Big Data Security Analytics for Phishing URLs Detection

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Ugyen Dorji (191452)

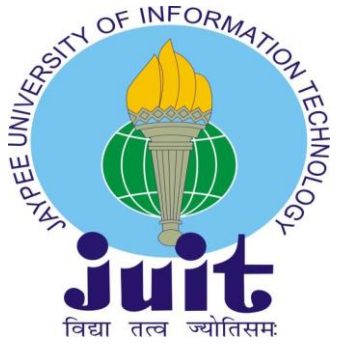
&

Tenzin Gyaltshen (191453)

Under the supervision of

Dr. Deepak Gupta

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Big Data Security Analytics for Phishing URLs Detection"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **Dr. Deepak Gupta**, Asst. Prof.(SG), Department of Computer Science and Engineering & Information Technology.
I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Data Science**.
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Ugyen Dorji, 191452

(Student Signature)
Tenzin Gyaltshen, 191453

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Deepak Gupta
Asst. Prof. (SG)
Computer Science and Engineering & Information Technology
Dated: 8th May 2023

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……..

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……..(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Acknowledgement

Firstly, we would like to thank the divine Kencho-Sum (The Triple Gem) for always keeping us in the best health and mentality to carry on with this project. We are always grateful and indebted to be under the protection of Mahakala, the Buddhist Protector Deity. We pray our faith will keep us humble and down to earth always.

We would like to acknowledge our parents next for being our biggest supporters, both morally and financially, even though we are far from each other. You remain our greatest motivation and inspiration in life to do well and enjoy a good life. We hope to make you proud and are always grateful for all opportunities in life.

We want to acknowledge our project supervisor Dr. Deepak Gupta for his expertise and care in supervising us for this project. He has always been approachable and helpful, even to the extent of giving us copies of his research works, which has been beyond helpful and extremely informative. We are always grateful.

We would also like to thank the Major Project coordinator, the institution, and all the important executive members of the institution for this wonderful opportunity to work on a project that would obviously lead us to better doors in our career.

Finally, we would like to thank ourselves for working hard to make this project a success. We are happy with ourselves because we have done the course of this project successfully.

This project has been carried out in our best interests and hope it becomes helpful to anyone who refers to it in the future.

Ugyen Dorji
191452

Tenzin Gyaltshen
191453

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviation	Full Form
PCA	Principal Component Analysis
LR	Logistic Regression
SVM	Support Vector Machine
KNN	k-Nearest Neighbors
DT	Decision Tree
IG	Information Gain
RF	Random Forest
Sklearn	Scikit Learn
URLs	Uniform Resource Locators
TPR	True Positive Rate
TNR	True Negative Rate
ES	Ensemble Stacking
MI	Mutual Information

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Phishing is a type of cybercrime in which uninformed internet users are duped into disclosing personal information such as login credentials and credit card information. Unlike software vulnerabilities, phishing attempts target human weaknesses and can be difficult to detect. We created multiple machine learning models in this research to detect phishing URLs based on their attributes. The models were trained in three ways: with all features, with feature selection, and with feature reduction. We employed approaches such as principal component analysis, ensemble modelling, mutual information gain, and stacking development to increase the performance of the models.

The models were trained using support vector machine and decision tree classifiers, and the model with the highest accuracy was chosen for deployment. We employed a confusion matrix to assess the model's performance, which had an accuracy rate of 96% in detecting phishing URLs. The chosen model was cloud-deployed and incorporated into a Chrome extension utilizing manifest 3, allowing for real-time detection of phishing URLs.

The research emphasizes machine learning's potential for improving online security and lowering the dangers connected with phishing attempts. The usage of feature reduction was discovered to be effective in improving the model's performance. The study also emphasizes the significance of ongoing monitoring and detection of phishing assaults, as people might fall victim to these attacks even when utilizing reputable websites.

Finally, the research demonstrates how machine learning approaches can be applied to detect phishing URLs and how feature reduction can improve model performance. The Chrome extension created as part of this research is a valuable tool for detecting phishing URLs in real time and adds to ongoing efforts to improve internet security.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

The evolution of tech-based commerce has immensely benefitted many people, allowing them to make any transaction in the comfort of their own homes. The technology has changed the entire commerce industry, making it very easy for buying and selling products with the tap of a finger. However, the new changes and developments have also proved to be a potential threat space with cybercriminals going all out to steal information from the uninformed internet fellow, also known as phishing.

Phishing is an already existing and growing cybercrime where cybercriminals use deceptive methods like fraudulent emails, messages, or any other possible means to deceive users into exposing their private information like card details, login information etc. to use for unethical purposes. The most common type of phishing is one in which cybercriminals develop webpages that look like legitimate sites and prompt the users to add their credentials. The users feel the site is legitimate and shares their credential not knowing that their personal information is now in the hands of cybercriminals.

The Anti-Phishing Working Group reports that there were more than 220,000 phishing attacks in the first quarter of 2021 alone. These attacks have been targeted at a range of industries, such as finance, healthcare, and retail. Further research data proves that the rate of phishing crimes have been increasing over the years continuously, implying that this cybercrime cannot be diminished completely anytime soon.

To protect oneself from phishing attacks, both individuals and organizations should take certain measures of precautions. Users must be careful and questioning when clicking on links in emails or messages when the sender is

unknown. They should use strong passwords which are hard to decode, and also take security measures offered by most websites these days like Two-Factor Authentication and adding recovery/secondary emails.

On the flipside, organizations should implement necessary security measures like utilizing anti-phishing software, training their personnel on identification and reporting phishing attacks when one occurs. With such security measures, organizations can drastically reduce the chances of phishing attacks and help safeguard their sensitive data.

This project is aimed at developing the best machine learning model capable of detecting phishing URLs in real time. For training this model, we use a dataset from Kaggle, which 11430 data points, each having 87 features that are broadly divided into URL features, content features, and several external features. The models will be compared based on evaluation metrics in three different scenarios (Without applying any Dimensionality Reduction, Applying Feature Selection and Applying Feature Reduction) with six different models (Logistic Regression, Support Vector Machine, Decision Tree, k-Nearest Neighbours, Random Forest and Ensemble Stacking). After the models have been trained and tested, we can deploy the most effective model on the cloud to build a Chrome extension that can identify whether a URL is phishing or legitimate. This project is a significant advancement in mitigating the growing risk of phishing attacks and protecting users from identity theft and financial loss.

1.1.1 PHISHING STATISTICS AND DELIVERY METHODS

With over 5 billion people online every day, risks of phishing increase with a lot of shady websites already on the World Wide Web. The problem is, even the most tech-fluent personnel can fall into this trap of scammers. The frequency of phishing attacks has been on the rise since March 2020, with IRONSCALES [2] reporting that 81% of organizations worldwide are attacked by this method of crime. Also, a disheartening report shows that only one in five businesses

have phishing awareness workshops once in a year, even though the threat it poses to this era of technology is immeasurable.

Phishing attacks are usually known only to occur with email services when it is taken at face value. However, now as social media branches to different variants, phishing attacks are launched on all possible mediums. While phishing threats from emails are still prominent with almost one-third occurring with the above-mentioned method, they are also delivered in different platforms like video conferencing platforms, instant messaging platforms including work-based platforms, cloud-based file storage services and also through SMS.

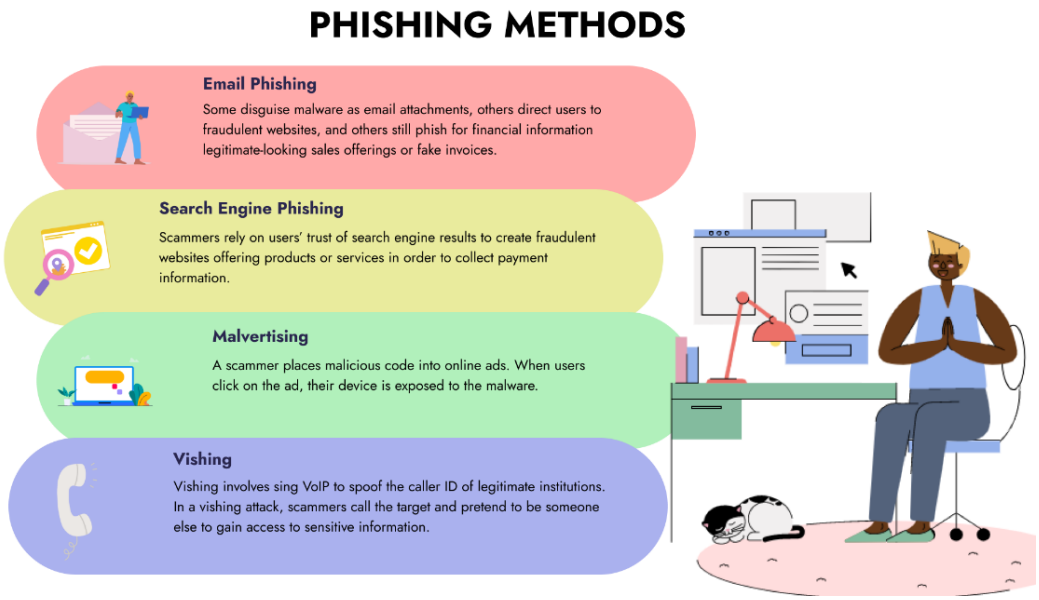


Fig 1.1 Phishing Delivery Methods

According to a report by Gone Phishing Tournament [3] in 2020, they found that 19.8% of users clicked on phishing links sent to them from which 67.5% gave sensitive information that could risk their personal information and credentials.

A report from ESET [4] concluded that, in the third quarter of 2020, the most common type of malicious files attached to phishing attacks through email were

Windows .exe files (74%), script files (11%), MS Office Documents (5%), compressed archives (4%), PDF documents (2%), Java files (2%), batch files (2%), shortcuts (2%), and Android executables (more than 1%).

