**A**

## Project Report On

**“PHISHING METER”**

## Submitted By,

Mr. Harshvardhan Ravindra Pasalkar

**MCA-II SEM-III, DIV- ‘C’**

## Roll No: 23345

Under The Guidance Of

## Prof. Dipali Patil

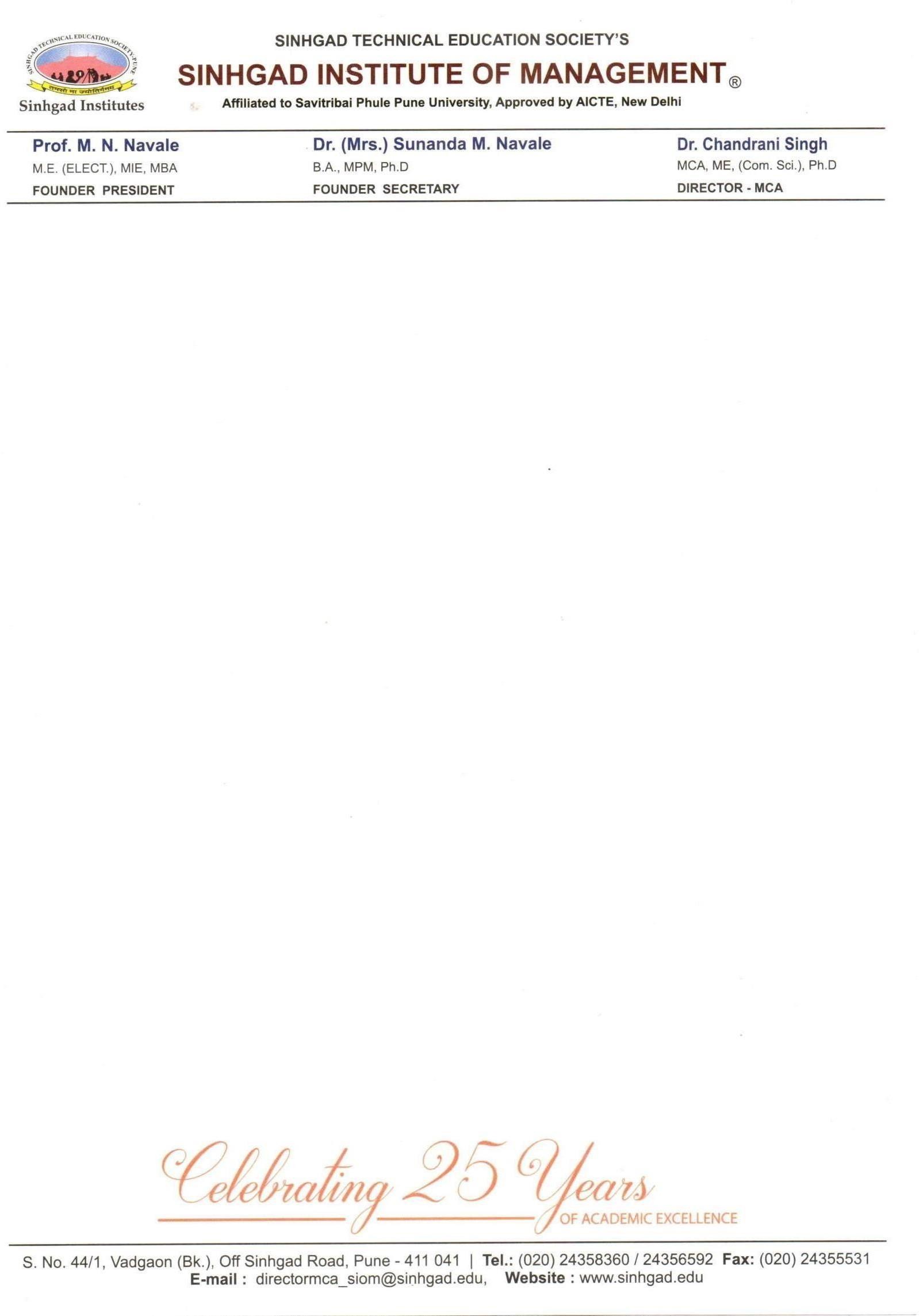
**For the Academic Year 2024-25**



**SUBMITTED TO,**

***Sinhgad Technical Education Society’s* Sinhgad Institute of Management Vadgaon Bk Pune 411041**

**(Affiliated to SPPU Pune & Approved by AICTE New Delhi)**



Date:

**CERTIFICATE**

This is to certify that MR. Harshvardhan Ravindra Pasalkar has successfully completed his project work entitled **“Phishing Prevention”** in partial fulfillment of MCA – II SEM –III Mini Project for the year 2024-2025. He has worked under our guidance and direction.

Prof. Dipali Patil Dr. Chandrani Singh

##### Project Guide Director, SIOM-MCA

Examiner 1 Examiner 2

**Date: Place:** Pune

**DECLARATION**

I certify that the work contained in this report is original and has been done by me under the guidance of my guide.

* The work has not been submitted to any other Institute for any degree or diploma.
* I have followed the guidelines provided by the Institute in preparing the report.
* I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
* Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

**Name and Signature of Project Team Members**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Seat No.** | **Name of students** | **Signature of students** |
| **1** |  | **Harshvardhan Ravindra Pasalkar** |  |

**ACKNOWLEDGEMENT**

It is very difficult task to acknowledge all those who have been of tremendous help in this project. I would like to thank my respected guide **Prof. Dipali Patil** for providing me necessary facilities to complete my project and also for their guidance and encouragement in completing my project successfully without which it wouldn’t be possible. I wish to convey my special thanks and immeasurable feelings of gratitude towards **Dr. Chandrani Singh, Director, SIOM-MCA.** I wish to convey my special thanks to all teaching and non-teaching staff members of **Sinhgad Institute of Management, Pune** for their support.

Thank You Yours Sincerely,

Harshvardhan Ravindra Pasalkar

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Chapter** | **Page No.** |
| 1 | **CHAPTER 1: INTRODUCTION** | 1 |
| 1.1 | Abstract | 2 |
| 1.2 | Existing System and Need for System | 3-4 |
| 1.3 | Scope of System | 5 |
| 1.4 | Operating Environment Hardware and Software | 6 |
| 1.5 | Brief Description of Technology Used | 7-8 |
| 1.6 | Machine Learning Approach | 9-12 |
| 2 | **CHAPTER 2: PROPOSED SYSTEM** |  |
| 2.1 | Feasibility Study | 1-15 |
| 2.2 | Objectives of the proposed system | 16 |
| 2.3 | Users of the system | 17-18 |
| 3 | **CHAPTER 3: ANALYSIS AND DESIGN** |  |
| 3.1 | Use Case Diagrams | 20 |
| 3.2 | Activity Diagram | 21 |
| 3.3 | Sequence Diagram | 22 |
| 3.4 | Dataset Description | 23-24 |
| 3.5 | Evaluation Metrics | 25-26 |
| 3.6 | Experimental Result | 27-30 |
| 3.7 | Sample Input and Output Screens  (Screens must have valid data. All reports must have at-least 5 valid records.) | 31-34 |
| 4 | **CHAPTER 4: CODING Sample code** | 35-36 |
| 5 | **CHAPTER 5: TESTING** | 37-41 |
| 6 | **CHAPTER 6: LIMITATIONS OF SYSTEM** | 42 |
| 7 | **CHAPTER 7: PROPOSED ENHANCEMENTS** | 43 |
| 8 | **CHAPTER 8: CONCLUSION** | 44 |
| 9 | **CHAPTER 9: BIBLIOGRAPHY** | 45 |

# CHAPTER 1: INTRODUCTION

**1.1 Abstract**

Phishing attacks are a widespread and evolving cyber threat that targets users by imitating legitimate websites to steal sensitive information. Unsuspecting users, especially those unfamiliar with the intricacies of cybersecurity, are particularly vulnerable to these deceptive sites. This project addresses the issue by developing a Chrome extension that uses machine learning to detect and block phishing URLs in real time, providing users with a proactive layer of security while they browse. Unlike traditional blacklist-based systems, which rely on databases of known malicious sites that may be outdated or incomplete, the proposed system dynamically analyses URL features and patterns to determine the likelihood of phishing.

The core functionality of this tool is powered by several machine learning algorithms, including Support Vector Machines (SVM), Random Forest, k-Nearest Neighbours (kNN), XGBoost, and Artificial Neural Networks (ANNs). These algorithms classify URLs based on key indicators associated with phishing, such as URL length, IP address usage, SSL certification status, and other suspicious patterns often present in phishing sites. By training on datasets like the UCI Phishing Website Dataset, the model learns to recognize and generalize phishing characteristics, making it capable of detecting previously unknown phishing URLs.

This solution also aims to educate users by providing alerts and explanations when suspicious sites are encountered, helping them recognize risky URLs. The Chrome extension, acting as the user interface, ensures that protection is seamlessly integrated into the browsing experience, requiring no additional action from users. Built in Python, the backend model leverages powerful libraries such as scikit-learn and TensorFlow, enabling efficient real-time detection without significant computational strain.

The system’s adaptability is a crucial feature. As phishing tactics evolve, the model can be retrained with new data, allowing it to respond to emerging threats. Furthermore, it can be expanded to support other browsers and integrate with enterprise security infrastructures, offering scalable protection to a wide audience. This project thus presents a robust, flexible, and user- friendly solution for combating phishing attacks, providing a critical safeguard against one of today’s most prevalent online threats.

* 1. **Existing System and Need for System**

#### Existing System:

Phishing detection in most current systems relies heavily on blacklists, where known malicious URLs are flagged and blocked. These blacklists are often compiled based on previous reports of phishing sites, meaning they can only address known threats. However, with phishing tactics constantly evolving, blacklists quickly become outdated as attackers create new, unlisted phishing sites. This approach also introduces latency, as there is often a delay in identifying and adding new phishing sites to the list, leaving users vulnerable in the interim.

Other existing systems may use heuristics or rule-based methods that flag URLs based on patterns, keywords, or reputation scores. For example, URLs with unusually long strings, suspicious characters, or known phishing indicators may be flagged. While useful, these rules are often simplistic and cannot reliably detect sophisticated phishing tactics. This limitation results in high rates of false positives, where legitimate sites are wrongly flagged, or false negatives, where phishing sites go undetected.

In addition, many users are unaware of common phishing tactics and may not recognize suspicious signs, such as subtle URL changes or visual similarities with legitimate sites. This makes it challenging for non-technical users to spot phishing attempts, even when given a warning. Overall, these traditional methods provide limited protection and fail to keep up with the fast-paced development of new phishing tactics.

#### Need for a System

The limitations of existing phishing detection systems highlight the need for a more adaptive, real-time solution. As phishing tactics evolve, attackers frequently create new sites with slight modifications that bypass traditional blacklists and rule-based systems. This constant innovation by attackers requires a detection method that can identify phishing attempts based on behavior and patterns rather than static rules.

Machine learning offers a dynamic approach that can analyze URLs and detect phishing by learning from data patterns rather than relying on static lists. A system based on machine learning can continuously adapt to new phishing strategies by retraining on updated datasets. This provides a scalable solution that can identify unknown phishing threats, offering better protection than traditional methods.

Moreover, a machine learning-based system can deliver real-time detection, offering immediate feedback to users. By integrating this technology into a Chrome extension, users can be alerted of potential risks instantly while browsing, reducing the likelihood of phishing success. This approach not only improves security but also educates users by highlighting suspicious elements of phishing sites. Such a system is essential to meet the growing need for proactive, flexible protection against one of the most prevalent cyber threats today.

