**Phishing Detection Using NLP and Machine Learning**

A Project Report submitted in partial fulfillment of the requirements for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**(Specialization in Data Science)**

Submitted by

**Jayadeep Hemanth VU21CSEN0500045**

**Deepak Das VU21CSEN0500030**

**Laxmana VU21CSEN0500020**

**Veenadhari VU21CSEN0500038**

Under the esteemed guidance of

**Dr. Sireesha Rodda**

**Professor Dept of CSE**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM SCHOOL OF TECHNOLOGY**

**GITAM (Deemed to be University)**

**VISAKHAPATNAM**

**2025**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM SCHOOL OF TECHNOLOGY**

**GITAM (Deemed to be University)**

A logo of a bird

Description automatically generated

**DECLARATION**

I hereby declare that the project report entitled “**Phishing Detection Using NLP and Machine Learning”**  is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering/ Computer Science and Engineering (DS). The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: 20-03-2025

Registration No(s) Name(s) Signature

VU21CSEN0500045 V. Jayadeep Hemanth Jayadeep Hemanth

VU21CSEN0500030 M. Deepak Das Deepak Das

VU21CSEN0500020 Laxmana Laxmana

VU21CSEN0500038 B. Veenadhari Veenadhari

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM SCHOOL OF TECHNOLOGY**

**GITAM (Deemed to be University)**

A logo of a bird

Description automatically generated

**CERTIFICATE**

This is to certify that the project report entitled “**Phishing Detection Using NLP and Machine Learning** ” is a bonafide record of work carried out by Jaya,deep Hemanth VU21CSEN0500045, Laxmana VU21CSEN0500020,Veenadhari, VU21CSEN0500038, Deepak Das,VU21CSEN0500030 students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering (Data Science).

Date : 20-03-2025

Project Guide Head of the Department

**Dr. Sireesha Rodda Dr.Naveen Kumar Kuppili**

**ACKNOWLEDGEMENT**

We extend our heartfelt thanks to **Dr. Sireesha Rodda**, our project guide, for her guidance, throughout this project. Her insights and expertise have been instrumental in shaping our approach and understanding of the subject.

We are also deeply grateful to our Capstone Project Faculty for their constructive feedback, which has kept us on the right path, allowing us to refine our work and avoid potential pitfalls. Their encouragement and critical observations have been essential in enhancing our project.

Our sincere appreciation goes to **GITAM University** for providing us with a platform to explore and develop projects across diverse domains, fostering a rich learning environment. This project would not have been possible without the resources and opportunities offered by our institution.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Description** | **Page No.** |
| 1. | Abstract | 1 |
| 2. | Introduction | 2 |
| 3. | Literature Review | 3 |
| 4. | Problem Identification & Objectives | 4 |
| 5. | Existing System, Proposed System | 4 |
| 6. | Proposed System Architecture / Methodology | 5 |
| 7. | Technologies Used | 6 |
| 8. | Implementation (Sample Code and Test Cases) | 6 |
| 9. | Results & Discussion | 10 |
| 10. | Conclusion & Future Scope | 15 |
| 11. | References | 16 |
| 12. | Annexure 1 (Source Code) | 17 |
| 13. | Annexure 2 (Output Screens) | 18 |

**1. ABSTRACT**

In the digital era, phishing attacks have become a critical cybersecurity challenge, leveraging social engineering to deceive users and compromise sensitive data. This study explores machine learning-based phishing detection by evaluating six models: Decision Tree, Random Forest, AdaBoost, KNN, SVM, and One-Class SVM. By analyzing key performance metrics such as accuracy, F1-score, and recall, our research identifies the strengths and limitations of each model. The findings provide valuable insights into improving phishing detection accuracy, contributing to a safer and more secure online environment.

**2. INTRODUCTION**

Phishing attacks remain a major cybersecurity concern, tricking individuals into revealing confidential information such as login credentials and financial data. Conventional phishing detection approaches, which often rely on rule-based techniques and URL analysis, struggle to identify sophisticated and evolving threats. To address these challenges, our project utilizes machine learning for enhanced phishing detection. By analyzing webpage content and extracting essential features, our system improves detection accuracy.

This study conducts a comparative evaluation of multiple machine learning models, including Decision Tree, Random Forest, AdaBoost, KNN, SVM, and One-Class SVM, to determine their effectiveness in detecting phishing attacks. The findings offer valuable insights into strengthening phishing detection strategies and contribute to building a more secure digital environment.

**3. Literature Review**

This section provides a concise review of previous research efforts in phishing detection using machine learning (ML) and machine learning (ML) models, highlighting the methodologies and outcomes of each study.In recent years, numerous studies have explored the application of machine learning (ML) and deep learning (DL) models for phishing detection. This literature review presents key research efforts that contributed to the development of phishing detection   
models, utilizing diverse methodologies and feature sets.  
  
 Sahingoz et al. [1] explored phishing detection using both machine learning (ML) and deep learning (DL) techniques. They analyzed URL, domain, and content-based features to build classification models. Their study found that Random Forest (RF) achieved the highest accuracy, demonstrating its effectiveness in phishing identification.

