Phishing Detection System through Hybrid Machine Learning

K Shireesha, Jagini Jaishna, Siddagoni Pavani
[1]Associate Professor, Department Of Cse, Bhoj Reddy Engineering College For Women, India.
[2,3,4,5]B. Tech Students, Department Of Cse, Bhoj Reddy Engineering College For Women, India.

ABSTRACT

The internet is in nowadays used to coordinate a variety of types of cybercrimes. Therefore, the primary subject of this study is phishing attacks. Phishing uses email distortion as its fundamental tactic. To gather the necessary data from the concerned parties, challenging correspondences are followed by mock sites. There is currently no complete and effective method for preventing phishing attacks, despite the fact that various studies have published their work on prevention, detection, and knowledge of these attacks. As a result, machine learning is essential in the fight against online crimes like phishing. The proposed study is based on the phishing URL-based dataset, which is a collection of phishing and legitimate URL features collected from more than 11000 website datasets. Several machine learning methods have been used after preprocessing in order prevent phishing URLs and provide user protection. This study utilizes several machine learning models like decision trees, linear regression, random forests, naive Bayes, gradient boosting classifiers, support vector classifiers, and a proposed hybrid LSD model that combines decision trees, support vector machines, and logistic regression with both soft and hard voting to effectively and accurately defend against phishing attacks. The proposed LSD model makes use of the GridSearchCV hyper parameter optimization technique and the canopy feature selection technique with cross fold validation.

Keywords: *Phishing, LSD Model, GridSearchCV, Canopy Feature Selection.*

1-INTRODUCTION

The speed at which internet technologies have advanced is both a welcome benefit and a welcomed threat to the digital landscape. Phishing is one of the most well-known cyber threats today and is understood as misleading malicious actors impersonating legitimate services or individuals to extract personal sensitive data, such as usernames, passwords or financial details. Phishing generally works through emails and fake websites designed to trick individuals into entering their personal information. Although there are multiple techniques available for detection including blacklists and heuristic based filters, phishing detection and the practical issue of protecting users continues to be a burden due to its evolving nature and rapid appearance of zero-day attacks. Since traditional detection systems can be limited in their abilities, I will employ a smarter and more adaptive approach by using hybrid machine learning approaches. Starting with a URL based phishing dataset with over 11,000 records, I will assess the structure and semantic patterns of web links and classify them as being either phishing or legitimate. The model will be generated using supervised training models such as Decision Trees, Random Forest, Support Vector Machines, Naive Bayes, Gradient Boosting, and Logistic Regression. Most importantly, will be an ensemble model called Hybrid LSD which combines logistic regression, an SVM, and Decision Trees together for predictions using both soft and hard voting methods.

2-LITERATURE REVIEW

- Phishing Detection Leveraging Machine Learning and Deep Learning[1]:
  Authors: Dinil Mon Divakaran et al.
  Phishing remains a serious cybersecurity issue, usually deceiving people into revealing sensitive information like login credentials or bank account details. This research investigates how cutting-edge technologies—namely machine learning and deep learning—coupled with the strength of big data, can be utilized to detect and counter phishing attacks. By examining high levels of online activity and distilling patterns, these smart systems can learn to identify suspicious activity and deceiving content, providing a proactive solution to protecting users from online fraud.
- Cyber Security: The Lifeline of Information and Information Technology[2]:
  Authors: Kameshwar Prasad and Deepak Rawat.
  With the worldwide boom in digital connectivity and the mass adoption of mobile and smart technologies, maintaining cybersecurity has never been more important. This book gives a thorough overview of the new security threats from modern technological environments, such as IoT devices, smart infrastructure, and the transition from 4G to 5G networks. It also examines how emerging technologies such as blockchain and artificial intelligence can be used to help develop safer and more intelligent digital landscapes. As an invaluable guidebook, the book provides real-world insights for both researchers and professionals who want to learn about and deal with the intricacy of today's cyber threats.
- An Analysis of Phishing Blacklists: Google Safe Browsing, OpenPhish, PhishTank[3]:
  Authors: S. Bell and P. Komisarczuk
  This research performs a thorough examination of three popular phishing blacklists—Google Safe Browsing, OpenPhish, and PhishTank—during the 75-day monitoring period. The study identifies how each site handles identification and delisting of malicious URLs and finds that phishing websites typically have short existences. Surprisingly, some URLs were discovered to reappear soon after delisting, which sparks worries about premature delisting and repeated threats. The results also indicate that OpenPhish in all cases flagged for threats sooner than PhishTank, despite having a smaller total list. There was a significant overlap, with roughly 12% of the URLs found on both OpenPhish and PhishTank, and OpenPhish flagging more than 90% of the shared URLs first. The findings reinforce the necessity for regular updates and better management of blacklist systems to improve user security against rapidly evolving phishing scams.
- Physical Attributes Significant in Preserving the Social Sustainability of the Traditional Malay Settlement[4]:
  Authors: Nor Zalina Harun, Nor Jariah Jaffar, and Patricia S. J. Kassim
  Traditional settlements are places where both the built environment and inhabitants still exhibit long lasting cultural practices and traditional skills. Nevertheless, the urban growth forces and modernization are starting to transform these heritage areas, particularly in Malaysia. This study seeks to determine the physical features that advance social cohesion and cultural continuity in historic Malay settlements. Using a qualitative research approach, the study examined settlements in Kuala Terengganu and determined three significant features that advance social sustainability: street pattern, property boundaries, and common open spaces. These aspects were discovered to be crucial in promoting community interaction and maintaining cultural identity. The research highlights the value of

careful planning and spatial arrangement in making certain that traditional communities are able to deal with contemporary stresses without fraying their social fabric.