According to a report by KnowBe4 [5], "Phishing by Industry," it was found that the highest number of victims of phishing attacks were from the industries of healthcare and pharmaceuticals, construction, and technology. From the above mentioned, the healthcare industry suffered most in terms of cost of attacks.

All these statistics prove the need for a solution to detection and security from phishing URLs before it can cause any more damage. These attacks can damage a lot to an industry from which the most dreadful are bankruptcy, delayed service, reputation damage, and loss of trust from their clients.

This project is aimed to help in contributing to fighting for a safer internet experience by developing a model that detects phishing URLs using a machine learning model trained using the best techniques.

1.2 PROBLEM STATEMENT

Phishing attacks are a major concern in today's technological age and the most vulnerable targets for phishing attackers are businesses and their employees. The attacks are targeted on employees through fraudulent emails and text messages, which misleads them to sharing their personal information, financial information or login credentials. These attacks are also targeted on huge businesses in the finance, healthcare and retail sectors with the goal of extracting huge sums of money and to destroy the public reputation of companies.

Phishing attacks are growing increasing complex with time as cybercriminals continue to develop and improve advanced social engineering tactics and

exploit loopholes in software and systems. This raises a high alert for internet users and businesses, and it is mandatory that they implement necessary security measures to protect themselves from these attacks. It is however surprising to see that most business personnel are uninformed and lack the skills to detect and report phishing attacks, while organizations also fail to implement a good security system due to limited resources.

The ever-changing nature of technology implies that phishing methods and techniques are always improving and possibly, new ones are being developed. This problem requires a combined effort from everyone in every level in a business setting. One can educate themselves and others through workshops and training on how to recognize and report a phishing attack. These problems also call for technological solutions to be able to detect phishing attacks in real time.

By addressing these problems, users and businesses can keep themselves safe from any phishing attack and not fall victim to loss of sensitive information, while minimizing the effects of cybercrime on society.

1.3 OBJECTIVES

The objectives of the project are:

- Compare and contrast the performance of machine learning models before and after dimensionality reduction in the context of detecting phishing URLs.
- Create a user-friendly API for the best performing machine learning model that can extract features from URLs in real-time.
- Implement the model and API in a browser extension for easy access and use by individuals.
- Evaluate the performance of the model and API using various metrics and real-world data to assess their effectiveness in detecting phishing websites.

1.4 METHODOLOGY

The project was carried out according to the following figure:

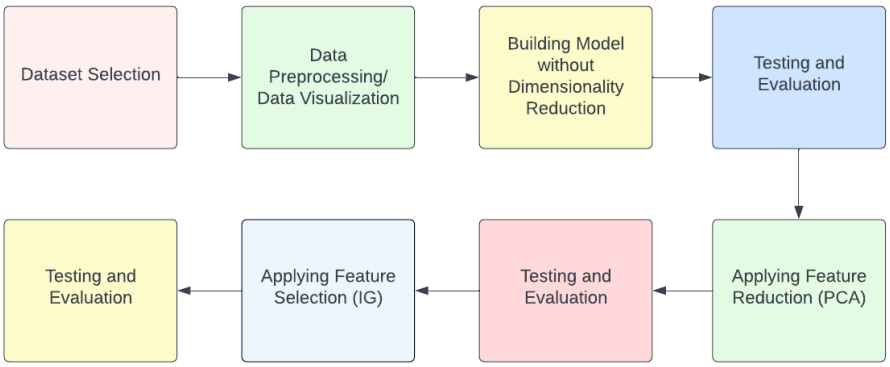


Fig 1.2 Flowchart showing Methodology for the Model

The field of cybersecurity is a constantly evolving landscape with new problems to solve every new day. Therefore, it is essential that there exist technological solutions to detect phishing attacks. With machine learning algorithms, classification of data has become very easy, and it is a great tool to use in the detection of phishing URLs. We discuss the methodology implemented to create a solution for phishing URLs detection using machine learning and the development of a Chrome Extension to use the best machine learning model in this section.

Firstly, we selected a database from Kaggle, "Web Page Phishing Detection Dataset" which contains 11430 data points, 87 extracted features for each data point under three classes: URL features, Page Content and External Features.

After selecting the dataset, the next step was initializing the database and pre-process the data. The dataset was cleaned, checked for null values, non-numerical labels were made numerical and ensured it was perfect for a machine learning model.

Data is best understood when visualized, hence we used visualization techniques to see the trends in data using different visualizations like scatter plots and heatmaps.

Next, we built machine learning models using six different classifiers: Logistic Regression, Random Forest, Naive Bayes, K-Nearest Neighbours, Support Vector Machines, and an Ensemble Stack. We built these models in three different situations: before any dimensionality reduction, after applying Principal Component Analysis, and after applying Information Gain feature selection.

We evaluated the performance of each model using various metrics such as accuracy, precision, recall, and F1-score. After evaluating the models, we selected the best performing model, which was the Random Forest model with 96.43% accuracy, 0.076 seconds test time, and 0.965 seconds train time. For creating a binary file for the model, we used the joblib library. We also wrote scripts for extracting features and hosted them on Flask, a Python-based micro web framework. We created an endpoint in Flask, which can be used to communicate with the model.

We used Postman API Platform to test the Flask endpoint, and once we were satisfied with the results, we hosted the Flask application on Railway app, a platform for hosting web applications on the cloud.

Finally, we integrated the Flask application on Railway with the Chrome extension, which allows the system to classify URLs in real-time. The extension automatically detects the active tab's URL and sends it to the Flask endpoint, which then interacts with the model and supplies the classification result.

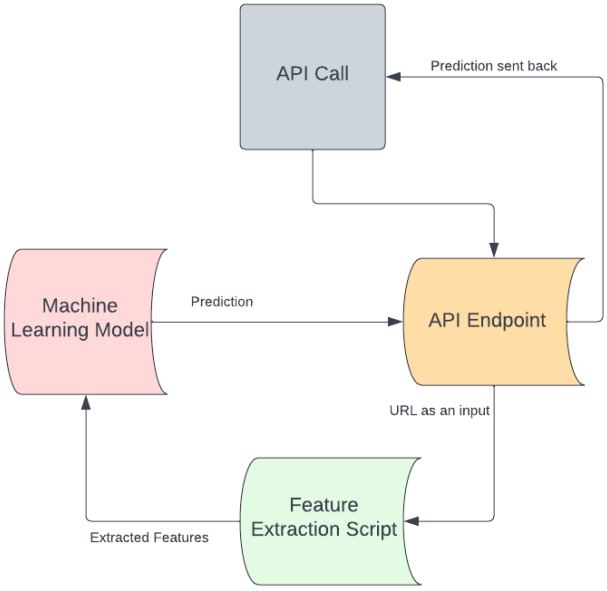


Fig. 1.3 The Architecture of the Flask App

.

In conclusion, the development of the phishing detection system involved several steps, from selecting the dataset to integrating the system with the Chrome extension. The system uses machine learning algorithms and has achieved high accuracy in detecting phishing URLs. The system's future scope includes adding more features, improving the model's accuracy, and scaling the system to handle more extensive datasets.

1.4.1 Initial Steps towards the Project

The first step towards the project was to set a suitable dataset that would best fit our requirements of having high-dimensional data so that we could perform various dimensionality reduction methods. After some comparisons and discussions, we selected a dataset from Kaggle [1], which had around 11400 data points with 87 features.

After obtaining the dataset, we uploaded the dataset into a Google Colab notebook. We used a big data analytics tool for python, PySpark to preprocess the data efficiently. We also used PyMongo, a python database management

tool, to store the dataset. Using this tool in our project helped us by letting us perform various data manipulation tasks on the dataset, which in turn leads to the improvement of any machine learning algorithm.

1.4.2 Data Pre-Processing and Visualization

Our project's data pre-processing and visualisation step intended to prepare the dataset for modelling by cleaning, transforming, and selecting features. At the start, we inspected the dataset for missing or null values and used appropriate approximation techniques to fill up the gaps. This made sure the dataset was finished and prepared for further processing.

To work with categorical data, we employed a label encoder function to transform non-numerical data into a numerical format. This helped to guarantee that the data was uniform throughout the dataset and could be evaluated more efficiently. To simplify the analysis and eliminate ambiguity in the data, we removed any unrelated columns from the dataset.

We visualised the dataset after cleaning and transforming it to gain insights and identify patterns and trends. To do this, we employed a variety of techniques, including a search for categorical features through the examination of unique values. If the number of unique values is less than 0.002%, the feature is classified as categorical. This assisted us in determining which features were continuous and which were categorical, which was crucial for further analysis.

Next, we created a heatmap to visualize the correlation between distinctive features in the dataset. It assisted us in finding which features were highly correlated with each other and which were not. While we did not drop any features based on the visualization, we could get a clear picture of how the features correlated to each other.

In order to learn about the distribution of features across target variables, we visualized the distribution of mean value of each feature against the target variable. We also visualized the graph for page rank against their count for each status value. There were some more interesting trends in the data that we found, but overall, this process assisted us in understanding how features correlate to their status values.

1.4.3 Building the Model

In building a machine learning model for phishing detection, a crucial step involves dividing the dataset into dependent and independent sets. The independent variables or features are the inputs to the model, while the dependent variable or label is the output, we want the model to predict. This is done using the train_test_split() function from scikit-learn, which splits the dataset into a testing set and a training set.

To ensure that the model can accurately predict new and unseen data, it is necessary to scale all the features of the training and testing datasets using a minmax scaler. Doing so ensures that all features have a similar scale and reduces the impact of outliers on the model's performance. In addition, to create the model, it is necessary to convert the independent sets into a one-dimensional array using the ravel() function from numpy.

Once the data is properly prepared, it is time to build the model. During this stage, a confusion matrix function is defined using seaborn's heatmap plot, which allows for easy visualization of the model's performance. Additionally, a model evaluation function is created using several evaluation metrics such as accuracy, precision score, recall, and f1 score. These metrics help assess the model's performance on the testing set and ensure that it is correct and robust.

