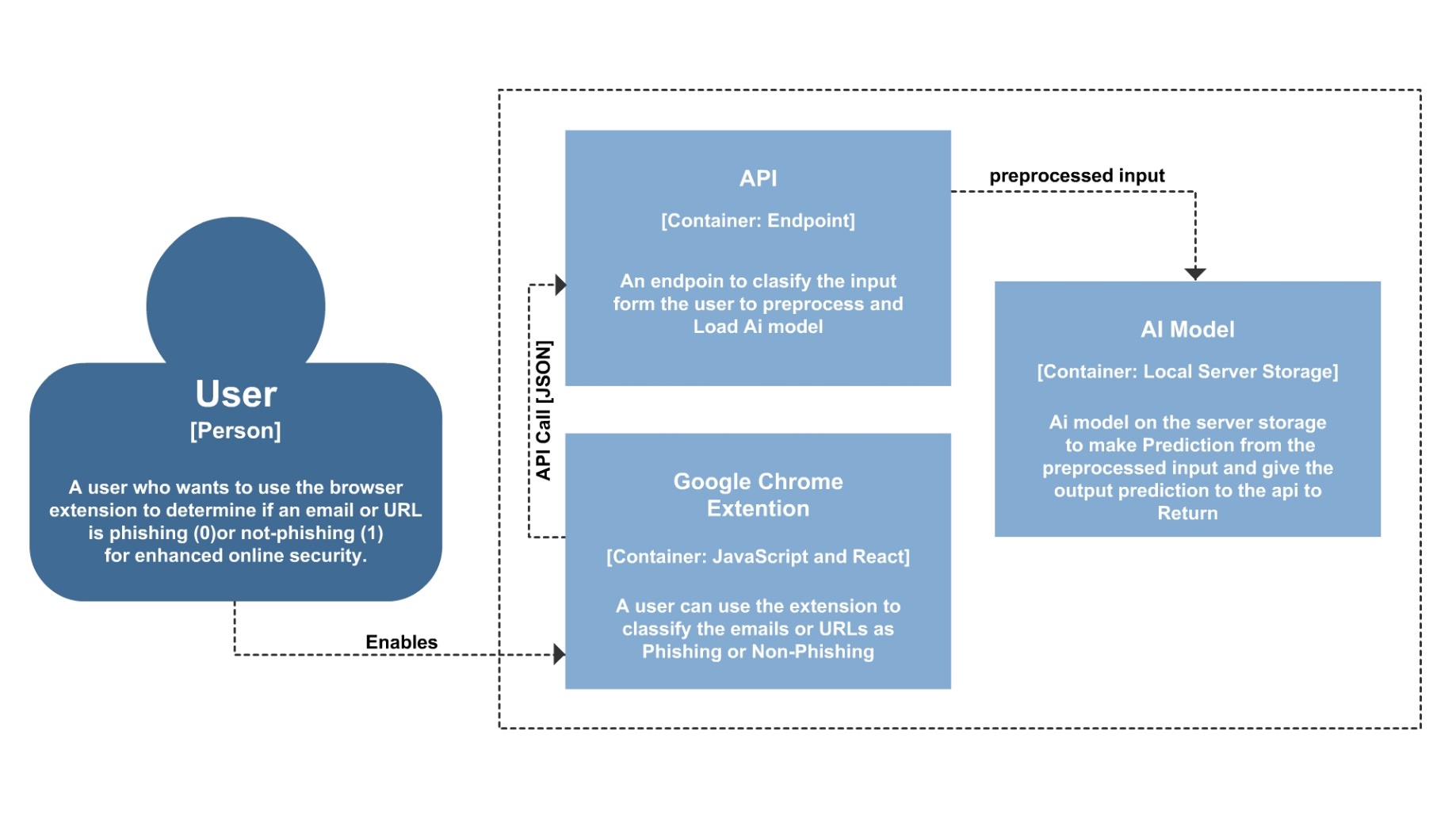
# **System Architecture**

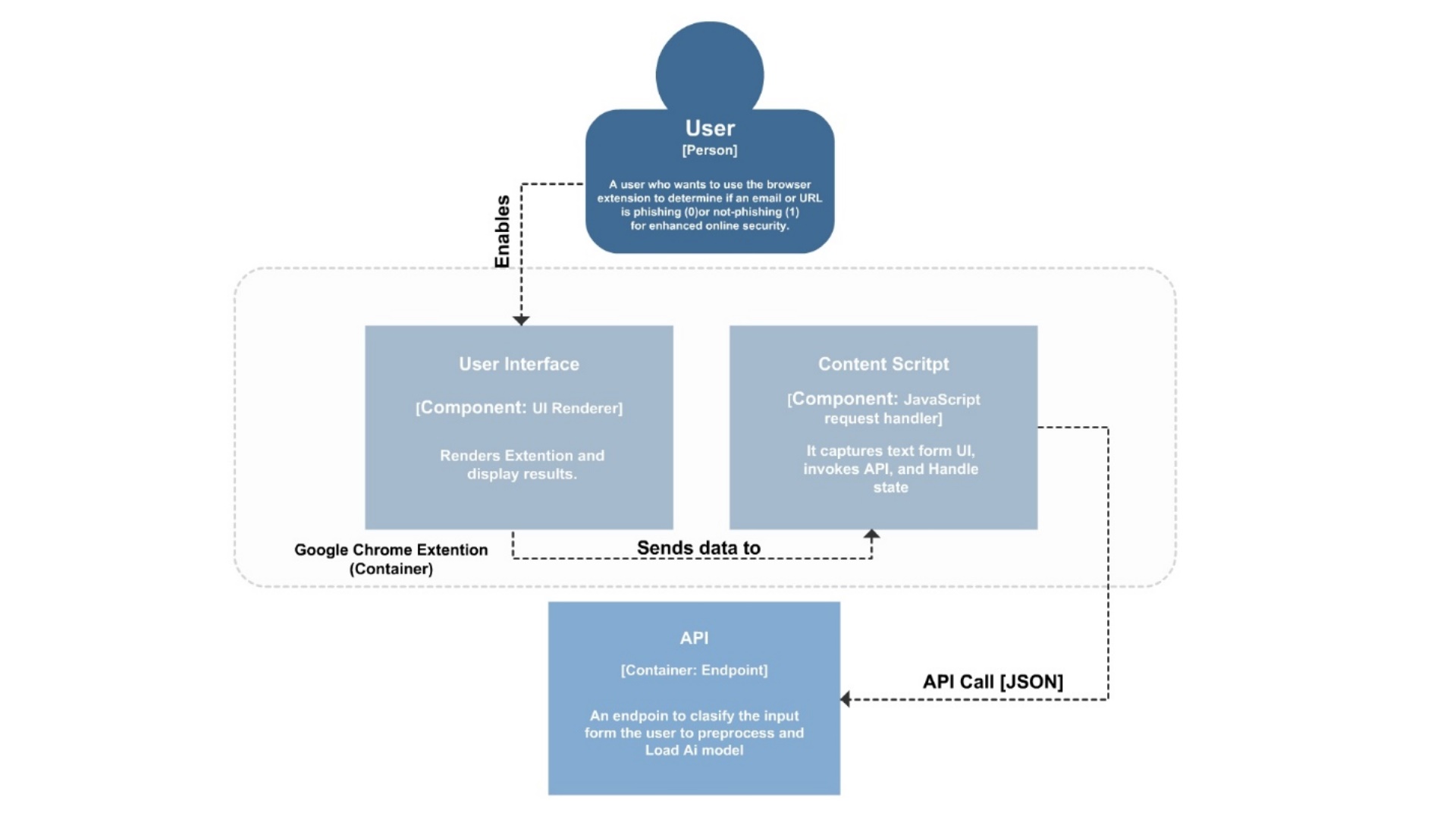
This section includes the context, container and component diagrams for the system:

# **Context Diagram:**



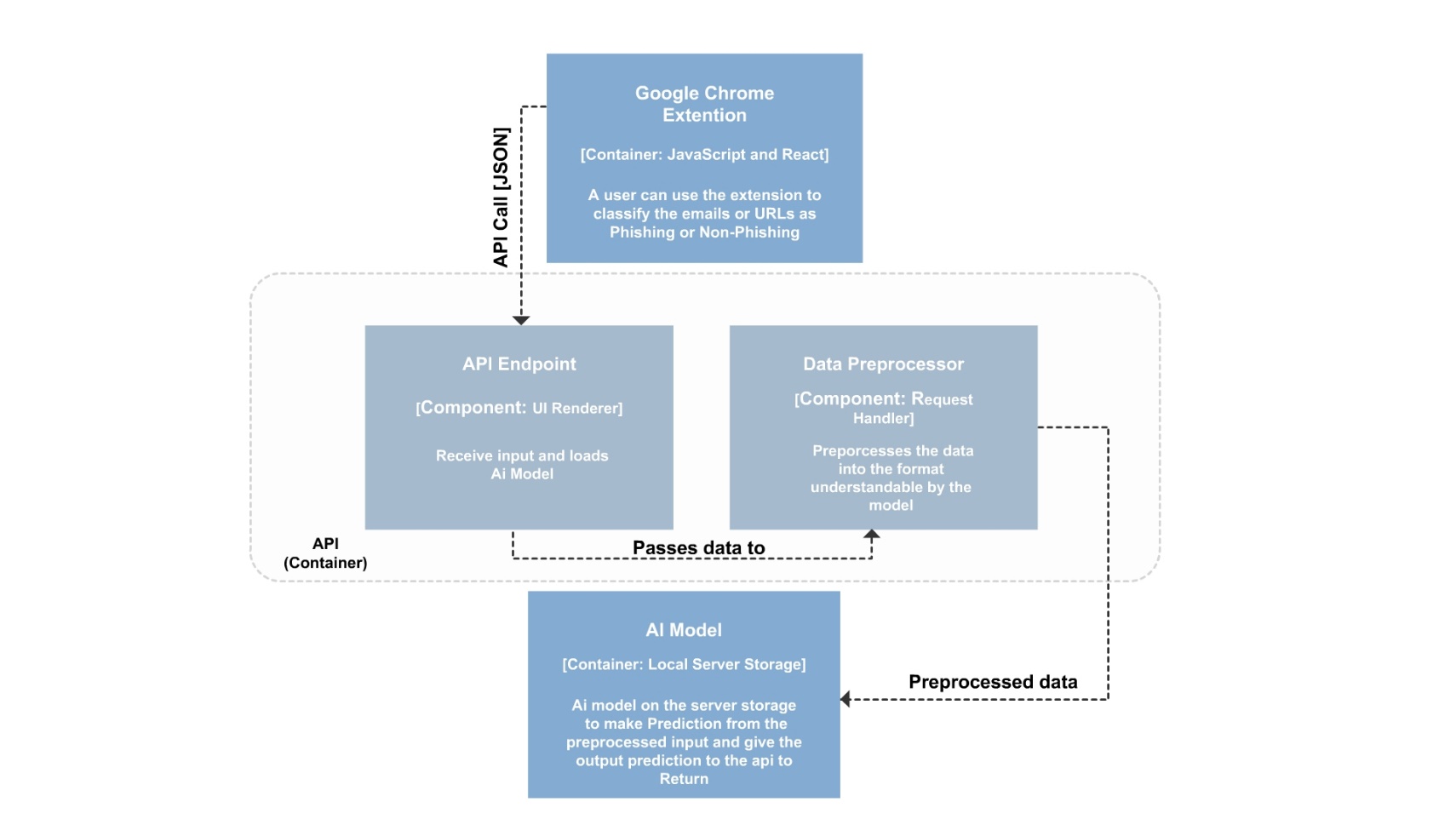
# **Container Diagram:**



**Component Diagrams:**

# **Google Extension Container expanded:**

# **API Container expanded:**



# **Flow of interaction**

Given below is the flow of interaction of the system:

## **User:**

Initiates the process by using the browser extension to check if an email or URL is phishing or not.

**Google Chrome Extension:**

The user interacts with this extension, which captures the user's input (email or URL) when he selects a URL or Email header.

**API:**

The extension sends the input to the API endpoint, in a JSON format, which is designed to classify the input.

**AI Model:**

The API preprocesses the input and then interacts with the AI model, which is stored on a local server to perform the prediction.

**API:**

The AI Model returns the prediction result back to the API.

**Google Chrome Extension:**

The API sends the result back to the extension.

**User:**

The extension displays the prediction result to the user, indicating whether the input was phishing or not.

Note: Our system does not include any class or Entity-Relationship Diagram (ERD) as we do not employ classes and do not store information in a database.

# **API Design:**

Given below is the API design of the system:

## **API**

**Endpoint:** /predict

**Method:** POST

## **Description:**

This endpoint receives a selected text snippet from the browser extension, encapsulated into a JSON structure, to evaluate whether the text represents phishing content.

## **Request Headers**

Content-Type: application/json (Required)

## **Request Body**

{

"text": "The selected text to analyze for phishing."

}

## **Response**

The API will respond with a JSON object that includes whether the text is likely phishing and any relevant details.