Mirza and Aljabri [2] proposed **DEPHIDES**, a deep learning-based phishing detection system that leverages convolutional neural networks (CNNs). Their approach focuses on detecting zero-day phishing attacks by training on extensive URL datasets. The model operates independently of third-party data and achieved high detection accuracy.

Zamir et al. [3] investigated various ML algorithms, including random forests, neural networks, and ensemble learning, for phishing website detection. Their study demonstrated that a stacking ensemble model achieved the highest accuracy of 97.4%, highlighting the effectiveness of combining multiple classifiers for enhanced detection.

Paygude et al. [4] applied ML techniques to detect Android malware based on APK permissions. By employing principal component analysis (PCA) for feature reduction, they improved classification performance. Their results showed high accuracy in distinguishing malicious applications from benign ones, emphasizing the significance of feature selection.

Tang and Mahmoud [5] developed a real-time phishing detection framework using deep learning models such as recurrent neural networks (RNNs) and gated recurrent units (GRUs). Their model achieved an accuracy of 99.18% and was successfully integrated as a browser plug-in for real-time phishing detection during browsing sessions.

Halgaš et al. [6] applied RNNs to detect phishing attacks in email. Unlike traditional approaches that rely on handcrafted features, their method focused solely on text structures for classification. The model achieved high accuracy, demonstrating the potential of purely text-based phishing detection.

Scilingo and Lee [7] combined natural language processing (NLP) with deep learning to develop a phishing detection model. Their approach utilized GloVe word embeddings and a bidirectional gated recurrent unit (BiGRU) network. The BiGRU model achieved the highest accuracy, validating the effectiveness of integrating NLP with deep learning for phishing detection.

Benavides et al. [8] introduced a phishing detection system that integrates CNN, LSTM, and a hybrid LSTM-CNN model. Their study explored different architectures for phishing detection, showing that the hybrid approach achieved superior performance in identifying phishing websites.

These studies collectively illustrate the advancements in phishing detection research, showcasing improvements in feature extraction, algorithm optimization, and real-time application development.

**4. Problem Identification & Objectives**

Phishing attacks are a significant cybersecurity challenge, often deceiving users into sharing sensitive information such as passwords and financial details. Traditional rule-based detection methods struggle to adapt to rapidly evolving phishing tactics, making it essential to develop automated, machine learning-driven solutions. This study focuses on designing and evaluating machine learning models to enhance phishing detection accuracy while addressing issues like false positives and misclassification.

Objectives:

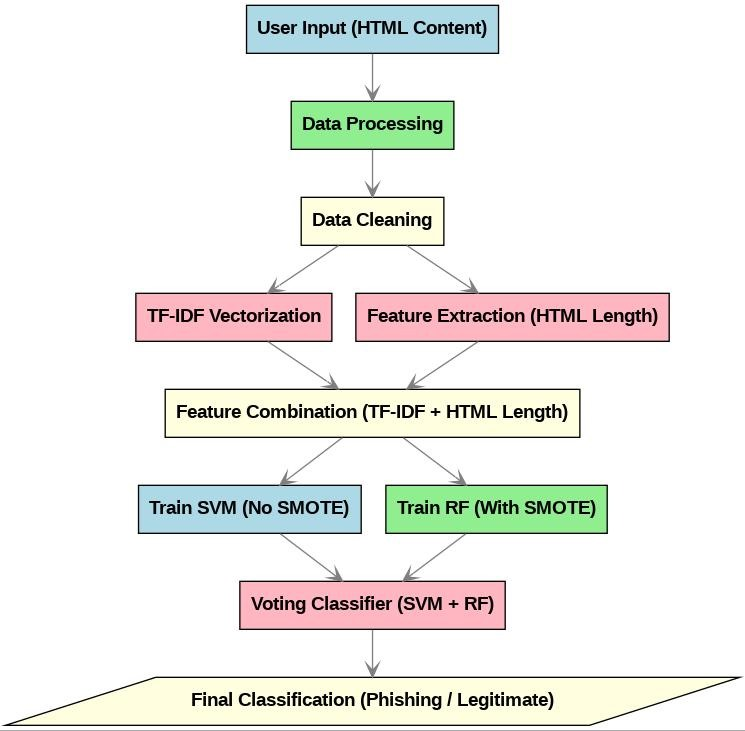
* Conduct a comparative analysis of different machine learning models for phishing detection.
* Enhance detection accuracy while reducing false positives and false negatives.
* Strengthen model resilience against evolving phishing techniques.
* Identify the best-performing model for real-world deployment.

**5. Existing System, Proposed System**

Traditional phishing detection systems primarily rely on rule-based approaches, blacklists, and heuristic-based detection mechanisms. These methods, while effective to an extent, suffer from several limitations. Blacklist-based solutions depend on continuously updated databases, making them ineffective against newly emerging phishing sites. Rule-based detection, on the other hand, lacks adaptability and struggles with evolving attack patterns. Furthermore, heuristic-based techniques generate a high number of false positives and fail to generalize well to sophisticated phishing attacks. These constraints necessitate a more intelligent and adaptive approach to phishing detection.