1.4.4 Without Feature Selection

Then the following models are built before any feature selection/reduction.

Table 1.1. Models' Parameters

Model	Parameters
Logistic Regression	max_iter =1000, random_state = 2
Support Vector Model	c = 175.0
Decision Tree	default parameters
K Nearest Neighbors	n = 3
Random Forest Classifier	default parameters
Ensemble Stack	estimators = selected based on accuracy, final_estimator = LR

1.4.5 Principal Component Analysis

In order to further perfect the performance of the machine learning model, a technique called Principal Component Analysis was applied. This technique is statistical, and it reduces the number of features and creates principal components of features which help preserve the most essential information. We started by using a MinMaxScaler to standardize the dataset and then applying PCA to reduce the number of features from 87 to 20.

After reducing the number of features, the model became less complex and easier to interpret, while still keeping most of the relevant information. The resulting model was then evaluated using the same machine learning algorithms as before, and the performance was compared to the earlier models that used all 87 features.

1.4.6 Information Gain/Mutual Information

We used Information gain as a method for feature selection. IG basically gives the relevance of any feature to the target variable. To see if our problem could be solved by dropping irrelevant features, and the dependence of the status of a phishing URL on its features, it was a very suitable technique.

To apply Information Gain, we used the mutual_info_classif() function from scikit-learn's feature selection module. This function calculates the mutual information score of each feature in relation to the target variable and returns a list of scores. We then set a threshold of 0.07 to only select the features with a score above the threshold. In total, 22 features were selected using this method.

This feature selection technique was applied to all the models implemented in the first part of the project, allowing us to compare the performance of models with all features, models with reduced features using PCA, and models with features selected through Information Gain.

1.5 ORGANIZATION

1.5.1 External Interface Requirements

1.5.1.1 Hardware Requirements

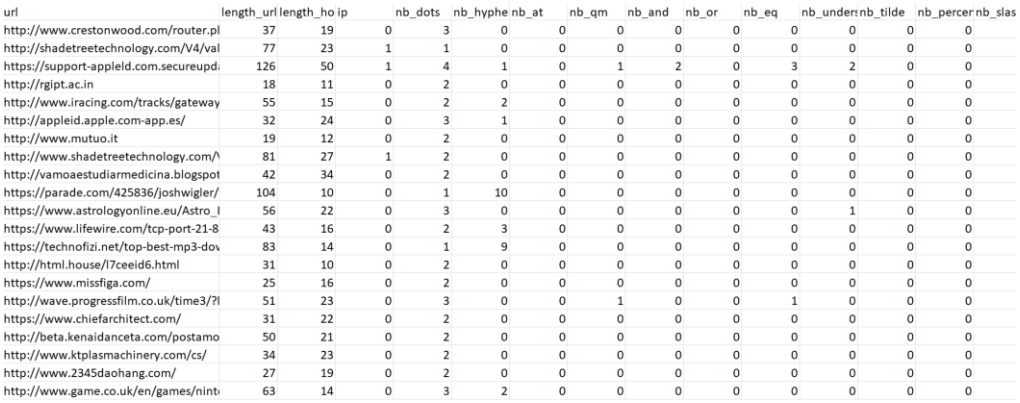
The system on which the following project is running has the following hardware specifications:

OS	macOS Monterey
CPU	Apple M2 chip with 8-core CPU and 8-core GPU
RAM	8 GB unified memory
Storage Type	512 GB SSD
RAM Type	Unified memory architecture
RAM Clock Speed	2133 MHz
Dedicated GPU	None (integrated graphics)

1.5.2 Software Requirements

Following software requirements have been found for the implementation of the phishing website detection system:

IDE	Visual Studio Code, Google Colab Notebook
Language	Python 3.10, JavaScript, HTML5, CSS
Libraries	NumPy, Pandas, PySpark, PyMongo, Seaborn Plot, Matplotlib, Joblib, Scikit Learn
Software/Services	Railway.app, Postman
Framework	Flask framework
Dataset	Web Phishing Dataset from Kaggle



url	length_url	length_ho	ip	nb_dots	nb_hyphe	nb_at	nb_qm	nb_and	nb_or	nb_eq	nb_under	nb_tilde	nb_percer	nb_slas
http://www.crestonwood.com/router.pl	37	19	0	3	0	0	0	0	0	0	0	0	0	
http://shadetreetechnology.com/V4/val	77	23	1	1	0	0	0	0	0	0	0	0	0	
https://support-appleId.com.secureupd	126	50	1	4	1	0	1	2	0	3	2	0	0	
http://rgipt.ac.in	18	11	0	2	0	0	0	0	0	0	0	0	0	
http://www.iracing.com/tracks/gateway	55	15	0	2	2	0	0	0	0	0	0	0	0	
http://appleid.apple.com-app.es/	32	24	0	3	1	0	0	0	0	0	0	0	0	
http://www.mutuo.it	19	12	0	2	0	0	0	0	0	0	0	0	0	
http://www.shadetreetechnology.com/\	81	27	1	2	0	0	0	0	0	0	0	0	0	
http://vamoaestudiarmedicina.blogspot	42	34	0	2	0	0	0	0	0	0	0	0	0	
https://parade.com/425836/joshwigler/	104	10	0	1	10	0	0	0	0	0	0	0	0	
https://www.astrologyonline.eu/Astro_I	56	22	0	3	0	0	0	0	0	0	1	0	0	
https://www.lifewire.com/tcp-port-21-8	43	16	0	2	3	0	0	0	0	0	0	0	0	
https://technofizi.net/top-best-mp3-dov	83	14	0	1	9	0	0	0	0	0	0	0	0	
http://html.house/l7ceeid6.html	31	10	0	2	0	0	0	0	0	0	0	0	0	
https://www.missfiga.com/	25	16	0	2	0	0	0	0	0	0	0	0	0	
http://wave.progressfilm.co.uk/time3/?l	51	23	0	3	0	0	1	0	0	1	0	0	0	
https://www.chiefarchitect.com/	31	22	0	2	0	0	0	0	0	0	0	0	0	
http://beta.kenaidanceta.com/postamo	50	21	0	2	0	0	0	0	0	0	0	0	0	
http://www.ktplasmachinery.com/cs/	34	23	0	2	0	0	0	0	0	0	0	0	0	
http://www.2345daohang.com/	27	19	0	2	0	0	0	0	0	0	0	0	0	
http://www.game.co.uk/en/games/ninte	63	14	0	3	2	0	0	0	0	0	0	0	0	

Fig 1.4 Screenshot of Dataset Snippet

1.5.3 Overall Description

1.5.3.1 Product Perspective

The world today is characterized by rapid technological development and unlimited connectivity. leading to the increase of security breaches and data loss. The rate of internet related threats like cybercrimes, fraudulent commerce, social engineering and phishing among many others have been growing at a high rate owing to the exponential growth of users of the internet and online

services. With human weakness being their primary weapons, cybercriminals develop new techniques and more sophisticated tools for performing cybercrime, hence posing a huge threat to the users of online services.

Phishing has become one of the most common types of cybercrimes, where end users are targeted using fraudulent links masked to seem legitimate causing a significant financial and data loss. Therefore, detecting and preventing phishing websites/urls have become an important field in the field of cybersecurity. Many researchers and scientists have implemented machine learning techniques to detect phishing URLs and create efficient solutions for the problem at hand.

Amidst these developments, our project aims to contribute to this field by developing a machine learning-based solution for phishing URL detection. Our approach analysing a large dataset of URLs to find features that are indicative of phishing attacks. We then train machine learning models to detect these features and classify URLs as either legitimate or phishing.

The overall goal of this project is to supply a reliable and efficient tool for detecting phishing attacks and preventing users from falling victim to these malicious schemes put on by people who want to exploit uneducated internet users.

1.5.3.2 Product Functions

With all the problems revolving around the cybersecurity space, it becomes a mandate for all scholars in this field to work on a product that helps the greater population on the internet to remain safe and calm as they browse the web.

With our product, we believe we can make a change which albeit small, can contribute to the greater goal of preventing phishing attacks on uninformed users. While our product may not be perfect, we have done our best in developing it and we believe it can serve a good cause for a safe browsing experience.

Our core product is a Chrome Browser Extension that checks the URL in the address bar of the browser. The extension executes every time the user refreshes a webpage or when a new page is loaded. The extension detects any URL for phishing by all extracting 87 features (aforementioned) in real time and alerts the user if the page they are visiting is suspected to be a phishing URL.

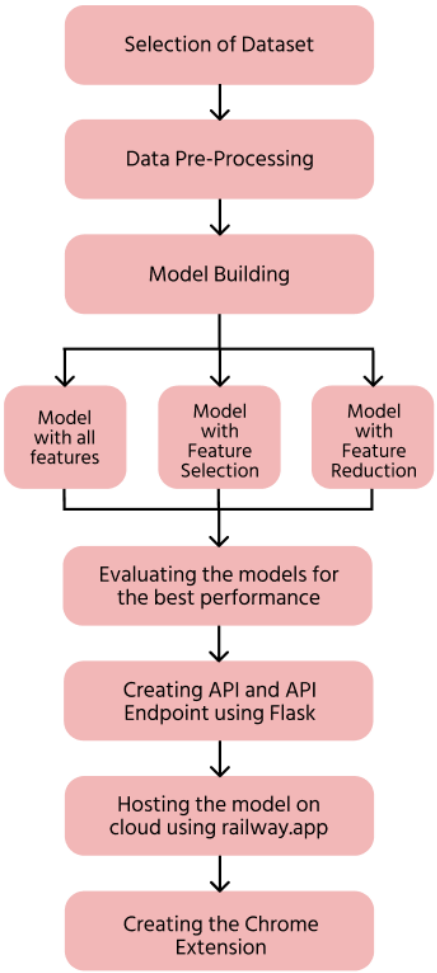


Fig. 1.5 Overall Pipeline of the Model

1.5.3.3 User Classes and Characteristics

We tried to classify the URLS into phishing URLs and the safe URLs:

- **Phishing URLs**: The victim is sent a link to an authentication page that seems legitimate but when you log in, the credential of the user is collected, giving the scammers access to the real account.
- **Safe URL**s: The URLs with minimum risk from hackers.