- Exploring a robust Machine Learning Classifier for detecting phishing domains using SSL certificates[5]:

Authors: Akanchha Akanchha

Due to the phishing sites appearing genuine to users, detecting them is a challenging task. The SSL certificate that is generally used to secure and encrypt the communication, can also be generated for the phishing sites. Phishing websites continue to be a difficult challenge to identify, as they are made to look legitimate to innocent users. One of the underhand methods is the utilization of SSL certificates, which people often equate with secure and trusted sites, but malicious players can now also acquire them for phishing websites. This misuse of HTTPS can make users believe in fraudulent sites simply because they see a padlock icon or a secure connection symbol in the browser. This research explores how phishing websites exploit SSL certificates and looks at the key certificate attributes that can help distinguish between genuine and fake domains. To address this issue, a detection system was developed that utilizes machine learning algorithms to analyze SSL certificate data. Amongst the models tested, the choice of decision tree algorithm was made for its interpretability and robust performance. The algorithm produces a set of decision rules to classify websites as legitimate or phishing based on patterns in their SSL certificate features.The system was proved to be highly accurate, classifying nearly 97% of test cases correctly. To make this model usable in real-time, a Web API was also created. This API enables users to input a domain and obtain a classification decision based on the rules of the decision tree. The test results validate that the suggested solution is efficient and robust, providing a realistic approach towards phishing detection by taking advantage of the very SSL signs usually exploited for misdirecting users.

- Modelling Hybrid Feature-Based Phishing Websites Detection Using Machine Learning Techniques[6]:

Authors: Sumitra Das Gupta et al.

This paper proposes a hybrid model using both URL and hyperlink features for detecting phishing sites. The authors build a real-time detection system that operates independently of third-party tools like search engines. They introduce a unique dataset with multiple ML algorithms to validate the effectiveness of hybrid features in improving model predictions and handling zero-hour attacks

- About Performance of NLP Transformers on URL-based Phishing Detection for Mobile Devices[7]:

Authors: Hossein Shirazi et al.

This research assesses transformer-based models, like BERT and RoBERTa, for phishing detection on mobile devices. The study uses URL-only input to streamline the model. The researchers analyze performance trade-offs between accuracy and computational efficiency -- ultimately choosing MobileBERT as an efficient option for real-time mobile detection, without reliance on server-side processing.

- Machine Learning for Detecting Phishing Websites[8] :

Authors: Amani Alswailem et al.

This research focused on the value of browser extensions for phishing prevention. The research utilized feature selection to create a ML-based browser tool that could warn user in real time. The study discussed the growing issue developing phishing strategies and the need to continually evolve the feature set so that models remain functional.

3-SYSTEM REQUIREMENT SPECIFICATIONS

Software requirements relate to specifying software resource requirements and pre- installation conditions required to a computer that can installed software with a minimal level of performance. These requirements or pre-conditions are not contained in the software installation package and have to be installed independently before the installation of the software.

Operating system is arguably the first requirement specified when defining system requirements (software). Software applications may not run on different versions of the same line of operating systems, although some level of backwards compatibility is generally maintained. The majority of applications written utilizing newer features of Linux Kernel v2.6 do apparently not run (or compile, or function) properly utilizing Networks with Linux Kernel (2.2, or 2.4).

Platform - A platform is a framework of some kind, either in hardware or software, allowing software applications to run. Examples of a platform include programming language and runtime libraries.

APIs and drivers - Software applications severely dependent on utilizing special hardware devices, particularly high, high-end display adapters, requires special API(s) or new device drivers.

PYTHON LANGUAGE

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python has high-level built in data structures, combined with dynamic typing and dynamic binding, which makes it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python uses simple, easy to learn syntax emphasizing readability which reduces the cost of program maintenance or development. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available for free in source or binary form for all major platforms, and can be freely distributed. Often programmers get attached to Python as a result of the increased productivity it allows. The edit-test-debug cycle is incredibly fast, and debugging Python programs is often an uneventful experience: a bug or bad input never causes a segmentation fault. When the interpreter hits an error, it raises an exception.

The interpreter prints a stack trace when an exception is not caught by the program. A source level debugger allows for inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through code a line at a time, and so on. The very fact that the source level debugger is written in Python speaks to the power of introspection in Python. However, the truth is that oftentimes, the fastest and simplest way to debug a program is to put a few print statements in the source code, and the fast edit-test-debug cycle makes this a very effective approach.