## **Success Response**

200 OK

{

"isPhishing": true or false,

"message": "Optional message with more details or suggestions"

}

## **Error Responses**

## **400 Bad Request**

{

"error": "Invalid request. No text provided."

}

## **422 Un-processable Entity**

{

"error": "Text could not be processed for prediction."

}

## **500 Internal Server Error**

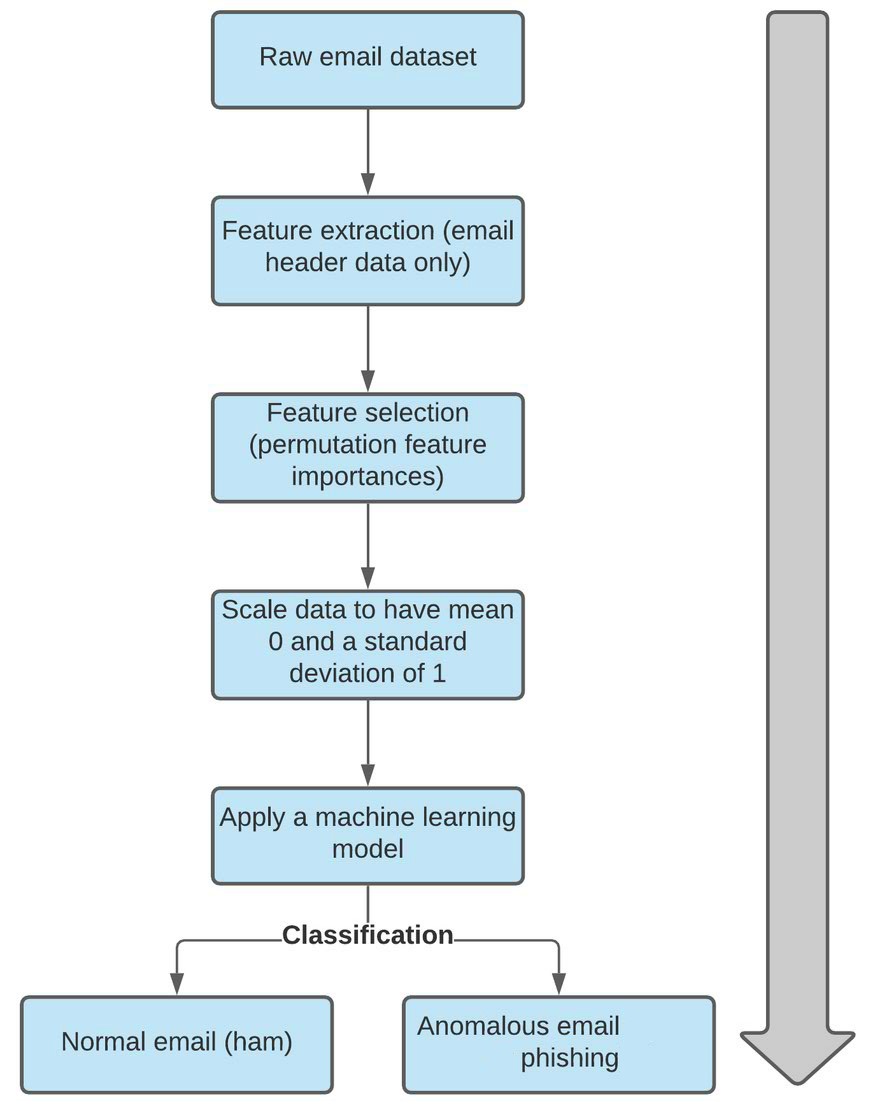
{

"error": "An error occurred while processing the request."

}

# **Email Phishing Detection (MI Component)**

# **Methodology:**



# **Data Collection:**

For the development of the machine learning model targeting email phishing, our approach centered around sourcing raw email headers that are labeled as 'phishy' or 'non-phishy'. This data is critical for feature engineering, which is the foundation of our model's ability to accurately identify potential phishing threats. To build a diverse and representative dataset, we decided to combine various sources.

# **Data Sources:**

* **TREC 2007 Corpus**: <https://plg.uwaterloo.ca/~gvcormac/treccorpus07/about.html>.
* **Fraudulent Email Corpus from Kaggle:** <https://www.kaggle.com/datasets/rtatman/fraudulent-email-corpus/code>
* **Monkey org:** <https://monkey.org/~jose/phishing/>

# **Example Data**

The dataset examples, both before and after feature engineering, has been zipped and included with the document due to its length and complexity, making it impractical to include directly in the document.

# **Feature Engineering**

Email headers serve as a rich source of information, having details about the origin, recipients, and journey of an email message. By looking into these fields, valuable features can be extracted, allowing for the identification of potential malicious and phishy behaviors. Each field contributes distinct information, telling us about the authenticity and intent behind an email.

These are among the header fields from which we extracted valuable information in feature extraction:

## **From:**

* Represents the sender of the email.
* Mandatory field that should be present in each email.
* The absence of the 'From' field may be indicative of spamming behavior.

## **To and Cc:**

* Indicates the recipient(s) of the email.
* The email can be sent to one or many recipients.
* The message should have at least one recipient address in either the 'To' or 'Cc' fields.

## **Received:**

* Contains information about servers that received and sent the message during its journey.

## **Return-Path:**

* Added by the final transport system that delivers the message to the recipients.
* Contains information about the address and the route back to the message originator.

**Date:**

* Indicates the date and time at which the message was sent, including the time zone.
* Added once the user submits the message.

## **Reply-To:**

* Defines the email address automatically inserted into the 'To' field when a user replies to an email message.

## **Error-To:**

* Contains the address to which notifications are to be sent.
* Includes a request to receive delivery notifications.

## **Sender:**

* Inserted by some systems if the actual sender is different from the text in the 'From' field.
* The address in the 'Sender' field represents an authenticated user or system.

## **References and In-Reply-To:**

* Identification fields for other correspondence.
* Hold the message identifier of the original and other messages when creating a reply.

