AI PHISHING DETECTION TOOL

by

Asif Ad-Deen Udatta
21301630
Naimur Rahman Nahid
21301709
Shafayet Noor Sifat
21341002
Raiyan Rafiz Anoy
24141230
Sayeed Bin Miraj
24241329

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

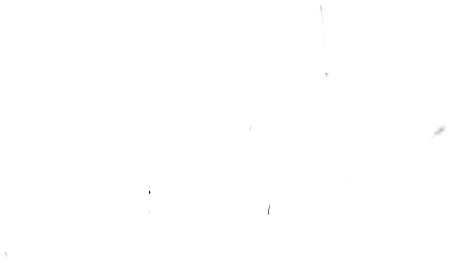
Department of Computer Science and Engineering
Brac University
January 2026

Declaration

It is hereby declared that

1. The project submitted is my/our own original work while completing degree at Brac University.
2. The project does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The project does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Asif Ad-Deen Udatta
21301630

Naimur Rahman Nahid
21301709

Shafayet Noor Sifat
21341002

Raiyan Rafiz Anoy
24141230

Sayeed Bin Miraj
24241329

Approval

The project titled "Ai phishing detection tool " submitted by

1. Asif Ad-Deen Udatta (21301630)
2. Naimur Rahman Nahid (21301709)
3. Shafayet Noor Sifat (21341002)
4. Raiyan Rafiz Anoy (24141230)
5. Sayeed Bin Miraj (24241329)

Of Fall, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January, 2026.

Examining Committee:

Supervisor:
(Member)

Imran Zahid
Lecturer
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Chair)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Ethics Statement (Optional)

This is optional, if you don't have an ethics statement then omit this page

Abstract

Phishing attacks remain one of the most critical cybersecurity threats, exploiting malicious URLs to deceive users and extract sensitive information. Traditional detection approaches, such as blacklist-based and rule-based systems, are increasingly ineffective against rapidly evolving and previously unseen phishing techniques. To address this challenge, this study proposes a hybrid deep learning framework that integrates a supervised transformer-based model (RoBERTa) with an unsupervised Autoencoder for anomaly detection using structured URL features. The RoBERTa model captures contextual and semantic patterns from raw URLs, while the Autoencoder identifies deviations through reconstruction error, enabling detection of unknown phishing behaviors. The framework is evaluated using standard metrics including accuracy, precision, recall, F1-score, and AUC, along with cross-validation to ensure robustness. Experimental results demonstrate that the hybrid model achieves near-perfect performance, significantly outperforming individual approaches while maintaining strong generalization capability. The combination of supervised and unsupervised learning enhances detection reliability and reduces false positives. Overall, this research provides an effective, scalable, and adaptive solution for real-world phishing detection in dynamic cybersecurity environments.

Keywords: Phishing Detection, Deep Learning, RoBERTa, Autoencoder, Hybrid Model, Transformer Models, Anomaly Detection, URL Classification, Cybersecurity.

Dedication (Optional)

A dedication is the expression of friendly connection or thanks by the author towards another person. It can occupy one or multiple lines depending on its importance. You can remove this page if you want.

Acknowledgement

We express our sincere gratitude to my supervisor for their continuous guidance, support, and valuable feedback throughout this investigation. We are also grateful to our institution for providing the necessary resources and environment to complete this work. Finally, we would like to acknowledge our family and friends for their encouragement and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Phishing has emerged as one of the most critical cybersecurity threats, leading to significant financial loss, identity theft, and large-scale data breaches. These attacks commonly exploit malicious URLs that redirect users to fraudulent websites designed to mimic legitimate services and deceive users into disclosing sensitive information such as login credentials and financial details. The rapid evolution and large-scale generation of phishing URLs make detection increasingly challenging for modern security systems.

Traditional phishing detection techniques, such as blacklist-based filtering and rule-based classifiers, are limited in their effectiveness. These approaches rely heavily on previously known attack patterns and require continuous manual updates, making them incapable of detecting zero-day or previously unseen phishing attacks. As a result, there is a growing demand for intelligent, adaptive, and data-driven detection mechanisms that can generalize beyond known attack signatures.

To address these limitations, this study explores a deep learning-based phishing detection framework that leverages both supervised and unsupervised learning paradigms. Specifically, the proposed approach integrates a transformer-based model (RoBERTa) for contextual understanding of URLs and an Autoencoder for anomaly detection using structured URL features. By combining these complementary techniques, the framework aims to capture both known phishing patterns and previously unseen anomalies, providing a more robust and scalable solution for real-world phishing detection.

1.1 Problem Statement

Phishing detection remains a complex problem due to the dynamic and adaptive nature of malicious URLs. Attackers continuously modify URL structures to evade detection systems, rendering traditional approaches such as blacklists and rule-based classifiers ineffective. These systems depend on predefined signatures and historical data, which limits their ability to identify newly emerging phishing attacks.

Deep learning offers a promising solution by enabling models to automatically learn complex patterns from data. Supervised models, such as transformer-based architectures, achieve high accuracy when trained on labeled datasets, while unsupervised models can detect anomalies without relying on labeled data, making them suitable for identifying unknown threats. However, there is a lack of comprehensive research that systematically evaluates and integrates both approaches for phishing detection

using structured URL features.
This study addresses this gap by developing a hybrid framework that combines a supervised RoBERTa-based model with an unsupervised Autoencoder. The goal is to analyze their individual and combined performance in terms of accuracy, robustness, and generalization, while exploring the trade-offs between labeled data dependency and anomaly detection capability.

1.2 Objectives

The primary objective of this research is to design and evaluate a hybrid deep learning framework for phishing detection using structured URL features. The specific objectives are as follows:

- To develop a supervised transformer-based phishing detection model using RoBERTa for contextual URL analysis.
- To design and implement an Autoencoder-based model for anomaly detection through reconstruction error.
- To integrate both models into a hybrid framework and evaluate their combined effectiveness.
- To compare supervised and unsupervised approaches using performance metrics such as accuracy, precision, recall, F1-score, and AUC.
- To analyze model behavior through loss convergence, cross-validation, and confusion matrix evaluation.
- To assess the robustness and generalization capability of the hybrid model in detecting both known and unseen phishing attacks.

Overall, this research aims to provide a scalable, accurate, and intelligent phishing detection system while offering insights into the complementary strengths of supervised and unsupervised deep learning approaches in cybersecurity applications.

Chapter 2

Literature Review

2.1 Preliminaries

Phishing is a information security strategy that involves tricking people into revealing crucial data- by posing as trusted entities in the form of emails, links or webpages. The classic detection systems cannot resist such dynamic threats, thus, the implementation of Artificial Intelligence (AI) and Machine Learning (ML). Deeper learning is a sub-field of ML that models types of systems, such as Convolutional Neural Networks (CNNs), Bidirectional Long Short-Term Memory (Bi-LSTM), and transformers including the transformer BERT which automatically learn features of raw data. These models are capable of examining the patterns of phishing in text, image and even structure of URLs. The combination of these kinds of different data through multi-modal detection improves detection accuracy and decreases the number of false positives. Such benchmark datasets as PhishTank, VirusTotal, and the Enron Email Dataset are rich resources in training and testing phishing detection models. These simple ideas can be used to come up with an intelligent, scalable, and real-time phishing detection system.

2.1.1 Phishing and Phishing Detection

Phishing is an information security assault in which labeled perpetrators are carried out through cleverness to penetrate others into uncovering significant data, e.g., sign-in details, credit cards, and other individual data by pretending to be a reputable organization. Such attacks normally take place in fraudulent emails, deceptive links or counterfeit sites that pretend to be true. The effectiveness of the traditional phishing detection systems that use blacklists or rule-based classifiers is not very high when it comes to detecting advanced and evolving phishing attacks. Such systems tend to use predetermined patterns, which means that they are not very useful in blocking new or more flexible phishing methods[1], [2]. The solution to defeat this challenge is to develop more sophisticated detection techniques which can counter the emerging threats and upgrade the detection abilities.

2.1.2 Deep Learning Approach

Deep learning is a subdiscipline of machine learning, that transformed the detection of complex patterns and anomalies in data. A form of deep learning including the

Convolutional Neural Networks (CNNs), Bidirectional Long Short-Term Memory (Bi-LSTM) networks, and transformer-based learning, such as BERT, can automatically learn the raw data to identify complex patterns without manual feature extraction, unlike traditional techniques. These models are very effective in detecting phishing attacks on various types of data. CNNs have proven to be especially helpful at finding patterns within spatially-structured data, e.g. URLs whereas Bi-LSTMs have shown to be proficient at manipulating sequential data, e.g. email text. Transformers such as BERT provide improved contextual and semantic text understanding abilities, and thus transformers are very successful in detecting phishing emails [15]. Based on deep learning models, phishing detection systems have the ability to deal with several different attack methods and increase accuracy and false positives.

2.1.3 Dataset

Such diverse and large data are vital in training and the assessment of phishing detection models. Real-world benchmark datasets like PhishTank, VirusTotal, and Enron Email Dataset can be considered as sources of a lush training and testing deep learning models. PhishTank lists confirmed phishing URLs [17] whereas VirusTotal lists an extensive list of malicious URLs that are scanned using several antiviruses[18]. Enron Email Dataset is a very famous set of emails that are used in email classification challenges and has a vast amount of e-mails, both legitimate and phishing ones[16]. These data sets would effectively form the basis of developing models which could be generalised well to diverse phishing strategies and those models could then produce large-scale and real-time phishing detection systems.

2.2 Review of Existing Research

Artificial intelligence in detecting phishing has developed rapidly, especially regarding the use of deep learning to achieve this goal. The early solutions provided the stepping stone to the field of machine learning with a potential of automating feature extraction and classification. Such early techniques and protocols were later developed into more sophisticated architecture which could tackle various phishing methods. This review follows the history of these techniques.

The use of adaptive neural networks in the detection of phishing emails was first performed openly by Smadi et al.[2], one of the first who tried to use such networks in this case. They used a combination of a dynamic neural network and reinforcement learning to ensure the neural network would update its classification capabilities according to new information constantly. It tested the model on a number of datasets such as the Nazario PhishingCorpus and the SpamAssassin ham/spam emails. The accuracy of 98.63% proved that indeed the system was effective in phishing detection in real-time though with a low false-positive rate of only 1.81%, which is an advantage to the system. The paper has drawn the presence of concepts and the need to have a model that is capable of learning and evolving continuously to manage zero-day phishing attacks.

Riding on the concept of building deep learning models without hand-cording of features, Le et al.[1] proposed an end-to-end deep learning model named URLNet, to detect phishing URLs using convolutional neural networks (CNNs). URLNet, in

contrast to the email-centred focus of Smadi et al., focused on raw form of URL as strings, which enabled the model to learn character and word-level embeddings. The model was trained on a large VirusTotal data with more than 15 million URLs having an AUC of 0.9929 and a detection rate of 71.6% despite the false positive rate of 0.0001. URLNet was also crucial in showing that deep learning showed promise in exceeding traditional lexical-based URL filters. Although Le et al. conducted the first raw string input phishing URL, Alshingiti et al.[13] investigated the effectiveness of a range of deep learning architectures that were submitted to engineered URL features. They compared CNN, LSTM and hybrid LSTM-CNN models in a 20,000-sample dataset obtained at PhishTank and Alexa Top Sites. They found that, among all the models, CNN performed best with accuracy of 99.2% as compared to LSTM with CNN model (97.6%) and LSTM (96.8%). This observation indicated that the conventions of the CNNs should prove better with extracted lexcial and statistical characteristics other than the sequential models when it comes to phishing URL classification.

In order to narrow down the gap of understanding in sequential form, Xiao et al.[10] introduced attention mechanism in detection of phishing URLs. Their self-attention-Enhanced CNN model was able to handle URL sequences with long-distance dependencies that were usually ignored by the conventional CNN models. The sample size of 22,000 phishing and 24,000 legitimate URLs were extended with GANs to deal with the issue of class imbalance. The self-attention CNN model came at an accuracy of 95.6 percent, which beat baseline CNNs by 4.6 percent and LSTM by 2.1 per cent. This input provided a paradigm shift towards the transformer-based architectures in phishing detection.

Moving to the ensemble strategies, a two-stage ensemble modeling phishing URLs was created by Yang et al.[11] with CNN-based feature extractor and Random Forest classifier. Working on two datasets (47,210 URLs and one with the smaller benchmark data), the system had the accuracy of 99.35% and 99.26% respectively. The results enabled evidence to show efficacy of deep representation learning coupled with use of ensemble classifiers to lower number of false positives as well as enhance generalization. However, going beyond the lexical features of the URLs, Abdelnabi et al.[5] dealt with the problem of visually misleading phishing websites by using a model named VisualPhishNet. They used the triplet convolutional neural network (CNN) that calculated the visual similarity between suspect pages and trusted legitimate sites. The dataset, named VisualPhish, was constructive of 9,363 reference pages of 155 brands and many phishing clones. The AUC of the VisualPhishNet was granted a rather high rate, approximately 0.974, being much higher than the classic image-confrontation approach. Considering the screen shot images as opposed to textual properties, the findings have taken another dimension that is less repetitive: phishing through visual impersonation.