Python is a dynamic, high-level, free open-source, interpreted programming language that supports procedural-oriented programming as well as object-oriented programming. In Python, we don’t need to explicitly state the type of variable because it is dynamically typed language.

HARDWARE REQUIREMENTS

- Processing power – The processing power of the central processing unit (CPU) is the first system requirement for any software. Most software that runs on x86 architecture defines processing power as the model and clock speed of the CPU. Many other characteristics of a CPU, such as bus speed, cache, and MIPS, influence its speed and processing power and are seldom discussed. This definition of power is often not accurate, because similar clock speed AMD Athlon and Intel Pentium CPUs often have significantly different throughput speeds. Intel Pentium CPUs have been very popular and will often come up in this category.

- Memory – All software, when running, is in the random access memory (RAM) of a computer. Memory requirement definition includes the requirements of the application, operating system, supporting software and files, as well as any other concurrent processes running on a multi-tasking computer system that can contribute to the hardware specification for the additional software.
- Secondary storage – Hard-disk requirements, in relation to the secondary storage of the software, will vary depending on the size of the software installation and the amount of temporary files it creates and manages while installing it or while it runs, as well as any swap space (if RAM is insufficient).

NON-FUNCTIONAL REQUIREMENTS

- **Usability**
  The interface should be user-friendly and intuitive for everyone, including those without much technical expertise.
- **Serviceability**
  The system should include tools to diagnose problems (misclassifications, loading errors, etc.) and logging to allow for debugging and maintenance.
- **Manageability**
  The model and web application should be easy to manage, update, and deploy, with clear delineation of responsibilities across front-end, back-end, and ML logic.
- **Recoverability**
  The system should be able to recover operations smoothly after a failure or crash, without losing user data or model accuracy.
- **Security**
  The system must provide secure user authentication and at the same time secure user inputs against all malicious entry. The system must protect itself against SQL Injection, URL-based attacks, and preventing unauthorized access to model predictions.
- **Data Integrity**
  The input data, the training data, and the output data from the model must be consistent and unmodified throughout all the phases of the system lifecycle.
- **Capacity**
  The system should be able to support a reasonable volume of simultaneous URL submissions and user logins, without causing performance degradation.