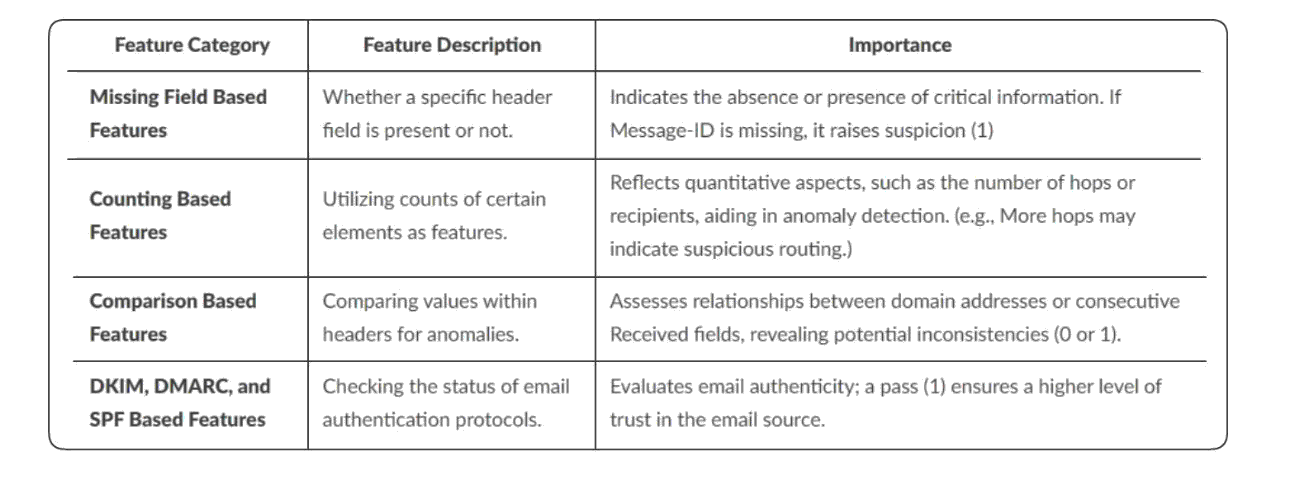
## **Message-ID (Optional):**

* A unique ID generated by the system for each message when it is first created.
* Can be useful in fault tracing if multiple copies of a message have been received.
* The domain of 'Message-ID' should generally match the domain in the 'From' field. Mismatching domains may be indicative of spamming behavior.

Our plan involved creating multiple features from these fields and organizing them into the categories discussed below. We incorporated insights from the literature we surveyed and integrated additional features based on our own considerations.

# **Feature Categories:**

We categorized the features into the following groups:



# **Missing Features:**

To enhance the understanding of email headers and identify potential anomalies, binary indicators (1 for anomaly, 0 for non-anomaly) denoting the presence or absence of specific fields were created. The absence of certain headers can be a significant marker of suspicious activity, such as spamming or phishing. Each missing field could suggest a deliberate attempt to avoid detection or hide the email's origin, making these indicators crucial for security analysis. The following features were engineered to capture this missing information:

* **missing\_from:** Binary indicator denoting the absence (0) or presence (1) of the 'From' field.
* **missing\_to:** Binary indicator denoting the absence (0) or presence (1) of the 'To' field.
* **missing\_cc:** Binary indicator denoting the absence (0) or presence (1) of the 'Cc' field.
* **missing\_received:** Binary indicator denoting the absence (0) or presence (1) of the 'Received' field.
* **missing\_return-path:** Binary indicator denoting the absence (0) or presence (1) of the 'Return-Path' field.
* **missing\_reply-to:** Binary indicator denoting the absence (0) or presence (1) of the 'Reply-To' field.
* **missing\_errors-to:** Binary indicator denoting the absence (0) or presence (1) of the 'Errors-To' field.
* **missing\_sender:** Binary indicator denoting the absence (0) or presence (1) of the 'Sender' field.
* **missing\_references:** Binary indicator denoting the absence (0) or presence (1) of the 'References' field.
* **missing\_in-reply-to:** Binary indicator denoting the absence (0) or presence (1) of the 'In-Reply-To' field.
* **missing\_message-id:** Binary indicator denoting the absence (0) or presence (1) of the 'Message-ID' field.

# **Numerical Features (counting based):**

Numerical features were created to offer additional insights into the email structure, aiding in phishing detection. These numerical indicators play a crucial role in phishing detection by providing quantitative metrics that can reveal patterns or anomalies indicative of potential phishing attempts. For instance, an unusually high number of hops or an unexpected distribution of recipients could signify malicious intent. These features include:

* **num\_hops:** Represents the number of servers the email passed through.
* **num\_recipients\_to:** Number of recipients in the 'To' field.
* **num\_recipients\_cc:** Number of recipients in the 'Cc' field.
* **num\_recipients\_from:** Number of recipients in the 'From' field.

# **Domain Matching Features (comparison based):**

To capture relationships between different email headers, domain matching features were created. For each domain pair, a binary feature indicates whether the domains match (1) or not (0). This approach provides a concise representation of domain relationships, facilitating the analysis of email header connections. Importantly, these features play a crucial role in phishing detection, as anomalies in domain matching may signify potential phishing attempts. Domain pairs include:

* **domain\_match\_message-id\_from:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'message-id' and 'from' headers.
* **domain\_match\_from\_return-path:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'from' and 'return-path' headers.
* **domain\_match\_message-id\_return-path:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'message-id' and 'return-path' headers.
* **domain\_match\_message-id\_sender**: Binary indicator indicating either a match (1) or mismatch (0) of domains in 'message-id' and 'sender' headers.
* **domain\_match\_message-id\_reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'message-id' and 'reply-to' headers.
* **domain\_match\_return-path\_reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'return-path' and 'reply-to' headers.
* **domain\_match\_reply-to\_to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'reply-to' and 'to' headers.
* **domain\_match\_to\_in-reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'to' and 'in-reply-to' headers.
* **domain\_match\_errors-to\_message-id:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'errors-to' and 'message-id' headers.
* **domain\_match\_errors-to\_from:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'errors-to' and 'from' headers.
* **domain\_match\_errors-to\_sender:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'errors-to' and 'sender' headers.
* **domain\_match\_errors-to\_reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'errors-to' and 'reply-to' headers.
* **domain\_match\_sender\_from:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'sender' and 'from' headers.
* **domain\_match\_references\_reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'references' and 'reply-to' headers.
* **domain\_match\_references\_in-reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'references' and 'in-reply-to' headers.
* **domain\_match\_references\_to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'references' and 'to' headers.
* **domain\_match\_from\_reply-to:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'from' and 'reply-to' headers.
* **domain\_match\_to\_from**: Binary indicator indicating either a match (1) or mismatch (0) of domains in 'to' and 'from' headers.
* **domain\_match\_to\_message-id:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'to' and 'message-id' headers.
* **domain\_match\_to\_received:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'to' and 'received' headers.
* **domain\_match\_reply-to\_received:** Binary indicator indicating either a match (1) or mismatch (0) of domains in 'reply-to' and 'received' headers.

# **Authentication Features:**

To assess the authenticity of an email, three binary features were created, indicating whether specific authentication mechanisms pass or not. The convention follows 1 for 'pass' and 0 for non-authentication, providing a clear indication of the verification status. This approach is vital for preventing email spoofing, phishing attacks, and unauthorized access, empowering users to make informed decisions about the legitimacy of incoming emails. The features are as follows:

* **pass\_dkim:** Evaluates if the 'dkim' field indicates a successful authentication ('pass').
* **pass\_spf:** Evaluates if the 'spf' field indicates a successful authentication ('pass').
* **pass\_dmarc:** Evaluates if the 'dmarc' field indicates a successful authentication ('pass').