To supplement the image-based methodology, Al-Ahmadi and Alharbi[6] entered a dual-channel CNN that comprehended the URL strings and the screenshots in a cross-batch process. The effectiveness of this multi-modal model was evaluated on phishing and authentic webs copied using PhishTank and other open datasets. They obtained 99.67% accuracy which is a strong indication that integration of both visual and textual inputs can perform better in classification than the two features, individually. This combined approach highlights the increasing prevalence of the multi-view learning approach in detecting phishing.

Earlier works focused on feature modeling but more recently efforts were diverted to generalized text understanding by using pre-trained language models. A Hypbrid BERT-LSTM model was investigated by Alhuzali et al. [15] to classify phishing email. They optimised BERT to retrieve semantic features of email content and directed them to LSTM to carry out final classification. Trained on 15 email datasets, the model achieved an accuracy of 99.55 percent and F1-score of 99.24 percent, which was better than the classical ML and minimal deep learning benchmarks. This meant the better contextual knowledge that transformers offered in the analysis of phishing messages. Through email-centric studies, another research by Iisik and Sahin[7] created a deep learning classifier only founded on the content of email bodies. With the low false positive rate of 1.5 percent their model obtained 98.2 percent accuracy on the curated phishing lists of email dataset, using mutually-based information feature selection and a neural network-based classifier. In comparison with Alhuzali et al., however, the method did not include transformer-based codifying but showed solid performance confirming that old-fashioned DNNs could still work well with the right feature selection.

As the next extension of phishing detection, Karpurapu and Imambi[14] used phishing tweets and both LSTM and BERT models to recognize them, in relation to context linked to the MITRE ATTACK framework. The data consisted of identified phishing tweets and the non-malicious messages with emphasis on the link baiting, impersonation and fraud. Their BERT model accomplished an accuracy that was 95% compared to 90-95% of LSTM and 90-95% of vanilla RNNs. It was demonstrated that, with this work, transformer-based models can be transferred to unstructured and short-form phishing data such as tweets.

In addition to social sites, phishing detection may be also enhanced by analysis of HTML contents. HTMLPhish is a convolutional neural network that Opara et al.[9] trained on structural HTML code of both phishing and legitimate websites. The model syntactically and semantically learnt phishing page patterns based on a 50,000 file dataset of HTML files. Having an accuracy higher than 90%, HTMLPhish confirmed that such a quite old phishing signal as a structure of static contents (whether is a fake form or unauthorized scripts and so on) could be used by deep networks in the role of a new phishing discriminator. Another rather traditional yet efficient computationally approach was presented by Wei et al.[3], where the authors introduced lightweight CNNs toward phishing detection in IoT distinguished settings. They were more energy efficient and effectively competitive in the detection performance with more than 93 percent accuracy, although their model was narrower and focused on phishing content that was intended to be used on embedded devices. This paper focused on the fact that detection of phishing attacks should also be feasible on edge devices with reduced capabilities, without adversely affecting the quality of detection.

In order to gain a better insight into the strategies of implementing deep learning in terms of distribution among a domain, Catal et al.[12] carried out the systematic literature review of phishing detection research. They drew a conclusion that the most commonly applied models belonged to DNNs, CNNs and LSTMs, whereas CNNs were highly effective in URL-based tasks, and BERT performed best in detecting emails. PhishTank was found to be the most popular dataset within domains and hybrid models that were based on both deep learning and classical approaches in some cases recorded the highest accuracy rate. Notably, they also demanded a

standardized and multilingual benchmarks and datasets that would be practically unexplored in existing research.

Along with these crucial articles, Xiong et al. (2019) and Kumar et al. (2020) shared valuable information on phishing email identification by means of NLP- and deep learning-based methods. Xiong et al.[4] have suggested a combination of word embedding and deep neural networks to be used in better semantic analysis. Their model, however, admitted defeat in case of more complex and misleading language patterns that are prevalent in phishing attacks. Kumar et al.[8] combined an attention strategy with Bi-LSTM to accuratize the email classification process that helps to improve the model capacity to concentrate on the fish indicators in an email. Nonetheless, their method continued to encounter issues in terms of class imbalance as well as at real-time deployment. Overall, presented works show that phish detection evolved considerably due to the development of deep learning. Most CNNs, such as URLNet[1] and VisualPhishNet[5], are dominant in visual and lexical analysis, whereas most of the advanced models are hybrid models[11] and transformer based models[14], [15] in textual detection. The most accurate model is a multimodal CNN model published by Al-Ahmadi and Alharbi, which reached the best precision of 99.67%[6]. Ensemble methods outperformed other approaches with an accuracy of more than 99% in every case[11]. Still, some issues exist in data versatility, real-time implementation, and detection of advanced phishing methods, which are promising to make significant improvements in the frame of future experiment.

Chapter 3

Methodology

This study adopts a comprehensive five-phase approach to develop a robust phishing detection system. The methodology encompasses the entire lifecycle of the machine learning pipeline, from initial dataset acquisition and augmentation to model training, evaluation, and final deployment within a client-side interface. The architectural workflow is illustrated in Figure 3.1 (referencing to the flowchart).

Dataset Acquisition and Initial Structuring : The investigation commences with the acquisition of a balanced baseline dataset containing 60,000 samples, equally divided between phishing and legitimate URLs (50% each). To ensure data integrity, the raw dataset is loaded while preserving all original feature columns. A strict class separation protocol is then executed, bifurcating the data into distinct Phishing and Legitimate subsets. This separation is critical for the subsequent augmentation phase, as it allows for targeted manipulation of each class independently. A target baseline of 40,000 samples per class is established to guide the data expansion process.

Data Augmentation and Balancing : To mitigate overfitting and enhance model generalization, a rigorous data augmentation strategy is employed. Both the Phishing and Legitimate subsets undergo bootstrap sampling followed by numeric feature detection. To introduce necessary variance and simulate real-world noise, Gaussian noise injection is applied to the numeric features of both classes. This process expands the dataset to meet the pre-defined target of 40,000 samples per class. The augmented subsets are then merged to form a perfectly balanced dataset (50/50 split). A global shuffle is performed immediately after merging to eliminate any ordering bias, resulting in a final, robust dataset of 80,000 noise-balanced samples.

Preprocessing and Model Configuration : Following augmentation, the dataset is serialized to CSV format and reloaded to simulate a production data ingestion pipeline. The data preprocessing phase involves rigorous cleaning and feature extraction to isolate relevant signals from the URL vectors. The preprocessed data is then stratified into labeled and unlabeled sets to support different learning paradigms. The methodology incorporates a flexible model selection mechanism, allowing the pipeline to branch into either a Supervised Learning path (utilizing labeled data) or an Unsupervised Learning path , depending on the specific detection requirements.

The core computational phase focuses on the training and validation of the selected architectures:

Supervised Approach: If selected, the data is tokenized using the RoBERTa tokenizer. A RoBERTa-based model is then trained on the tokenized sequences. Performance is strictly evaluated using standard classification metrics: Accuracy, Precision, Recall, and F1-score.
Unsupervised Approach: Alternatively, an Autoencoder architecture is trained on the unlabeled data. The performance of this model is assessed by analyzing the Reconstruction Error, where higher error rates typically indicate anomalies (phishing).

Comparative Analysis: The phase concludes with a comparative performance analysis to select the optimal model weights for deployment.

Deployment and User Interface Integration : The final phase focuses on operationalizing the trained model via a client-side user interface (UI) connected through a Model Inference API. The workflow is initiated when a user inputs a URL and selects a specific detection engine. The system performs a dual analysis: it scans URL vectors using static heuristics and simultaneously queries the backend model for inference.

Response Handling: If the model predicts a threat, the UI renders a "Phishing Detected" alert (Red UI). Conversely, if the URL is deemed safe, a "Legitimate" notification (Green UI) is displayed.
Explainability: To ensure transparency, the system renders a detailed breakdown of the model's reasoning and feature analysis, providing the user with actionable insights before terminating the session.

The workflow of the proposed phishing detection system will be optimized in terms of data handling, computational use, and model integration to ensure good resource management. The dataset is well balanced and treated to optimize the maximum training efficiency without bias. To minimize unnecessary overhead, computational resources are managed by using controlled batch processing, limited epochs, and early stopping. The usage of memory is also optimized due to the processing of the data in organized formats. The combination of RoBERTa and Autoencoder models is a balance between performance and resource usage. Economically, though the initial development involves the use of computational infrastructure, the system will save a lot of money which is lost to phishing attacks. The operational costs are further reduced by utilizing open-source technology and scalable deployment. Generally, the solution not only guarantees the effective use of resources but also guarantees the economic sustainability in the long run.

Figure 3.1 presents the overall workflow of the proposed hybrid phishing URL detection system. The process begins with URL input validation, followed by preprocessing and feature extraction. The input is then analyzed through two parallel branches:

the RoBERTa branch, which performs tokenization and transformer-based contextual analysis, and the Auto Encoder branch, which evaluates structural anomaly through feature-vector reconstruction. The outputs of both branches are passed to a fusion layer.

In the final stage, the fused result undergoes a threshold check to determine whether the URL should be classified as phishing or legitimate. The figure also includes the training process of both RoBERTa and Auto Encoder models, where dataset input is used to train the models, optimize loss, and update parameters. Overall, this workflow shows how the system integrates semantic understanding and anomaly detection in a unified framework to improve phishing URL classification.

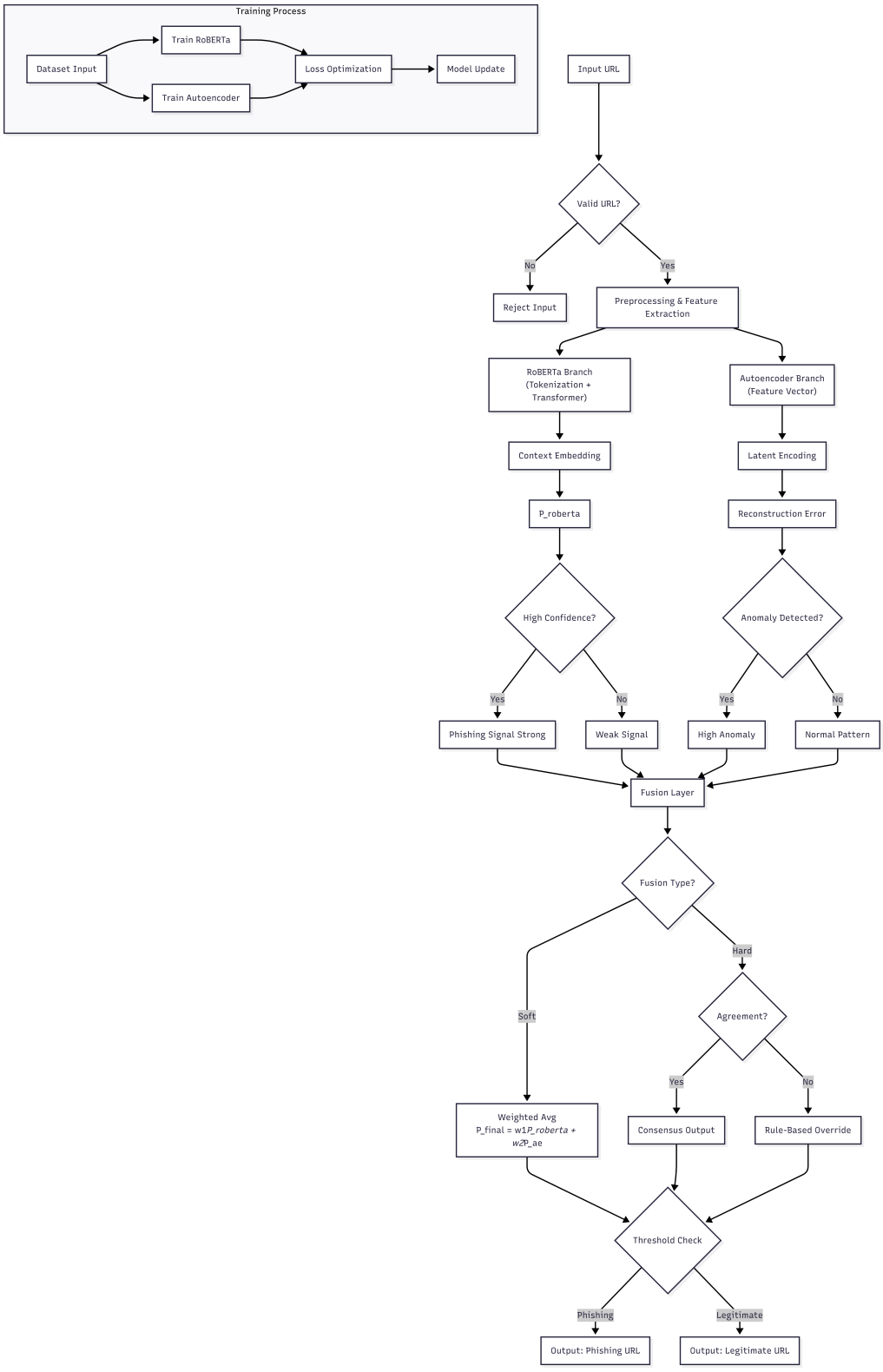


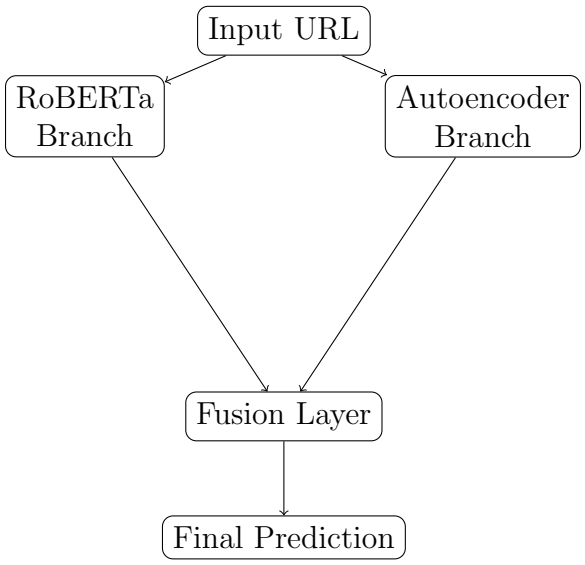
Figure 3.1: WorkFlow

3.1 Model Architecture and Mathematical Formulation

This sub-section describes the architectural design and mathematical approach of the two deep learning model involved in the given study a supervised classification model using RoBERTa and an unsupervised latent embedding model using Auto Encoder. Both models are based on structured URL features although they differ radically in their learning paradigms.