4-SYSTEM DESIGN

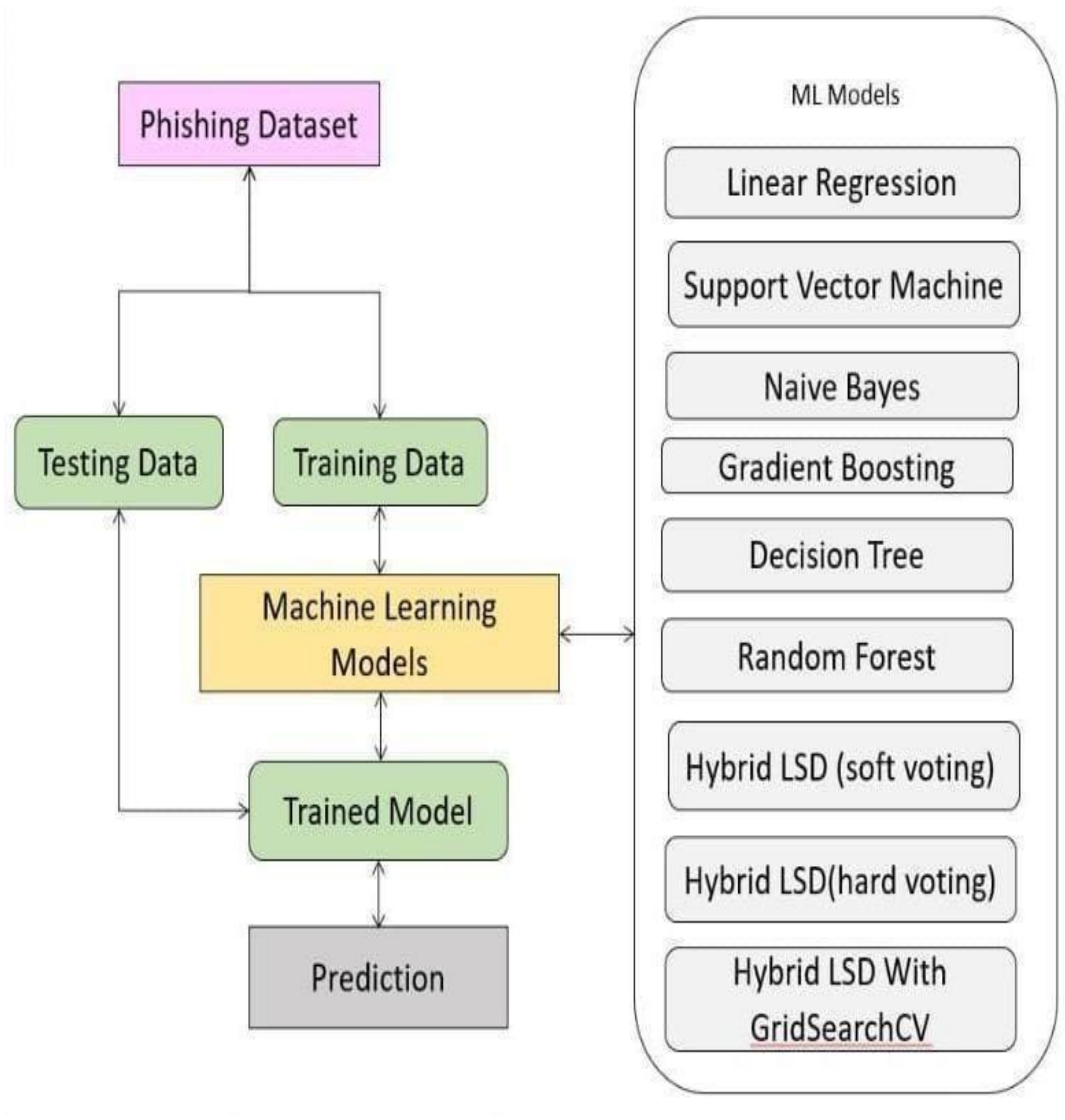


Fig 1: System architecture

DATA FLOW DIAGRAM

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
- DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
- DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction.

DFD may be partitioned into levels that represent increasing information flow and functional detail.

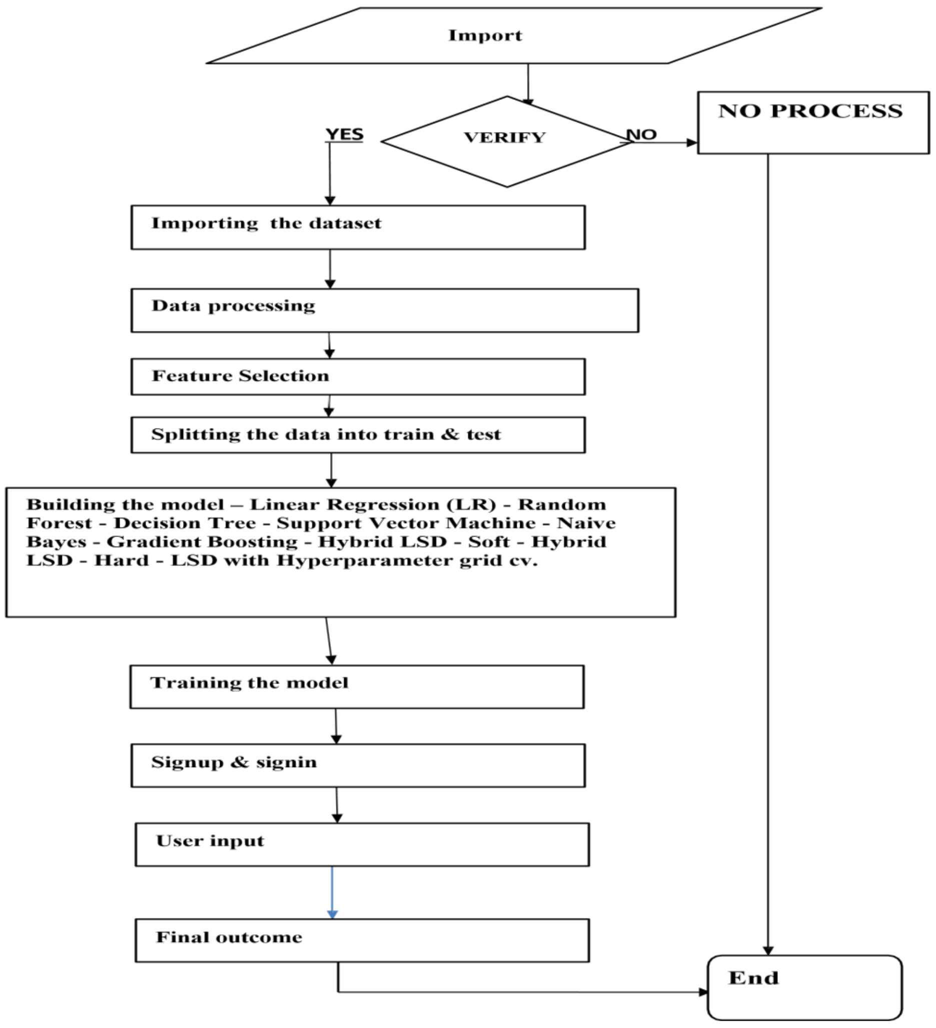

Import
VERIFY
YES
NO
NO PROCESS
Importing the dataset
Data processing
Feature Selection
Splitting the data into train & test
Building the model – Linear Regression (LR) - Random Forest - Decision Tree - Support Vector Machine - Naive Bayes - Gradient Boosting - Hybrid LSD - Soft - Hybrid LSD - Hard - LSD with Hyperparameter grid cv.
Training the model
Signup & signin
User input
Final outcome
End