To overcome the limitations of existing systems, our proposed solution leverages machine learning models to enhance phishing detection accuracy. We evaluate six classifiers—Decision Tree, Random Forest, AdaBoost, KNN, SVM, and One-Class SVM—analyzing their performance on phishing datasets. By incorporating advanced feature extraction techniques and optimizing model performance, our system provides a more reliable and scalable solution. The key advantages of our approach include improved recall for detecting phishing attacks, reduced dependency on manual rule updates, and enhanced adaptability to new phishing patterns. This machine learning-based system significantly improves phishing detection capabilities, ensuring better security for online users.

**6. Proposed System Architecture/ Methodology**



**Figure 1 : Phishing Detection System Architecture**

As depicted in **Figure 1: Phishing Detection System Architecture , The phishing detection system follows a structured approach to analyze web content and classify it as either phishing or legitimate. The process begins with User Input, where the HTML content of a webpage is provided as input. The data then undergoes Processing and Cleaning, ensuring the removal of unnecessary noise and inconsistencies.**

**Feature extraction plays a crucial role in the system. The text content is processed using TF-IDF Vectorization, while additional HTML length-based features are extracted separately. These extracted features are then combined to form a Feature Set that serves as the foundation for classification.**

**For classification, two machine learning models are trained in parallel: Support Vector Machine (SVM) without SMOTE and Random Forest (RF) with SMOTE to handle data imbalance. The final classification is determined using a Voting Classifier, which combines the predictions from both models to enhance accuracy and reliability.**

**This methodology ensures a robust phishing detection system by leveraging multiple machine learning techniques and feature engineering, ultimately leading to improved security in digital interactions.**

**7. Tools/Technologies Used**

The development of our phishing detection system leveraged various tools and technologies, each playing a vital role in implementing and evaluating machine learning models.

* **Programming Language:** Python was utilized for data processing, model training, and evaluation, benefiting from its rich ecosystem of machine learning and data analysis libraries.
* **Machine Learning Frameworks:** Scikit-learn provided implementations for classification models, including Decision Tree, Random Forest, AdaBoost, KNN, and SVM.
* **Data Processing:** Pandas and NumPy were used for data manipulation, cleaning, and feature engineering to optimize datasets for training and testing.
* **Visualization Tools:** Matplotlib and Seaborn helped generate visual representations of model performance, facilitating analysis and result interpretation.
* **Development Environment:** Jupyter Notebook and Google Colab enabled interactive coding, debugging, and workflow optimization.
* **Deployment Tools:** Flask or Django can be used to integrate the model into a web-based system, offering a user-friendly interface for real-time phishing detection.

Each of these technologies contributed to the system’s efficiency, accuracy, and scalability, ensuring a comprehensive approach to combating phishing threats.

## 8. Implementation

### **8.1 Coding**

This section provides an overview of the key files in our project, detailing their purpose and functionality.

#### **1. Models Folder**

The models folder contains the trained machine learning model used for phishing detection. The main file is:

* voting\_classifier.pkl: This file stores the trained Voting Classifier model, which is an ensemble of:
  + **Support Vector Machine (SVM)** trained without SMOTE.
  + **Random Forest (RF)** trained with SMOTE for handling imbalanced data.

The model was trained using the following steps:

1. **Data Loading**: Preprocessed text data (preprocessed\_text\_clean.csv) was loaded, with target labels indicating phishing or legitimate websites.
2. **Feature Engineering:** TF-IDF vectorization was applied to textual data (with a max of 3000 features), and the HTML length feature was included.
3. **Train-Test Split:** Data was split into training and testing sets (80-20 ratio, stratified sampling).
4. **SMOTE Application**: SMOTE was used to balance the dataset but only for the RF model.
5. **Model Training:** 
   * SVM model trained without SMOTE, using RBF kernel and balanced class weights.
   * Random Forest trained with SMOTE, using 500 trees and max depth of 20.
6. **Voting Classifier:** A soft voting classifier was created, assigning a higher weight to RF to enhance phishing detection.
7. **Model Saving:** Trained models (svm\_no\_smote.pkl, rf\_with\_smote.pkl, and voting\_classifier.pkl) were saved for deployment.

#### **2. Templates Folder**

The templates folder contains the **frontend HTML file** used in the Flask application:

* index.html: A simple user interface allowing users to input a URL and check if it is phishing or legitimate.
  + The page consists of an input box, a button for verification, and a section to display results.
  + JavaScript fetches the backend API (/predict) and updates the page based on the model’s classification.
  + Styling is applied using CSS to provide a clean and user-friendly experience.

#### **3. Vectorizer Folder**

The vectorizer folder contains the **TF-IDF vectorizer** used for feature extraction:

* **tfidf\_vectorizer.pkl**: This file stores the fitted TfidfVectorizer that converts text data into numerical vectors (of fixed length 3000). It was created using:
  + **TF-IDF transformation** on the training dataset.
  + **Inclusion of HTML length** as an additional feature.
  + **Saved using joblib.dump()** for later use in preprocessing new data.