## Scope of System

The proposed phishing detection system aims to provide comprehensive, real-time protection against phishing attacks through a Chrome extension powered by machine learning. This tool is designed to safeguard both individual users and potentially organizational networks by classifying URLs in real-time as either phishing or legitimate. The extension will analyze various URL characteristics, such as length, presence of suspicious keywords, and SSL status, to accurately identify phishing threats, even those that are new or previously unlisted.

The scope of this system includes both individual user protection and wider application within organizations, where employees might accidentally expose sensitive data to phishing sites. By providing immediate feedback on URL safety, the system minimizes the likelihood of successful phishing attacks. In addition, the Chrome extension format ensures a user-friendly interface, seamlessly integrated into the browsing experience without requiring any special technical knowledge.

Furthermore, the system’s machine learning models can be continuously updated with new data, allowing it to learn and adapt to emerging phishing techniques. This adaptability makes it a long- term solution, capable of evolving alongside phishing tactics. Beyond Chrome, the system has the potential to be expanded to other browsers and can also be integrated into broader cybersecurity infrastructures, such as network security tools or organization-wide email filters.

In summary, the scope of the system encompasses individual, organizational, and adaptable protection against phishing, aiming to create a more secure browsing environment. Its flexible and scalable architecture allows it to provide robust security across diverse user bases and contexts.

* 1. **Operating Environment-Hardware and Software:**

**Server-side requirement**

|  |  |
| --- | --- |
| **Software Requirement** | **Hardware Requirement** |
| Operating System: Windows 7 or above | Processor: Intel Core i3 or above |
| Front End: **python** | RAM: 4GB or above |
| Back End: **python** | HDD: 512 GB or above |
| Web Browser: Chrome, Mozilla Firefox |  |

**Client-side requirement**

|  |  |
| --- | --- |
| **Software Requirement** | **Hardware Requirement** |
| Operating System: Windows 7 or above | Processor: Intel Core i3 or above |
| Web Browser: Chrome, Mozilla Firefox | RAM: 2GB or above |
|  | HDD: 512 GB or above |

* 1. **Brief Description of Technology Used**

The phishing detection system employs a combination of machine learning algorithms and web technologies to provide real-time protection against phishing attacks. Here’s an overview of the primary technologies and their roles in the project:

1. **Python Programming Language:** Python is used for developing the machine learning models due to its simplicity, versatility, and extensive library support for data science and machine learning tasks. Key Python libraries include:
   * Scikit-learn: Provides tools for data preprocessing, model training, and evaluation with algorithms such as Support Vector Machines (SVM), k-Nearest Neighbors (kNN), and Decision Trees.
   * TensorFlow and Keras: Used for implementing neural networks and advanced machine learning models like Artificial Neural Networks (ANNs), which help detect complex patterns in phishing URLs.
   * Pandas and NumPy: Essential for handling large datasets and performing data manipulation and analysis.

##### Machine Learning Algorithms:

* + Support Vector Machines (SVM): A supervised learning model effective for binary classification problems, helping differentiate between phishing and legitimate URLs.
  + k-Nearest Neighbors (kNN): Classifies URLs based on similarity to known phishing examples, making it useful for finding patterns in new data.
  + Decision Trees and Random Forests: These algorithms split data based on attribute decisions, allowing the system to classify URLs based on multiple phishing indicators.
  + XGBoost: An advanced ensemble learning technique, XGBoost refines model predictions through gradient boosting, improving accuracy and resilience to new phishing tactics.
  + Artificial Neural Networks (ANNs): ANNs analyze non-linear relationships, enabling complex pattern detection, making them effective against sophisticated phishing techniques.

1. Chrome Extension: The Chrome extension serves as the user interface, integrating seamlessly into the browser to provide real-time feedback. When users navigate to a URL, the extension communicates with the backend machine learning model to classify the site as safe or phishing. Alerts are displayed to inform the user of potential risks, ensuring an intuitive and accessible browsing experience.
2. Data Sources: The system leverages datasets such as the UCI Phishing Website Dataset and Phish Tank data, which contain labeled phishing and legitimate URLs. This data enables accurate training of the machine learning models by providing features like URL length, IP address usage, and SSL certificate presence.
3. Model Updating and Continuous Learning: The system’s architecture supports ongoing updates to the models, allowing it to adapt as new phishing tactics emerge. This continuous learning capability ensures that the tool remains effective over time, providing users with long-term protection.

This blend of technologies creates a powerful, adaptable, and user-friendly solution for detecting and preventing phishing attacks, meeting the needs of both individual users and organizations.

* 1. **Machine Learning Approaches**

Machine learning provides simplified and efficient methods for data analysis. It has indicated promising outcomes in real-time classification problems recently. The key advantage of machine learning is the ability to create flexible models for specific tasks like phishing detection. Since phishing is a classification problem, Machine learning models can be used as a powerful tool. Machine learning models could adapt to changes quickly to identify patterns of fraudulent transactions that help to develop a learning-based identification system. Most of the machine learning models discussed here are

classified as supervised machine learning, this is where an algorithm tries to learn a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. We present machine learning methods that we used

in our study.

##### Logistic Regression

Logistic Regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, Logistic Regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes. Logistic regression works well when the relationship in the data is almost linear despite if there are complex nonlinear relationships between variables, it has poor performance. Besides, it requires more statistical assumptions before using other techniques.

##### K Near Neighbors

K-Nearest Neighbors (KNN) is one of the simplest algorithms used in machine learning for regression and classification problems which is non-parametric and lazy. In KNN there is no need for an assumption for the underlying data distribution. KNN algorithm uses feature similarity to predict the values of new datapoints which means that the new data point will be assigned a value based on how closely it matches the points in the training set. The similarity between records can be measured in many different ways. Once the neighbors are discovered, the summary prediction can be made by returning the most common outcome or taking the average. As such, KNN can be used for classification or regression problems. There is no model to speak of other than holding the entire training dataset.

##### Support Vector Machine

Support vector machines (SVMs) are one of the most popular classifiers. The idea behind SVM is to get the closest point between two classes by using the maximum distance between classes. This technique is a supervised learning model used for linear and nonlinear classification. Nonlinear classification is performed using a kernel function to map the input to a higher dimensional feature space. Although SVMs are very powerful and are commonly used in classification, it has some weakness. They need high calculations to train data. Also, they are sensitive to noisy data and are

therefore prone to overfitting. The four common kernel functions at the SVM are linear, RBF (radial basis function), sigmoid, and polynomial,

which is listed in Table I. Each kernel function has particular parameters that must be optimized to obtain the best result.

TABLE I

FOUR COMMON KERNELS [12]

Kernel Linear RBF

Sigmoid Polynomial

Type Formula

K (xn, xi) = (xn, xi)

K (xn, xi) = exp (−γ∥xn − xi∥2 + C) K (xn, xi) = tanh (γ (xn, xi) + r)

K (xn, xi) = (γ (xn, xi) + r)d

Parameter C,γ

C,γ

C,γ,r

C,γ,r,d

##### Decision Tree

Decision tree classifiers are used as a well-known classification technique. A decision tree is a flowchart-like tree structure where an internal node represents a feature or attribute, the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition based on the attribute value. It partitions the tree in a recursive manner called recursive partitioning. This particular feature gives the tree classifier a higher resolution to deal with a variety of data sets, whether numerical or categorical data. Also, decision trees are ideal for dealing with nonlinear relationships between attributes and classes. Regularly, an impurity function is determined to assess the quality of the division for each node, and the Gini Variety Index is used as a known criterion for the total performance. In practice, the decision tree is flexible in the sense that it can easily model nonlinear or unconventional relationships. It can interpret the interaction between predictors. It can also be interpreted very well because of its binary structure. However, the decision tree has various drawbacks that tend to overuse data. Besides, updating a decision tree by new samples is difficult.

##### Random Forest

Random Forest, as its name implies, contains a large number of individual decision trees that act as a group to decide the output. Each tree in a random forest specifies the class prediction, and the result will be the most predicted class among the decision of trees. The reason for this amazing result from Random Forest is because of the trees protect each other from individual errors. Although some trees may predict the wrong answer, many other trees will rectify the final prediction, so as a group the trees can move in the right direction. Random Forests achieve a reduction in overfitting by combining many weak learners that underfit because they only utilize a subset of all training samples Random Forests can handle a large number of variables in a data set. Also, during the forest construction process, they make an unbiased estimate of the generalization error. Besides, they can estimate the lost data well. The main drawback of Random Forests is the lack of reproducibility because the process of forest construction is random. Besides, it is difficult to interpret the final model and subsequent results, because it involves many independent decision trees.

##### Ada-Boost

From some aspects, Ada-boost is like Random Forest, the Ada-Boost classification like Random Forest groups weak classification models to form a strong classifier. A single model may poorly categorize objects. But if we combine several classifiers by selecting a set of samples in each iteration and assign enough weight to the final vote, it can be good for the overall classification. Trees are created sequentially as weak learners and correcting incorrectly predicted samples by assigning a larger weight to them after each round of prediction. The model is learning from previous errors. The final prediction is the weighted majority vote (or weighted median in case of regression problems). In short Ada-Boost algorithm is repeated by selecting the training set based on the accuracy of the previous training. The weight of each classifier trained in each iteration depends on the accuracy obtained from previous ones.

##### Gradient Boosting

Gradient Boosting trains many models incrementally and sequentially. The main difference between Ada-Boost and Gradient Boosting Algorithm is how algorithms identify the shortcomings of weak learners like decision trees. While the Ada-Boost model identifies the shortcomings by using high weight data points, Gradient Boosting performs the same methods by using gradients in the loss function. The loss function is a measure indicating how good the model’s coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimize.

##### XGBoost

XGBoost is a refined and customized version of a Gradient Boosting to provide better performance and speed. The most important factor behind the success of XGBoost is its scalability in all scenarios. The XGBoost runs more than ten times faster than popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. The scalability of XGBoost is due to several important algorithmic optimizations. These innovations include a novel tree learning algorithm for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning. Parallel and distributed computing make learning faster which enables quicker model exploration. More importantly, XGBoost exploits out of core computation and enables data scientists to process hundreds of millions of examples on a desktop. Finally, it is even more exciting to combine these techniques to make an end-to-end system that scales to even larger data with the least amount of cluster resources.

##### Artificial Neural Networks

Artificial neural networks (ANNS) are a learning model roughly inspired by biological neural networks. These models are multilayered, each layer containing several processing units called neurons. Each neuron receives its input from its adjacent layers and computes its output with the help of its weight and a non-linear function called the activation function. In feed-forward neural networks like in 2, data flows from the first layer to the last layer. Different layers may perform different transformations on their input. The weights of neurons are set randomly at the start of the training and they are gradually adjusted by the help of the gradient descent method to get close to the optimal solution. The power of neural networks is due to the non-linearity of hidden nodes. As a result, introducing non-linearity in the network is very important so that you can learn complex functions.