Fig 2: Data Flow Diagram

5-IMPLEMENTATION

The intended system is aimed at presenting an intelligent solution to phishing threat detection utilizing a hybrid machine learning method based on URL-based analysis. Using a dataset provided by Kaggle that is publicly available, the system employs several features derived from both legitimate and malicious URLs. For appropriate classification of these URLs, the system employs different machine learning algorithms, including Linear Regression, Decision Trees, Random Forests, Support Vector Machines, Naive Bayes, and Gradient Boosting. To enhance model performance further, an ensemble method is suggested— referred to as the LSD hybrid model—which makes use of Logistic Regression, SVM, and Decision Tree classifier decisions both through soft (probabilistic) and hard (majority) voting methods. In order to refine the configuration of the model to deliver optimal results with maximum accuracy, GridSearchCV is used to perform hyperparameter tuning so that the system is able to select the optimum parameters for each of the algorithms. To deploy the system, the entire system is built into a Flask-based web application. This web platform incorporates a SQLite driven user registration and login system for a minimal but secure interface. When a user provides a URL, it is subject to feature extraction with the help of tools like BeautifulSoup and WHOIS . The extracted features are then passed on to the machine learning model, which has been trained to forecast whether the URL is genuine or a phishing attempt.

Dataset

In this phishing detection system, the dataset plays a central role and is obtained from Kaggle's open-source repository. It contains 11,055 website entries, each labeled as either legitimate or phishing. Rather than relying on the raw URL, the dataset represents each entry through a collection of over 30 numerical features. These features reflect various characteristics related to the structure of the URL, domain details, and behavioral indicators. Examples include the use of IP addresses in place of domain names, URL shortening services, special characters such as the "@" symbol, hyphenated domain segments, embedded iframes, domain registration age, and estimated web traffic. The classification label, referred to as the Result, assigns a value of 1 to phishing URLs and -1 or 0 to legitimate ones. To prepare the dataset for machine learning, a series of preprocessing steps are carried out, and a stratified train-test split to preserve the ratio of classes. All feature values are encoded numerically to facilitate compatibility with classification algorithms. The structured nature of the dataset allows for efficient training of various models, enabling accurate phishing detection purely based on URL attributes, without the need for inspecting the full website content.

Feature Extraction

Even though the dataset already has a good set of features, we took it a notch up by deriving some more information from the external tools to improve the model accuracy. For this we used BeautifulSoup (a Python library for scraping the data from the HTML content of the web pages). This enabled detection of some common red flags typical on phishing sites, including hidden elements, automatic redirection via JavaScript, and deceptive anchor tags that don't go anywhere. At the same time, we conducted WHOIS lookups to collect valuable background information on each domain — such as when the domain was registered, who owns it and how soon it's designed to expire. This information matters because phishing sites are usually associated with domains that are newly created or registered for extremely short periods. Once we fetched this information, we converted all the features to numeric values for our machine learning models to process efficiently. We did some data scrubbing by filtering out features that had no overwhelming value and ensured the model is focusing on key aspects only needed in order to make accurate predictions.

Data Preprocessing

Before we move on to training the machine learning models, we need to ensure that our dataset is in prime condition. This involves going through some necessary preprocessing to make sure that all is clean, consistent, and ready for analysis. We go to the trouble of treating any missing or incomplete data entries—either by filling them in with suitable values or by eliminating them altogether to prevent any skewing of our findings. Since some algorithms, like Support Vector Machines, can be extremely sensitive to the range of input features, we do feature normalization by applying a standardization technique so that all values are within a uniform range. We also translate the classification labels, which tell us whether a URL is phishing or legitimate, to numerical form so they can lie alongside our machine learning models.

Finally, we split the dataset into training and test sets, typically in a 70- to-30 ratio, and use stratified sampling to ensure both phishing and legitimate examples are suitably represented in each subset. These preprocessing steps provide a solid foundation for accurate and efficient model training. A range of Python libraries were imported to handle different parts of workflow, from data processing to deployement. Pandas and Numpy were used for reading the dataset and performing data manipulation and numerical operations.To train and train the models we used scikitlearn. Flask and SQLite are used to deploy the system as web application.