1.5.3.4 Design and Implementation Constraints

We had a dataset of 11,430 data points with 87 dimensions for our phishing detection project. The dataset included 56 URL-based features, 24 content-based features, and 7 external service query features. Half of the sites in the dataset were legitimate, and the other half were scams. But, we knew that with new techniques and methods being developed by hackers every day, our model might not always be able to accurately predict the output.

When we were designing and implementing our phishing detection extension, we had to balance accuracy and performance. We wanted the extension to accurately detect phishing URLs, but we also wanted it to be lightweight enough not to slow down browsing.

One of the challenges we faced was writing scripts for feature extraction. These scripts needed to be accurate, efficient, and easy to maintain. They also required a deep understanding of the dataset and which features were relevant to phishing detection.

One of the biggest challenges we faced was finding a reliable dataset of active phishing URLs to test for unsafe URLs. Most of the public datasets had outdated or inactive URLs, making it hard to train our model accurately. Unfortunately, there weren't any open-source datasets or APIs we could use to get active phishing URLs, so we had to manually scrape the web for a limited number of them.

CHAPTER 2: LITERATURE SURVEY

2.1 LITERATURE REVIEW

This chapter presents the most recent research works about the classification of phishing website data using machine learning.

In [6] on detection of phishing, the authors underlined that when new approaches to counter different phishing attempts are put out, attackers adapt their strategy to get around the newly suggested phishing method. Consequently, it is strongly recommended to adopt hybrid models and machine learning-based techniques.

An approach based on human behaviour when exposed to rogue websites was proposed by Rao and Pais [7]. With the inclusion of heuristic screening, this human habit of inputting false credentials is automated. Using this method, they were able to achieve an accuracy of 96.38%.

The authors of [8] examined many phishing webpage characteristics and narrowed it down to the top 19 features. They employed support vector machine (SVM), random forest (RF), neural network, logistic regression, and naive bayes (NB) for categorization of phishing webpages after extracting the features from the webpage and achieved about 99% accuracy.

In the research [9] also compared the model building times before and after feature selection. They found out that building times for models were significantly decreased, with the slightest change in accuracy. They also concluded that Random Forest classification technique was the method that had the most accuracy in both times before selecting the features and after. They urge researchers to use ensemble methods since they have the highest evaluation scores.

Another research [10] compared URL, Page, and HTML features, where they found URL features were the most effective in detection for phishing. They also conclude with deciding that RF is the best method to detect phishing, an ensemble technique.

In order to increase the effectiveness of their system, Almseidin et al. [11] published a research paper in which they used several ML algorithms and feature selection techniques. The studies used a phishing dataset with 48 attributes that included 5000 legitimate and 5000 malicious websites. They concluded that the accuracy is best for the RF algorithm with only 20 features.

Using classification methods and characteristics from natural language processing (NLP), Sahingoz [12] suggested a real-time system (NLP). According to experimental findings, the RF classifier, which uses NLP-based features, has the greatest accuracy of 97%.

Using methods for features selection, Zamir [13] examined the characteristics of the phishing data. They suggested two characteristics by fusing different traits. For the purpose of finding phishing websites, they employed a variety of machine learning and stacking approaches.

Using URL-based properties, Basnet and Doleck [14] suggested a heuristic technique. A collection of 138 characteristics that were retrieved from 31,000 malicious and 16,000 phishing websites was used in the experiments. The four categories of these 138 characteristics are search engine, reputation, lexical, and keyword based. For the goal of categorization, they used seven distinct machine learning methods. The most correct method was random forest.

Khanna and Gupta [15] highlighted the use of machine learning algorithms in detecting phishing attacks owing to their high accuracy and efficiency while Islam et al. [16] proposed for usage of an ensemble learning approach that

combined several machine learning models, leading to higher performance scores. Zhao and Liu [18] have also done a similar research, although they included feature selection with ensemble learning, and also by achieving high accuracy scores on a huge dataset.

Roy and Singh [17] compared various machine learning algorithms for phishing detection and concluded that decision tree algorithm had the best performance among others. Yadav et al. [19] took a novel approach towards detecting phishing attacks through Convolutional Neural Networks (CNNs) with a high detection rate. In Singh et al. [20] research, they found out ensemble-based techniques were more accurate than other models comparatively.

Yang et al. [21] used both machine learning and website features to detect for phishing and achieved high accuracy scores. In [22] Dam et al. introduced phishing detection methods that used different models like SVM, LR, DT and Artificial Networks (ANNs). Kumar et al. [23] proposed that the random forest algorithm when implemented after feature selection gave high phishing detection accuracy.

2.2 PROJECTED APPROACH

This project is aimed at using big data analytics tools and applying machine learning. The preprocessing step includes the use of a big data tool PySpark (Apache Spark). After processing, models will be fit to their train and test data sets in 6 different algorithms: LR, SVM, DT, KNN, RF and ES.

The models will be evaluated in three scenarios: with all features included, after applying PCA, after applying IG. The best model from the evaluated list will be used for a Chrome Extension. To create the Chrome Extension which checks URLs for phishing in real time through a machine learning model, an API will

be developed. The extension would use the API to live extract URL features and check their status for phishing.

This approach would enable us to build an effective phishing detection system using machine learning and big data tool.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 DESIGN OF THE PROJECT

The project is divided into three parts while developing the machine learning model. The information on each part is given below:

3.1.1 Before Feature Selection

The dataset used in the project was sourced from a MongoDB database and loaded into PySpark for analysis through Google Colab. The dataset consists of 11,430 data points with 87 dimensions, engineered to cater to the need of phishing detection.

To prepare the dataset for machine learning algorithms, it underwent a series of pre-processing steps. Firstly, the dataset was converted into a Spark data frame, which is a distributed collection of data organized into named columns. This allowed for efficient processing of large datasets across a distributed computing environment.

Next, categorical variables in the dataset were handled using one-hot encoding, a technique that converts categorical variables into binary features that can be easily used in machine learning models. The dataset was also checked for missing values, and any missing values were imputed using the mean value of the respective feature.

After pre-processing, the dataset was scaled using a min-max scaler, a normalization technique that scales features to a range between 0 and 1. This helped to ensure that features with larger ranges did not dominate the machine learning algorithms.

The pre-processed dataset was then split into independent and dependent variables. The independent variables are the features that the machine learning algorithms use to predict the dependent variable, which is the target variable that shows whether a website is legitimate or phishing. The dataset was further split into training and testing datasets using an 80-20 split.

The dataset after pre-processing was then fitted into six machine learning models LR, KNN, SVM, DT, RF and ES. Each model was trained on the training dataset from the split and evaluated on the testing dataset from the split.

We evaluated the performance of these models and recorded the results, with all the performance metrics plus the test and train times.

```
x[:8]
```

	avg_word_host	avg_word_path	avg_words_raw	brand_in_path	brand_in_subdomain	char_repeat	dns_record	domain_age	domain_in_brand	domain_in_title	...	sho
0	7.0	4.500000	5.750000	0	0	4	1	-1	0	0	...	
1	19.0	14.666667	15.750000	0	0	4	0	5767	0	1	...	
2	8.4	8.142857	8.250000	0	0	2	0	4004	0	1	...	
3	5.0	0.000000	5.000000	0	0	0	0	-1	0	1	...	
4	5.0	7.000000	6.333333	0	0	3	0	8175	0	0	...	
5	4.5	0.000000	4.500000	0	0	3	0	-1	0	1	...	
6	4.0	0.000000	4.000000	0	0	3	0	7529	0	0	...	
7	11.0	14.666667	13.200000	0	0	8	0	5767	0	1	...	

Fig 3.1. Before MinMax Operation

```
[93] scaler = MinMaxScaler()

     x_sc =  scaler.fit_transform(x)
```

```
x_sc[:8]
```

```
array([[1.57894737e-01, 1.80000000e-02, 2.97029703e-02, 0.00000000e+00,
        0.00000000e+00, 2.73972603e-02, 1.00000000e+00, 8.53639609e-04,
        0.00000000e+00, 0.00000000e+00, 1.54207174e-03, 1.00000000e+00,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 7.14285714e-02,
        1.53468386e-02, 2.85714286e-02, 8.00000000e-01, 0.00000000e+00,
        1.63934426e-01, 7.23763571e-03, 1.08827086e-02, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 8.69565217e-02, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00, 3.64885168e-03, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 3.22580645e-02, 0.00000000e+00, 0.00000000e+00,
        1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 5.00000000e-01,
```

Fig 3.2. After performing the MinMax Operation

3.1.2 Principal Component Analysis

```
# print the explained variances
print("Variances (Percentage):")
print(pca.explained_variance_ratio_ * 100)
print()
```

```
Variances (Percentage):
[1.61165758e+01 1.31189761e+01 8.55395378e+00 6.72554271e+00
 5.77818405e+00 4.73446873e+00 4.21368396e+00 3.85719558e+00
 3.79514777e+00 3.54698730e+00 2.83673255e+00 2.72028819e+00
 2.24411081e+00 2.13536330e+00 2.04711536e+00 1.80349624e+00
 1.66030866e+00 1.54751866e+00 1.33152339e+00 1.09943583e+00
 1.02444950e+00 1.00367368e+00 9.43615709e-01 6.43436289e-01
 6.03324131e-01 5.67055679e-01 4.83115326e-01 4.57474565e-01
 4.19901377e-01 3.90600896e-01 3.64110222e-01 3.43668742e-01
 2.91278052e-01 2.08952439e-01 2.04036006e-01 1.80152840e-01
 1.69280654e-01 1.67212807e-01 1.53515780e-01 1.42921058e-01
 1.31602427e-01 1.18648360e-01 1.09811877e-01 9.31732155e-02
 9.09438568e-02 7.44727808e-02 6.68659173e-02 6.24818590e-02
 5.92754316e-02 5.57102472e-02 5.09526831e-02 4.25259280e-02
 3.93125701e-02 3.80863543e-02 3.66679125e-02 3.12127739e-02
 3.00687418e-02 2.72518581e-02 2.50783616e-02 2.17206507e-02
 1.94429760e-02 1.84718422e-02 1.80227019e-02 1.74669351e-02
 1.39470284e-02 1.34997554e-02 1.14001747e-02 1.11581137e-02
 9.69286715e-03 7.64798562e-03 5.51841846e-03 4.92096221e-03
 4.43515709e-03 3.23395968e-03 2.88739809e-03 2.33493031e-03
 1.05441258e-03 5.18071106e-04 9.80021640e-05 6.49165605e-32
 6.49165605e-32 6.49165605e-32 6.49165605e-32 6.49165605e-32
 6.49165605e-32]
```

Fig 3.3. Variance of each feature (%)

PCA is a method for reducing the number of dimensions in the data. In our study, we scaled our dataset using a min-max scaler before applying PCA on it. The use of PCA makes it easier to identify the key characteristics or elements that are primarily responsible for the variation in the data. The goal is to maintain the majority of the data while reducing the number of features in the dataset.