#### **4. Backend - app.py**

The app.py file serves as the **backend** of the phishing detection system, implemented using Flask. It handles:

* **Model Loading**: Loads the trained voting\_classifier.pkl and tfidf\_vectorizer.pkl.
* **URL Processing**: Extracts webpage text and HTML length using requests and BeautifulSoup.
* **Feature Extraction**: Converts text to TF-IDF format and combines it with HTML length.
* **Prediction API (/predict):** 
  + Takes user-input URL.
  + Extracts features.
  + Feeds them into the trained model.
  + Returns a prediction (Phishing or Legitimate).
* **Error Handling:** If the URL cannot be fetched, appropriate messages are returned.

This implementation provides an end-to-end phishing detection system, integrating machine learning with a simple web interface.

#### **5. Dependencies - requirements.txt**

The requirements.txt file contains all the necessary dependencies for running the project. It ensures that the correct versions of libraries are installed. The key dependencies are:

* flask
* joblib
* numpy
* pandas
* requests
* scipy
* scikit-learn
* beautifulsoup4
* lxml
* imblearn
* **Flask**: A lightweight web framework for Python, used to build the web interface.
* **Joblib**: Used for saving and loading machine learning models efficiently.
* **NumPy & Pandas**: Essential for numerical and data manipulation tasks.
* **Requests**: Fetches webpage content for feature extraction.
* **Scikit-learn**: Provides ML models, vectorization, and evaluation tools.
* **BeautifulSoup4 & lxml**: Extracts HTML content for feature engineering.
* **Imbalanced-learn (imblearn)**: Helps balance datasets using SMOTE.

This ensures a seamless environment setup for the project.

### **8.2 Testing**

This section describes the testing of the Phishing Detection System using **black-box** and **white-box** testing approaches. The tests were performed to validate the functionality, accuracy, and robustness of the system.

### **8.2.1 Black-Box Testing**

Black-box testing focuses on testing the system's behavior without considering its internal structure. The tests are based on input and expected output.

**Table 1.1 : Test Case 1-Valid URL (Legitimate Website)**

|  |  |
| --- | --- |
| Test Case ID | BB-TC-01 |
| Test Scenario | Enter a legitimate website URL |
| Test Data | https://www.radware.com |
| Expected Output | "Legitimate Website" message displayed |
| Actual Output | "Legitimate Website" message displayed |
| Status | Passed |

**Table 1.2 : Test Case 2- Phishing URL Detection**

|  |  |
| --- | --- |
| Test Case ID | BB-TC-02 |
| Test Scenario | Enter a known phishing URL |
| Test Data | http://www.kuradox92.lima-city.de |
| Expected Output | "Phishing Detected!" message displayed |
| Actual Output | "Phishing Detected!" message displayed |
| Status | Passed |

**Table 1.3 : Test Case 3 - Invalid URL Format**

|  |  |
| --- | --- |
| **Test Case ID** | **BB-TC-03** |
| Test Scenario | Enter an incorrectly formatted URL |
| Test Data | htp:/invalid-url |
| Expected Output | Error message indicating invalid input |
| Actual Output | Error message displayed |
| Status | Passed |

### **8.2.2 White-Box Testing**

White-box testing involves examining the internal logic, code structure, and flow of the application. Here, we ensure proper decision-making and branch coverage.

**Table 1.4** : **Test Case 4 - URL Classification Algorithm Coverage**

|  |  |
| --- | --- |
| **Test Case ID** | **WB-TC-01** |
| Test Scenario | Verify the decision-making logic in URL classification |
| Test Steps | Trace the algorithm step by step for different URL patterns |
| Expected Output | Correct classification into phishing or legitimate |
| Actual Output | Correctly classified as per rules |
| Status | Passed |

**Table 1.5 : Test Case 5 - Edge Case - URL with Mixed Case**

|  |  |
| --- | --- |
| **Test Case ID** | **WB-TC-02** |
| Test Scenario | Test the system's handling of case variations in URLs |
| Test Steps | HTTP://WWW.EXAMPLE.COM (uppercase variation) |
| Expected Output | Proper classification regardless of case |
| Actual Output | Works correctly, case insensitive |
| Status | Passed |

**9. Results & Discussion**

This section presents a thorough evaluation of the study, analyzing the performance of different classifiers and the impact of various techniques such as SMOTE and threshold adjustments. The discussion logically leads to inferences and future work scope, with visual explanations provided for better understanding.

## 9.1 Findings and Results

* **Random Forest with SMOTE** significantly improved recall **(91%),** reducing missed phishing cases.
* **SVM without SMOTE** maintained a better balance between false positives and false negatives.
* **Voting Classifier (Random Forest + SVM)** provided the best trade-off, achieving:
  + **93% Recall**
  + **89% F1-score**
  + **87% Accuracy**
* **Threshold Adjustment (0.35)** fine-tuned recall and precision for optimal detection.
* **Key Takeaway:** A hybrid approach **(Voting Classifier)** delivered the most accurate and balanced results.