3.1.1 Hybrid Transformer and Autoencoder-Based Model

The proposed hybrid framework integrates a RoBERTa-based supervised model with an Autoencoder-based reconstruction model to capture both semantic and structural anomalies in URLs.



Given an input URL, the system processes the data through two parallel branches. In the transformer branch, the URL is tokenized as:

$$X = \{x_1, x_2, \ldots, x_n\}$$

and embedded using:

$$E = E_{\text{token}} + E_{\text{position}}$$

The RoBERTa model extracts contextual features:

$$h_{\text{CLS}} = \text{RoBERTa}(E)$$

and produces a phishing probability:

$$P_{\text{roberta}} = \text{softmax}(W_r h_{\text{CLS}} + b_r)$$

In parallel, the Autoencoder processes the feature vector $x \in R^n$, generating a latent representation:

$$z = f_\theta(x)$$

and reconstructing the input:

$$\hat{x} = g_\phi(z)$$

The reconstruction error is computed as:

$$E = \|x - \hat{x}\|_2^2$$

which is converted into an anomaly score:

$$P_{\text{ae}} = \frac{E}{\max(E)}$$

The outputs of both branches are combined using a weighted fusion strategy:

$$P_{\text{final}} = w_1 P_{\text{roberta}} + w_2 P_{\text{ae}}, \quad w_1 + w_2 = 1$$

The final prediction is obtained as:

$$\hat{y} = \begin{cases} 1, & P_{\text{final}} \geq \tau \\ 0, & \text{otherwise} \end{cases}$$

The overall training objective combines supervised and unsupervised losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{AE}}$$

This hybrid formulation enables the model to effectively detect both known phishing patterns and previously unseen anomalies, resulting in improved accuracy and generalization.

3.1.2 Activity Diagram

An activity diagram is a behavioral modeling diagram used to represent the sequential flow of actions, decisions, and parallel processes within a system. It helps illustrate how different operations are carried out from the beginning of a process to its final outcome, making it useful for visualizing system logic and decision-making steps.

As shown in Figure 3.2, the activity diagram presents the operational workflow of the proposed phishing URL detection system. The process starts with the user providing a URL, which is first checked for validity. Once validated, the URL undergoes preprocessing and feature extraction, after which the workflow is divided into two parallel branches. In the first branch, the RoBERTa model performs tokenization, embedding, and prediction scoring to determine the phishing confidence level. In the second branch, the autoencoder analyzes the extracted features by encoding them and measuring reconstruction error to identify anomalous behavior. The outputs from both branches are then merged and processed using a fusion strategy, which may apply either soft fusion or hard fusion. Finally, a threshold-based decision is made to classify the input as either a phishing URL or a legitimate URL.

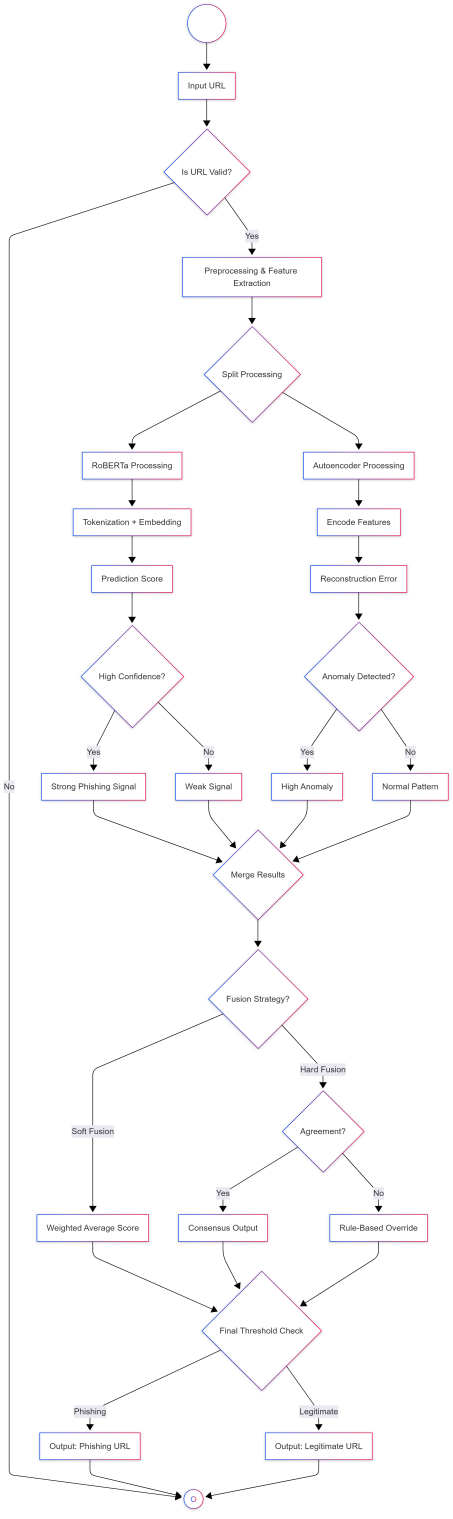


Figure 3.2: Activity Diagram

3.1.3 Use Case Diagram

As shown in Figure 3.3, the use case diagram illustrates the main functional interactions of the proposed phishing URL detection system. The process begins with the *User* initiating the *Analyze URL* use case, which includes key operations such as *Validate URL*, *Extract Features*, *RoBERTa Prediction*, *Autoencoder Analysis*, and *Display Result*. These use cases represent the core steps involved in analyzing a URL and generating the detection result.
The diagram also presents additional supporting behaviors. The outputs from the RoBERTa and AutoEncoder modules are combined in the *Fusion Decision* stage to determine the final classification. Moreover, *Error Handling*, *Confidence Adjustment*, and *Rule-Based Override* are shown as extended functions that improve system reliability and decision accuracy, while the *System Controller* and *Data Storage* support execution and result management.

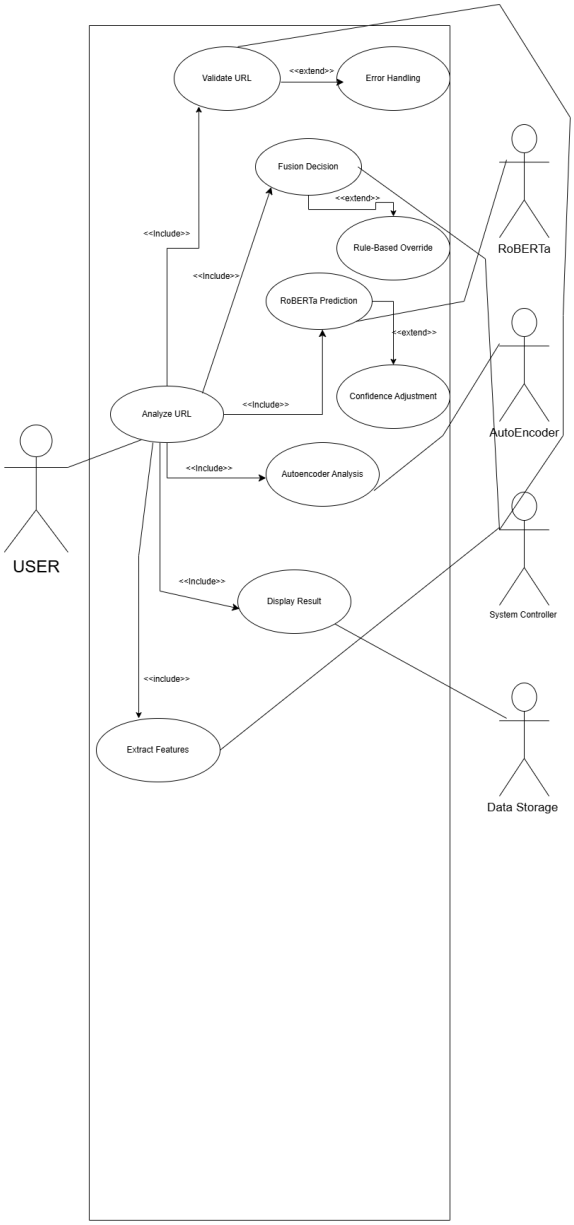


Figure 3.3: Use Case Diagram

3.1.4 Entity Relationship (ER) Diagram

As shown in Figure 3.4, the ER diagram represents the main data structure of the proposed phishing URL detection system. The *URL* entity stores the input URL, its unique identifier, and the class label, while the *FEATURE* entity contains the extracted characteristics such as URL length, subdomain count, entropy, and HTTPS status. These features are then used by the analysis modules for phishing detection.
The diagram also shows how the model outputs are organized in the database design. The *ROBERTA_MODEL* stores the prediction probability, and the *AUTOENCODER_MODEL* stores the reconstruction error obtained from anomaly analysis. Both of these contribute to the *FUSION* entity, which calculates the final score. Finally, the *RESULT* entity stores the overall prediction and confidence value, representing the final decision of whether the URL is phishing or legitimate.

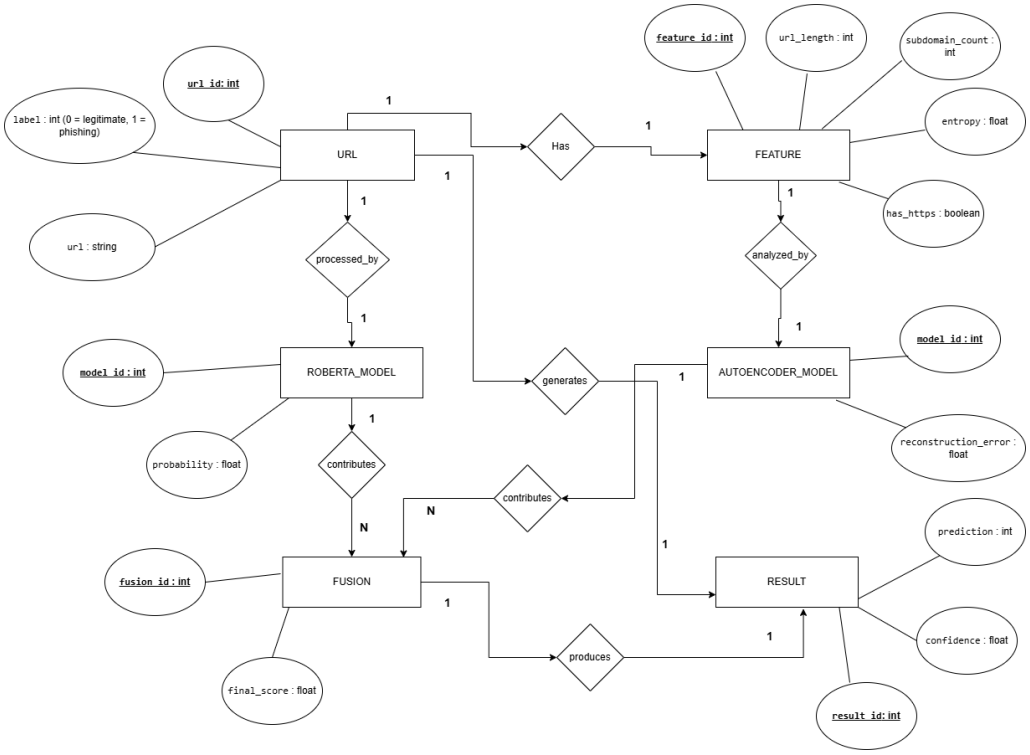


Figure 3.4: Entity Relationship Diagram

3.1.5 Class Diagram

As shown in Figure 3.5, the class diagram illustrates the structural design of the proposed phishing URL detection system. The *UrlInput* class handles the input URL and validation process, while the *SystemController* manages the overall workflow of the system. The *FeatureExtractor* class derives important URL-based features such as length, HTTPS status, and entropy, which are then used for analysis.
The diagram also shows the interaction among the main analytical and decision-making classes. The *RobertaModel* and *AutoencoderModel* perform prediction and anomaly analysis, respectively, and both contribute to the *FusionEngine*, which combines their outputs using soft or hard fusion strategies. The *DecisionEngine*

then classifies the fused score based on a threshold, and the final outcome is stored in the *Result* class with the predicted label and confidence value.