We used PySpark to construct the covariance matrix, eigenvectors, and eigenvalues of the matrix after scaling the dataset. The highest data variance's directions are represented by the eigenvectors, and its size is represented by the eigenvalues. Based on the cumulative explained variance ratio, we identified the top 20 components.

We were able to reduce the dimensionality of the dataset from 87 to 20 by using PCA, which greatly reduced the computation time required for model training and prediction. The reduced dataset was then divided into training and testing datasets and fitted into the six previously mentioned models.

In summary, PCA was applied to our dataset with the purpose of extracting the most important features that contribute the most to the variation in the data. This helped to reduce the number of features in the dataset and significantly reduced the computation time needed for model training and prediction. The reduced dataset was then used to train and evaluate our models.

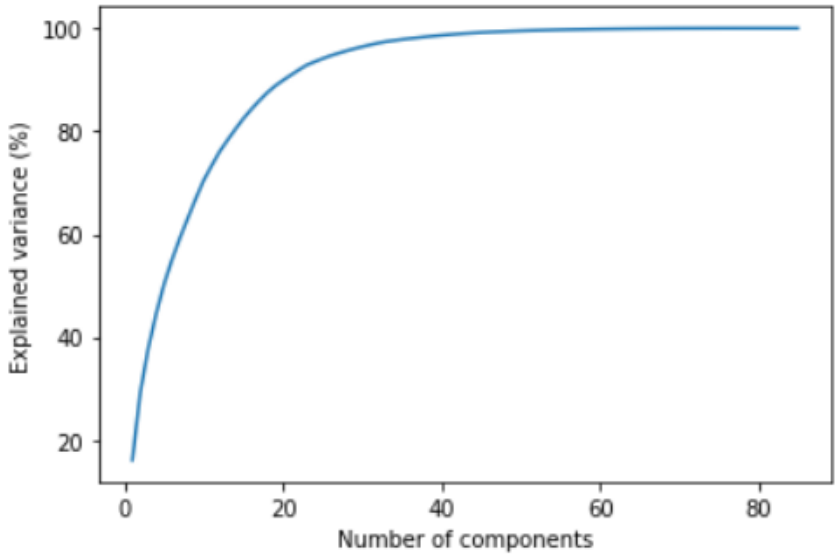


Fig 3.4: Cumulative Variance of all the features

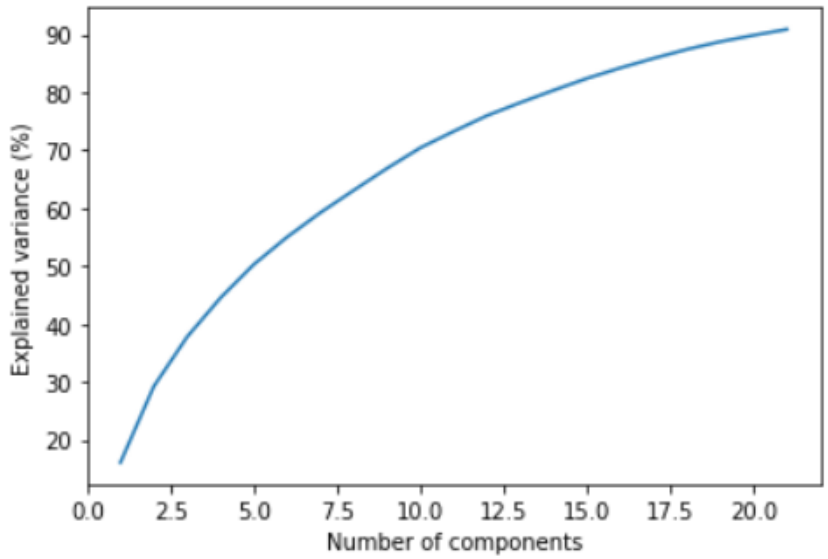


Fig 3.5: Cumulative Variance of Top 21 Components

```
Top 4 most important features in each component
================================================
Component 0: ['ratio_extErrors', 'external_favicon', 'domain_with_copyright', 'ratio_extHyperlinks']
Component 1: ['empty_title', 'ratio_extHyperlinks', 'ratio_extErrors', 'domain_registration_length']
Component 2: ['domain_registration_length', 'empty_title', 'domain_in_brand', 'external_favicon']
Component 3: ['external_favicon', 'iframe', 'port', 'domain_age']
Component 4: ['domain_registration_length', 'domain_with_copyright', 'domain_in_brand', 'nb_underscore']
Component 5: ['ratio_extHyperlinks', 'right_clic', 'port', 'iframe']
Component 6: ['iframe', 'nb_star', 'domain_age', 'sfh']
Component 7: ['domain_in_brand', 'right_clic', 'length_words_raw', 'external_favicon']
Component 8: ['external_favicon', 'domain_registration_length', 'port', 'domain_age']
Component 9: ['sfh', 'port', 'domain_in_brand', 'domain_age']
Component 10: ['nb_star', 'length_words_raw', 'ratio_extHyperlinks', 'nb_underscore']
Component 11: ['iframe', 'ratio_extErrors', 'length_words_raw', 'ratio_intHyperlinks']
Component 12: ['punycode', 'length_words_raw', 'empty_title', 'domain_age']
Component 13: ['punycode', 'sfh', 'ratio_intHyperlinks', 'right_clic']
Component 14: ['sfh', 'submit_email', 'domain_registration_length', 'empty_title']
Component 15: ['submit_email', 'links_in_tags', 'domain_age', 'shortest_words_raw']
Component 16: ['ratio_extErrors', 'nb_star', 'tld_in_path', 'suspecious_tld']
Component 17: ['links_in_tags', 'domain_with_copyright', 'length_words_raw', 'nb_underscore']
Component 18: ['nb_underscore', 'iframe', 'ratio_extErrors', 'submit_email']
Component 19: ['dns_record', 'suspecious_tld', 'tld_in_path', 'ratio_extErrors']
Component 20: ['nb_underscore', 'tld_in_path', 'domain_age', 'suspecious_tld']
```

Fig 3.6: Top 4 Features of Each Component

3.1.3 Information Gain

We used the mutual information (MI) score feature selection approach in addition to the PCA method previously discussed to increase the precision of our classification of phishing websites. The amount of information that a feature gives about the target variable is measured using MI, a feature selection technique that is frequently utilised. The dataset was trimmed down to 22 features after the most informative features were chosen, those with scores above 0.07.

We may avoid the dimensionality curse and enhance the performance of our models by reducing the dimensionality of the dataset using feature selection techniques like MI and PCA. This aids in lowering overfitting and increases the models' generalizability.

The new dataset, which was once more split into separate training and testing sets, was fitted with the same models that had previously been employed. We were able to evaluate how the feature selection techniques affected the accuracy and potency of our models as a result. In order to determine if the models could distinguish between legitimate and phishing websites, they were evaluated based on a variety of performance parameters, including accuracy, precision, recall, and F1-score.

The performance of our models can be significantly improved by using feature selection methods like MI and PCA, which also offer a more effective and precise method for classifying phishing websites.

3.1.4 Selecting the Model

We assessed a number of models, including Logistic Regression, Decision Tree, Naive Bayes, and Random Forest after pre-processing and feature selection. The Random Forest model, which had a train time of 0.965 seconds and a test time of 0.07 seconds, outperformed the others with a test set accuracy of 96.43%. Random Forest is an ensemble learning technique that makes the final prediction by combining the results from various decision trees.

It is renowned for its resistance to overfitting and capacity for handling large-scale, multidimensional data. The model performed better as a result of our feature selection process's contribution to reducing the data's dimensionality and choosing the most crucial characteristics for the model. So, for our phishing detection system, we decided to employ the Random Forest model.

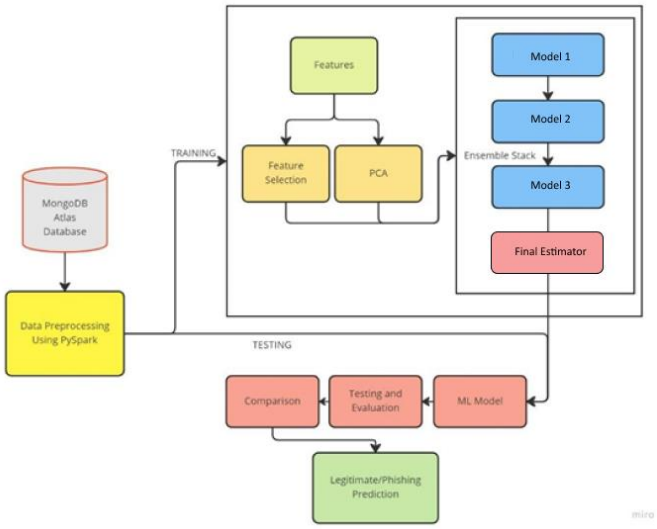


Fig. 3.8. Architecture of the ML Model

3.1.5 Creating the Chrome Extension

For our study, we made advantage of the Kaggle dataset, which already had certain feature extraction tools written. To function with the most recent version of Python, many of the scripts needed to be changed because they were deprecated.