# **Feature Importance**

## The determination of feature importance involved an approach that included permutation importance calculation, aggregated metrics, and a ranked feature presentation.

## Firstly, permutation importance was computed for different models including Random Forest, SVM, and MLP models and others, identifying the top features for each model individually. Subsequently, aggregated importance metrics were established through a counter dictionary, facilitating the aggregation of feature importance across all models. Cumulative importance was then calculated based on the top features from each model, ensuring a comprehensive evaluation.

## The final step involved presenting the results in a structured manner. The aggregated results were sorted to provide a clear understanding, and a ranked list of the most influential features was generated.

# **Model Short-Listing**

To develop a stacked ensemble model, we initially shortlisted base learners based on their individual performance metrics. Random Forest (RF), Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) were selected for their superior validation results.

Logistic Regression (LR) was chosen as the meta-learner for its effectiveness in interpreting and integrating the predictions from the base learners, thereby enhancing the final decision-making process.

# **Training**

During the training phase, 80% of the dataset was used to train the StackingClassifier, a method that leverages the collective strengths of various algorithms. We structured our approach into four distinct stacks:

* **BaseLearner\_S1:** Consists of Random Forest (RF), Multilayer Perceptron (MLP), and K-Nearest Neighbors (KNN).
* **BaseLearner\_S2:** Comprises Random Forest (RF), Multilayer Perceptron (MLP), and Support Vector Machine (SVM).
* **BaseLearner\_S3:** Includes Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).
* **BaseLearner\_S4:** Features Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).

Each base learner stack was trained independently, and the meta-learner was trained to effectively combine their strengths to improve the overall prediction accuracy.

# **Testing and Evaluation**

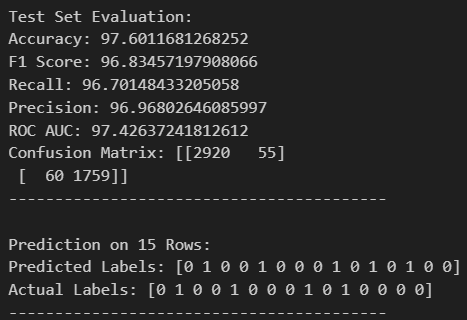
Testing was performed on previously unseen test data, divided into 20% of the dataset before training. Each base learner stack (BaseLearner\_S1, BaseLearner\_S2, BaseLearner\_S3, BaseLearner\_S4) was individually tested, and the subsequent test evaluation results for each base learner are as follows:

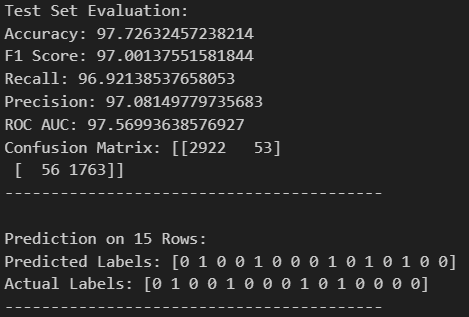
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **---** | **Accuracy** | **F1\_Score** | **Recall** | **Precision** |
| **BL\_S1** | 97.72 | 97.00 | 96.92 | 97.08 |
| **BL\_S2** | 97.68 | 96.94 | 96.81 | 97.07 |
| **BL\_S3** | 97.51 | 96.70 | 96.15 | 97.27 |
| **BL\_S4** | 97.60 | 96.83 | 96.70 | 96.96 |

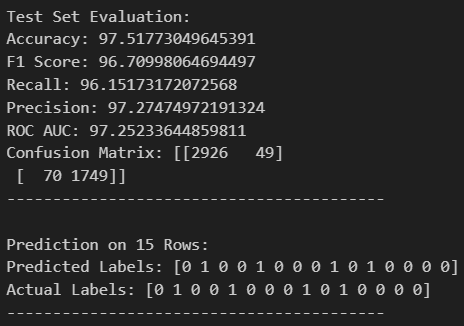
# **Demo and Results**

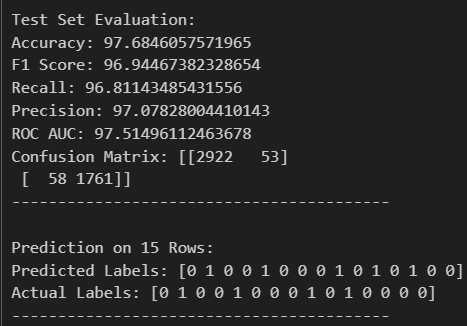
The provided screenshots indicate the performance of various base learners in a stacked ensemble model tested on unseen data. Each base learner's metrics—accuracy, F1 score, recall, and precision—are documented, showcasing their individual contributions to the ensemble's overall performance.

A specific set of 15 rows was also independently predicted to demonstrate the model's practical application.









# **Evaluation matrix**

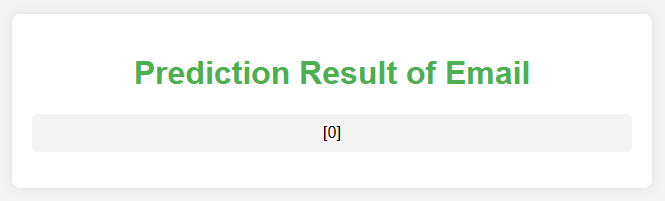
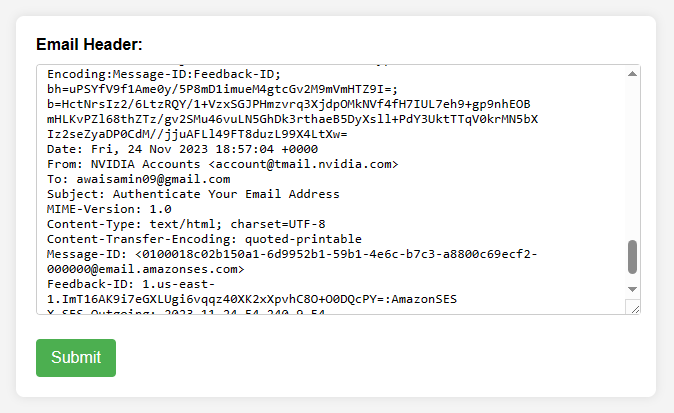
As advised by EBRYX, our evaluation includes accuracy and precision, although in our project, we have also included other metrics such as recall and F1 score.