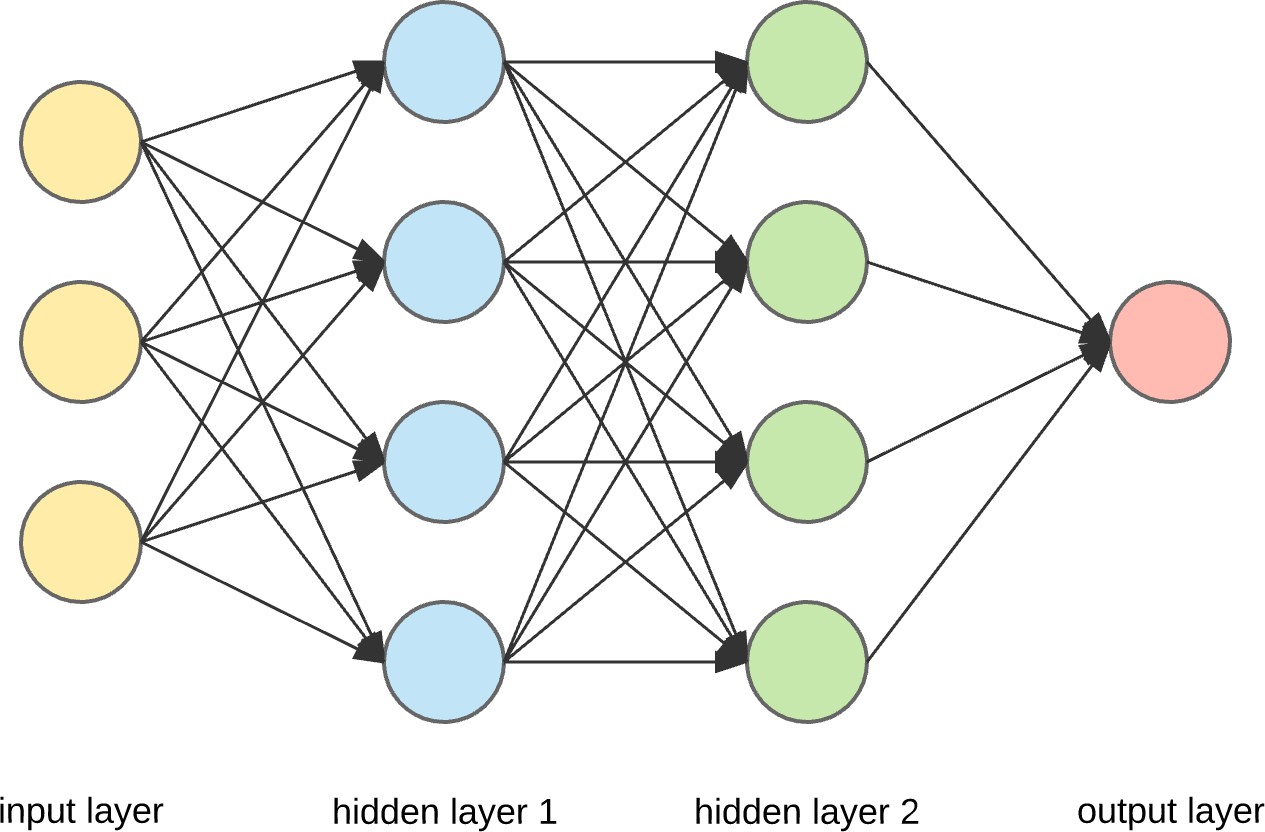


Fig. Artificial Neural Network

# Chapter 2: PROPOSED SYSTEM

## Feasibility Study

##### Technical Feasibility

The phishing detection system is technically feasible, as it leverages widely-used machine learning algorithms and robust development tools.

* + Machine Learning Models: Algorithms such as Support Vector Machines (SVM), Decision Trees, Random Forest, k-Nearest Neighbors (kNN), and Artificial Neural Networks (ANNs) are effective for phishing classification. These models have been validated in other cybersecurity applications, ensuring reliability in phishing detection.
  + Python Libraries and Tools: Python libraries like scikit-learn, TensorFlow, and Keras provide essential functions for machine learning, model training, and evaluation. These libraries are open-source and optimized for performance, minimizing development time and costs.
  + Dataset Availability: Pre-existing datasets, including the UCI Phishing Website Dataset and Phish Tank, contain labeled phishing and benign URLs with features such as URL length, IP usage, SSL certificates, and domain age. These datasets support the initial training and validation of machine learning models without the need for new data collection.
  + Chrome Extension API: The Chrome API enables the development of browser extensions that interact with user browsing in real-time. This allows the phishing detection model to classify URLs on-the-fly and alert users without interrupting the browsing experience.
  + Scalable Infrastructure: Cloud platforms like AWS, Google Cloud, or Azure offer scalable resources for model deployment, allowing the system to handle varying levels of demand without overloading. Cloud-based machine learning environments facilitate updates and maintenance, enhancing system adaptability to evolving phishing tactics.

##### Economic Feasibility

The project is economically feasible, as it balances reasonable development costs with strong potential benefits.

* + Development Costs: Costs include salaries for a team skilled in machine learning, web development, and cybersecurity. Using open-source software (e.g., TensorFlow, scikit-learn) significantly reduces software costs. Additionally, pre-existing datasets eliminate data collection expenses.
  + Infrastructure Costs: Ongoing costs will involve cloud hosting, data storage, and potential fees for serverless functions or virtual machines. Cloud providers offer flexible pricing models, allowing costs to scale with demand, which keeps operating expenses manageable over time.
  + -Revenue Potential: The project can generate revenue through subscription models for advanced or enterprise versions, licensing to organizations, or partnerships with cybersecurity firms. These revenue streams are attractive to companies aiming to reduce phishing risks and protect sensitive information.
  + Cost Savings: For businesses, this system helps prevent phishing-related incidents, reducing potential costs from data breaches, legal fines, and productivity losses. For individual users, it enhances online security at no additional cost, increasing its marketability.

##### Operational Feasibility

The operational design and user experience make this system feasible to deploy and manage effectively.

* + Ease of Use: The system is designed as a Chrome extension, which integrates seamlessly with the browsing experience and requires minimal user input. Users only need to install the extension; phishing detection runs automatically in the background.
  + Maintenance and Updates: Machine learning models can be periodically retrained with updated phishing data to ensure detection accuracy. The use of cloud infrastructure facilitates these updates without manual intervention from users, allowing the system to adapt to new phishing tactics.
  + Scalability: Cloud services support scaling as the user base grows, ensuring the system remains responsive and effective under higher usage. This scalability is especially beneficial for enterprise deployments, where many users may access the system concurrently.
  + Target User Base: The system serves both individual users, who can use the Chrome extension for personal browsing, and organizations, who can deploy it to protect employees’ browsing activities. This dual usability broadens the impact and operational feasibility of the system.
  1. **Objectives of Proposed System**

The objectives of the proposed phishing detection system are centered on enhancing online security, providing real-time protection, and ensuring ease of use. Here are the primary objectives:

##### Real-Time Phishing Detection

The system aims to detect phishing URLs in real-time as users navigate the web. By analyzing URLs dynamically, it prevents users from accessing phishing sites before any harm can occur, reducing the chances of data breaches and credential theft.

##### High Detection Accuracy Using Machine Learning

To maximize effectiveness, the system will use machine learning models trained on extensive phishing datasets. Algorithms like Support Vector Machines (SVM), Random Forests, and Neural Networks will help classify URLs with high precision, minimizing false positives (legitimate sites flagged as phishing) and false negatives (phishing sites undetected).

##### User-Friendly Chrome Extension

The system will be implemented as a Chrome extension, providing an intuitive, easy-to-install tool that integrates seamlessly into the browsing experience. Users will be alerted about potentially dangerous URLs without disrupting their browsing flow, making it accessible to both technical and non-technical users.

##### Continuous Learning and Adaptability

The system is designed to stay effective over time by adapting to new phishing tactics. By retraining models with updated datasets, it will continuously improve detection capabilities, adapting to the evolving nature of phishing attacks and maintaining a high level of security.

##### Data Privacy and Compliance

The system will be built with data privacy as a priority. It will not collect personal information, and only processes URL features needed for classification, ensuring compliance with data protection regulations like GDPR and CCPA. User transparency will be maintained to enhance trust and adoption.

##### Scalability for Individual and Organizational Use

While initially targeted at individual users, the system will be designed to scale for organizational use, allowing businesses to deploy the tool across employee networks. This will help organizations reduce phishing risks by protecting employees’ web activities and enhancing overall security.

##### Educational Alerts to Raise User Awareness

The system will include alerts that inform users why a URL is flagged as phishing, helping them understand common phishing indicators. This feature aims to educate users on phishing risks, empowering them to recognize threats and make safer browsing choices in the future.

* 1. **Users of the system**

The phishing detection system is designed to serve a broad range of users, from individuals to organizations, each benefiting uniquely from the system’s features. Here are the primary user groups:

##### Individual Internet Users

General internet users, including those with limited technical knowledge, can install the Chrome extension to protect themselves from phishing attacks. The system provides real-time alerts when a suspicious URL is detected, helping individuals safely navigate the internet and avoid scams targeting personal information and credentials.

##### Corporate Employees

Organizations can deploy this system across their networks to protect employees from phishing attacks during work. By installing the extension on company devices, businesses can reduce the risk of data breaches and unauthorized access, safeguarding sensitive company information and customer data.

##### Small and Medium-Sized Enterprises (SMEs)

SMEs often lack dedicated cybersecurity resources and are more vulnerable to phishing attacks. This system offers an affordable, accessible solution for SMEs to secure their online activities, helping prevent losses from phishing-related scams without significant investment in specialized cybersecurity infrastructure.

1. **IT and Security Teams in Large** Organizations

In larger organizations, IT and security teams can use the system as an additional layer of cybersecurity, integrating it with existing security protocols. The system’s continuous learning capability enables it to adapt to new phishing trends, offering proactive protection and supporting cybersecurity teams in maintaining organizational security.

##### Educational Institutions

Schools, universities, and other educational institutions can implement this system to protect students and staff from phishing threats. By deploying it across their networks, institutions help reduce cybersecurity risks and provide students with a safer online environment, particularly in digital learning platforms.

##### Financial and E-commerce Businesses

Financial institutions and e-commerce businesses, which are frequent targets of phishing attacks, can use this system to protect both employees and customers. The system reduces the risk of phishing scams aimed at stealing financial data, helping protect the organization’s reputation and customer trust.

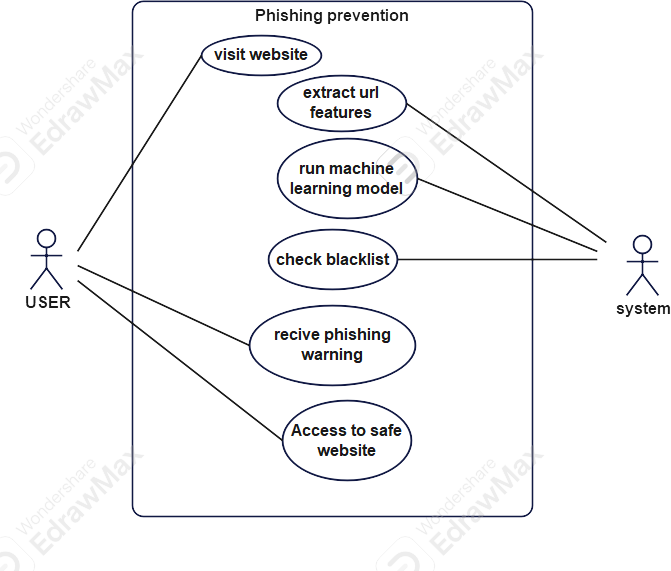
##### Cybersecurity Researchers and Analysts

Cybersecurity researchers can leverage the system’s machine learning models and phishing data to study phishing techniques and test detection algorithms. This use of the system may lead to improvements in phishing detection methods and inform the development of future cybersecurity solutions.