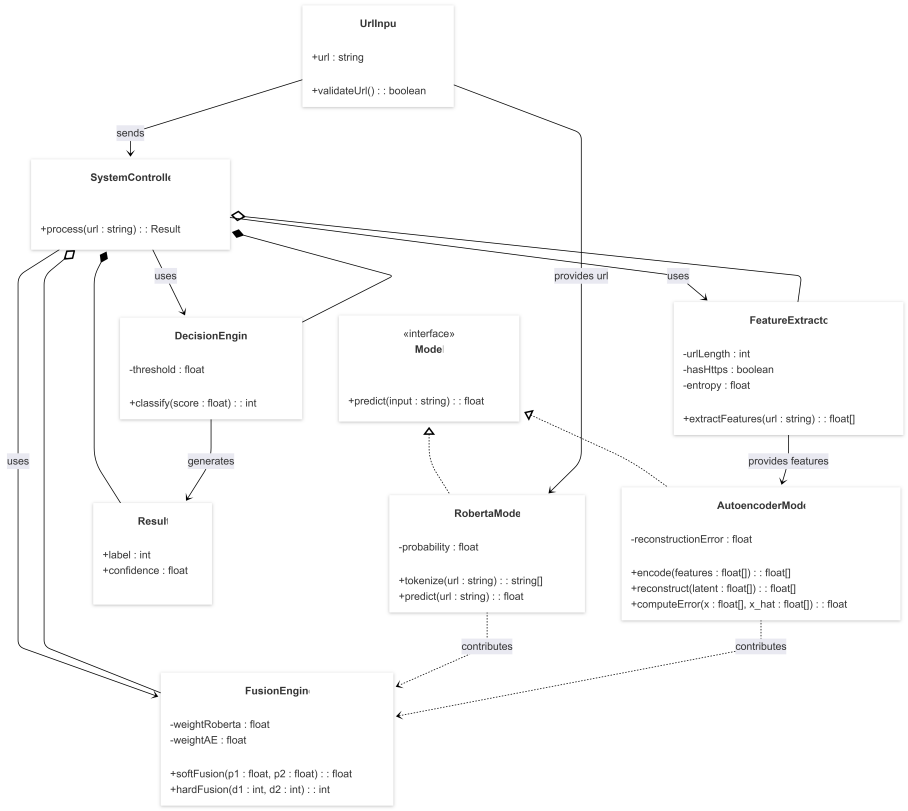


Figure 3.5: Class Diagram

3.2 Web Application Analysis

3.2.1 User Interface

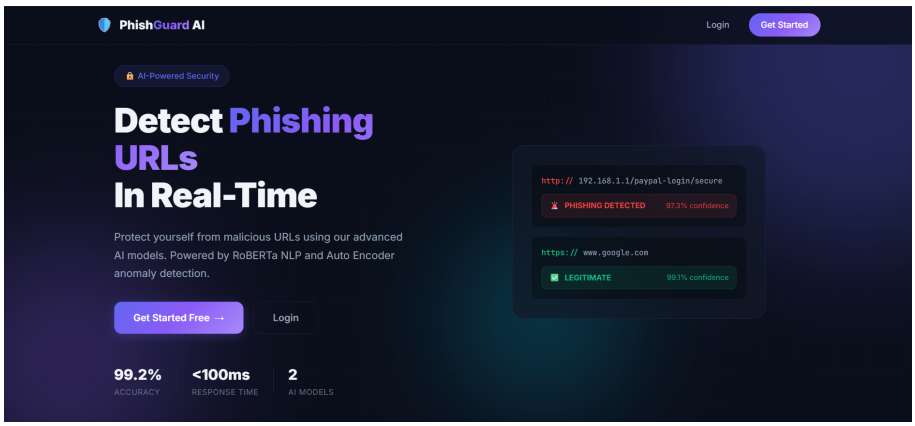


Figure 3.6: User Interface of the Web Application

The user interface of the proposed PhishGuard AI system is designed to provide a clear, modern, and user-friendly environment for phishing URL detection. The

landing page features a dark-themed dashboard with visually distinct sections, including a navigation bar, authentication options, and a prominent hero section that highlights the core purpose of the platform. The main heading, "Detect Phishing URLs in Real-Time" immediately communicates the objective of the system, while the supporting text briefly introduces the use of RoBERTa-based natural language processing and Auto Encoder anomaly detection models. Call-to-action buttons such as **Get Started Free** and **Login** make the interface intuitive and accessible for users. The interface includes a visual panel showing example URL classification results, helping users understand the system quickly. Suspicious URLs are flagged with a phishing warning and confidence score, while safe URLs are marked as legitimate. With clear color coding, structured layout, and performance indicators like accuracy and response time, the design ensures usability, clarity, and effective real-time phishing detection.

3.2.2 Account Creation and User Sign-In

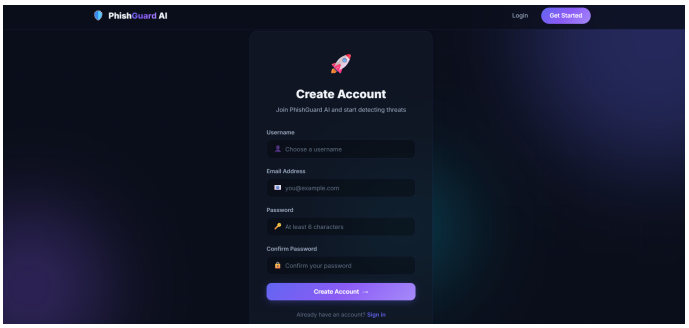


Figure 3.7: Account Creation of the Web Application

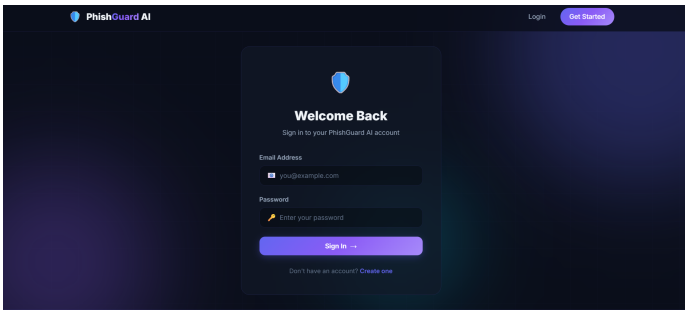


Figure 3.8: User Log in of the Web Application

Figures 3.7 and 3.8 illustrate the account creation and user sign-in interfaces of the PhishGuard AI web application. These two authentication components are essential for controlling access to the system and ensuring that only registered users can use the phishing detection platform. The account creation page allows new users to register by providing a username, email address, password, and password confirmation, while the login page allows existing users to access their accounts using their registered email and password.
The interfaces are designed with a consistent dark-themed layout that matches the overall visual style of the application. Both pages use a centered card-based structure, clearly labeled input fields, supportive icons, and highlighted action buttons

to make the authentication process simple and intuitive for users. As shown in Figure 3.7, the registration page provides a structured form for creating a new account, whereas Figure 3.8 presents a streamlined login form for returning users. Together, these features improve usability, maintain interface consistency, and provide a secure entry point to the PhishGuard AI system.

3.2.3 RoBERTa-Based URL Detection and Detailed Analysis

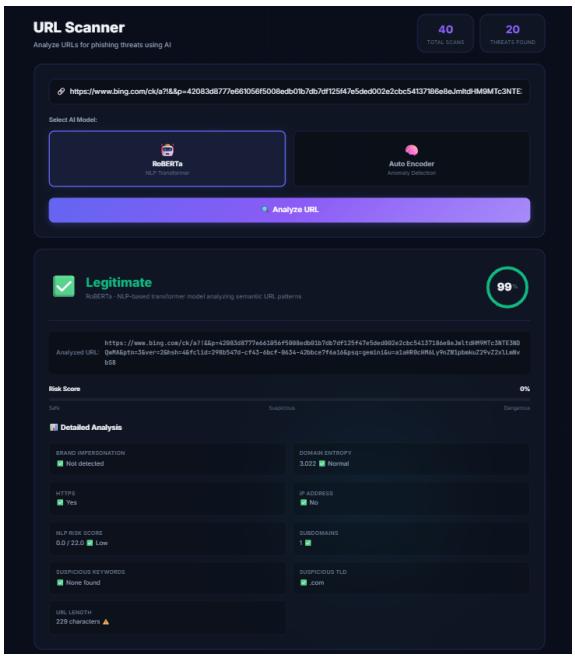


Figure 3.9: RoBERTa-based detection result for a legitimate URL

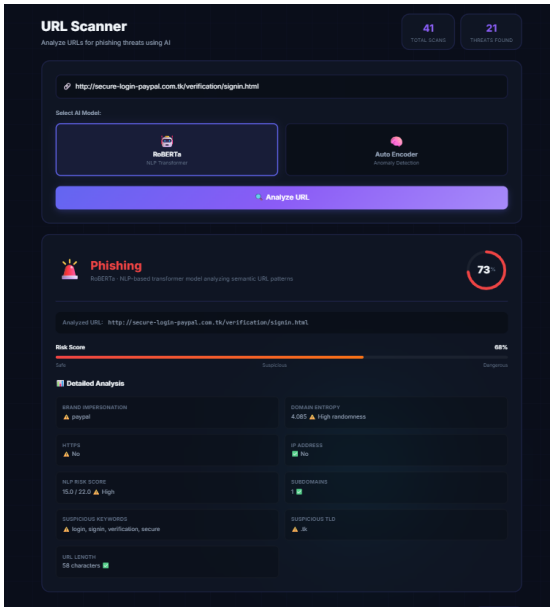


Figure 3.10: RoBERTa-based detection result for a phishing URL

Figures 3.9 and 3.10 demonstrate how the RoBERTa-based detection module classifies URLs as either legitimate or phishing by analyzing their textual and structural patterns. The interface first accepts a URL as input and then applies the selected RoBERTa model to predict the security status of the link. In addition to the final classification result, the system also provides a confidence score, a risk score, and a detailed feature-level explanation. This improves the interpretability of the detection process by allowing users to understand why a URL is considered safe or suspicious rather than only viewing a binary output.

As shown in Figure 3.9, the model identifies the submitted URL as *Legitimate* with a very high confidence score of 99% and a risk score of 0%. Although the URL is relatively long, the detailed analysis indicates several characteristics commonly associated with safe links. No brand impersonation is detected, the domain entropy remains within a normal range, HTTPS is enabled, no raw IP address is used, and no suspicious keywords are found in the URL. The suspicious top-level domain (TLD) check also shows a common and widely trusted extension, `.com`. These combined indicators contribute to a very low NLP risk score, which supports the model's decision that the URL is benign.

In contrast, Figure 3.10 presents a URL that is classified as *Phishing* with a confidence score of 73% and a risk score of 68%. The detailed analysis reveals multiple suspicious properties that strengthen this decision. First, the URL appears to imitate a trusted brand name, *PayPal*, which is a common phishing strategy used to deceive users. Second, the link does not use HTTPS, which reduces its trustworthiness. Third, the presence of highly suspicious terms such as *login*, *signin*, *verification*, and *secure* increases the likelihood that the URL was intentionally crafted to trick users into revealing sensitive information. In addition, the TLD `.tk` is often considered more suspicious in phishing-related contexts, while the high domain entropy suggests unusual or less natural domain construction. These factors collectively result in a high NLP risk score and support the phishing classification.

The detailed analysis section in Figure 3.9 and Figure 3.10 beneath each prediction result is especially important because it adds an explainable layer to the detection system. Rather than depending only on the final model output, the application highlights meaningful URL characteristics such as HTTPS usage, suspicious keywords, domain entropy, URL length, IP address presence, brand impersonation, and TLD behavior. This makes the system more transparent and practical for real-world cybersecurity applications, as users can visually inspect the reasons behind each decision and gain greater confidence in the phishing detection process.