6-RESULT

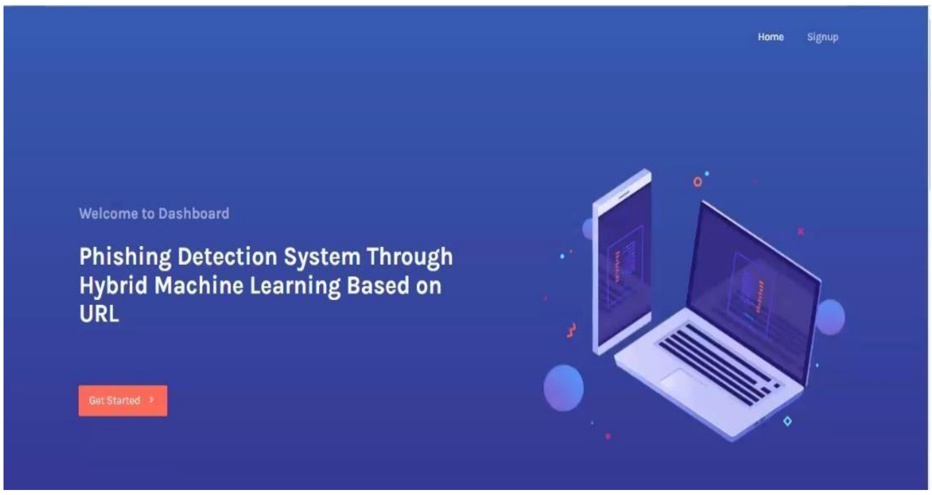

Home
Signup
Welcome to Dashboard
Phishing Detection System Through Hybrid Machine Learning Based on URL
Get Started

Fig 1: Dashboard

This is the homepage/dashboard interface of the phishing detection system. It provides users with:

- A welcome screen
- Navigation options (signup/home)
- A clear call to action to begin using the detection system

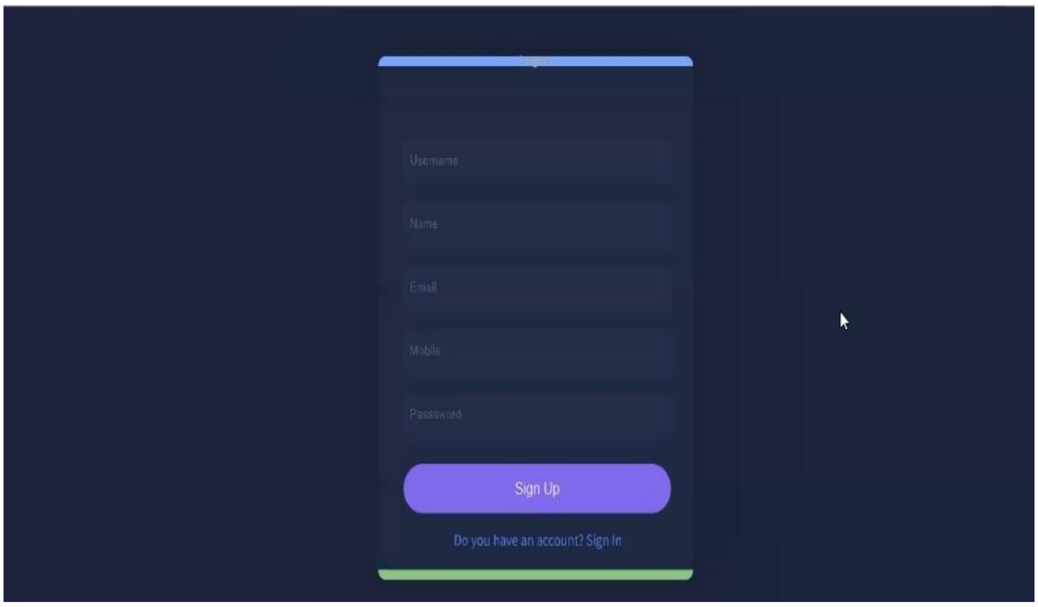


Fig 2: Sign Up Page

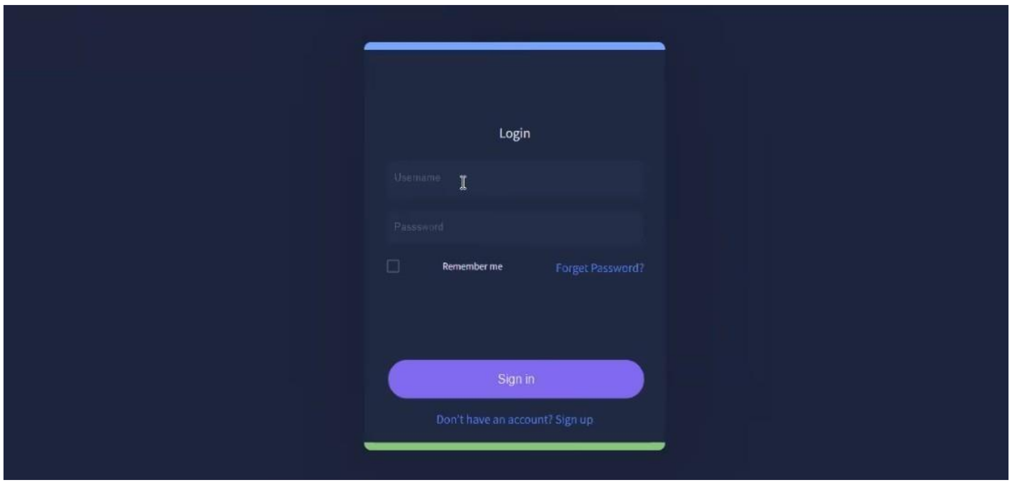


Fig 3: Login page

This Login Page acts as the gateway to the secure, authenticated portion of the phishing detection system