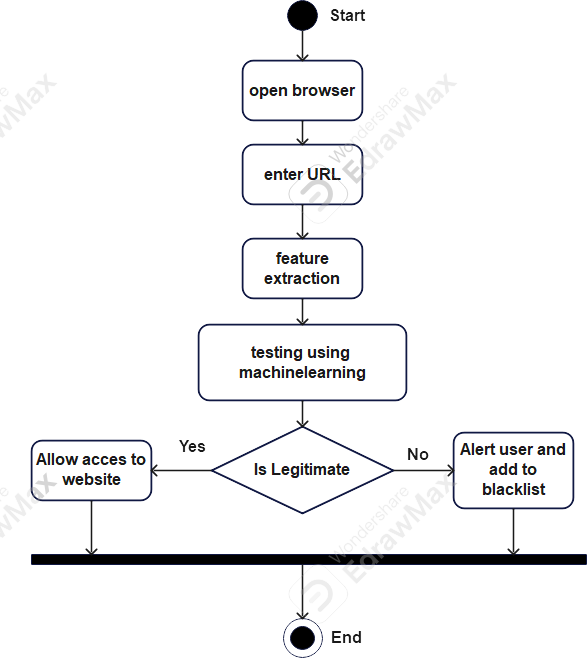
By serving these diverse user groups, the system helps create a safer online environment for individuals and organizations alike, reducing the prevalence and impact of phishing attacks across multiple sectors.

# Chapter 3: ANALYSIS AND DESIGN

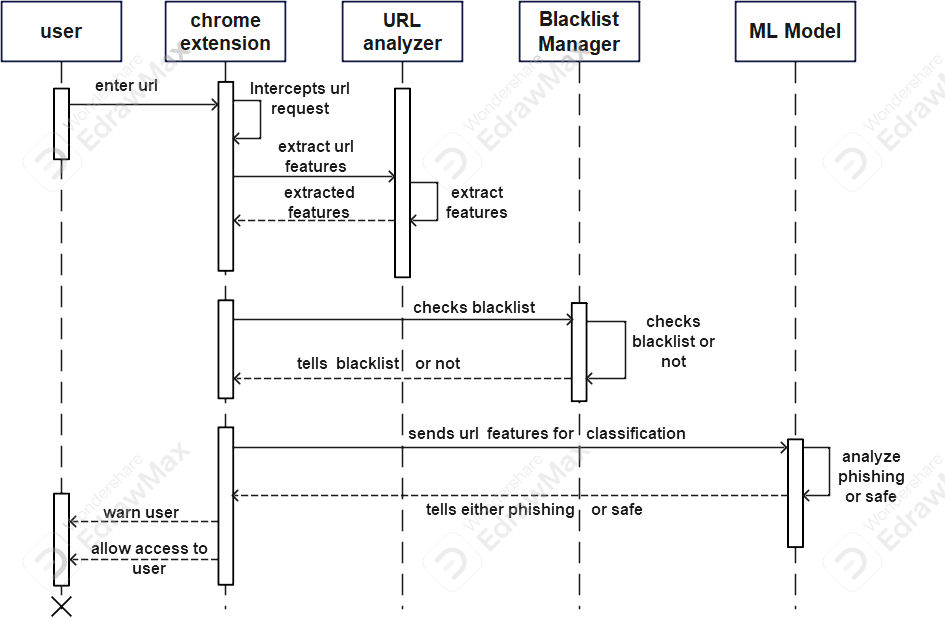
* 1. **Use Case Diagram**



* 1. **Activity Diagram**



* 1. **Sequence Diagram**



* 1. **Dataset Description**

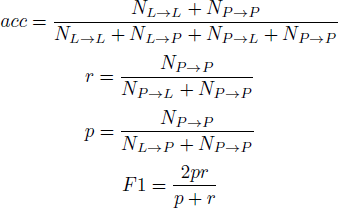
One of the main challenges in our research was the scarcity of phishing dataset. Although many scientific papers about phishing detection have been published, they have not provided the dataset on which they used in their research. Moreover, another factor that hinders finding a desirable dataset is the lack of a standard feature set to record characteristics of a phishing website. The dataset that we used in our research was well researched and benchmarked by some researchers. Fortunately, the accompanying wiki of the dataset comes with a data description document which discusses the data generation strategies taken by the authors of the dataset. For updating our dataset with new phishing websites, we have also implemented a code that extracts features of new phishing websites that are provided by the PhishTank website. The dataset contains about 11,000 sample websites, we used 10% of samples in the testing phase. Each website is marked either legitimate or phishing. The features of our dataset are as follows:

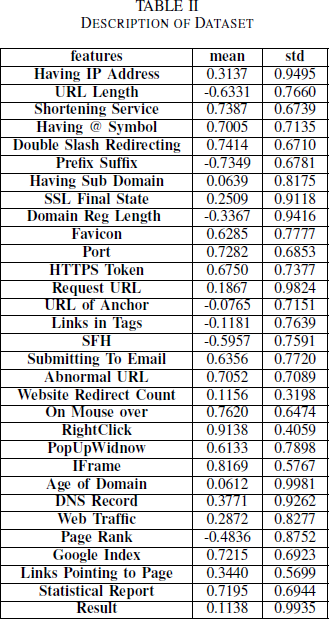
1. Having IP Address: If an IP address is used instead of the domain name in the URL, such as “[http://217.102.24.235/sample.html”.](http://217.102.24.235/sample.html)
2. URL Length: Phishers can use a long URL to hide the doubtful part in the address bar.
3. Shortening Service: Links to the webpage that has a long URL. For example, the URL “[http://sharif.hud.ac.uk/”](http://sharif.hud.ac.uk/) can be shortened to “bit.ly/1sSEGTB”.
4. Having @ Symbol: Using the “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol
5. Double Slash Redirection: The existence of “//” within the URL which means that the user will be redirected to another website
6. Prefix Suffix: Phishers tend to add prefixes or suffixes separated by (-) to the domain name so that users feel that they are dealing with a legitimate webpage. For example, http://www.Confirme- paypal.com.
7. Having Sub Domain: Having subdomain in URL.
8. SSL State: Shows that website use SSL
9. Domain Registration Length: Based on the fact that a phishing website lives for a short period
10. Favicon: A favicon is a graphic image (icon) associated with a specific webpage. If the favicon is loaded from a domain other than that shown in the address bar, then the webpage is likely to be considered a Phishing attempt.
11. Using Non-Standard Port: To control intrusions, it is much better to merely open ports that you need. Several firewalls, Proxy and Network Address Translation (NAT) servers will, by default, block all or most of the ports and only open the ones selected
12. HTTPS token: Having deceiving “https” token in URL. For example, “[http://https-www-mellat-phish.ir”](http://https-www-mellat-phish.ir/)
13. Request URL: Request URL examines whether the external objects contained within a webpage such as images, videos, and sounds are loaded from another domain.
14. URL of Anchor: An anchor is an element defined by the < a > tag. This feature is treated exactly as “Request URL”.
15. Links In Tags: It is common for legitimate websites to use! Meta. tags to offer metadata about the HTML document! Script. tags to create a client-side script; and

! Link. tags to retrieve other web resources.

1. Server Form Handler: If the domain name in SFHs is different from the domain name of the webpage.
2. Submitting Information to E-mail: A phisher might redirect the user’s information to his email.
3. Abnormal URL: It is extracted from the WHOIS database. For a legitimate website, identity is typically part of its URL.
4. Website Redirect Count: If the redirection is more than four-time
5. Status Bar Customization: Use JavaScript to show a fake URL in the status bar to users
6. Disabling Right Click: It is treated exactly as “Using onMouseOver to hide the Link”
7. Using Pop-up Window: Showing having pop-up windows on the webpage.
8. IFrame: IFrame is an HTML tag used to display an additional webpage into one that is currently shown.
9. Age of Domain: If the age of the domain is less than a month.
10. DNS Record: Having the DNS record
11. Web Traffic: This feature measures the popularity of the website by determining the number of visitors.
12. Page Rank: Page rank is a value ranging from “0” to “1”. PageRank aims to measure how important a webpage is on the Internet.
13. Google Index: This feature examines whether a website is in Google’s index or not.
14. Links Pointing to Page: The number of links pointing to the web page.
15. Statistical Report: If the IP belongs to top phishing IPs or not.
    1. **Evaluation Metrics**

For evaluating phishing classification performance, we use accuracy(acc) recall(r), precision(p), F1 score, test time, and train time of classifiers. Recall measures the percentage of phishing websites that the model manages to detect (model’s effectiveness). Precision measures the degree to which the phishing detected websites are indeed phishing (model’s safety). F1 score is the weighted harmonic mean of precision and recall. Let NL→L be the number of legitimate websites classified as legitimate, NL→P be the number of legitimate websites misclassified as phishing, NP→L be the number of phishing misclassified as legitimate and NP→P be the number of phishing websites classified as phishing. Thus, the following equations hold





* 1. **Experimental Results**

In our experiments, we used 10-fold cross-validation for model performance evaluation. we divided the data set into 10 sub-samples. A sub-sample is used for testing data and the rest is used for training models. Since phishing detection is a classification problem we must use a binary classification model, we consider “-1“as a phishing sample and “1“as a legitimate one.

In our study, we used various machine learning models for detection phishing websites which are Logistic regression, Ada booster, random forest, KNN, neural networks, SVM, Gradient boosting, XGBoost. We evaluate the accuracy, precision, recall, F1 score, training time, and testing time of these models and we used different methods of feature selection and hyperparameters tuning for getting the best results. Table

II shows the comparison between accuracy, precision, recall, and F1 score of these models.

For finding the best performance from support vector machine we have tested four kinds of kernel:

* Linear kernel
* Polynomial kernel
* Sigmoid kernel
* RBF kernel

In our experience Linear, Polynomial, and RBF kernels would work equally well on this dataset but we get the best performance from the RBF kernel. The choice of the kernel and regularization parameters can be optimized with a cross-validation model selection. With more than a few hyperparameters to tune, automated model selection is likely to result in severe over-fitting, due to the variance of the model

selection criterion. In the absence of expert knowledge, the RBF kernel makes a good default kernel when our problem requiring a non-linear classifier. In Figure 4 performance of SVM with the different kernel are presented.

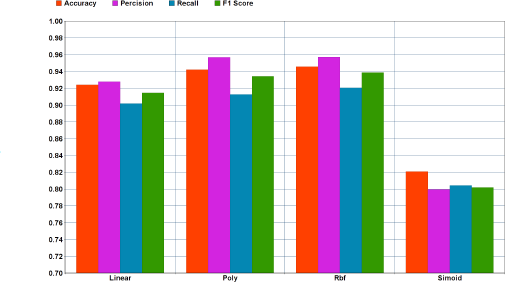


Fig. 4. Performance of SVM classfier with various kernels

We found that Random Forest is highly accurate, relatively robust against noise and outliers, it is fast, simple to implement and understand, and can-do feature selection implicitly. Being unaffected by noise is the main advantage of Random Forest over AdaBoost. According to Central Limit Theorem, Random Forest reduces variance by increasing the number of trees. However, the main disadvantage of Random Forests that we faced in implementing our model was the high number of hyperparameters to tune for getting the best performance. Moreover, Random Forest introduces randomness into the training and testing data which is not suitable for all data sets.

In KNN classification we found out the best performance is acquired when we set k to 5. In KNN classification there is no optimal number to set k that is suitable for all kinds of datasets. According to the KNN result which is shown in Figure 5 the noise will have a higher impact on the result when the number

of neighbors is small, moreover, a large number of neighbors make it computationally expensive to acquire the result. Our result has also shown that a small number of neighbors is the most flexible fit which will have low bias but the high variance plus a large number of neighbors will have a smoother decision boundary which means lower variance but higher bias.

The main advantage of XGBoost is its fast speed compared to other algorithms, such as ANN and SVM, and its regularization parameter that successfully reduces variance. But even aside from the regularization parameter, this algorithm leverages a learning rate and subsamples from the features like random forests, which increases its ability to generalize even further. However, XGBoost is more difficult to understand, visualize, and to tune compared to AdaBoost and Random Forests. There is a multitude of hyperparameters that can be tuned to increase performance. XGBoost is a particularly interesting algorithm when speed as well as high accuracies are of the essence. Nevertheless, more resources in training the model are required because the model tuning needs more time and expertise from the user to achieve meaningful outcomes.