In order to extract all 87 features from any given URL and utilise them to train our machine learning models, we updated the deprecated scripts included with the dataset. Information about the URLs, the webpage's content, and external features were among these features.

a. Flask for API Endpoint

Our phishing detection model's API endpoint was built using the Python web framework Flask. Flask provided us with a simple interface for managing HTTP requests and responses as well as the capacity to build the necessary paths and methods for our API.

Our trained machine learning model was saved as a binary file using the joblib package, which we later imported into our Flask application. We created a POST route that would accept a JSON payload containing the features of a brand-new web page and would then return the page's estimated phishing chance. With Flask, we had no trouble deploying our API to Railway.app, a cloud hosting service.

b. Railway.app for Hosting the API on the Cloud

Our Flask application and machine learning model were quickly deployable thanks to the cloud hosting service Railway.app. We started a brand-new project on Railway.app and linked it to the Flask application code repository on our GitHub account. We received a public URL from Railway.app for our API

endpoint, which we included into our Chrome extension to make calls to the cloud-based Flask programme.

c. Postman API Platform for Testing

Prior to cloud deployment, we tested our Flask API using Postman, a well-known API testing tool. We were able to quickly send HTTP calls to our API and observe the responses with the help of Postman. We tested our API's accuracy in estimating phishing probability for brand-new web pages using both legitimate and incorrect feature inputs.

d. The Chrome Extension

We created a Chrome extension to allow users to quickly and easily check the phishing probability of any web page they are currently browsing. The extension consists of a popup window that appears when the user clicks on the extension icon in their browser. We used Manifest 3 to define the extension's properties and permissions. We also used a background.js script to manage communication with the Flask API endpoint hosted on Railway.app. Finally, we used popup.html and internal CSS to design the layout and appearance of the extension popup.

3.1.6 API Structure

Our project's API structure is a crucial component of its functionality. We've created a Flask app to act as a server that receives a URL string in the request body via a POST request. Our binary machine learning model, which we trained to detect phishing URLs (the RF model), processes the URL to determine whether it's legitimate or malicious based on the extracted features using the script.

After processing the URL, the app then returns the status of the URL. The possible statuses of the URL are 1, 0, and -1. The 1 status indicates that the URL

is an active phishing webpage, 0 indicates that the URL is an active safe webpage while -1 indicates the URL is inactive or inaccessible.

We designed the endpoint for our API using Flask, a web framework that enables us to build and deploy web apps easily and quickly. Flask makes handling HTTP requests straightforward, which is ideal for creating REST APIs. Our endpoint is designed to accept POST requests, which is the most common method used to send data to a web server. This allows us to data securely over the internet.

Our endpoint is hosted on railway.app, a project hosting and deployment site. Railway.app enables us to deploy our project rapidly and without needing to worry about server setup or configuration. This enables us to concentrate on developing our project, rather than infrastructure.

Overall, our project's API structure offers a simple and effective way for users to verify URLs' safety when browsing the internet. Our binary machine learning model quickly identifies phishing sites, safeguarding users against malicious attacks. Flask provides an elegant solution for handling HTTP requests, and railway.app enables us to deploy our project with ease.

3.1.7 Apache Spark on Small Datasets

Apache Spark is a powerful tool that's commonly used for large-scale data processing, including detecting phishing URLs for Big Data Security Analytics. However, since our project had a relatively small dataset of only 3.4 MB, using Spark in a major role wasn't practical. For small datasets, it can actually be more efficient to use simpler tools for data processing since the overhead of using Spark can outweigh the benefits. Despite this, we still used Spark's PySpark framework for pre-processing our data, even though it turned out to be slower than expected due to our incorrect usage. We did this because we wanted to fulfil the requirement of using a big data tool in our project.

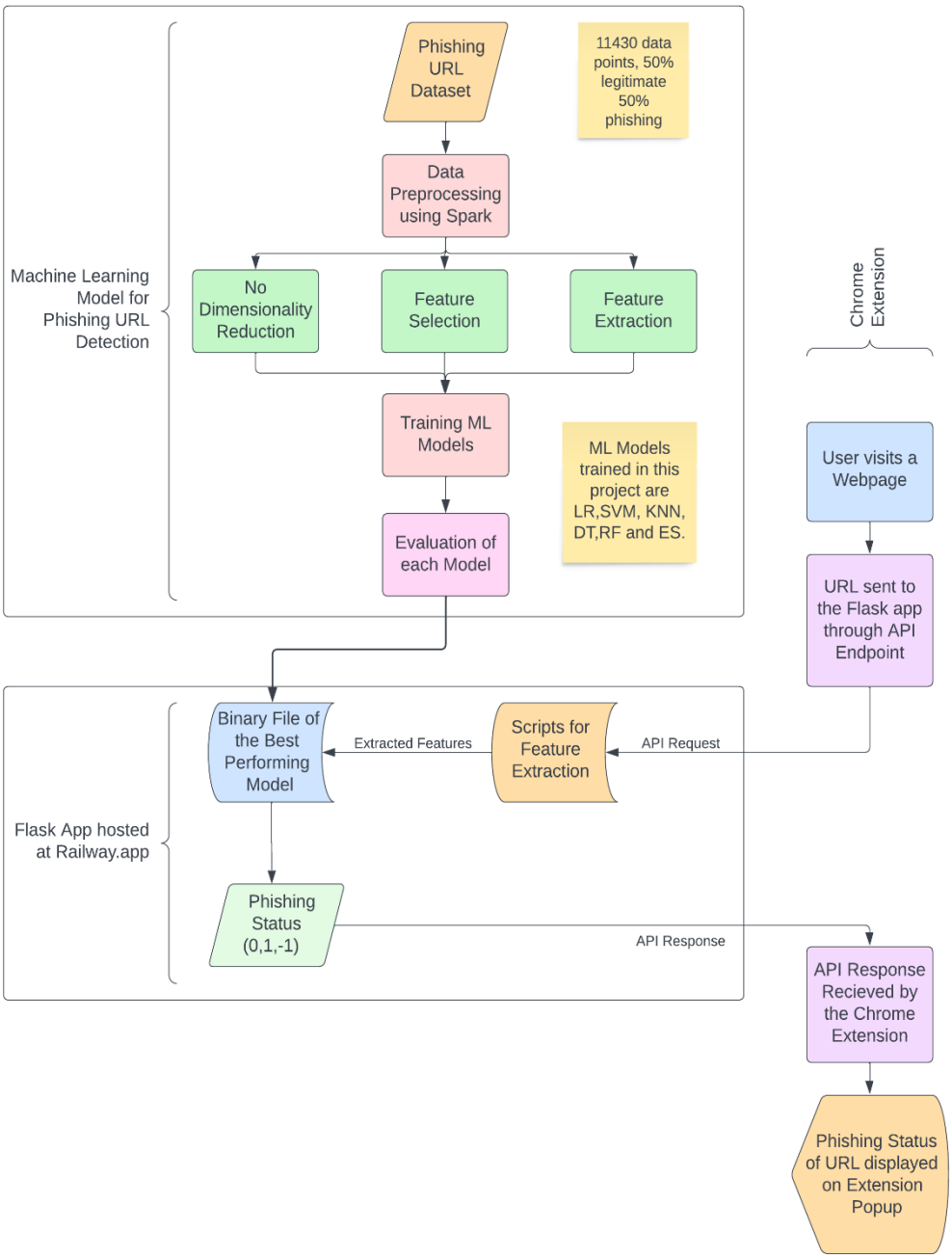


Fig 3.9 Block Diagram of the Project

3.2 ANALYSIS AND DEVELOPMENT OF ALGORITHMS

3.2.1 Machine Learning Models

3.2.1.1 Logistic Regression

Logistic Regression is a type of supervised learning in machine learning. It is used to decide the likelihood of a binary event if it will occur (yes/no). The use of machine learning to predict if a stock price will increase or decrease is an example of logistic regression. Binary classification implies that there are only two possible outcomes to an event, in the example of the stock price above, it should either increase or decrease.

In the example, the likelihood that the stock price of a share will change may depend on the company's reputation, financial state of the people, trust of public etc. These independent variables may have an impact on the stock price of the share of a company.

a. Where is Logistic Regression used?

Logistic regression is majorly used in binary classification of data where it classifies a data into a YES or NO class.

Logistic regression is used in:

- the sports sector to decide whether a team is likely to win or lose.
- the trading sector to decide if a share price goes up or down.
- News verification to decide if any news is fake or legitimate.

b. The Mathematics Behind the Algorithm

The probability is constantly between 0 (does not occur) and 1 (happens). Using our Phishing Detection example, the chance of the URL being legitimate and not legitimate will total to 1 in the event of binary categorization. In logistic regression, probability is calculated using the logistic function or the sigmoid function. A straightforward S-shaped curve called the logistic function is used to translate data into a value between 0 and 1.

The equation representing logistic regression is:

$$h\theta(x) = \frac{1}{1 + e^{-(\beta o + \beta 1 X)}}$$

where:

$h\theta(x)$ is the output of the logistic function, where $0 \leq h\theta(x) \geq 1$

$\beta 1$ is the slope.

βo is the y-intercept.

X is the independent variable.

3.2.1.2 K-Nearest Neighbours

A supervised machine learning (ML) technique known as K-nearest neighbours (KNN) can be used for both classification and regression problems. However, the main application in the industry still stays for mostly for classification and prediction problems. The next two characteristics would accurately describe KNN:

- KNN is a slow learning algorithm (lazy learner) because it does not have an allocated training time and makes use of the complete data during classification for learning.
- Since there are no assumptions made about the data, K Nearest Neighbours method is also parameter free.

a. How does this Algorithm work?