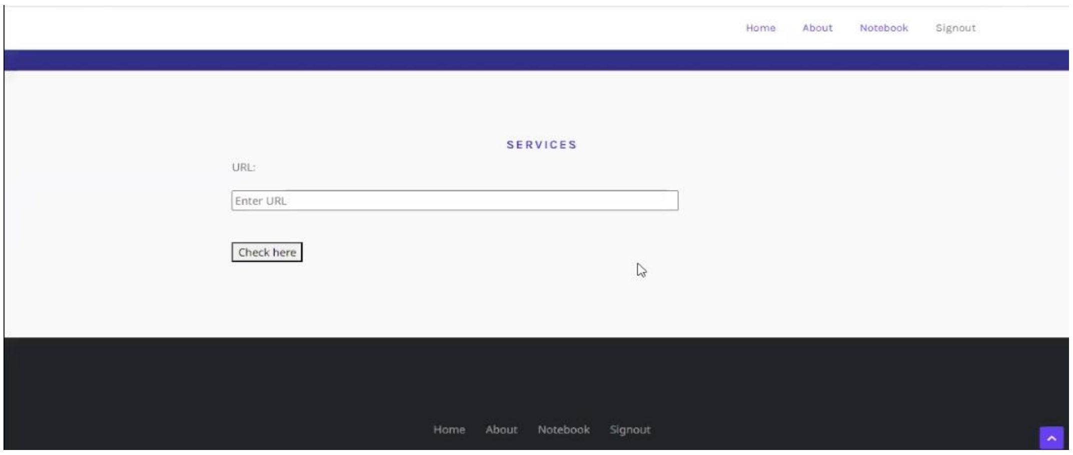
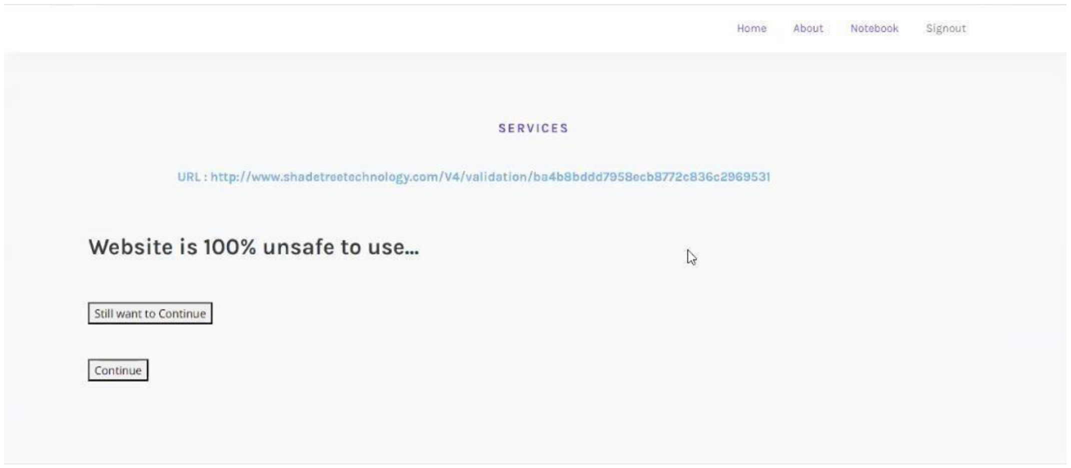


Fig 4: URL input

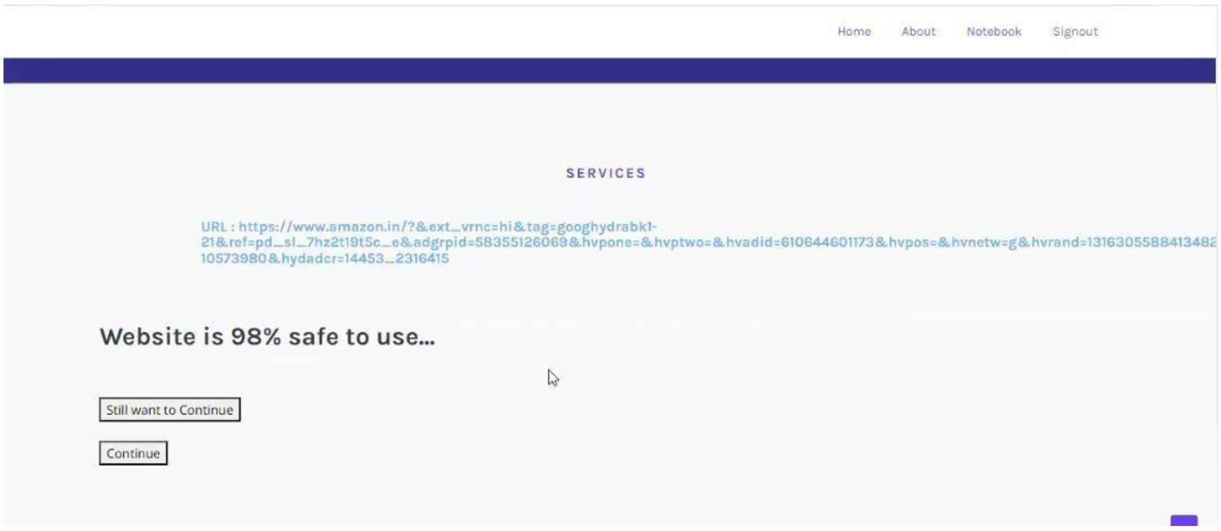
To deliver the main service of the application detecting phishing attempts through URL analysis using a hybrid machine learning approach.