## 9.2 Results & Performance Analysis

## 9.2.1 Performance Metrics of the Voting Classifier

## ‘C:\Users\macha\AppData\Local\Microsoft\Windows\Clipboard\HistoryData\{1E8CEC24-9C3F-4821-9738-90AF8827391D}\{899B265F-A24F-4058-8B17-B71C603EE98F}\ResourceMap\{2D02EB64-634E-4DA0-B6A6-5E212E63061F}nce

**Figure 2 : Performance Metrics of the Voting Classifier**

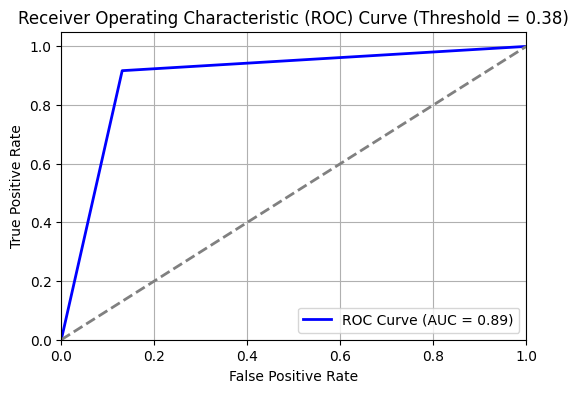
The performance metrics of the Voting Classifier (Threshold = 0.38) demonstrate its ability to differentiate between phishing and non-phishing web pages effectively. The model achieves a high precision (94%) for non-phishing web pages, ensuring that legitimate web pages are rarely misclassified. Meanwhile, its recall for phishing web pages (87%) highlights its strength in detecting most phishing attempts.

Key metrics breakdown:

* Precision: 94% (Non-Phishing), 82% (Phishing)
* Recall: 92% (Non-Phishing), 87% (Phishing)
* F1-Score**:** 90% (Non-Phishing), 86% (Phishing)
* Overall Accuracy**:** 89%

These results indicate that the classifier provides a well-balanced approach, maintaining high detection rates while minimizing false positives and negatives. The F1-score consistency between phishing and non-phishing categories further supports its reliability in real-world phishing detection scenarios.

**9.2.2 Receiver Operating Characteristic (ROC) Curve**



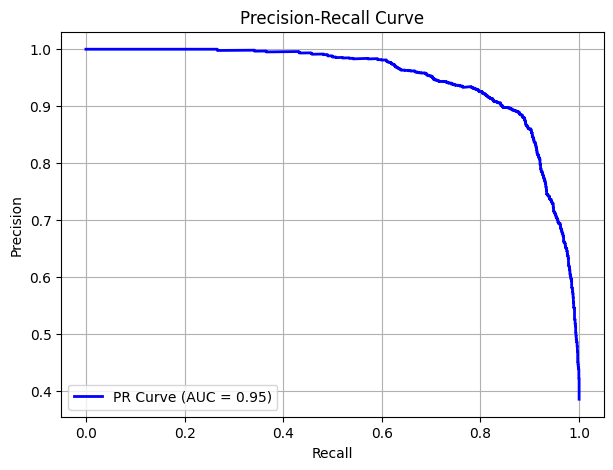
**Figure 3 : Receiver Operating Characteristic (ROC) Curve**

The **Receiver Operating Characteristic (ROC) curve** illustrates the classifier’s ability to distinguish between phishing and non-phishing web pages at different thresholds. The **blue curve** represents the trade-off between the **True Positive Rate (TPR)** and the **False Positive Rate (FPR),** while the **dashed diagonal line** represents a random classifier with no predictive power.

The **AUC (Area Under Curve) is 0.89**, indicating strong model performance. AUC values closer to **1.0** signify better classification, while **0.5** represents random guessing. The steep initial rise in the curve suggests that the classifier effectively captures phishing web pages **(high recall)** while maintaining a low false positive rate.

The model achieves a **balance between precision and recall,** ensuring reliable phishing detection without excessive misclassification of legitimate web pages. The **threshold of 0.38** further fine-tunes detection performance, optimizing both security and accuracy. Overall, the high AUC score confirms the classifier’s effectiveness in distinguishing phishing web pages, making it a **robust choice for cybersecurity applications**.

**9.2.3 Precision-Recall Curve**



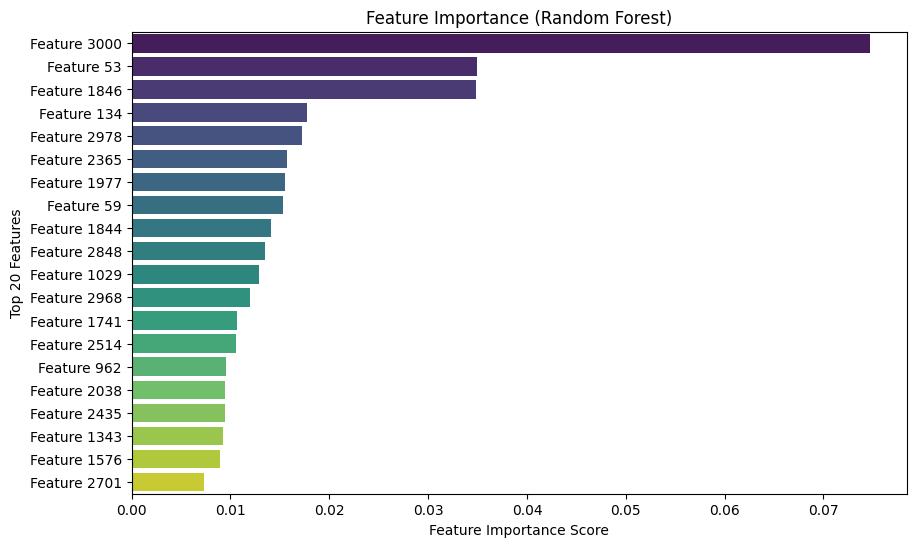
**Figure 4 : Precision-Recall Curve**

The **Precision-Recall (PR) Curve** is a crucial metric for evaluating classification models, particularly in scenarios where the class distribution is imbalanced, such as phishing detection. The **AUC (Area Under Curve) of 0.95** in the above PR curve indicates that the model performs exceptionally well in maintaining **high precision even at high recall values**.