The K-nearest neighbours (KNN) method predicts the values of a new data to be predicted based on how alike their features are which resembles that the datapoint's class will be predicted based on how similar they are with the existing data in the training dataset. By the help of the processes below, we may understand how the algorithm functions in steps:

1. Load Training and Test dataset
2. Choose a value of K (any integer, preferably odd)
3. Calculate distance with a suitable norm (L1: Manhattan, L2: Euclidean, and Hamming Distance)
4. Sort data points based on distances in ascending order
5. Choose the first k number of rows in the sorted array.
6. Assign a label (0 or 1) to the data.

The distances can be calculated by substituting p for 1 (Manhattan Distance) and 2 (Euclidean Distance) in the Minkowski Distance equation. n is the number of data points, x and y are the values of the data. The Minkowski Distance equation for KNN is:

$$d = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

3.2.1.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a popular machine learning technique that can generate highly accurate results for both regression and classification applications. It is, nevertheless, more efficient with classification difficulties. It is a supervised learning method that predicts the class of a given set of data points. It is used for linear and non-linear classification, regression, and outlier detection.

The main idea of SVM is to determine the best line, known as the hyperplane, that splits the data into numerous groups. The hyperplane is designed in such a way that the margin between the two classes is maximised. The margin in SVM is the separation between each class's closest data points and the hyperplane. The class distinction is better the larger the margin.

By translating the input data into a higher-dimensional space, the kernel trick, a mathematical approach used by the SVM algorithm, makes it simple to segregate the data points into the two classes. In this higher-dimensional space, the input variables are combined linearly to form the hyperplane. SVM is able to manage complex and non-linear decision boundaries as a result.

The closest data points to the hyperplane are called support vectors, which are then utilised to establish the hyperplane. These points are utilised to calculate the margin and act as reference points for building the hyperplane. In order to effectively categorise the data points while maximising the margin, SVM searches for the ideal hyperplane.

To sum up, SVM is a strong and flexible machine learning technique that is frequently used for classification tasks. It operates by locating the ideal hyperplane that divides the data into various classes using support vectors.

Because it can handle non-linear decision constraints and is immune to noise and outliers, SVM is a preferred option in many applications.

a. Hyperplanes and Support Vectors

Hyperplanes: The user's task is to choose the best line that will aid in classifying the data in a dataset with n features because more than one line can frequently be used to divide the data points presented in the graph. The hyperplane is this line. The hyperplane is always built with the highest distance between the classes formed by the data points on a graph.

Support Vectors: These are the vectors that stay nearest to the hyperplane, so it has a high effect on the location of the hyperplane.

b. Working of Support Vector Machine

Let us use the example of the below image, which shows two classes, class A: ball, and class B: pyramid, to better understand how SVM functions. We now want to use the SVM algorithm to decide the ideal hyperplane that separates the two classes.

The data points are all read by the Support Vector Machine, and then creates a line that separates the classes from each other. This line is the hyperplane or even called as a decision boundary. All data points that fall into class A in Fig. 3.10 are identified as balls and the points in the other class are pyramids.

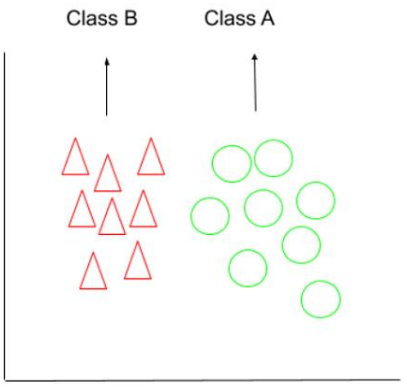


Fig 3.10 Class A and B

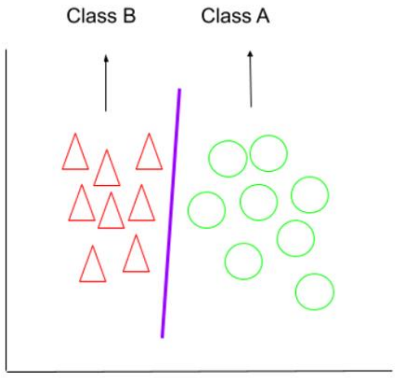


Fig 3.11 Labelled Data

One may see that there are several hyperplanes that can divide the data points, but as mentioned earlier the best hyperplane is the one that is the furthest distance from the data points in both classes. Hence, the algorithm searches for the best one.

Different dimensions are possible, and they solely depend on the features we have. When there are more than three features, it is difficult to visualize.

Fig 3.12 Class A & Class B

Looking at Fig. 3.12 let us consider that there are two classes differentiated based on colour, where the first class is for red colour and the second one for yellow.

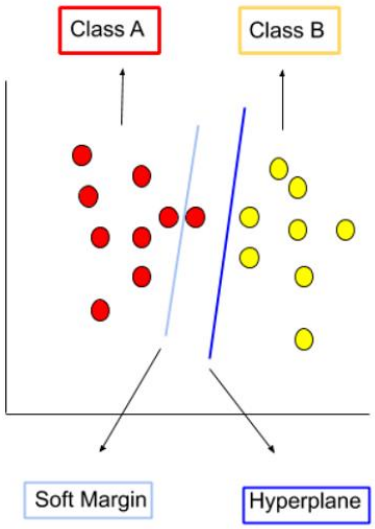


Fig 3.13 Hyperplane and a Soft Margin

When a soft margin is distributed the work of hyperplane, some data points are classified incorrectly. It also gets stuck in a cycle of balance and imbalance by

moving itself to prevent more incorrect classifications and trying to widen the gap between the support vectors.

c. Advantages of using of SVM

- Since it depends on support vectors and is independent of the data points, the model gains high stability.
- Outliers do not affect the performance of the model.
- There are no assumptions on the dataset unlike other ML algorithms.
- Numerical predictions can be easily classified by SVM.

d. Disadvantages of using SVM

- Blackbox method
- There is a good chance of an overfitting problem occurring.
- It requires a relatively higher computational power as compared to other models.

3.2.1.4 Decision Tree

Decision Tree algorithm is a type of supervised learning algorithm. This algorithm is designed to cater to both classification and regression problems easily.

The algorithm learns the rules that derive from data that it has worked with earlier and builds a model that can classify a datapoint.
The decision making of the algorithm stems from the root of the decision tree, where it tries to predict the class using the feature values stored in it with the features that is in the dataset. The point then moves to the branch that correlates to the value then to the node that decides what class it belongs to following the prediction rules.

a. How a Decision Tree Works

The ability of selecting and making divisions in the dataset is of significant importance to the performance of the model. There are different decision trees for both classification and regression.

The decision to split a node in the tree into multiple sub nodes is accompanied by a number of techniques used by the decision tree. The similarity of freshly formed derived nodes is further increased when another node is derived from it. The tree forms the nodes by going through all the features available first before deriving new nodes.

The method we use in this project is the ID3 method, which is an extension of D3, from which ID3 is derived from ID3 algorithm, in the absence of backtracking, forms decision trees by traversing the space of highly homogenous branches due to it being a top-down greedy search algorithm (An algorithm that always selects the best choice).

b. ID3 Algorithm Steps:

- Set S at the beginning is the root node.
- In each iteration of the unused attribute of set S, this algorithm finds the entropy (H) and information gain (IG).
- The feature is then selected based on entropy and information gain, where a low entropy and high information gain is desired.
- The selected feature is then used to divide the subset S.
- Only features that have not been selected earlier are considered because the algorithm iterates over all the subsets one by one.

c. Information Gain as an Attribute and Underlying Mathematics

Information gain is a statistical method that measures how much a feature is efficient in classifying a data point into the original class. A feature that gives minimum entropy and largest IG is essential for the model to perform well.

Information gain is inversely proportionate to entropy. IG calculates the difference in entropy before and after splitting a set based on the selected feature. The method ID3 (Iterative Dichotomiser) of decision tree uses the information gain.

Mathematically, IG is represented as:

$$Information\ Gain\ (T,X) = Entropy\ (T) - Entropy\ (T,X)$$

Therefore, IG is also defined as

$$Information\ Gain = Entropy\ (before) - \sum_{j=1}^{K} Entropy\ (j, after)$$

Where:

Entropy (before) before refers to the set before division,

K refers to the number of divisions,

and (j, after) refers to a divided subset j after division

3.2.1.5 Random Forest

Random Forest (RF) classifier is a machine learning algorithm that uses a collection of DTs to improve the accuracy and generalization of the model. Instead of relying on a single DT, RF leverages predictions from multiple DTs and predicts the class based on the majority of predicted classes. This method is known as bagging or bootstrap aggregating.

The RF algorithm builds multiple decision trees on several random subsets of the original dataset and aggregates the results by finding the mean or mode of the predicted classes. By creating multiple decision trees, RF reduces the chances of overfitting the model to the training data and improves the accuracy and robustness of the model.

RF algorithm is highly efficient while working with high-dimensionality data. It is also lesser chance of overfitting than a single decision tree, as each DT in the RF is trained on a different subset of data.

The RF algorithm is illustrated in the diagram below, which shows a simple example of how the algorithm works.