# **Local Deployment**

We initiated the local deployment process by creating a dummy website designed, for our convenience until the development of our browser extension, which included a form allowing us to input relevant email header data. This website was built with Flask powering the backend, exposing APIs to facilitate communication between the front end and the server. Upon receiving user inputs, the system underwent preprocessing steps to ensure proper formatting for subsequent analysis. Following the preprocessing phase, the system invoked a machine learning model to predict whether the provided email header was indicative of a phishing attempt (1) or not (0). Following the machine learning model prediction, the results are dynamically showcased on the website, indicating whether the provided email header is identified as a phishing attempt (1) or not (0).

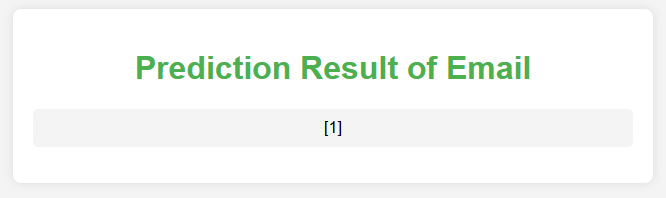
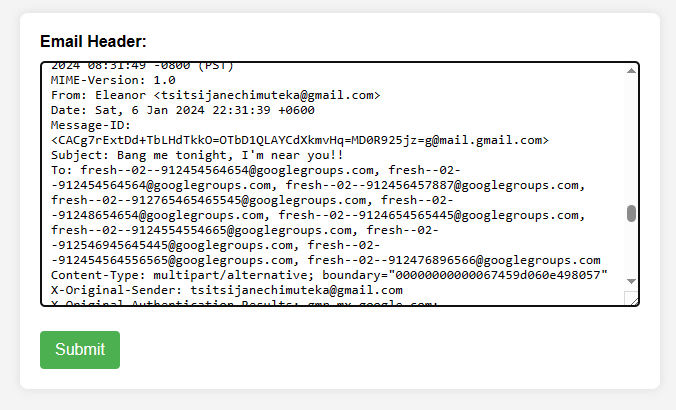
# **Prediction on non-phishing header (0):**

We generated predictions using our own emails we collected for the analysis.



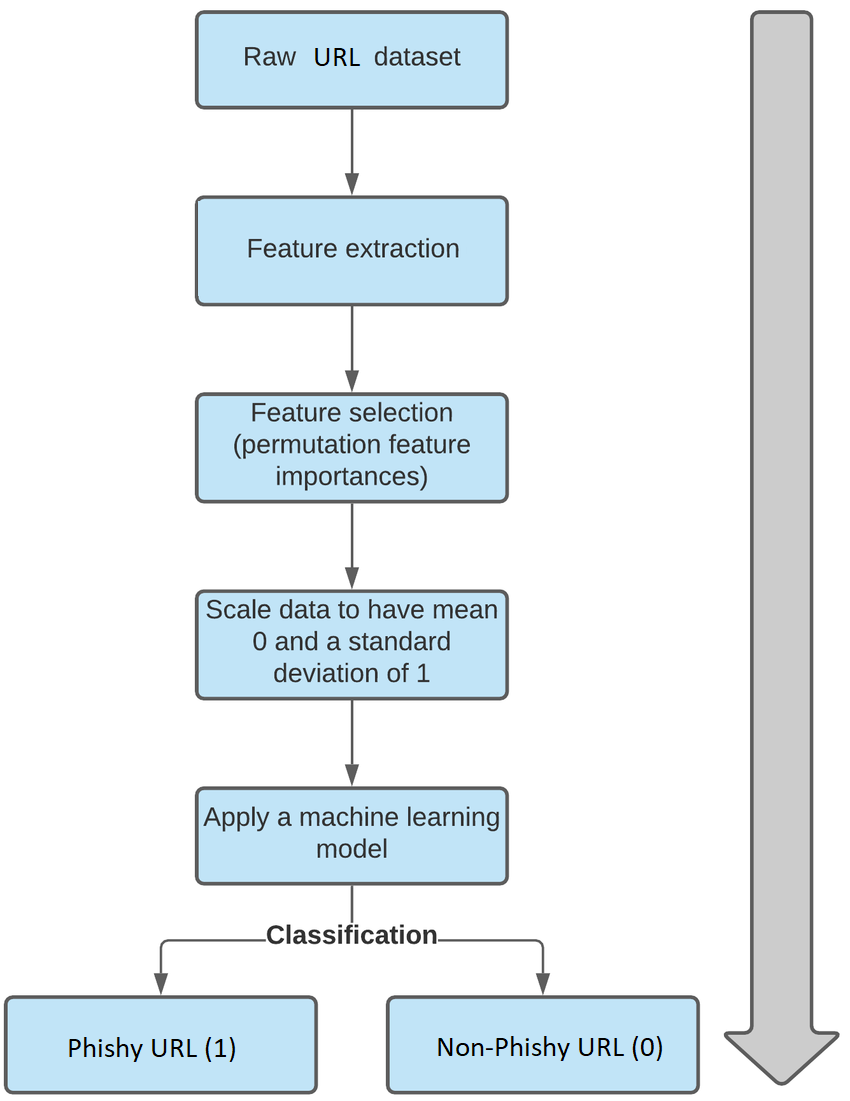
# **Prediction on phishing Email header (1):**

We generated predictions using our own emails we collected for the analysis.



# **URL Phishing Detection (ML Component)**

# **Methodology:**



# **Data Collection:**

For the development of the machine learning model targeting URL phishing, our approach centered around sourcing raw URLS that are labeled as 'phishy' or 'non-phishy'. This data is critical for feature engineering, which is the foundation of our model's ability to accurately identify potential phishing threats. To build a diverse and representative dataset, we decided to combine various sources.

# **Data Sources:**

* **PhishTank**: [PhishTank](https://www.phishtank.com/developer_info.php).
* The University of New Brunswick's open datasets: [UNB - URL Dataset](https://www.unb.ca/cic/datasets/url-2016.html).
* **Kaggle Phishing Dataset for Machine Learning**: <https://www.kaggle.com/datasets/shashwatwork/phishing-dataset-for-machine-learning>
* **Phishing Websites Dataset from FCSIT, UNIMAS**: <https://www.fcsit.unimas.my/phishing-dataset#:~:text=The%20Phishing%20Websites%20Dataset%20contains,different%20folder%20for%20each%20sample>
* **Kaggle Website Phishing Dataset**: <https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset>

# **Example Data**

The dataset examples, both before and after feature engineering, has been compressed and included with the document due to its length and complexity, making it impractical to include directly in the document.