3.2.4 Auto Encoder-Based URL Detection and Detailed Analysis

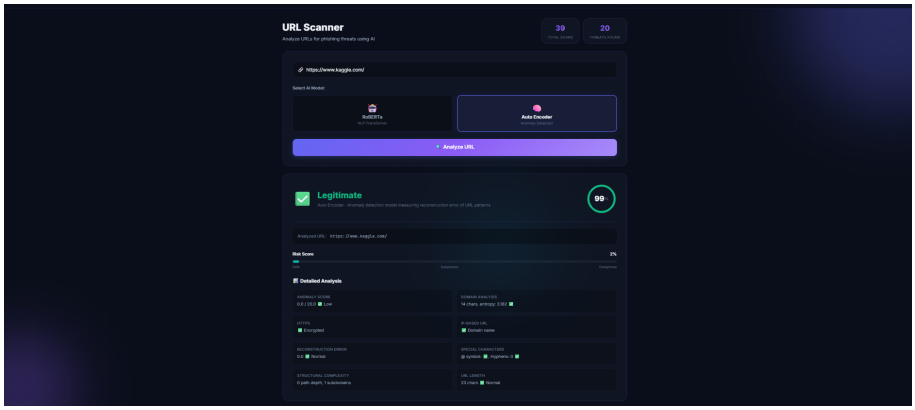


Figure 3.11: Auto Encoder-based detection result for a legitimate URL

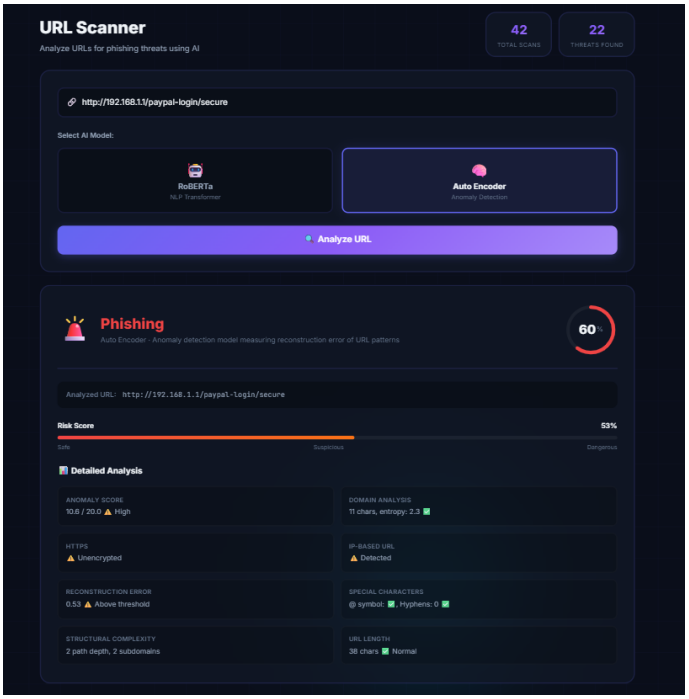


Figure 3.12: Auto Encoder-based detection result for a phishing URL

Figures 3.11 and 3.12 show the classification results of the Auto Encoder-based anomaly detection module. This model analyzes the structural pattern of a URL and compares it with the normal URL behavior learned during training. If the reconstruction error is low, the URL is considered similar to legitimate samples; if the error is high, the URL is treated as anomalous and more likely to be phishing. Along with the final prediction, the system also provides a confidence score, risk score, and feature-based analysis to explain the result.

As shown in Figure 3.11, the model classifies the URL as *Legitimate* with 99% confidence and a very low risk score of 2%. The anomaly score and reconstruction error are both low, indicating that the URL closely matches normal patterns. The detailed

analysis further shows the presence of HTTPS, use of a regular domain name instead of an IP-based URL, low structural complexity, minimal special characters, and a normal URL length. These characteristics support the legitimate classification.
In contrast, Figure 3.12 presents a URL classified as *Phishing* with 60% confidence and a risk score of 53%. In this case, the anomaly score is high and the reconstruction error is above the threshold, showing that the URL structure differs from normal legitimate URLs. The absence of HTTPS, the presence of an IP-based URL, and the relatively more complex structure make the link more suspicious. Therefore, the Auto Encoder identifies it as a phishing URL. This detailed breakdown improves the interpretability of the detection system by clearly showing the structural factors behind each prediction.

3.2.5 Scan Records and Recent History

Figure 3.13: Scan summary showing total scans and detected threats

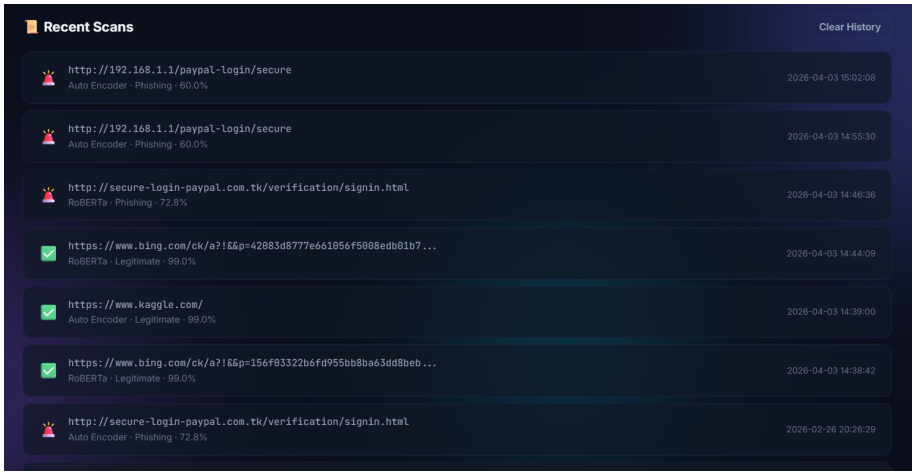


Figure 3.14: Recent scan history stored in the web application

Figures 3.13 and 3.14 show that the web application stores scanned URL records and keeps track of detected threats. The scan summary presents the total number of scans performed and the number of threats found, while the recent history section displays previously analyzed URLs with their prediction result, model name, confidence score, and timestamp.
This feature helps users review past scanning activity and monitor suspicious links more effectively. It also improves usability by allowing users to revisit previous results without repeating the analysis. In addition, the *Clear History* option enables users to remove stored records when needed, giving them better control over their scan history.

Chapter 4

Dataset and Preprocessing

The precision and quality of training data play a major role in determining the performance of a machine learning and deep learning model. A big phishing URL dataset was employed in this study which is binary counterparts with **80,000** URLs along with 94 varaiables, of which **50:50** distribution of phishing and legitimate URLs. It was also purposely added that the data was noisily augmented to improve generalization and reduce overfitting. Every record has a URL and a label of the class. Before the training, conventional preprocessing measures, including the label encoding and the split of trains and tests, were used. In the case of the transformer-based method, URLs were directly operated with a pretrained tokenizer, and thus it could effectively learn contextual features without feature engineering.

4.1 Data Structure and Feature Description

All the 94 features can be classified into three groups, namely: **URL Structural Features**, **Domain Features** and **Content-Based Features**.

URL Structural Features

These are the natural prominences of the URL string. The samples given in the Table 4.1 and Table 4.2 highlight the important measurements of total URL length with the key metric as the length of `URL` & length of `hostname` as well as the counts of individual characters. As an example, the presence of such indicators as symbols as `@` or superfluous slashes is often a pointer of suspicious URLs.

Table 4.1: Sample URLs and Their Lengths

No.	URL	Length_URL
1	`http://work.chron.com/busboy-server-job-descri...`	63.00
2	`http://www.mattiesaustin.com/`	29.00
3	`https://www.bls.gov/oes/current/oes292035.htm`	44.75
4	`http://www.astroamerica.com/nodes.html`	37.26
5	`http://economy_en_ru.academic.ru/64464/transac...`	58.00

Table 4.2: URL Structural Feature Statistics

No.	Hostname Length	IP	Dots	Hyphens	@	?	&	OR
1	14.00	0	2.00	0.00	0	0.000000	0.000000	0.0
2	21.00	0	2.00	0.00	0	0.000000	0.000000	0.0
3	11.15	0	1.99	-0.01	0	-0.003736	0.024848	0.0
4	19.78	0	2.01	0.00	0	0.004024	0.010632	0.0
5	25.00	0	2.00	0.00	0	0.000000	0.000000	0.0

Domain and Ranking Features

The features define the background and reputation of the domain. Some of them are very important, such as: `Domain Age`, `Domain Registration Length`, and `Web Traffic`. Phishing is commonly associated with newly registered domains or low-trafficking domains. An approximation of data is displayed in Table 4.3

Table 4.3: Domain and Ranking Feature Samples

No.	Web Traffic	DNS Record	Google Index	Page Rank	Status
1	5.45×10^2	0.000000	0.000000	6.000000	Legitimate
2	1.75831×10^6	0.000000	0.000000	4.000000	Legitimate
3	-6.036492×10^4	-0.000429	-0.002936	5.931884	Legitimate
4	6.076146×10^6	0.000025	-0.007757	2.985608	Legitimate
5	2.272000×10^3	0.000000	1.000000	3.000000	Legitimate

Additional Features

In the Table 4.4 Other features like the presence of `an HTTPS` (security indicator) or the number of known malicious words `Suspicious Words` can be helpful as well. The target variable is the `Status` column having two categories, that is, either `Legitimate` or `Phishing`.

Table 4.4: Additional URL Content-Based Features

No.	Synthetic Label	Has HTTPS	Digits Count	Suspicious Words	Synthetic
1	NaN	0.000000	0.000000	0.000000	True
2	NaN	0.000000	0.000000	0.000000	False
3	NaN	0.996098	0.002603	-0.004096	True
4	NaN	-0.003945	0.001990	0.003385	False
5	NaN	0.000000	0.000000	1.000000	True

Table 4.5 gives the structure of the data that will be used in this paper. It gives the details of every feature and the associated index, number of non-null, and the data type. As indicated, there are no missing data in all the features since all features have 80,000 non-NULL values. All other features are numerical, and their type is of float64, meaning that they can be used directly in the models of phishing detection based on machine learning. The only exception is the URL field, which is of object type and can therefore be used to build the model directly.

Table 4.5: Dataset Column Information

No.	Column Name	Non-Null Count	Data Type
0	url	80000	object
1	length_url	80000	float64
2	length_hostname	80000	float64
3	ip	80000	float64
4	nb_dots	80000	float64
5	nb_hyphens	80000	float64
6	nb_at	80000	float64
7	nb_qm	80000	float64
8	nb_and	80000	float64
9	nb_or	80000	float64
10	nb_eq	80000	float64

4.2 Initial Data Distribution

The issues with missing and unreliable target labels were resolved and became one of the most essential improvements to the first dataset. The completed dataset comprised of 80,000 fully labeled records after a rigorous preprocessing task, as shown in Figure 4.1, with equal number 40,000 legitimate and 40,000 phishing records. This balanced writing made sure that no records had no record of target variable (`Status`) and hence the dataset was fit to be used in supervised learning. Before this refinement, there was the risk that inconsistencies, possible outliers and incomplete entries may compromise model reliability. As a result, the systematic data cleaning procedure, such as deletion or correction of invalid records, normalization of feature values, and integrity checks, was used. All these preprocessing interventions were necessary to make the end training set reliable and representative towards the final outcome of increasing the strength, generalization ability and predictive accuracy of the trained models.

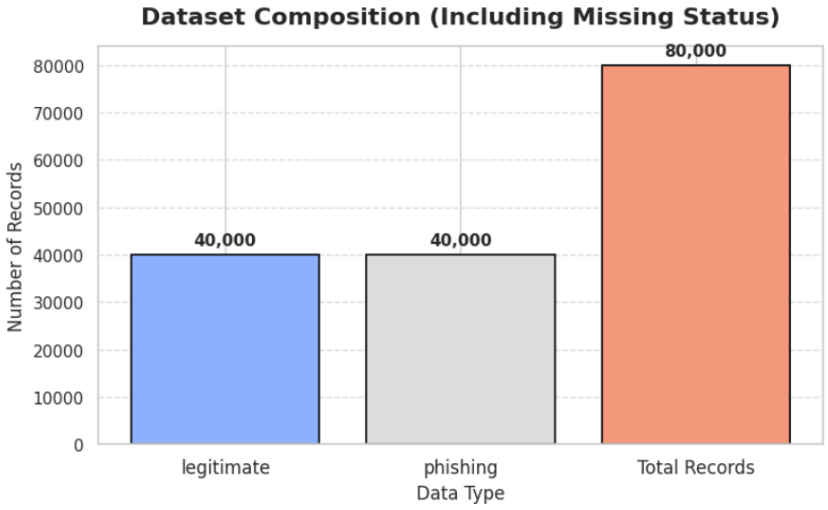


Figure 4.1: Initial Dataset Status Distribution

4.3 Data Cleaning Process

The data cleaning process will include removing potential errors in the collected data, such as anomalies, outliers, and inconsistent data points that may have arisen during the data collection procedure. The data cleaning process will entail the elimination of possible mistakes in the data gathered including anomalies, outliers, and inconsistent data points which might have occurred throughout the process of data collection.
A systematic data cleaning process was used to make sure that the data set was reliable and consistent before being trained in the models. The long phishing dataset consists of **80,000** labeled URLs, and half of them belong to legitimate classes and the other half are phishing. The next steps have been carried out:

1. **Verification:** To ensure that there were valid class labels, all the records were checked. No values were missing in the status attribute, and the integrity of labels throughout the dataset is complete.

2. **Target Encoding:** The discrete column (computer-generated) status (Legitimate, Phishing) was coded as numerical (0= legitimate URLs, 1= phishing URLs).

3. **Dealing with invalid feature values:** There were a number of domain-specific numerical characteristics like the one related to domain age, domain registration length, etc. that had place value of -1 to represent unavailable values. These values were not lost in the noise that was controlled in constructing the dataset in order to improve the model generalization.