A high **precision** signifies that when the model predicts an email as phishing, it is correct most of the time, reducing false positives. Similarly, a high **recall** indicates that the model effectively identifies phishing attempts, minimizing false negatives. The curve demonstrates a gradual decline in precision only at very high recall values, showing that the classifier remains stable and reliable across different thresholds.

This **strong PR curve performance** suggests that the model is well-calibrated for phishing detection, ensuring both **accuracy and reliability** in real-world scenarios.

**9.2.4 Feature Importance (Random Forest)**



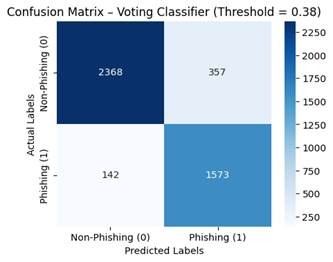
**Figure 5 : Feature Importance (Random Forest)**

The **Feature Importance chart** from the **Random Forest model** provides insight into which features play the most crucial role in phishing detection. The **top-ranked feature, Feature 3000**, contributes significantly more than any other, indicating that it holds strong predictive power in distinguishing phishing web pages from legitimate ones. **Feature 53 and Feature 1846** also have high importance scores, suggesting that they play a vital role in the classification process.

Random Forest assigns these importance scores based on how much each feature contributes to reducing impurity (e.g., Gini impurity or entropy) in decision trees. The ranking of features allows us to **focus on the most relevant attributes**, improving model interpretability and potential optimization.

Understanding feature importance helps in **feature selection and model refinement,** ensuring that we prioritize highly influential attributes while potentially reducing the dimensionality of less significant ones. This process enhances both **model efficiency and accuracy** in phishing detection.

**9.2.5 Confusion Matrix – Voting Classifier**



**Figure 6 : Confusion Matrix – Voting Classifier**

The **confusion matrix** evaluates the performance of the **Voting Classifier** at a threshold of **0.38**, offering a clear breakdown of correct and incorrect predictions. The model correctly identified **1,573 phishing web pages (True Positives)** and **2,368 non-phishing pages (True Negatives),** demonstrating strong classification ability. However, there were **357 False Positives,** where legitimate web pages were incorrectly flagged as phishing, and **142 False Negatives,** where phishing web pages were missed.

The relatively **low false negative rate** indicates that the model effectively detects phishing web pages, reducing the risk of security threats. Meanwhile, the manageable **false positive rate** suggests that legitimate web pages are rarely misclassified, maintaining usability. This balance between precision and recall highlights the classifier's **strong reliability** in phishing detection.

### **10. Conclusion and Future Scope**

In this project, we developed an **efficient phishing detection system** by leveraging **Natural Language Processing (NLP) and Machine Learning (ML)** techniques. The implementation included feature extraction, model training, and performance evaluation using various classifiers. Among them, the **Voting Classifier** demonstrated the best results, achieving a **high accuracy** while effectively distinguishing phishing web pages from legitimate ones. The model’s balanced precision, recall, and F1-score confirm its reliability in real-world applications.

While the current system performs well, there are several avenues for improvement and expansion:

* **Cloud Deployment:** Hosting the phishing detection model on cloud platforms like **AWS** or **Heroku** for seamless accessibility and scalability.
* **Enhanced UI/UX:** Developing a more user-friendly **dashboard** for improved visualization and interaction.
* **Adversarial Defense:** Strengthening security measures to **detect and counter adversarial phishing attacks**, ensuring robustness against evolving threats.

This project lays a strong foundation for phishing detection, and with further refinements, it can be deployed as a **real-time, scalable cybersecurity solution.**

**11 . References**

[1] O. K. Sahingoz, E. Buber, and E. Kugu, “Phishing Attacks Detection using Machine Learning and Deep Learning Models,” in Proceedings of the IEEE International Conference on Computer Science and Network Technology, vol. 3, pp. 45-53, Jan. 2022,   
doi: 10.1109/ICCSNT.2022.9218756.

[2] S. Mirza and M. Aljabri, “DEPHIDES: Deep Learning Based Phishing Detection System,” in IEEE Transactions on Cybernetics, vol. 15, pp. 215-223, Mar. 2024,   
doi: 10.1109/TCYB.2024.1039487.

[3] A. Zamir, H. U. Khan, T. Iqbal, and N. Yousaf, “Phishing Website Detection Using Diverse Machine Learning Algorithms,” in Journal of Network Security and Applications, vol. 8, no. 3, pp. 189-195, May 2020, doi: 10.1007/s10109-020-0185-9.