# **Feature Engineering**

Feature engineering in URL phishing detection involves the extraction of valuable information from the URL to discern patterns and characteristics indicative of phishing behavior. This approach allows us to transform raw URL data into meaningful features, enhancing our ability to detect potential phishing threats.

Our plan involved creating multiple features and organizing them into the categories discussed below. We incorporated insights from the literature we surveyed and integrated additional features based on our own considerations.

# **Feature Categories:**

We categorized the features into the following groups:

# 

# **Address Bar Based Features):**

These features assess characteristics derived from the URL address bar, generating binary indicators (1 or 0) to highlight specific attributes. Their importance lies in detecting patterns associated with phishing attempts and revealing potential malicious intent

## **Have\_IP:**

* **Usage:** Is 1 if the URL contains an IP address, and 0 otherwise.
* **Importance:** Detects if the URL contains an IP address, which could be indicative of phishing.

## **Have\_At:**

* **Usage:** Is 1 if the URL contains the "@" symbol, and 0 otherwise.
* **Importance:** Identifies URLs that may misuse the "@" symbol, a common tactic in phishing.

## **URL\_Length:**

* **Usage:** Is 1 if the length of the URL is greater than or equal to 54 characters, and 0 otherwise.
* **Importance:** Longer URLs may be associated with phishing attempts, making this a useful length-based indicator.

## **URL\_Depth:**

* **Usage:** Is the depth of the URL path.
* **Importance:** Provides insight into the structure of the URL, aiding in the detection of suspicious patterns.

## **Redirection:**

* **Usage:** Is 1 if there is a redirection in the URL, and 0 otherwise.
* **Importance:** Identifies URLs with redirection, a common technique in phishing attacks.

## **https\_Domain:**

* **Usage:** Is 1 if the domain part of the URL contains "https", and 0 otherwise.
* **Importance:** Detects whether the URL uses secure HTTP, a potential indicator of legitimacy.

## **TinyURL:**

* **Usage:** Is 1 if the URL is a shortened URL, and 0 otherwise.
* **Importance:** Flags URLs using shortening services, often associated with phishing.

## **Prefix/Suffix:**

* **Usage:** Is 1 if the domain part of the URL contains a hyphen (-), and 0 otherwise.
* **Importance:** Identifies URLs with unconventional domain structures, a potential sign of phishing.

# **Domain Based Features:**

These features focus on characteristics related to the domain of the URL, utilizing binary indicators (1 or 0) to evaluate specific attributes. They play a crucial role in assessing the legitimacy and potential risks associated with the domain.

## **DNS\_Record:**

* **Usage:** Checks the presence of a DNS record for the domain**.**
* **Importance:** Determines whether the domain has a valid DNS record, a key aspect in assessing legitimacy**.**

## **Domain\_Age:**

* **Usage**: Is 1 if the age of the domain is less than 6 months, and 0 otherwise.
* **Importance:** Older domains are often more trustworthy; hence, age is a factor in phishing detection.

## **Domain\_End:**

* **Usage:** Is 1 if the time until the domain expiration is less than 6 months, and 0 otherwise.
* **Importance:** Expiring domains might be associated with malicious activities; hence, this is a temporal indicator.

# **HTML & Javascript Based Features:**

These features analyze attributes derived from HTML and Javascript, providing binary indicators (1 or 0) to highlight specific behaviors. They are instrumental in uncovering potential malicious actions within the webpage.

## **iFrame:**

* **Usage:** Is 1 if there is an IFrame redirection, and 0 otherwise.
* **Importance**: Detects if the URL uses iframes, a common technique in phishing for hiding malicious content.

## **Mouse\_Over:**

* **Usage:** Is 1 if there is a mouse-over effect on the status bar, and 0 otherwise.
* **Importance:** Identifies if mouse-over actions may lead to malicious behaviors.

## **Right\_Click:**

* **Usage:** Is 1 if the right-click attribute is enabled, and 0 otherwise.
* **Importance:** Detects if the right-click functionality is restricted, which can be a sign of phishing.

## **Web\_Forwards:**

* **Usage:** Is 1 if there are more than 2 forwardings in the response, and 0 otherwise.
* **Importance:** Identifies URLs with multiple forwardings, a potential indicator of phishing.

## The determination of feature importance involved a approach that included permutation importance calculation, aggregated metrics, and a ranked feature presentation.

## Firstly, permutation importance was computed for different models including Random Forest, SVM, and MLP models and others, identifying the top features for each model individually. Subsequently, aggregated importance metrics were established through a counter dictionary, facilitating the aggregation of feature importance across all models. Cumulative importance was then calculated based on the top features from each model, ensuring a comprehensive evaluation.

## The final step involved presenting the results in a structured manner. The aggregated results were sorted to provide a clear understanding, and a ranked list of the most influential features was generated.

# **Model Short-Listing**

To develop a stacked ensemble model, we initially shortlisted base learners based on their individual performance metrics. Random Forest (RF), Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) were selected for their superior validation results.

Logistic Regression (LR) was chosen as the meta-learner for its effectiveness in interpreting and integrating the predictions from the base learners, thereby enhancing the final decision-making process.

# **Training:**

During the training phase, 80% of the dataset was used to train the StackingClassifier, a method that leverages the collective strengths of various algorithms. We structured our approach into four distinct stacks:

* **BaseLearner\_S1:** Consists of Random Forest (RF), Multilayer Perceptron (MLP), and K-Nearest Neighbors (KNN).
* **BaseLearner\_S2:** Comprises Random Forest (RF), Multilayer Perceptron (MLP), and Support Vector Machine (SVM).
* **BaseLearner\_S3:** Includes Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).
* **BaseLearner\_S4:** Features Multilayer Perceptron (MLP), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM).

Each base learner stack was trained independently, and the meta-learner was trained to effectively combine their strengths to improve the overall prediction accuracy

# **Testing and Evaluation:**

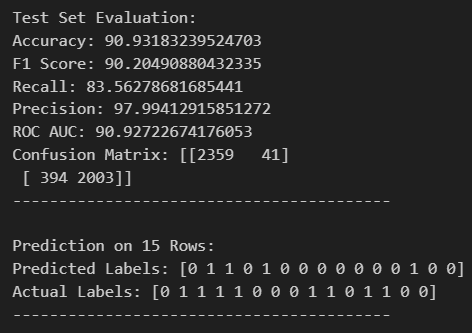
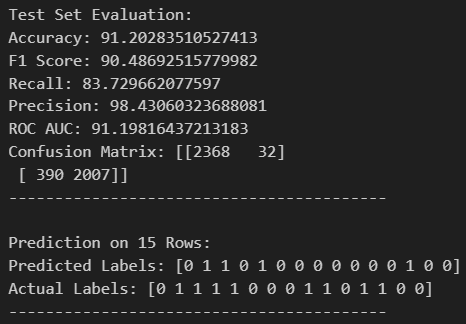
Testing was performed on previously unseen test data, divided into 20% of the dataset before training. Each base learner stack (BaseLearner\_S1, BaseLearner\_S2, BaseLearner\_S3, BaseLearner\_S4) was individually tested, and the subsequent test evaluation results for each base learner are as follows:

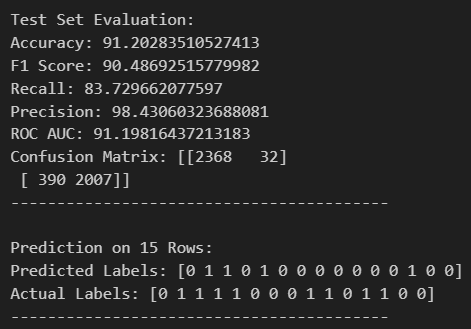
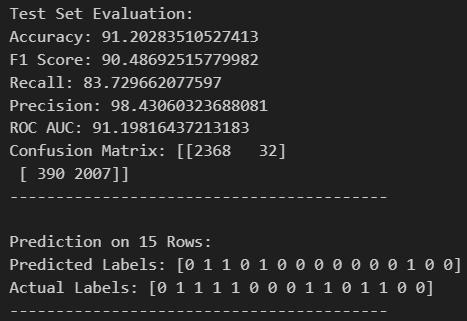
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **---** | **Accuracy** | **F1\_Score** | **Recall** | **Precision** |
| **BL\_S1** | 91.20 | 90.40 | 83.70 | 98.43 |
| **BL\_S2** | 91.20 | 90.48 | 83.70 | 98.44 |
| **BL\_S3** | 91.20 | 90.48 | 83.72 | 98.43 |
| **BL\_S4** | 90.93 | 90.20 | 96.56 | 97.99 |

# **Demo and Results**

The provided screenshots indicate the performance of various base learners in a stacked ensemble model tested on unseen data. Each base learner's metrics—accuracy, F1 score, recall, and precision—are documented, showcasing their individual contributions to the ensemble's overall performance.

A specific set of 15 rows was also independently predicted to demonstrate the model's practical application.





# **Evaluation matrix**

As advised by EBRYX, our evaluation includes accuracy and precision, although in our project, we have also included other metrics such as recall and F1 score.

# **Enhancing URL Phishing Detection Model**

# Although the model showed high accuracy on test datasets, however, when we extended our testing to include real-world data from our browsing histories, the performance was not as expected. This led us to conduct a thorough analysis to identify and address the underlying issues.

# **Analysis: Challenges and Solutions**

Following are the challenges and their solutions we identified in our analysis:

## **Challenge 1: Inconsistent Performance on Real-World Data**

# **Problem:** Although the AI model showed high accuracy on test datasets, its performance significantly dropped when applied to browsing history of our own, achieving only an 31.9% accuracy rate.

# **Solution:** We expanded our dataset to include a more diverse set of URLs by scraping and exporting the browsing history from multiple Google accounts and devices. This approach aimed to train our model on a dataset that better represents the variety of URLs encountered in real-world browsing.

## **Challenge 2: Inadequate Data Preprocessing**

# **Problem:** Initial preprocessing steps resulted in inconsistent outputs, which impacted the model's learning process negatively.

# **Solution:** To address inconsistencies, we systematically reviewed and tested each feature individually to correct any discrepancies that occurred.

## **Challenge 3: Incorrectly Labeled Dataset**

# **Problem:** The dataset initially used for training was found to have inaccuracies in its labeling, particularly some instances were mislabeled regarding their phishing status.

# **Source:** https://www.kaggle.com/datasets/sid321axn/malicious-urls-dataset

# **Solution:** We switched to a more reliable dataset for phishing website detection (with the same instances labelled correctly)

# **Source:** Ariyadasa, Subhash; Fernando, Shantha; Fernando, Subha (2021), “Phishing Websites Dataset”, Mendeley Data, V1, doi: 10.17632/n96ncsr5g4.1

## **Challenge 4: Inefficient Data Processing**

# **Problem:** The preprocessing of URLs was initially time-consuming, as each URL had to be individually processed, leading to significant delays.

# **Solution:** To accelerate data processing, we implemented multithreading, launching multiple threads to preprocess data in parallel. This approach significantly reduced the preprocessing time, enabling us to handle larger datasets more efficiently.

## **Challenge 5: Data Corruption Due to File Locking**

# **Problem:** When processing data in parallel, issues with file locking led to data corruption or loss, as multiple threads attempted to write to the same file.

# **Solution:** We modified our approach to have each thread generate its output in a separate CSV file. This eliminated the issues with file locking and data corruption. After preprocessing, we combined all the separate files into a single dataset for training the model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **---** | **Accuracy** | **F1\_Score** | **Recall** | **Precision** |
| **BL\_S1** | 87.28 | 84.34 | 81.98 | 86.85 |
| **BL\_S2** | 87.28 | 85.10 | 83.70 | 85.55 |
| **BL\_S3** | 90.20 | 84.34 | 83.72 | 86.69 |
| **BL\_S4** | 90.93 | 84.34 | 81.98 | 86.85 |

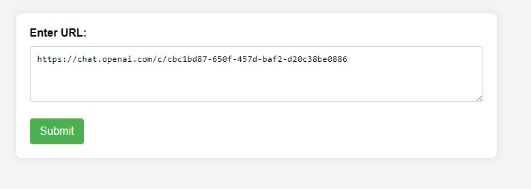
# **Updated Testing and Evaluation:**

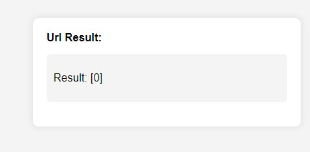
# **Local Deployment**

We initiated the local deployment process by creating a dummy website designed, for our convenience until the development of our browser extension, which included a form allowing us to input URL data. This website was built with Flask powering the backend, exposing APIs to facilitate communication between the front end and the server. Upon receiving user inputs, the system underwent preprocessing steps to ensure proper formatting for subsequent analysis. Following the preprocessing phase, the system invoked a machine learning model to predict whether the provided URL was indicative of a phishing attempt (1) or not (0). Following the machine learning model prediction, the results are dynamically showcased on the website, indicating whether the provided URL is identified as a phishing attempt (1) or not (0).

# **Prediction on non-phishing header (0):**

We generated predictions using our own browsing history (URLS) we collected for the analysis.





# **Prediction on non-phishing header (0):**

We generated predictions using our own browsing history (URLS) we collected for the analysis.

# 

# 

# 