URL : http://www.shadetreetechnology.com/V4/validation/ba4b8bddd7958ecb8772c836c2969531

Website is 100% unsafe to use...

Fig 5: URL analysis result page



URL : https://www.amazon.in/?&ext_vrnc=hi&tag=googhydrabk1-21&ref=pd_sl_7hz2t19t5c_e&adgrpid=58355126069&hvpone=&hvptwo=&hvadid=610644601173&hvpos=&hvnetw=g&hvrand=1316305588413482
10573980&hydadcr=14453_2316415

Website is 98% safe to use...

Fig 6: URL analysis result page

This page is meant to inform the user of the model’s classification decision and provide a strong deterrent against visiting malicious sites. By giving both a warning and an option to proceed, it maintains a balance between security and user control.

7-CONCLUSION

Phishing remains one of the most prevalent and harmful issues in the field of cybersecurity by taking advantage of human trust and biases in computer systems. In this project we approached this problem by developing and implementing a smart automated phishing detection system that employs hybrid machine learning. The proposed framework can analyze URL-based features to distinguish the legitimacy of a website in real-time and provide users with a useful defense against phishing attacks. The system draws on multiple machine learning models (Logistic Regression, Support Vector Machine, Decision Tree, Random Forest, Gradient Boosting, and Naive Bayes) to deliver a dependable classification engine. Most notably, the project has implemented a Hybrid LSD (Logistic, SVM, Decision Tree) ensemble model with soft and hard voting techniques. The ensemble method improves detection reliability, while also reducing the probability of false positives and false negatives. In conclusion, this phishing detection system illustrates how modern machine learning can be successfully bundled and optimized for current end users as a practical defense against online threats. This thesis implements URL based detection, however future improvements could include content-based analysis of web pages, scraping web page behavior in real time, and including browser extensions for improved usability.

REFERENCES

[1] D. M. Divakaran and A. Oest, ‘‘Phishing detection leveraging machine learning and deep learning: A review,’’ 2022, arXiv:2205.07411.

[2] K. Prasad and D. Rawat, “Cyber Security: The Lifeline of Information and Communication Technology.” Cham, Switzerland: Springer, 2020.

[3] S. Bell and P. Komisarczuk, ‘‘An analysis of phishing blacklists: Google safe browsing, OpenPhish, and PhishTank,’’ in Proc. Australas. Comput. Sci. Week Multiconf. (ACSW), Melbourne, VIC, Australia. New York, NY, USA: Association

for Computing Machinery, 2020, pp. 1–11, Art. no. 3, doi: 10.1145/3373017.3373020.

[4] N. Z. Harun, N. Jaffar, and P. S. J. Kassim, “Physical attributes significant in preserving the social sustainability of the traditional Malay settlement,” in Reframing the Vernacular: Politics, Semiotics, and Representation, Springer, 2020, pp. 225–238.

[5] A. Akanchha, ‘‘Exploring a robust machine learning classifier for detecting phishing domains using SSL certificates,’’ Fac. Comput. Sci., Dalhousie Univ., Halifax, NS, Canada, Tech. Rep. 10222/78875, 2020.

[6] S. D. Gupta, A. S. Ghosh, A. S. Chakraborty, and P. Ghosh, “Modelling hybrid featurebased phishing websites detection using machine learning techniques,” Journal of King Saud University - Computer and Information Sciences, 2022.

[7] H. Shirazi, A. Ghiasi, and R. Jalili, “About performance of NLP transformers on URLbased phishing detection for mobile devices,” in Proc. IEEE Int. Conf. Big Data, 2021.

[8] A. Alswailem, M. Alshamrani, and A. Almulhem, “Machine learning for detecting phishing websites,” in Proc. 5th Int. Conf. Inf. Syst. Security Priv., 2020, pp. 435–440.

[9] G. Sonowal and K. S. Kuppusamy, ‘‘PhiDMA—A phishing detection model with multi-filter approach,’’ J. King Saud Univ., Comput. Inf. Sci., vol. 32, no. 1, pp. 99–112, Jan. 2020.

[10] A. K. Jain and B. Gupta, ‘‘PHISH-SAFE: URL features-based phishing detection system using machine learning,’’ in Cyber Security. Switzerland: Springer, 2018, pp. 467–474.