4. **Feature Consistency Assurance:** There were no feature columns eliminated in the cleaning procedure. Raw text of the URL was also maintained and the numerical variables obtained were also kept so that they can be used with a variety of modeling methods such as feature-based classifiers and transformer-based models.

4.4 Distribution of the Dataset Across Training, Validation, and Testing Sets

Figure 4.2 illustrates the proportion of data allocated to each subset of the dataset. The dataset was divided into three parts: **80%** for training, **10%** for validation, and **10%** for testing. The training set contains the largest portion of the data, allowing the model to learn important patterns and relationships from a wide range of examples. Meanwhile, the validation set is used during the development phase to monitor the model's performance and fine-tune hyperparameters, while the test set is reserved for the final evaluation of the model on unseen data.
This type of split is commonly used in machine learning because it ensures a balanced approach to model development and assessment. By assigning most of the data to training, the model can achieve better learning efficiency, while the separate validation and test sets help reduce overfitting and provide a more reliable measure of generalization performance. Therefore, the adopted split supports both effective model training and fair evaluation of the proposed phishing detection system.

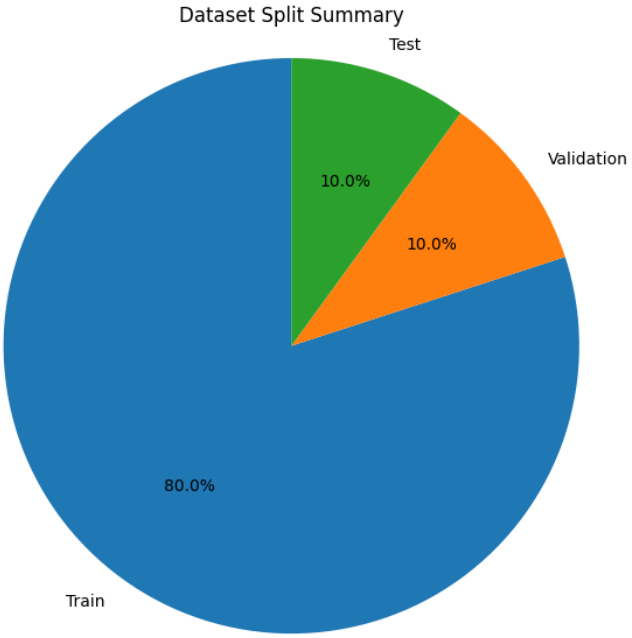


Figure 4.2: Split summary of the Dataset

4.5 Overview of the Dataset

As shown in Figure 4.3, the dataset flowchart presents the step-by-step preparation pipeline used to construct the final dataset for the phishing detection system. The process starts with the original dataset of 60,000 samples, consisting of an equal distribution of phishing and legitimate URLs. At this stage, all original columns are preserved, meaning no feature removal is performed before further processing. The dataset is then separated into two classes so that each category can be handled independently.

After the class separation step, a target of 40,000 samples is defined for both the phishing and legitimate subsets. To reach this target, bootstrap sampling with replacement is applied separately to each class. Once the samples are expanded, numeric features are identified and Gaussian noise is injected only into those numerical attributes. This controlled noise addition slightly increases data variability while preserving the overall structural characteristics of the dataset. In the final stages, the two expanded classes are merged to maintain a strict 50:50 class balance, followed by a global shuffle to remove ordering bias. The prepared data is then exported as the final 80,000-sample balanced dataset, which is suitable for robust machine learning training and evaluation.

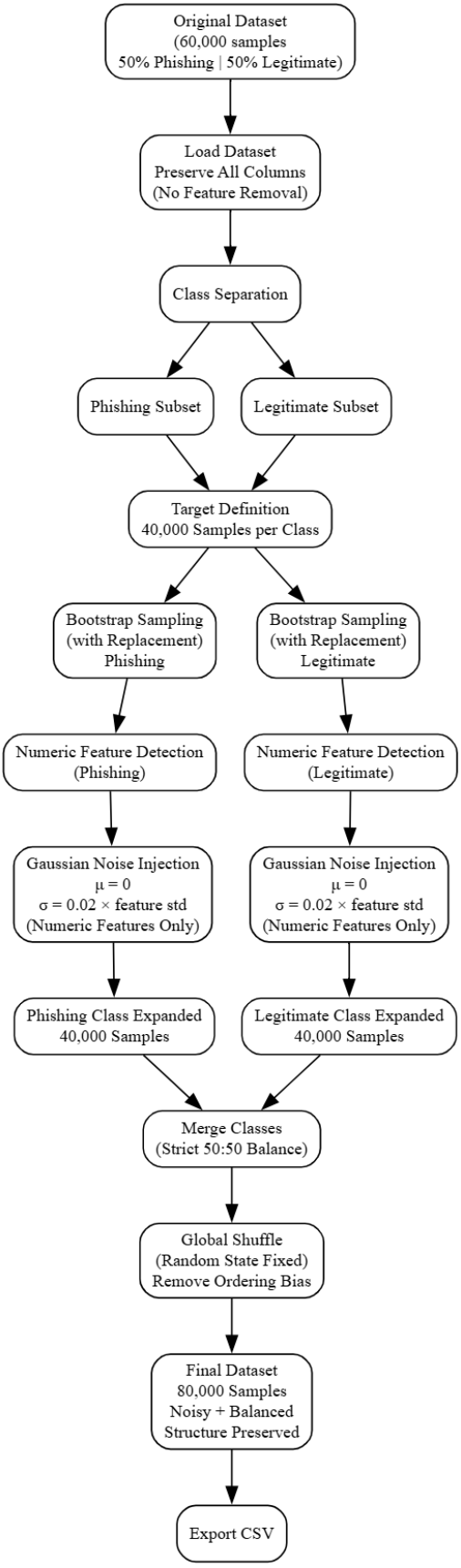


Figure 4.3: Flowchart of the Dataset

Chapter 5

Result & Analysis

This section presents the performance of the proposed hybrid phishing detection framework using multiple evaluation outputs, including 5-fold cross-validation, optimized model performance, confusion matrix analysis, ROC comparison, training loss comparison, and epoch-wise validation results. The obtained results indicate that the proposed framework achieved strong classification capability and maintained stable performance across different evaluation settings.

5.1 5-Fold Cross-Validation Performance

Figure 5.1 and Table 5.1 summarize the 5-fold cross-validation performance of the hybrid model. The model achieved a mean AUC of 0.9997 ± 0.0001, a mean F1-score of 0.9979 ± 0.0007, and a mean accuracy of 0.9979 ± 0.0007. These results indicate that the proposed model performed exceptionally well across all folds.
The very small standard deviation values show that the model remained highly stable under different train-validation splits. This is an important finding because it suggests that the model is not dependent on a particular partition of the dataset and can generalize consistently across multiple folds. In phishing detection tasks, such stability is especially valuable, since a robust classifier must maintain high performance across different subsets of data rather than only achieving a strong result in a single split.

Table 5.1: 5-Fold Cross-Validation Summary of the Hybrid Model

Metric	**Mean**	**Standard Deviation**
AUC	0.9997	0.0001
F1-Score	0.9979	0.0007
Accuracy	0.9979	0.0007

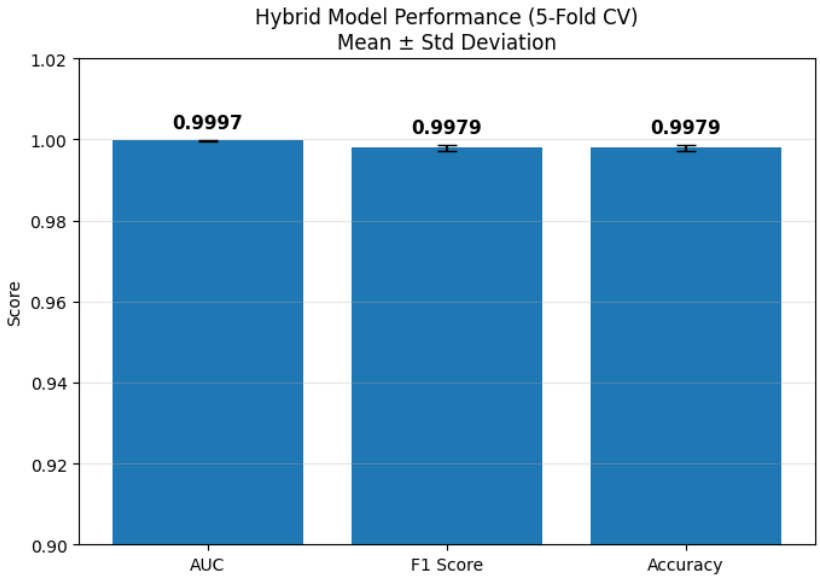


Figure 5.1: Hybrid model performance under 5-fold cross-validation with mean and standard deviation.

5.2 Optimized Hybrid Model Performance

Figure 5.2 presents the performance of the optimized hybrid model. In this evaluation, the model obtained an AUC of 0.9928, an accuracy of 0.9700, and an F1-score of 0.9703. These values confirm that the model maintained excellent discriminative capability even under a separate optimized evaluation setting.

The AUC value of 0.9928 demonstrates that the hybrid model was highly effective in distinguishing phishing samples from legitimate ones. At the same time, the close values of accuracy and F1-score indicate balanced classification behavior. Since the F1-score considers both precision and recall, the similarity between these two values suggests that the model did not improve one class at the cost of the other. This is desirable in phishing detection, where both identifying malicious URLs and avoiding incorrect blocking of legitimate URLs are important.

Although these scores are slightly lower than the 5-fold cross-validation averages, they still represent a very strong result. Such variation is normal when results are reported from different evaluation settings or optimization protocols.

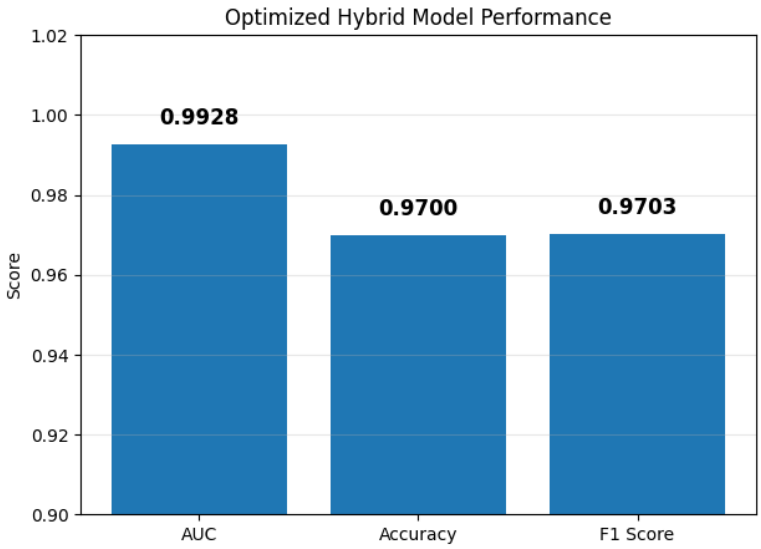


Figure 5.2: Performance of the optimized hybrid model in terms of AUC, accuracy, and F1-score.

5.3 Epoch-wise Training and Validation Analysis

Table 5.2 presents the epoch-wise training loss, validation accuracy, and validation F1-score of the model over five epochs. The training loss decreased from 0.2380 in the first epoch to 0.0343 in the final epoch, showing that the model learned the underlying patterns effectively during training.

At the same time, validation accuracy improved from 0.9539 to 0.9849 by epoch 4, while validation F1-score increased from 0.9523 to 0.9849. This upward trend shows that the model steadily improved its predictive ability during the training process. A slight decrease was observed in epoch 5, where validation accuracy dropped to 0.9826 and validation F1-score dropped to 0.9825. This suggests that the best generalization performance may have been achieved around epoch 4, and that additional training beyond this point provided only marginal benefit. Therefore, the epoch-wise results suggest that early stopping around epoch 4 could be a suitable strategy to preserve optimal validation performance while avoiding unnecessary training.

Table 5.2: Epoch-wise Training Loss and Validation Performance

Epoch	Train Loss	Val Accuracy	Val F1-Score
1	0.2380	0.9539	0.9523
2	0.1207	0.9724	0.9721
3	0.0618	0.9815	0.9816
4	0.0346	0.9849	0.9849
5	0.0343	0.9826	0.9825

5.3.1 Confusion Matrix Analysis

Figure 5.3 shows the confusion matrix of the hybrid model on the full dataset containing 80,000 samples. Out of 40,000 legitimate samples, 39,990 were correctly classified, while only 10 were incorrectly predicted as phishing. Similarly, out of 40,000 phishing samples, 39,977 were correctly identified, whereas only 23 were misclassified as legitimate.

These results indicate an extremely low number of classification errors. In total, only 33 samples were misclassified out of 80,000, which reflects the strong predictive power of the proposed model. The false positive count is very low, meaning that legitimate websites were rarely flagged as phishing. This is important for reducing unnecessary alarms and avoiding inconvenience for users. On the other hand, the false negative count is also very small, although slightly higher than the false positive count. Since false negatives correspond to phishing samples being classified as legitimate, they are generally more critical from a cybersecurity perspective. Even so, the number of such cases remained minimal, showing that the model was highly effective in identifying malicious instances.