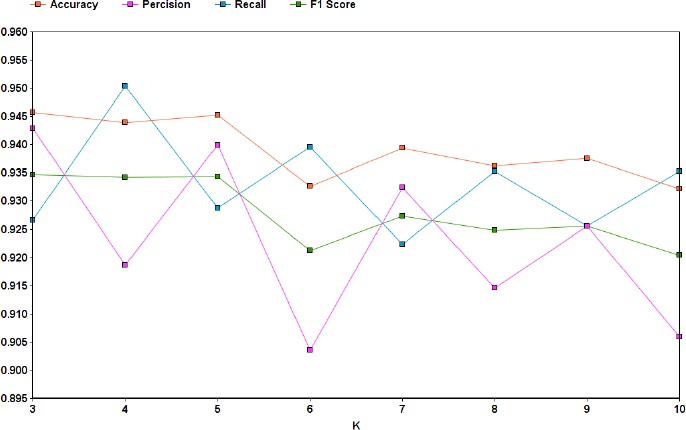


Fig. 5. KNN with different K

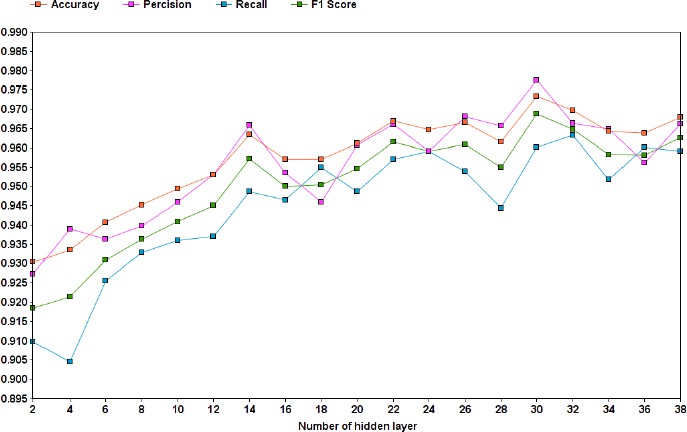
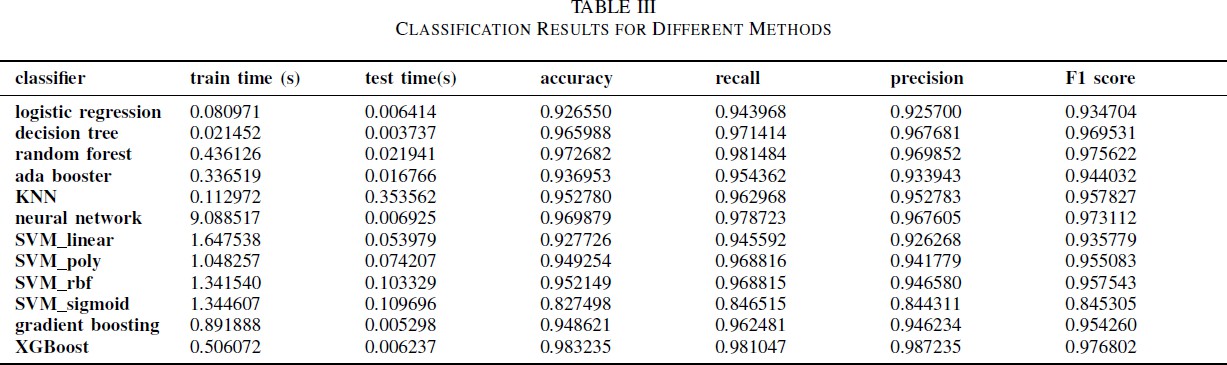


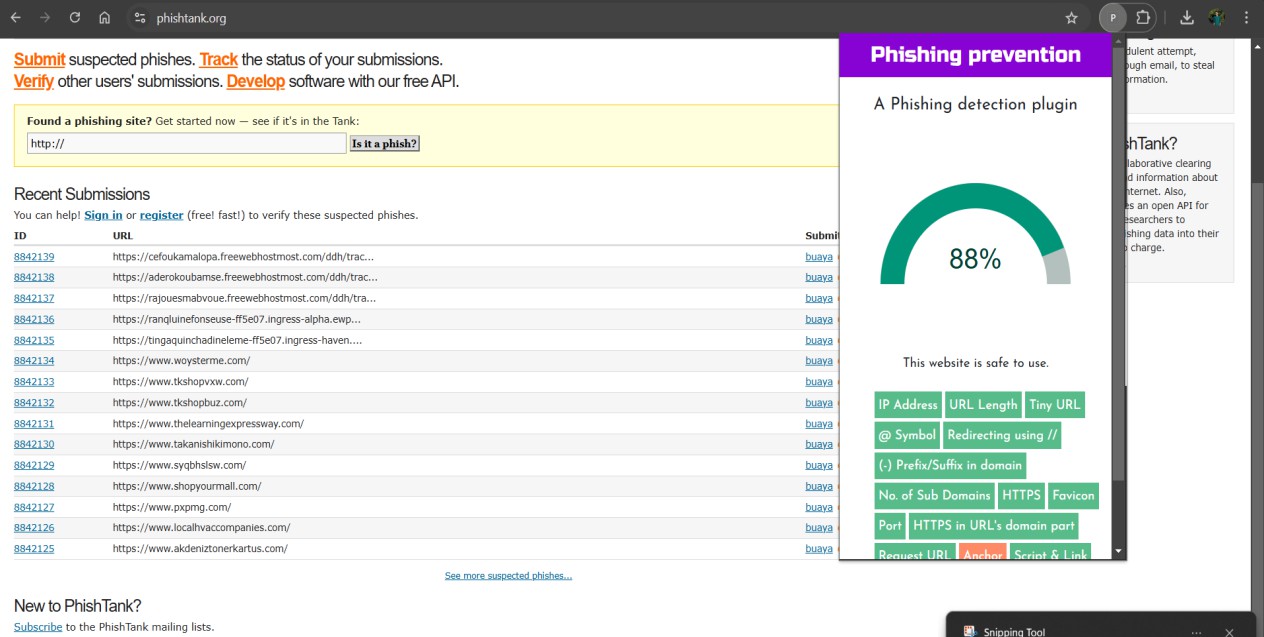
Fig. 6. Neural Network with different depth

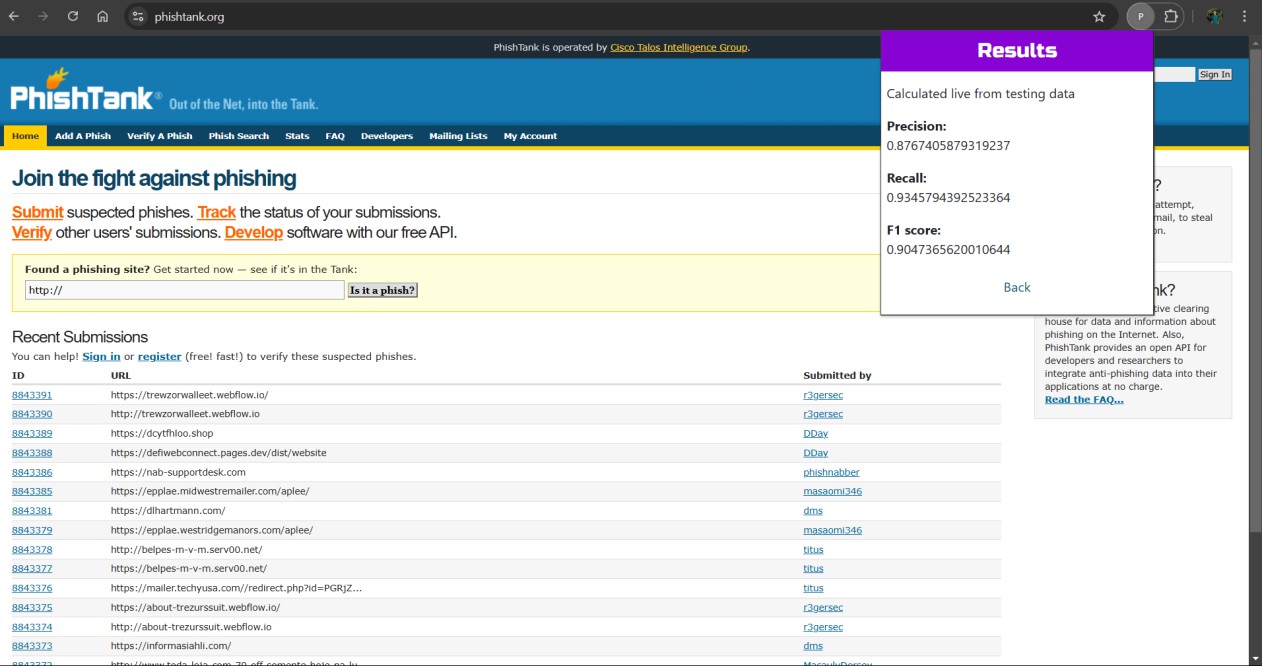
As expected, neural network’s training time was considerably higher compared to other machine learning models. XGBoost’s F1 score was slightly better compared with neural network’s. This is due to the fact that our training data size is small. Unlike XGBoost, neural network model is also unable to explain why it have predicted a website as a phishing one. The explainability will help us to specify key features more easily. In the implementation of neural networks, we use Adam optimizer and relu activation function in the hidden layer, figure 6 shows the performance of the neural network with a different number of the hidden layer, we get the best performance with 30 hidden layers. We trained our model on

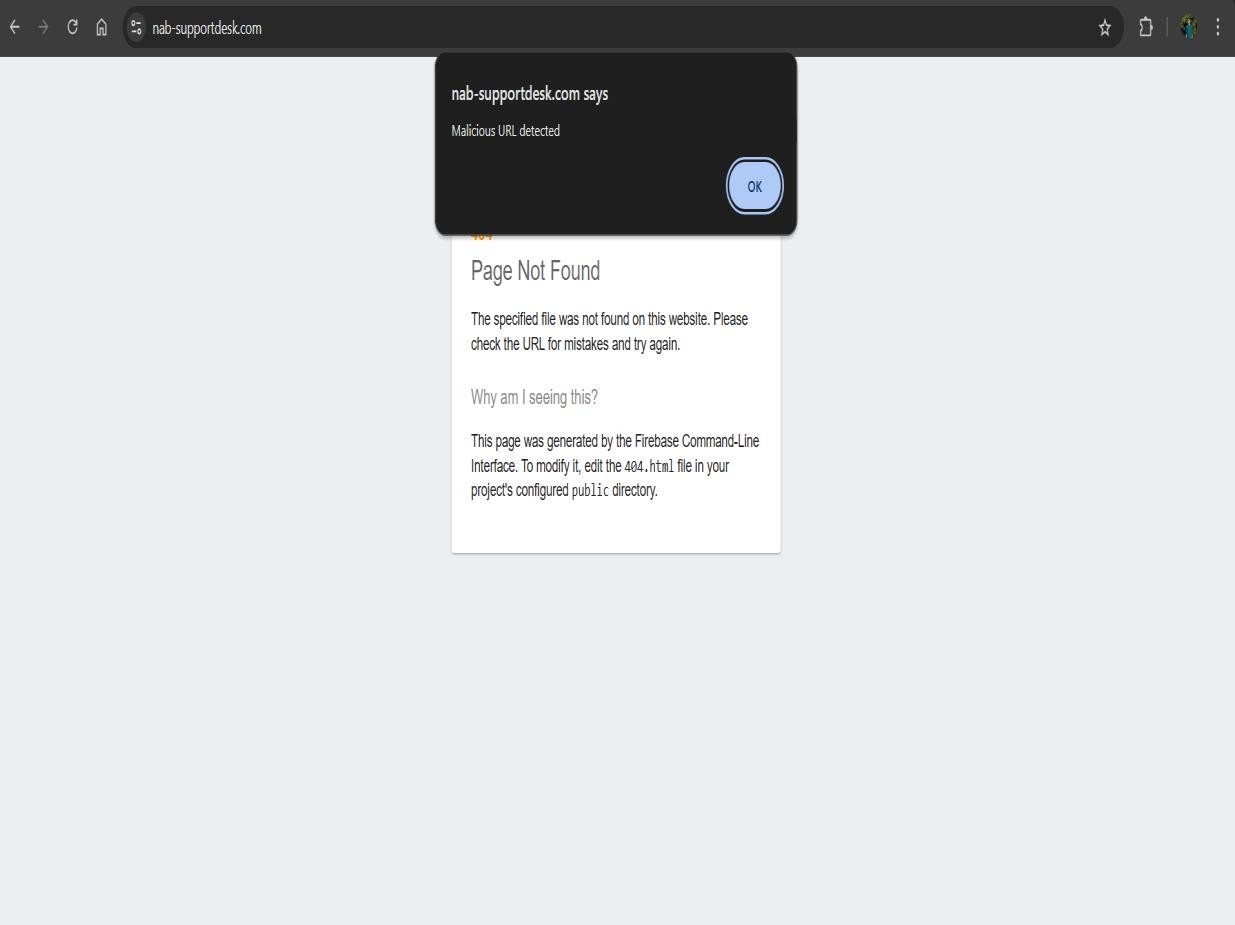
500 epochs with early stopping.