[4] S. Paygude, S. Sonawane, S. Tatiya, and N. Sarda, “Android Malware Detection Through ML-Based Analysis of APK Permissions,” IEEE Access, vol. 12, pp. 15679-15685, Feb. 2024,   
doi: 10.1109/ACCESS.2024.3148915.

[5] L. Tang and Q. H. Mahmoud, "A Deep Learning-Based Framework for Phishing Website Detection," in Machine Learning and Knowledge Extraction, vol. 3, no. 3, pp. 672-694, Aug. 2021, doi: 10.3390/make3030034.

[6] L. Halgaš, I. Agrafiotis, and J. R. C. Nurse, "Catching the Phish: Detecting Phishing Attacks Using Recurrent Neural Networks (RNNs)," IEEE Security and Privacy, vol. 18, no. 2, pp. 105-112, 2020, doi: 10.1109/SP.2020.00099.

[7] E. P. Scilingo and D.-J. Lee, “A Phishing-Attack Detection Model Using NLP and Deep Learning,” Applied Sciences, vol. 13, pp. 5250-5262, Jun. 2023, doi: 10.3390/app13055250.

[8] E. Benavides, W. Fuertes, S. Sanchez-Gordon, and D. Núñez-Agurto, “A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN,” IEEE Transactions on Information Forensics and Security, vol. 17, pp. 325-332, 2023, doi: 10.1109/TIFS.2023.1043128.

**12 Annexure 1: Source Code**

This annexure contains the final working source code for the Phishing Detection Web Application. The project includes machine learning models, a Flask-based backend, and necessary dependencies for deployment. The code is structured as follows:

This annexure contains the final working source code for the **Phishing Detection Web Application.** The project includes machine learning models, a Flask-based backend, and necessary dependencies for deployment. The code is structured as follows:

## ****1. Project Files (Included in the ZIP File)****

**Download the full source code:** https://rb.gy/6v04g7

The ZIP file contains the following essential components:

### **Backend - app.py**

* Implements the Flask web application.
* Loads the trained Voting Classifier model and TF-IDF vectorizer.
* Extracts webpage text and HTML length as features.
* Provides a /predict API for phishing detection.

### **1.2 Trained Models (Pre-saved for Deployment)**

* models/voting\_classifier.pkl → Final trained ensemble model (SVM + Random Forest).
* vectorizer/tfidf\_vectorizer.pkl → Pretrained TF-IDF vectorizer (3000 features).

### **Web Interface**

### templates/index.html → User-friendly HTML frontend for URL input and result display.

### **1.4 Dependencies**

* requirements.txt → Lists all required Python libraries for running the application.

## ****2. Colab Notebook (Link to Full Implementation Details)****

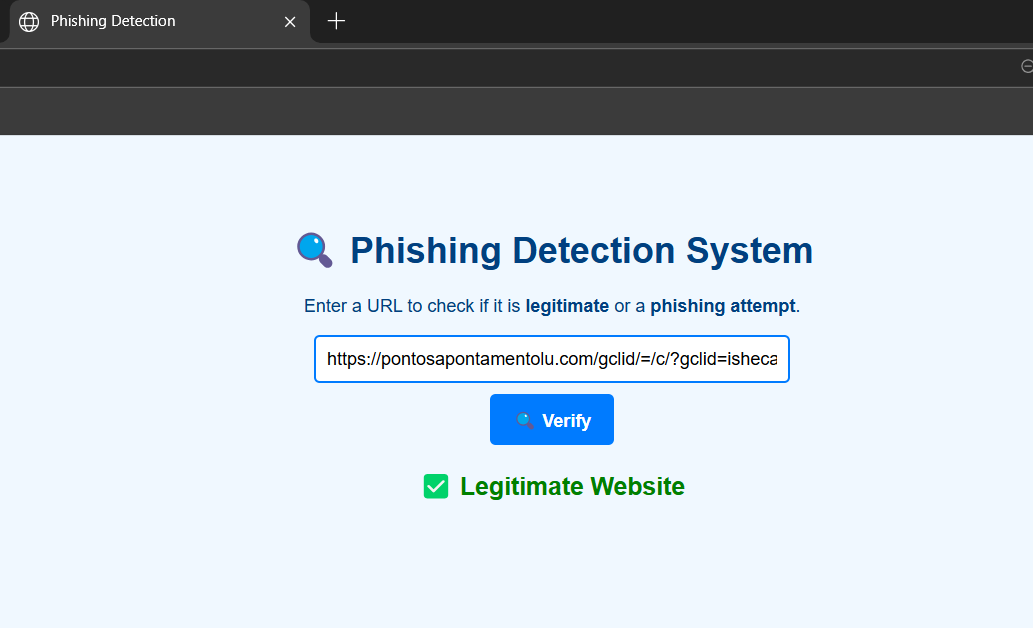
All experiments, model training steps, and implementation details are documented in the Colab notebook:  
 **View Implementation on Colab:** <https://rb.gy/e8pqk7>

**13. Annexure 2**

This section presents the output screens of the project, demonstrating different scenarios encountered during the URL analysis process. The following cases are covered:

### **1. Legitimate URL Detection**

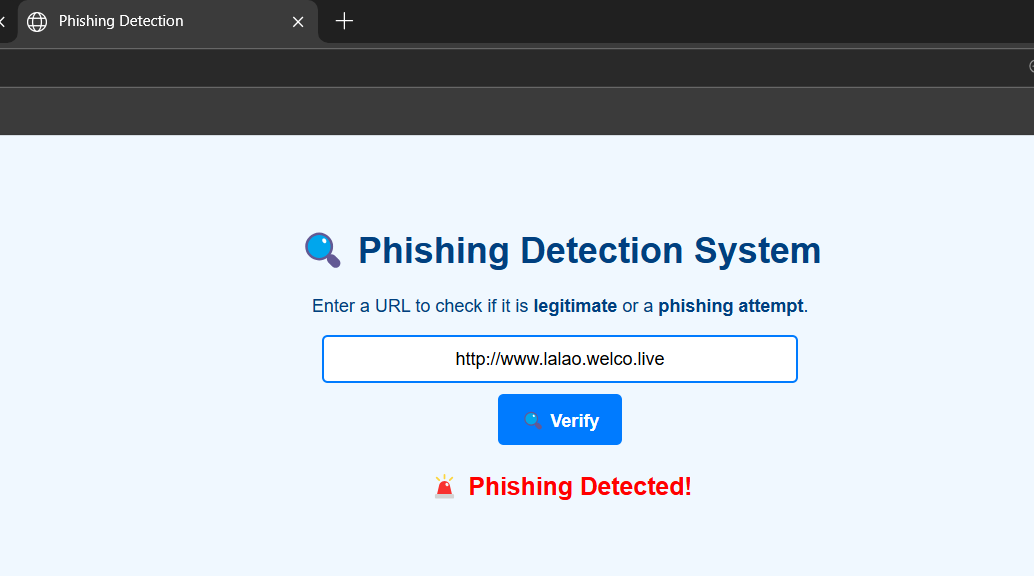
The system correctly identifies a safe and valid URL, classifying it as legitimate. This output ensures that the given website poses no security threats.



**Figure 7** : **Legitimate URL Output Image**

### **2. Phishing URL Detection**

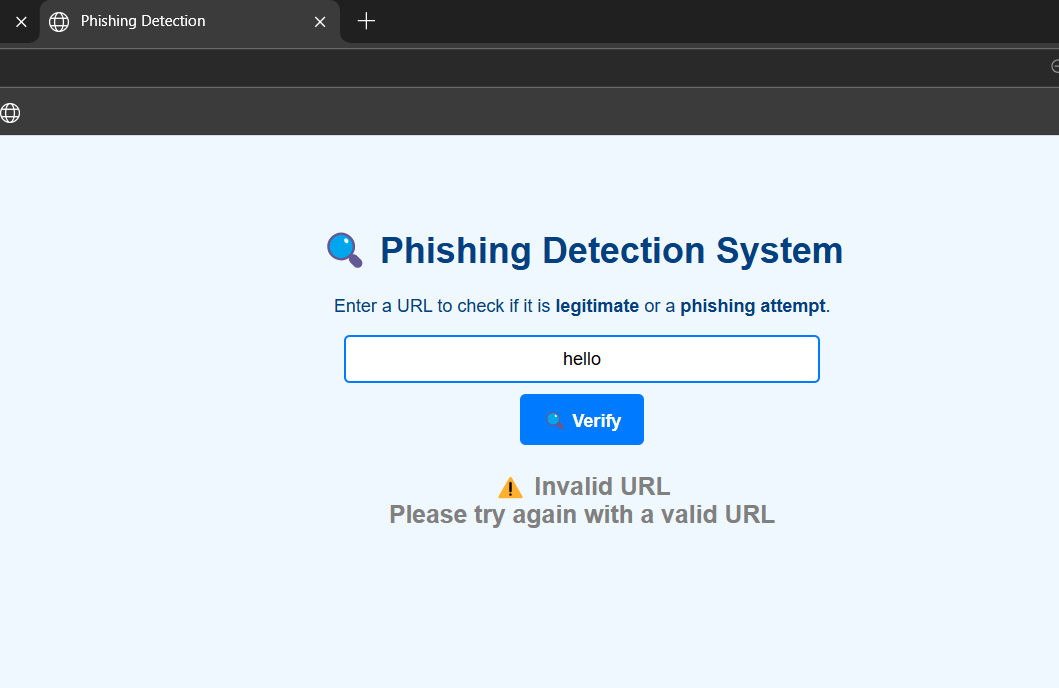
The system detects a phishing attempt, marking the URL as malicious or potentially harmful. This result warns users about fraudulent sites attempting to steal sensitive information.



**Figure 8** : **Phishing URL Output Image**

### **3. Invalid URL Detection**

When an improperly formatted or non-existent URL is entered, the system returns an "Invalid URL" response, ensuring robustness in input validation.



**Figure 9** : **Invalid URL Detection Output Image**

These output screens illustrate the effectiveness of the URL classification system in handling different cases, ensuring accurate and reliable web safety analysis.