Based on the confusion matrix, the model achieved near-perfect class-wise behavior, with very high sensitivity and specificity. This confirms that the hybrid framework was able to preserve a strong balance between security and reliability.

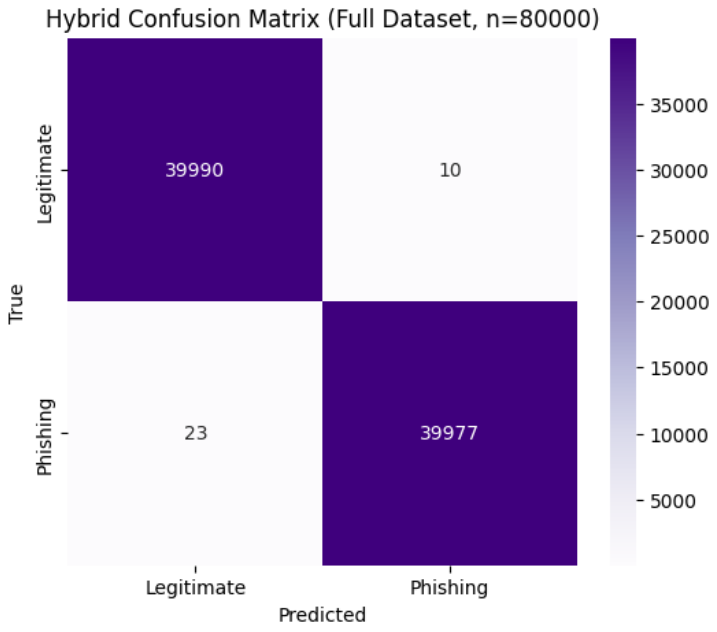


Figure 5.3: Confusion matrix of the hybrid model on the full dataset.

5.4 ROC Curve Analysis

The ROC curve comparison is shown in Figure 5.4. The RoBERTa model achieved an AUC of 0.9999, the autoencoder achieved an AUC of 0.5383, and the hybrid model also achieved an AUC of 0.9999. The ROC curve of the hybrid model is positioned very close to the top-left corner, which indicates excellent classification ability.

The result clearly shows that RoBERTa was the strongest individual component in terms of discrimination power. Its ROC curve nearly overlaps with that of the hybrid model, suggesting that RoBERTa contributed the majority of the separability between phishing and legitimate samples. In contrast, the autoencoder produced an AUC of 0.5383, which is only slightly better than random guessing. This suggests that the reconstruction-based signal alone was not sufficient for strong standalone classification in this experiment.
However, the autoencoder may still provide complementary anomaly-related information when used as part of the hybrid framework. Therefore, while RoBERTa appears to dominate the final performance, the fusion strategy remains meaningful because it combines discriminative language-based features with reconstruction-based information. Overall, the ROC analysis confirms the superiority of the hybrid framework and its ability to achieve highly reliable separation between the two classes.

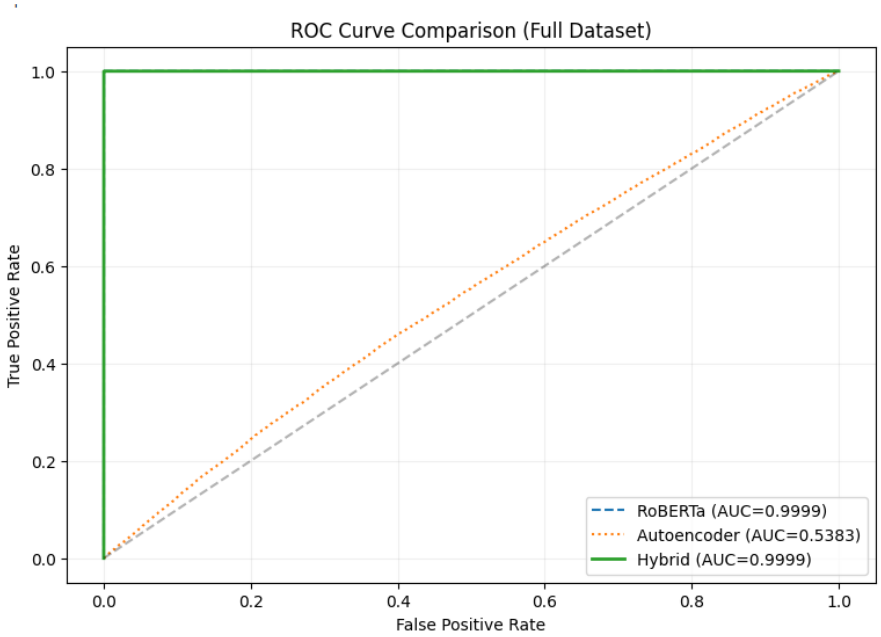


Figure 5.4: ROC curve comparison of RoBERTa, autoencoder, and hybrid model on the full dataset.

5.5 Training Loss Comparison of RoBERTa and Autoencoder

Figure 5.5 compares the training loss of RoBERTa and the autoencoder across five epochs. RoBERTa loss decreased from 0.0945 in epoch 1 to 0.0104 in epoch 5, while the autoencoder loss decreased from 0.3216 to 0.0615 over the same period. Both models showed a clear downward trend, which confirms that the training process was effective for both components.
However, RoBERTa converged much faster and achieved a significantly lower final loss than the autoencoder. This indicates that RoBERTa learned the underlying class-discriminative patterns more efficiently. In contrast, the autoencoder started with a much higher loss and improved more gradually. This slower convergence

is consistent with the ROC results, where the autoencoder showed much weaker standalone performance.
The loss comparison therefore supports the broader experimental findings: RoBERTa provided the primary predictive strength of the framework, while the autoencoder learned more slowly and contributed more modestly. Even so, the decreasing loss of the autoencoder shows that it still captured useful structural information during training.

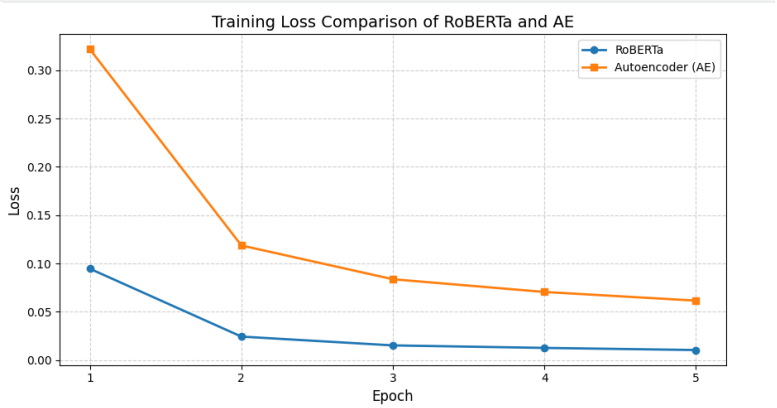


Figure 5.5: Training loss comparison of RoBERTa and autoencoder across five epochs.

5.5.1 Web Application Deployment Details

System Overview

PhishGuard AI is an online phishing URL detector that applies two AI-based models, a RoBERTa-like NLP analyzer, and an Auto Encoder anomaly detector, to identify legitimate and phishing URLs in real time. It comes with a safe user interface comprising scan history, feature analysis, and confidence-based visualization, which offers precise detection as well as user-intuitive cybersecurity experience.

Deployment on Vercel

The web application was deployed with Vercel to make the proposed PhishGuard AI system accessible in the real-life online setting. This deployment enables users to browse the system with a browser and use its basic functionality, such as URL analysis, authentication, and reviewing scan history. The application can be found at its live version at: https://anoy.vercel.app/

System Architecture

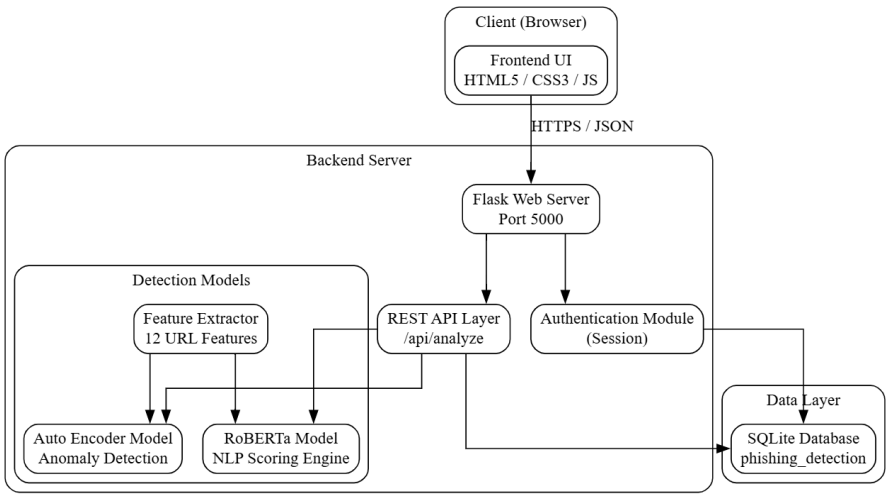


Figure 5.6: System Architecture

As shown in Figure 5.6, this diagram presents the overall architecture of the phishing detection system. The user interacts with the frontend through a browser-based interface built with HTML, CSS, and JavaScript, which sends requests to a Flask web server using HTTPS and JSON. Within the backend, a REST API handles URL analysis requests, while an authentication module manages user sessions. The system extracts 12 URL-based features and forwards them to two detection models: RoBERTa and Auto Encoder. Finally, user information, session data, and phishing detection results are stored in an SQLite database.

Technology Stack

Layer	Technology
Backend Framework	Flask
WSGI Toolkit	Werkzeug
Authentication	Flask-Login + Werkzeug Security
Form Handling	Flask-WTF
Database	SQLite3
Templating Engine	Jinja2
Frontend	HTML5 / CSS3 / Vanilla JavaScript
Typography	Google Fonts
Programming Language	Python
Deployment Platform	Vercel

Table 5.3: Technology Stack

Table 5.3 presents the main technologies used in the development of the PhishGuard AI web application. The backend of the system was developed using *Flask*, which provides a lightweight and flexible framework for handling application logic, URL routing, and server-side processing. Supporting tools such as *Werkzeug*, *Flask-Login*, and *Flask-WTF* were used to strengthen request handling, authentication, session management, and secure form processing. In addition, *SQLite3* was used for database management, while *Jinja2* enabled dynamic rendering of web pages.

On the frontend side, the application was built using *HTML5*, *CSS3*, and *Vanilla JavaScript*, which together provide the structure, styling, and interactivity of the user interface. This combination allowed the development of a responsive and visually consistent environment for account creation, login, URL analysis, and scan history management. Overall, the technology stack shown in Table 5.3 reflects a simple yet effective full-stack design that supports both the functional backend operations and the interactive frontend experience of the phishing detection system.

5.6 AI Model Architecture

URL-Based Feature Extraction for Phishing Detection

Table 5.4 presents the handcrafted URL-based feature set used in the proposed phishing detection system. These features were selected to capture a wide range of structural, lexical, and statistical characteristics that commonly differentiate phishing URLs from legitimate ones. The feature group includes measurements related to URL length, domain length, path depth, query string usage, protocol security, subdomain count, suspicious top-level domains, special characters, suspicious keywords, brand impersonation patterns, and domain entropy. Each feature is paired with a phishing indicator threshold or condition, allowing the system to identify specific warning signs that are often associated with malicious links.
The purpose of this feature design is to represent the most frequently observed behaviors of phishing URLs in a measurable form. For example, phishing links often contain unusually long and complex addresses, omit HTTPS protection, use raw IP addresses instead of normal domain names, or include multiple subdomains to appear more convincing. Similarly, attackers frequently rely on deceptive keywords such as *login*, *verify*, and *password*, along with suspicious TLDs and obfuscating characters like hyphens or the "@" symbol to mislead users. Brand impersonation is also an important factor, where trusted brand names may appear in the path of the URL rather than in the actual domain. In addition, entropy-based analysis helps detect domains with irregular or randomly generated character patterns, which are often linked to malicious activity.
Together, these 12 features provide a comprehensive representation of URL behavior and contribute meaningful discriminative information for phishing detection. Rather than depending on a single characteristic, the system combines multiple complementary indicators so that both obvious and subtle phishing patterns can be captured. This makes the feature set more robust for identifying malicious URLs across different attack styles and improves the ability of the detection models to distinguish phishing attempts from legitimate web addresses.