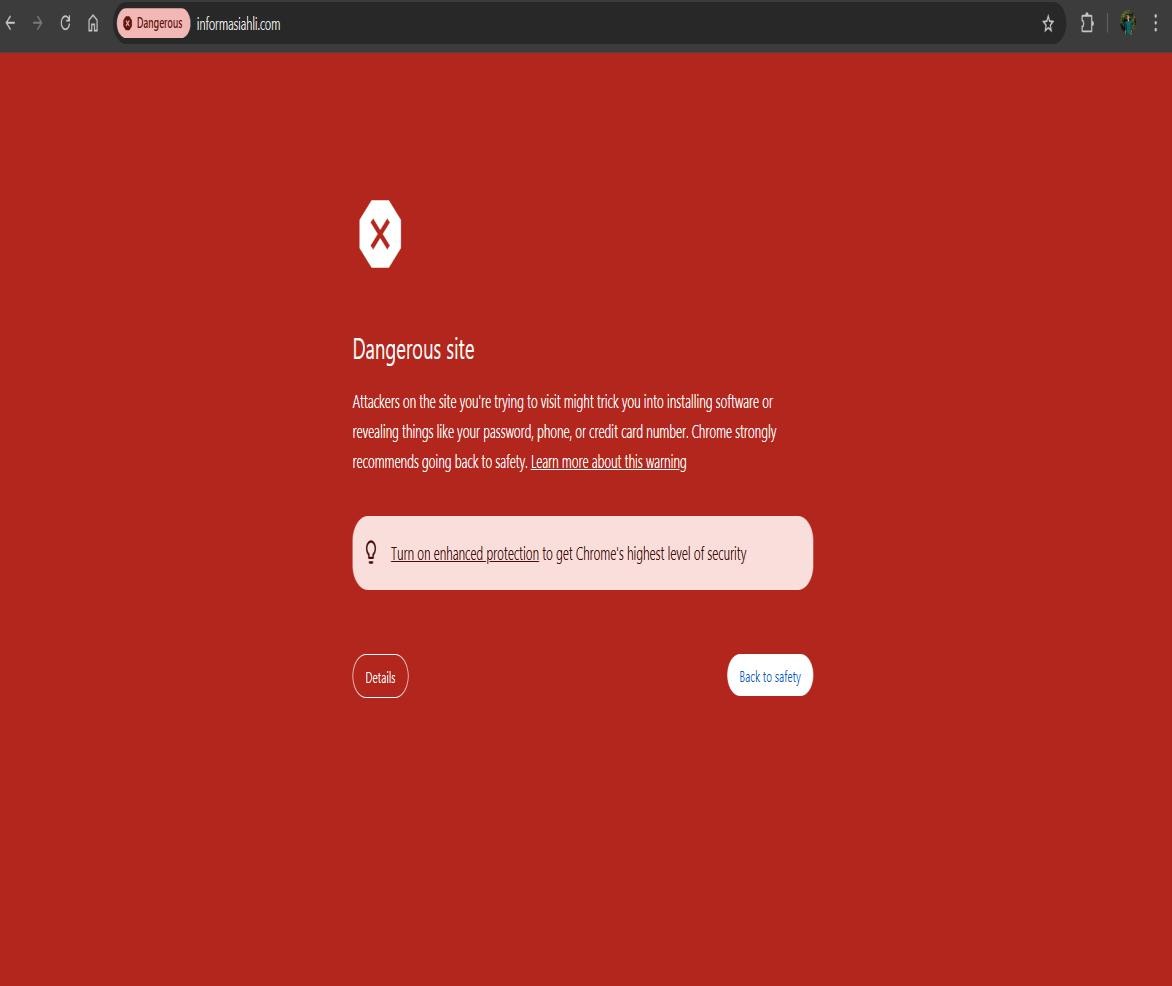


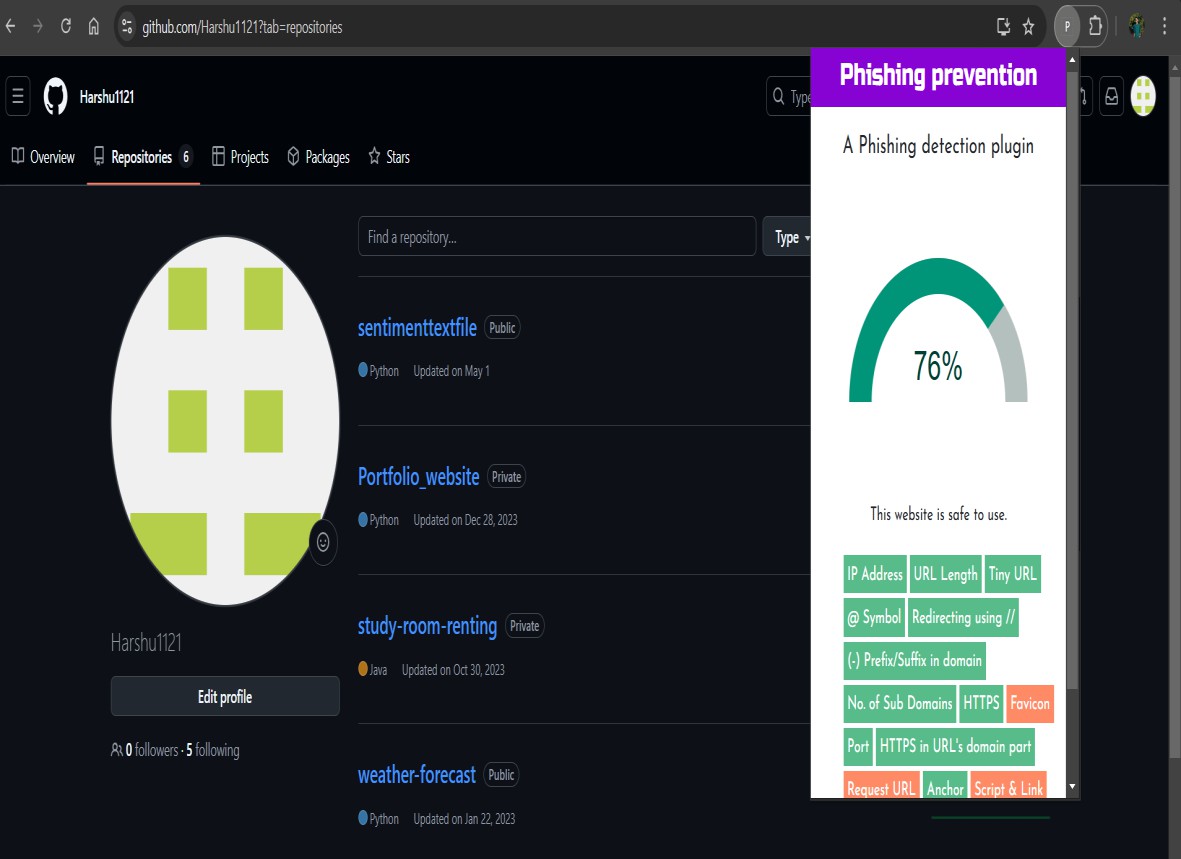
* 1. **Sample Input and Output Screens**

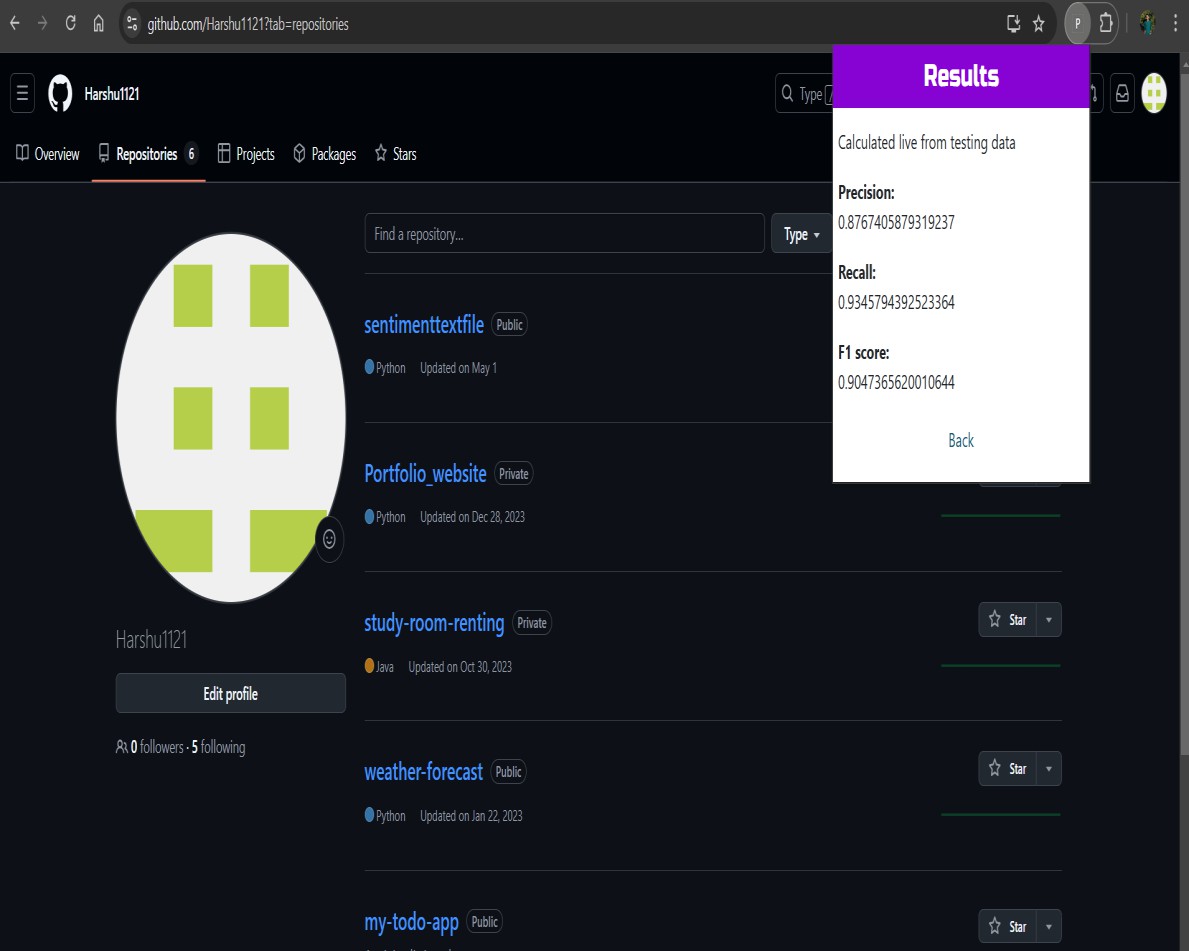


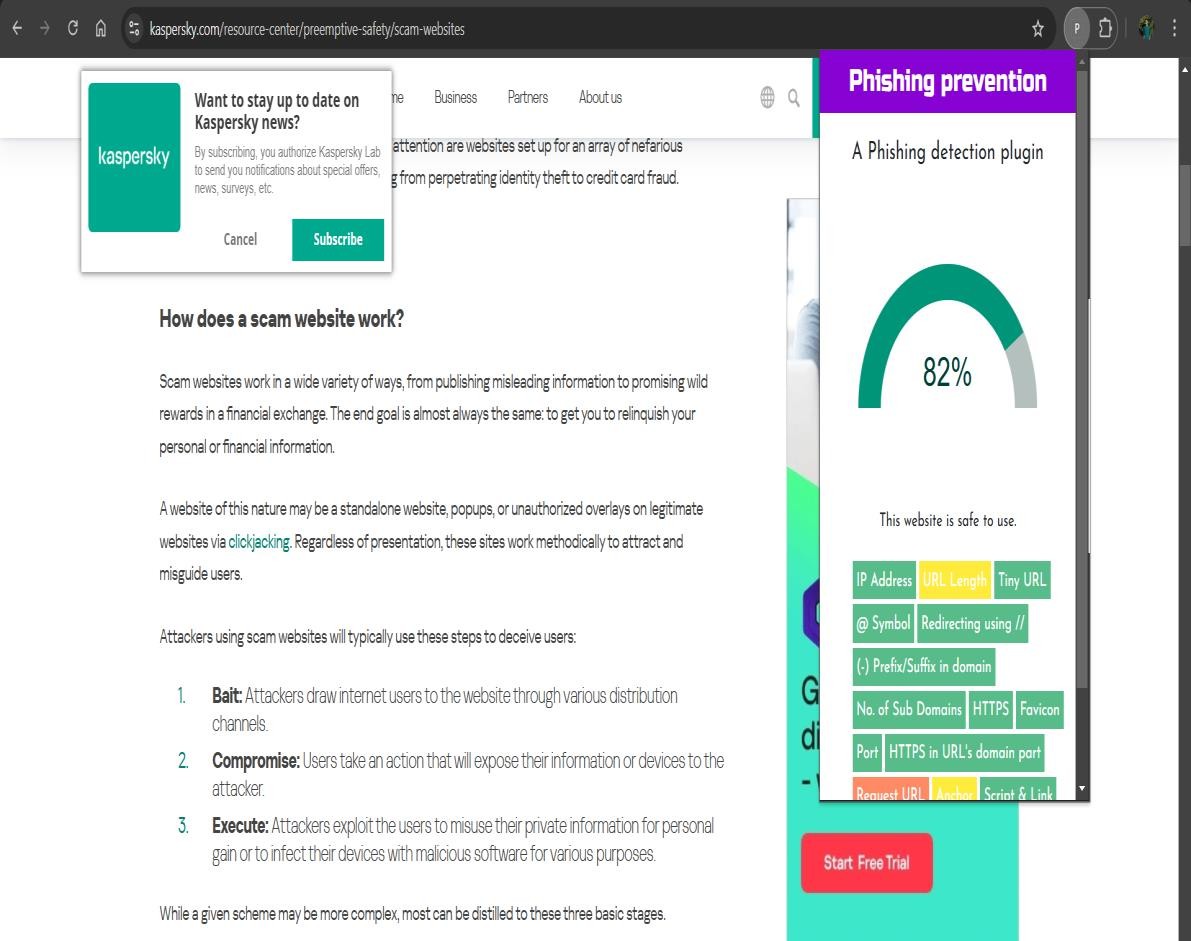


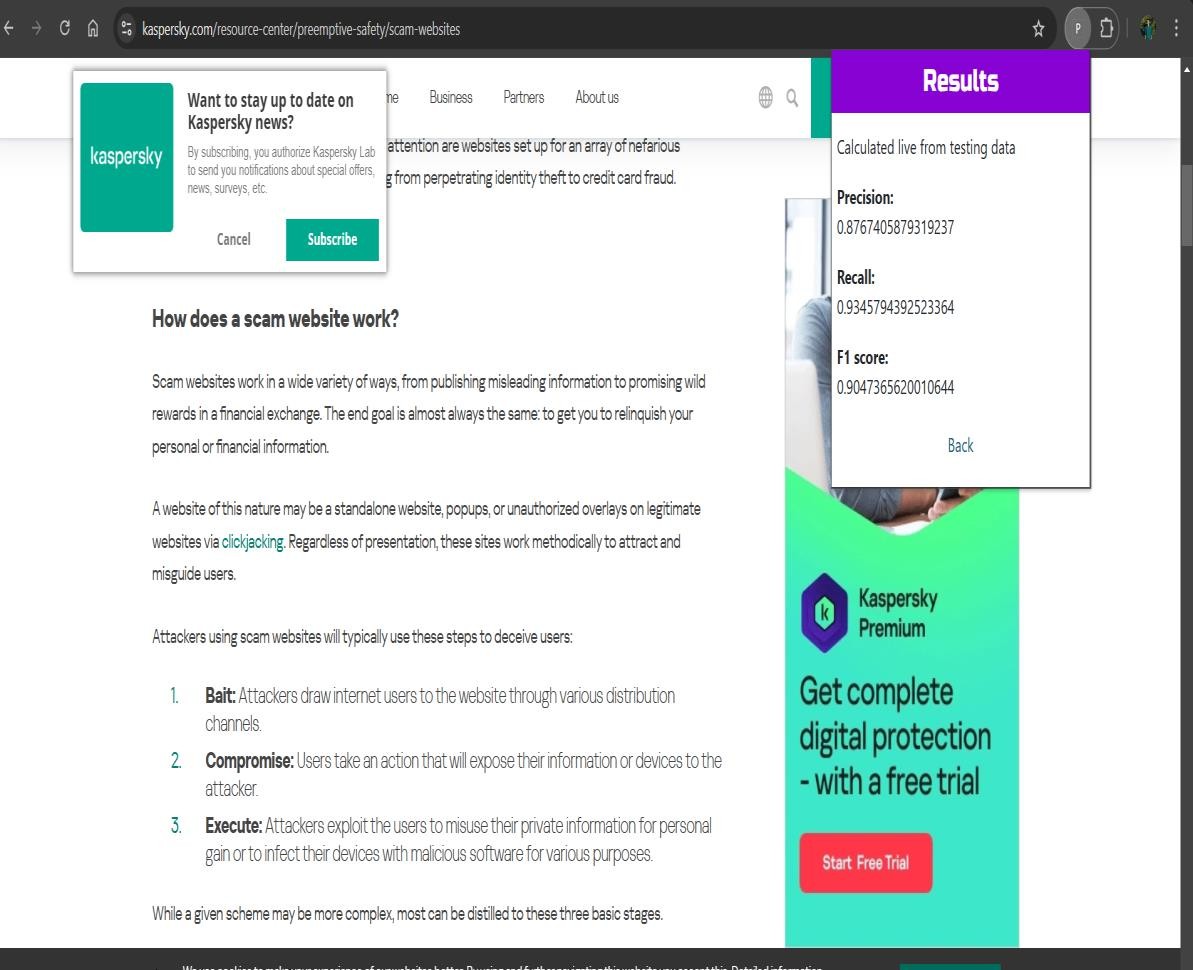












### CHAPTER 4: CODING SAMPLE CODE

##### training.py

from sklearn.ensemble import RandomForestClassifier from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import cross\_val\_score from sklearn.metrics import accuracy\_score

import numpy as *np*

import json import dump

*X\_train* = np.load('../dataset/X\_train.npy')

*y\_train* = np.load('../dataset/y\_train.npy')

print('X\_train:{0}, y\_train:{1}'.format(*X\_train*.shape, *y\_train*.shape))

*clf* = RandomForestClassifier()

print('Cross Validation Score: {0}'.format(np.mean(cross\_val\_score(*clf*, *X\_train*, *y\_train*, cv=10))))

*clf*.fit(*X\_train*, *y\_train*)

*X\_test* = np.load('../dataset/X\_test.npy')

*y\_test* = np.load('../dataset/y\_test.npy')

*pred* = *clf*.predict(*X\_test*)

print('Accuracy: {}'.format(accuracy\_score(*y\_test*, *pred*)))

#print(forest\_to\_json(clf))

json.dump(dump.forest\_to\_json(*clf*), open('../../static/classifier.json', 'w'))

##### dump.py

from sklearn.tree import \_tree def tree\_to\_json(*tree*):

*tree\_* = *tree*.tree\_ *feature\_names* = range(30) *feature\_name* = [

*feature\_names*[*i*] if *i* != \_tree.TREE\_UNDEFINED else "undefined!" for *i* in *tree\_*.feature

]

def recurse(*node*):

*tree\_json* = dict()

if *tree\_*.feature[*node*] != \_tree.TREE\_UNDEFINED:

*tree\_json*['type'] = 'split'

*threshold* = *tree\_*.threshold[*node*]

*tree\_json*['threshold'] = "{} <= {}".format(*feature\_name*[*node*], *threshold*) *tree\_json*['left'] = recurse(*tree\_*.children\_left[*node*])

*tree\_json*['right'] = recurse(*tree\_*.children\_right[*node*]) else:

*tree\_json*['type'] = 'leaf'

*tree\_json*['value'] = *tree\_*.value[*node*].tolist() return *tree\_json*

return recurse(0)

def forest\_to\_json(*forest*):

*forest\_json* = dict()

*forest\_json*['n\_features'] = *forest*.n\_features\_ *forest\_json*['n\_classes'] = *forest*.n\_classes\_ *forest\_json*['classes'] = *forest*.classes\_.tolist() *forest\_json*['n\_outputs'] = *forest*.n\_outputs\_ *forest\_json*['n\_estimators'] = *forest*.n\_estimators

*forest\_json*['estimators'] = [tree\_to\_json(*estimator*) for *estimator* in *forest*.estimators\_] return *forest\_json*

### CHAPTER 5: TESTING

* 1. **Testing Strategy**

To ensure the quality and effectiveness of the Phishing Detection System, a comprehensive testing strategy was designed to evaluate individual components and their integration. The testing process was structured as follows:

* + 1. Requirement Analysis
       - Reviewed the system's functional and non-functional requirements.
       - Defined objectives for detecting phishing URLs, model accuracy, and alert mechanisms.
    2. Test Planning
       - Developed test cases for feature extraction, backend model integration, and user interface.
       - Included scenarios for functional, performance, and security testing.
    3. Test Execution
       - Tested feature extraction using various URL structures.
       - Evaluated model predictions for phishing and legitimate URLs.
       - Verified that the extension displays appropriate alerts.
    4. Usability Testing
       - Assessed the extension’s ease of use, feedback clarity, and non-intrusiveness.
    5. Performance and Compatibility Testing
       - Tested API latency under high loads.
       - Ensured compatibility across different operating systems and Chrome versions.
    6. Regression Testing
       - Validated previously successful test cases after code updates to ensure no regressions.
  1. **Unit Testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| UT- 01 | Validate IP address detection | 1. Visit the URL [http://192.168.0.1.](http://192.168.0.1/) 2. Extract URL features using the content script. | [http://192.168.0.1](http://192.168.0.1/) | Feature "IP Address" = "1" | Feature "IP  Address"  = "1" | Pass |
| UT- 02 | Validate absence of IP address | 1. Visit the URL https://example.co m. 2. Extract URL features using the content script. | https://example.com | Feature "IP Address" = "-1" | Feature "IP  Address"  = "-1" | Pass |
| UT- 03 | Validate HTTPS  detection | 1. Visit the URL https://secure- site.com. 2. Extract URL features using the content script. | https://secure-site.com | Feature "HTTPS"  = "-1" | Feature "HTTPS " = "-1" | Pass |
| UT- 04 | Detect special character "@" in URL | 1. Visit the URL [http://example@ph](http://example@ph/) ishing.com. 2. Extract URL features using the content script. | [http://example@phishin](http://example@phishin/) g.com | Feature "@ Symbol" = "1" | Feature "@ Symbol"  = "1" | Pass |
| UT- 05 | Validate backend classification | 1. Send extracted features of phishingsite.com to the backend API. | Features from phishingsite.com | Prediction  =  "phishing" | Predictio n = "phishin g" | Pass |

* 1. **Integration Testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| IT-01 | Test data flow from frontend to backend | 1. Open a website. 2. Use content.js to extract features. 3. Send features to the backend API. | Features of [http://phishingsite.co](http://phishingsite.co/) m | Backend receives features correctly | Backend receives features correctly | Pass |
| IT-02 | Test backend response | 1. Send features to the backend API. 2. Wait for classification result. | Features from any URL | Backend returns prediction (e.g., "phishing") | Backend returns "phishin g" | Pass |
| IT-03 | Test frontend alert | 1. Open [http://phishingsite](http://phishingsite/)   .com.   1. Wait for classification result. 2. Check for alert. | [http://phishingsite.co](http://phishingsite.co/) m | Alert displayed: "Warning: This website may be phishing!" | Alert displaye d: "Warnin g: This website may be phishing! " | Pass |
| IT-04 | Handle invalid backend response | 1. Simulate backend unavailability. 2. Open a URL in Chrome. 3. Observe extension behaviour. | Backend unavailable | Error displayed: "Unable to classify URL" | Error displaye d: "Unable to classify URL" | Pass |

* 1. **Performance Testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| PT-01 | Backend API response time | 1. Send feature data from https://google.com to the backend API. | Features from https://google.com | Response time < 1 second | Response time = 0.8 seconds | Pass |
| PT-02 | Handle 100 simultaneous requests | 1. Send 100 requests to the backend API concurrently. | 100 feature JSONs | No crashes; all responses returned | No crashes; all responses returned | Pass |
| PT-03 | Test extension memory usage | 1. Open 10 tabs with the Chrome extension active. | URLs from multiple tabs | Memory usage < 100MB | Memory usage = 92MB | Pass |

* 1. **User Acceptance Testing**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| UAT- 01 | Test legitimate URL detection | 1. Visit https://google.com 2. Observe extension behaviour. | https://google.com | No alert displayed | No alert displaye d | Pass |
| UAT- 02 | Test phishing URL detection | 1. Visit [http://phishingsite.](http://phishingsite/) com. 2. Observe extension behaviour. | [http://phishingsite.co](http://phishingsite.co/) m | Alert displayed: "Warning: Phishing URL" | Alert displaye d: "Warnin g: Phishing URL" | Pass |
| UAT- 03 | Test borderline  suspicious URL | 1. Visit a site with   mixed features (e.g., long URL | URL with mixed indicators | Alert:  "Suspicious URL" | Alert:  "Suspici ous | Pass |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Test Steps** | **Input** | **Expected Output** | **Actual Output** | **Status** |
|  |  | with HTTPS).  2. Observe result. |  |  | URL" |  |
| UAT- 04 | Test feedback clarity | 1. Open extension popup after visiting a flagged site. 2. Check for feature breakdown. | [http://phishingsite.co](http://phishingsite.co/) m | Clear explanation of phishing risk displayed | Clear explanati on of phishing risk displaye d | Pass |

### CHAPTER 6: LIMITATIONS OF SYSTEM

##### Detection of Evolving Phishing Techniques

While the system uses machine learning to detect phishing, attackers constantly evolve their tactics. Advanced techniques, such as visually spoofed websites or using legitimate URLs as intermediaries, can sometimes bypass detection, limiting the system’ s ability to identify all phishing attempts.

##### Dependence on Training Data Quality

The effectiveness of the machine learning models heavily relies on the quality and diversity of training data. If the dataset used is outdated or lacks certain phishing patterns, the model may struggle to accurately identify newer or less common phishing techniques.

##### False Positives and False Negatives

Machine learning models can sometimes misclassify URLs, resulting in false positives (flagging legitimate sites as phishing) or false negatives (missing actual phishing sites). False positives may inconvenience users, while false negatives can leave users exposed to potential threats.

##### Limited to Chrome Browser

The system is currently implemented as a Chrome extension, limiting its availability to Chrome users. Users of other browsers, such as Firefox, Safari, or Edge, are not able to benefit from the system unless it is expanded to those platforms.

##### Privacy and Security Concerns

Although the system is designed to protect user data privacy, some users may have concerns about data processing and URL analysis. Ensuring transparency in data handling and privacy compliance is essential to address any user hesitations regarding data privacy.

### CHAPTER 7: PROPOSED ENHANCEMENTS

##### Expanded Browser Compatibility

To reach a wider audience, the system can be extended to support additional browsers like Firefox, Safari, and Edge. This enhancement would make phishing protection accessible to users regardless of their preferred browsing platform.

##### Integration with Email and Messaging Platforms

Phishing attacks are common in emails and messaging apps. By integrating the system with popular email clients (e.g., Gmail, Outlook) and messaging platforms, users can be alerted to phishing links within these services, expanding protection beyond web browsing.

##### Improved Machine Learning Model with Continuous Learning

Implementing continuous learning by periodically updating the model with new phishing data can improve detection accuracy. This adaptive approach will ensure the system remains effective against evolving phishing tactics and emerging threats.

##### Enhanced User Education and Training

Adding educational features, such as interactive phishing awareness modules or detailed explanations of phishing indicators, can help users understand phishing tactics. Educated users are more likely to recognize suspicious links, reducing their risk of falling victim to phishing.

##### Customizable Sensitivity Settings

Allowing users to adjust the sensitivity of phishing detection can improve user experience by tailoring the system to individual preferences. For example, users could choose a more conservative setting to minimize false positives or a stricter setting to maximize protection.

##### Real-Time Threat Intelligence Integration

Integrating real-time threat intelligence feeds from cybersecurity databases would enhance the system’s accuracy. This feature would enable the detection of newly discovered phishing sites almost instantly, reducing users’ exposure to fresh threats.

##### Enterprise-Level Dashboard and Analytics

For organizational use, a centralized dashboard providing phishing detection analytics and threat reports would benefit IT and security teams. This feature would allow administrators to monitor phishing threats within their organization and proactively address vulnerabilities.

These enhancements aim to improve the system’s accessibility, accuracy, adaptability, and user experience, creating a more robust and versatile phishing detection tool.

### CHAPTER 8: CONCLUSION

The proposed phishing detection system provides an innovative, machine learning-based approach to tackle phishing, one of today’s most prevalent cybersecurity threats. Unlike traditional systems that rely solely on blacklists or static rule-based filtering, this system employs dynamic machine learning models to analyze and classify URLs in real time. This allows it to proactively protect users by blocking phishing sites before they can steal sensitive information, which is especially beneficial given the rapid evolution of phishing tactics.

The system is implemented as a Chrome extension, making it easily accessible and intuitive for a broad range of users, including individual internet users, corporate employees, SMEs, and educational institutions. Through this interface, users receive immediate alerts about suspicious websites, helping them avoid potentially harmful interactions. The use of well-established machine learning algorithms like Support Vector Machines, Decision Trees, and Neural Networks enhances the accuracy and efficiency of phishing detection, providing a high level of security with minimal disruption to the user’s browsing experience.

While effective, the system has certain limitations. Its reliance on the quality and recency of training data means that it may occasionally misclassify URLs, resulting in false positives or false negatives. Additionally, as a Chrome-only solution, its reach is limited to users of this browser. However, these limitations are addressed in the proposed future enhancements, such as expanding support to additional browsers, integrating with email platforms, and enabling continuous learning. These improvements will ensure the system remains adaptive and relevant as phishing techniques continue to evolve.

Proposed enhancements, such as customizable sensitivity settings, real-time threat intelligence integration, and a centralized dashboard for enterprise users, have the potential to make the system even more versatile. By adding features that allow the system to be tailored to different user needs and organizational requirements, the tool can be widely deployed across various environments, further increasing its impact.

In conclusion, this system offers a valuable and scalable solution to mitigate phishing risks. By providing users with both protection and education, it not only blocks malicious attempts but also raises awareness about phishing tactics, empowering users to navigate the digital world more safely. With its adaptive machine learning foundation and planned enhancements, the system promises to play a critical role in fostering a safer online environment and contributing to a proactive cybersecurity posture across individual and organizational users alike.

### CHAPTER 9: BIBLIOGRAPHY

1. V. B. et al, “study on phishing attacks,” International Journal of Computer Applications, 2018.
2. P. Kumaraguru, J. Cranshaw, A. Acquisti, L. Cranor, J. Hong, M. A.Blair, and T. Pham, “School of phish: a real-world evaluation of antiphishing training,” in Proceedings of the 5th Symposium on Usable Privacy and Security, pp. 1–12, 2009.
3. R. C. Dodge Jr, C. Carver, and A. J. Ferguson, “Phishing for user security awareness,” computers & security, vol. 26, no. 1, pp. 73–80, 2007.
4. R. Dhamija, J. D. Tygar, and M. Hearst, “Why phishing works,” in Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 581–590, 2006.
5. C. Ludl, S. McAllister, E. Kirda, and C. Kruegel, “On the effectiveness of techniques to detect phishing sites,” in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment,

pp. 20–39, Springer, 2007.

1. A. P. Rosiello, E. Kirda, F. Ferrandi, et al., “A layout-similarity-based approach for detecting phishing pages,” in 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007, pp. 454–463, IEEE, 2007.
2. S. Afroz and R. Greenstadt, “Phishzoo: Detecting phishing websites by looking at them,” in 2011 IEEE fifth international conference on semantic computing, pp. 368–375, IEEE, 2011.
3. K.-T. Chen, J.-Y. Chen, C.-R. Huang, and C.-S. Chen, “Fighting phishing with discriminative keypoint features,” IEEE Internet Computing, vol. 13, no. 3, pp. 56–63, 2009.
4. A. K. Jain and B. B. Gupta, “Phishing detection: Analysis of visual similarity-based approaches,” Security and Communication Networks, vol. 2017, 2017.
5. R. S. Rao and S. T. Ali, “A computer vision technique to detect phishing attacks,” in 2015 Fifth International Conference on Communication Systems and Network Technologies, pp. 596–601, IEEE, 2015.
6. B. B. Gupta, N. A. Arachchilage, and K. E. Psannis, “Defending against phishing attacks: taxonomy of methods, current issues and future directions,” Telecommunication Systems, vol. 67, no. 2, pp. 247–267, 2018.