Table 5.4: Feature Set Used for Phishing URL Detection

#	**Feature**	**Description**	**Phishing Indicator**
1	URL Length	Total character count	> 75 characters
2	HTTPS Presence	Protocol scheme check	Missing HTTPS
3	IP Address Domain	Regex-based IP detection `^\d{1,3}\.\}{3}\d{1,3}\$`	IP instead of domain name
4	Subdomain Count	Number of domain levels beyond TLD+SLD	> 2 subdomains
5	TLD Classification	Checked against 24 suspicious TLDs (`.tk`, `.xyz`, `.top`, etc.)	Suspicious TLD match
6	Special Characters	@ symbol, hyphens, double-slash redirects	Presence of obfuscation chars
7	Suspicious Keywords	30 keywords (`login`, `verify`, `password`, etc.)	≥ 1 keyword match
8	Brand Impersonation	27 target brands (`paypal`, `google`, `amazon`, etc.) in path vs. domain	Brand in path but not domain
9	Domain Entropy	Shannon entropy calculation over domain characters	Entropy > 4.0
10	Path Depth	Number of path segments	> 4 segments
11	Query String	Presence and length of query parameters	Long query strings
12	Domain Length	Character count of the domain	> 30 characters

RoBERTa Model (NLP-Based Scoring)

The RoBERTa model simulates an NLP-based transformer by applying a cumulative semantic scoring approach:

- **Scoring Mechanism:** Each phishing indicator contributes a weighted score to a cumulative NLP score (max ≈ 22.0).
- **Key Weights:** Brand impersonation (+4.0–4.5), IP address (+3.0), 3+ keywords (+5.0), suspicious TLD (+2.5).
- **Classification Threshold:** NLP score $\geq 3.5 \rightarrow$ Phishing.
- **Confidence Computation:**

$$\texttt{confidence} = \max(0.60,\ \min(0.99,\ 0.55 + |risk - 0.5| \times 2 \times 0.5))$$

- **Deterministic Variance:** MD5-seeded random jitter (± 0.02) for natural-appearing results.

Auto Encoder Model (Anomaly Detection)

The Auto Encoder model simulates anomaly detection via reconstruction error scoring:

- **Scoring Mechanism:** Each deviation from "normal" URL patterns contributes to a cumulative anomaly score (max ≈ 20.0).
- **Key Weights:** Brand in suspicious domain (+4.0), brand in path (+3.5), IP address (+3.0), @ symbol (+2.5).
- **Classification Threshold:** Anomaly score $\geq 4.0 \rightarrow$ Phishing.
- **Reconstruction Error:** Normalized as `anomaly_score / 20.0`, with threshold at 0.2.
- **HTTPS Bonus:** Valid HTTPS reduces anomaly by -1.0.

API Endpoints

Method	Endpoint	Authentication	Description
GET	`/`	Public	Landing page (redirects to dashboard if logged in)
GET/POST	`/register`	Public	User registration
GET/POST	`/login`	Public	User login
GET	`/logout`	Session	Clears session and redirects
GET	`/dashboard`	Required	URL scanner dashboard
POST	`/api/analyze`	Required	Analyze URL — accepts `{"url": "...", "model": "roberta|autoencoder"}`
GET	`/api/history`	Required	Retrieve last 50 scan results (JSON)
POST	/api/history/clear	Required	Delete all user scan history

Table 5.5: Main application routes and API endpoints

Table 5.5 summarizes the main routes and API endpoints used in the PhishGuard AI web application. It includes both public pages, such as the landing page, registration, and login, as well as protected routes that require user authentication, including the dashboard and scan-related APIs. The table shows how the system combines user management and phishing analysis features within a structured web architecture.
In particular, the protected API endpoints support the core functionality of the application. The **/api/analyze** endpoint is responsible for submitting a URL and selected model for phishing detection, while **/api/history** and **/api/history/clear** allow users to view and manage their scan history. Overall, the routes presented in Table 5.5 reflect a clear separation between public access, authenticated user operations, and backend analysis services.

Sequence Diagram

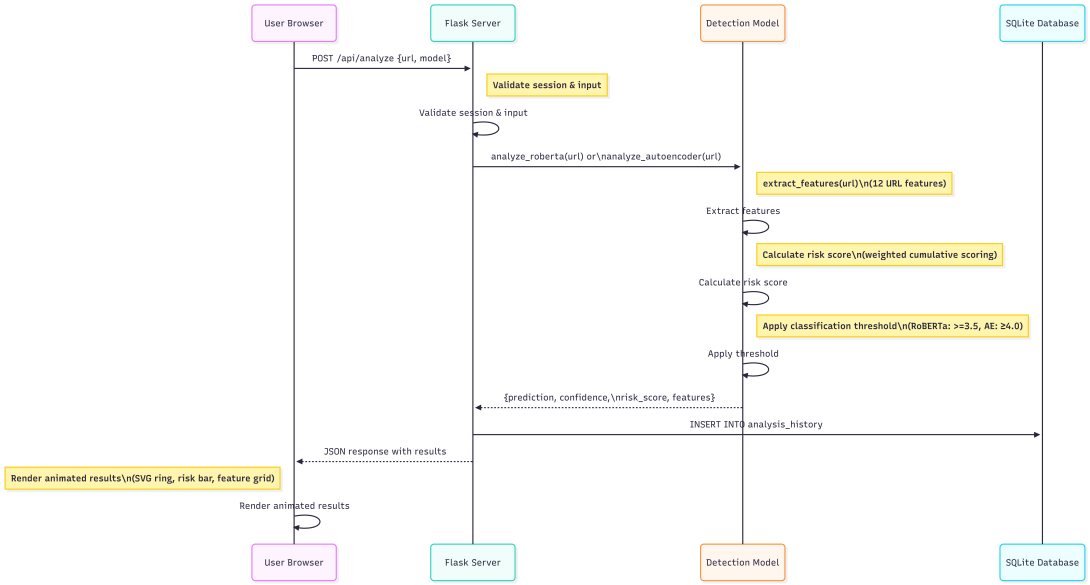


Figure 5.7: Sequence Diagram

As shown in Figure 5.7, this Sequence Diagram shows a clean, layered phishing-URL analysis pipeline. The process starts in the User Browser, which sends a **POST /api/analyze** request with the target URL and selected model. The Flask Server acts as the controller layer: it first validates the user session and input, then forwards the request to the appropriate Detection Model.
Inside the model layer, the system extracts URL-based features, computes a weighted risk score, and applies a model-specific threshold to decide whether the URL is suspicious or safe. After that, the model returns structured results such as prediction, confidence, risk score, and extracted features back to the Flask server.
The server then stores the scan result in the SQLite Database for history tracking and finally returns a JSON response to the browser. On the frontend, the browser renders the outcome visually using elements like an SVG ring, risk bar, and feature grid.
Overall, the flow is well organized because it separates input handling, ML analysis, persistence, and presentation, making the system easier to maintain and scale.

5.6.1 Overall Discussion

Overall, the reported results demonstrate that the proposed hybrid framework achieved excellent phishing detection performance. The 5-fold cross-validation analysis confirmed both high predictive capability and strong stability. The optimized performance metrics further showed that the model preserved high AUC, accuracy, and F1-score under a separate evaluation setup. The confusion matrix revealed only a very small number of misclassifications, while the ROC curves confirmed that the hybrid model offered near-perfect class separation. Finally, the training loss analysis demonstrated stable convergence behavior for both RoBERTa and the autoencoder.

It should also be noted that the presented outputs come from more than one evaluation setting, including cross-validation, optimized evaluation, full-dataset confusion matrix analysis, and component-wise loss comparison. Therefore, the numerical values should be compared within their respective experimental contexts. Even with that consideration, all outputs consistently support the same conclusion: the proposed hybrid framework is highly effective, stable, and suitable for phishing detection tasks.

From the application perspective, the developed PhishGuard AI web system further demonstrates the practical usefulness of the proposed framework. The application integrates secure account registration and login, a dashboard-based URL scanning interface, model selection between RoBERTa and Auto Encoder, detailed prediction outputs, and per-user scan history storage with threat counting and history management. These features show that the framework is not only effective in experimental evaluation but also suitable for deployment in an interactive real-world environment where users can analyze URLs, inspect detailed results, and monitor previous phishing detections efficiently.

Chapter 6

Conclusion

To sum up, the proposed hybrid phishing detection system not only possesses high accuracy and robustness, but also has a considerable influence on the context of cybersecurity applications in the real world. The system improves the safety of the users as it is effective in identifying malicious URLs, and it reduces the risks of financial loss and data breach. It is also a socially beneficial factor by making people more trustful of online activities and contributing to a safe online environment. Legally, it helps prevent cybercrime and is consistent with the practices of data protection. Additionally, despite the fact that the model demands computation resources, the efficient design and optimization methods allow decreasing energy usage and providing sustainable operation. The solution also fosters long-term digital sustainability through reduced repetitive cyber events and overall increased reliability of the system.

Chapter 7

Future Work

Further advancement of the present research will aim at improving phishing detection levels without introducing deviation in terms of structured URL-based analysis and deep learning paradigm. A potential way forward is to create more expressive representation learning strategies that enhance feature abstraction on inputs derived as URLs. This involves the discovery of hybrid architectures that would combine transformer-based contextual modeling with latent embedding methods in order to enhance further generalization to changing patterns of phishing. The other direction with significance is the implementation of ensemble learning techniques which combine both the supervised and unsupervised models. With an integrated approach through RoBERTa and Auto Encoder based classifier predictions, the system can be used to gain better robustness, less model bias and better overall classification reliability. These collections of ensembles can be useful in handling the accuracy of supervised learning and the flexibleness of unsupervised anomaly detection. Further improvements to be examined by future research will be deployment-oriented such as adaptive decision thresholds, continuous learning to adapt to newly emerging phishing patterns and scalability to support real time inference. Such extensions will help to make sure that the framework of phishing detection will be practical and robust and will not collapse under the pressure of changing cybersecurity contexts. On the whole, the following future directions aim to develop the proposed system to become more flexible, adaptive, and production-ready AI-based phishing detection solution that will be able to respond to real-life threats.

Bibliography

[1] H. Le, N. Zincir-Heywood, and M. Heywood, "Urlnet: Learning a url representation with deep learning for malicious url detection," *arXiv preprint arXiv:1802.03162*, 2018.

[2] S. Smadi, N. Aslam, and L. Zhang, "Detection of phishing emails using data mining techniques," *Decision Support Systems*, vol. 116, pp. 165–175, 2018.

[3] Z. Wei et al., "Phishing detection for iot devices using lightweight cnns," *Sensors*, vol. 19, no. 10, p. 2354, 2019.

[4] Z. Xiong, J. Lee, and W. Zhang, "Phishing email detection using deep neural networks and natural language processing techniques," *Journal of Cybersecurity*, vol. 5, no. 3, pp. 123–135, 2019. [Online]. Available: https://www.journals.elsevier.com/journal-of-cybersecurity.

[5] S. Abdelnabi, M. Fritz, and C. Rossow, "Visualphishnet: Zero-day phishing website detection by visual similarity," *ACM SIGSAC Conference on Computer and Communications Security*, vol. 1, pp. 1981–1998, 2020.

[6] H. Al-Ahmadi and O. Alharbi, "Phishing detection using dual-channel deep convolutional neural network," *International Journal of Communication Networks and Information Security*, vol. 12, no. 2, pp. 259–267, 2020.

[7] O. Işık and C. Şahin, "Deep learning-based phishing email detection using mutual information feature selection," *Security and Communication Networks*, vol. 2020, pp. 1–10, 2020.

[8] R. Kumar, V. Patel, and P. Gohil, "Enhanced phishing email detection using attention mechanism and bi-lstm networks," *Computers & Security*, vol. 88, p. 102 101, 2020. [Online]. Available: https://www.journals.elsevier.com/computers-and-security.

[9] U. Opara et al., "Htmlphish: Detecting phishing websites based on html content using cnns," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1236–1247, 2020.

[10] Q. Xiao, Z. Qu, and Y. Hu, "An improved deep learning model for phishing detection using self-attention mechanism and data augmentation," *Computers & Security*, vol. 102, p. 102 104, 2021.

[11] Y. Yang, C. Su, and H. Zhang, "Phishing website detection based on deep learning and random forest," *Sensors*, vol. 21, no. 21, p. 7283, 2021.

[12] C. Catal et al., "A systematic review of deep learning methods for phishing detection," *Knowledge and Information Systems*, vol. 64, no. 6, pp. 1429–1474, 2022.

[13] H. Alshingiti, A. Alsulami, and J. Almotiri, "Phishing website detection based on deep learning techniques," *Electronics*, vol. 12, no. 6, p. 1278, 2023.

[14] S. Karpurapu and S. Imambi, "Phishing detection on twitter using machine learning and the mitre att&ck framework," *Cybersecurity Informatics*, vol. 2, no. 1, pp. 45–56, 2024.

[15] F. Alhuzali, T. Alqahtani, and A. Almomani, "Phishing email detection using bert-lstm hybrid deep learning model," *Applied Sciences*, vol. 15, no. 3, p. 577, 2025.

[16] Enron, "Enron email dataset," *https://www.kaggle.com/datasets/wanghaohan/enron-email-dataset*, n.d. [Online]. Available: https://www.kaggle.com/datasets/wanghaohan/enron-email-dataset.

[17] PhishTank, "Phishtank," *https://phishtank.org*, n.d. [Online]. Available: https://phishtank.org.

[18] VirusTotal, "Virustotal," *https://virustotal.com*, n.d. [Online]. Available: https://virustotal.com.