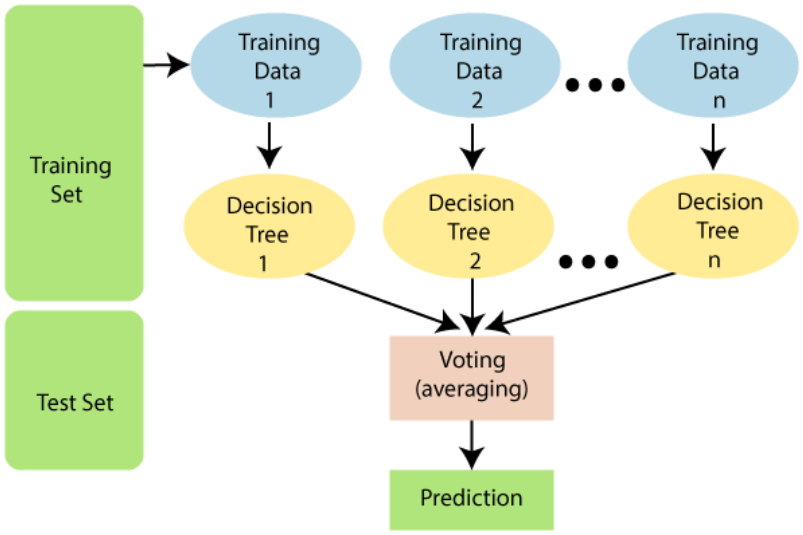


Fig 3.14 Decision Tree Algorithm (courtesy of *java point, Random Forest*)

The RF algorithm is mainly used over other algorithms due to the following reasons:

- The training time is very less compared to other ML algorithms.
- The algorithm's prediction accuracy is high, not taking into consideration the size of the data set.
- Even with a noticeable number of data missing, this algorithm can keep its performance regardless.

a. Advantages of using Random Forest in Machine Learning

- Both classification and regression tasks can be taken care of by Random Forest.
- Large datasets with excessive features can be handled by this algorithm.
- The performance of the model is raised, and overfitting is prevented.

b. Disadvantages of using Random Forest in Machine Learning

- It is not possible for regression problems as it is for classification problems.

3.2.1.6 Ensemble Stacking

Using output from several nodes (such as kNN or NB or LR), the ensemble learning method that creates a new model by stacking them is the Ensemble Stack. To make predictions on the test dataset, the final model is employed.

Stacking involves combining the outputs from several different machine learning models on the same dataset. A stacking model includes of two or more estimators, also referred to as level-0 models (model stack), and a final estimator, or level-1 model (meta learner), which makes the final prediction and joins the output predictions of the level 0 estimators.

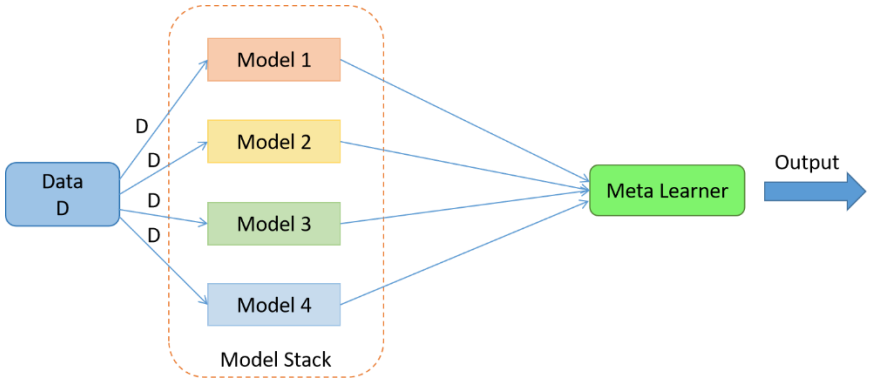


Fig 3.15 Ensemble Stacking

- Estimators to training data and whose predictions are received are referred to as Level-0 Models (Model Stack)
- Final estimator that learns the best way to integrate predictions from the model stack.

The model stacks' extrapolations from non-sample data are used to train the final estimator. In other words, data that was not utilised to train the base models is fed into the base models or level-0 models, which then make predictions and offer outcomes that serve as the input and output pairs of the training dataset to which the final estimator is fitted.

A popular method for creating the training dataset for the meta-model is to perform k-fold cross-validation on the level-0 models, using the out-of-fold predictions as the basis of prediction.

The final estimator model's training data may also contain base model inputs, such as training data input components. This can give the final estimator (meta model) more context for deciding how to combine its predictions.

The basic models may be trained on the complete original training dataset while the meta-model can be trained separately on this dataset after it has been constructed.

While several machine learning models perform well on a dataset but not on some, stacking is acceptable. We can conclude that there is little or no association between the model predictions and the mistakes in those predictions.

Estimators are consistently complex and different. Therefore, it is wise to adopt multiple complex models, such as linear models, DTs, SVMs, ANNs, and more, those that can predict how to obtain good predictions. As first estimators, different ensemble techniques like RFs can also be employed.

3.2.2 Dimensionality Reduction

3.2.2.1 Information Gain

Machine learning uses the feature selection technique to pick the most pertinent subset of the features that are present in a dataset. By doing this, the data noise can be reduced, and a more accurate prediction or classification can be made.

More features always do not mean more information for the model. Sometimes there are unnecessary features in the data set, which may cause the model to get trained with useless information. For example, how much a person reads in a day does not relate to whether they have diabetes or not.

a. The Mathematics behind the Algorithm

Information Gain is obtained as,

$$I(x) = -\log_2 p(x)$$

Entropy is a measure of system dysfunction. High entropy systems would be unpredictable, more disorganized, and less unexpected. On the other hand, a system with lower entropy would be more orderly and highly predictable. Information

Entropy measures how unpredictable a data distribution is in the context of data science.

Entropy is obtained as,

$$E(S) = \sum_{i=1}^{c} - p_{o\,i} \log_2 p_o$$

3.2.2.2 Principal Component Analysis

In machine learning, an unsupervised learning strategy known as PCA is used to reduce dimensionality. It is a statistical process that turns observations of correlated attributes into a collection of linearly uncorrelated data using orthogonal transformation. These newly revised features are the Principal Components. This is a popular tool for exploratory data analysis and predictive modelling. It is a strategy for detecting meaningful patterns in a given dataset by reducing variances.

PCA typically seeks the surface with the lowest dimensionality to project the high-dimensional data onto. PCA works by taking each characteristic's variance into account, because a high attribute demonstrates a firm divide between groups, lowering the dimensionality. PCA's practical applications include image processing, movie recommendation systems, and power allocation optimisation across numerous communication channels. Because it employs a feature extraction technique, it retains the significant variables while discarding the useless ones.

The PCA theory is backed by mathematical solutions like:

- Variance and Covariance
- Eigenvalues and Eigen Vectors

A few terms used often in the PCA algorithm:

- **Dimensionality:** It is the quantity of dimensions or features in the dataset in question. The dataset’s no. of columns makes this easier to decide.
- **Correlation:** This term describes the degree to which dual features are connected to each other. In any case, if one feature changes, the other features likewise are altered. The correlation score ranges under -1 and

+1. Here, -1 denotes an inverse relationship between the variables, while +1 implies a strong correlation between the variables.

- **Orthogonal:** It sets up that there is no connection between the variables, and as a result, there is none.
- **Eigenvectors:** If a nonzero vector v is supplied along with a square matrix M. In a case where Av is divisible by v's scalar, v is known to be an eigenvector.
- **Covariance Matrix:** It is a matrix having covariance between two variables.

The Principal Components are the newly altered characteristics or the result of PCA, as previously said. These principal components are either the same number or less than the first characteristics that were included in the dataset. Following are a few characteristics of these primary components:

- The linear combination of the original dimensions must be the principal part.
- There is zero correlation between any two variable/components owing to orthogonal nature of the components.
- If a case of n principal components, the first component is the most significant while the nth component is the least significant.

a. Steps for Principal Component Analysis

1. Picking the dataset and dividing it into dependent and independent sets.
2. We then stand for the dataset as a two-dimensional matrix.
3. Standardization of the dataset.
4. Calculate the covariance of matrix Z.
5. Calculate the Eigen Values and Eigenvectors
6. Sort the Eigenvectors.
7. Derive the principal components
8. Drop unnecessary components.

3.2.3 k-Fold Cross Validation

k-Fold Cross Validation is a technique used in machine learning to evaluate the performance of a model. It involves splitting the available dataset into k equal parts, or "folds." One of the folds is used as the testing dataset, while the other k-1 folds are used as the training dataset. This process is repeated k times, with each of the k folds being used once as the testing dataset.

The results from k evaluations are aggregated to produce an estimate of the model's performance. Doing so helps in reducing the variance in the approximation of the model's capabilities in terms of performance, while providing near accurate accuracy scores.

K-fold cross validation is mostly effective in situations where the size of the dataset is small, or when there is a possibility of overfitting to the training dataset. The general performance of the model can be improved by using multiple folds to train and test the model, which in our case, we have used the value of K as 10.

Fig 3.16 Working of K-fold Cross Validation

When the value of K is large, or when implemented on a large dataset, k-fold cross validation presents a drawback, in which the time taken to train, and test becomes large due to computational expensiveness. Nonetheless, there are techniques to make the process faster, like parallelizing the testing and training of the model.

3.2.4 Evaluation Metrics

In order to measure the performance of the model, the evaluation parameters below have been implied:

- **True Positive Rate:** The rate of correct predictions of scam (phishing) websites. It is also referred to as recall.

$$TPR = \frac{TP}{Actual\ Positive} = \frac{TP}{TP + FN}$$

- **False Positive Rate:** Rate of incorrectly predicted legitimate URLs.

$$FPR = \frac{FN}{Actual\ Positive} = \frac{FN}{TP + FN}$$

- **Precision:** Degree of exactness.

$$Precision = \frac{TP}{TP + FP}$$

- **F1 score:** Harmonic mean on precision and recall.

$$F1\ Score = 2.\frac{precision.recall}{precision + recall}$$

- **Accuracy (%):** Rate of correctly recognized legitimate and phishing websites.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Matthew's Correlation Coefficient (MCC):** It measures the correlation between predicted and actual data labels. It ranges from -1 to 1.

$$MCC = \frac{TPxTN - FPxFN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

CHAPTER 4: PERFORMANCE ANALYSIS

4.1 EXPERIMENTAL RESULT

4.1.1 Model Results

4.1.1.1 Before Feature Selection

The results obtained before as follows:

a. Logistic Regression:

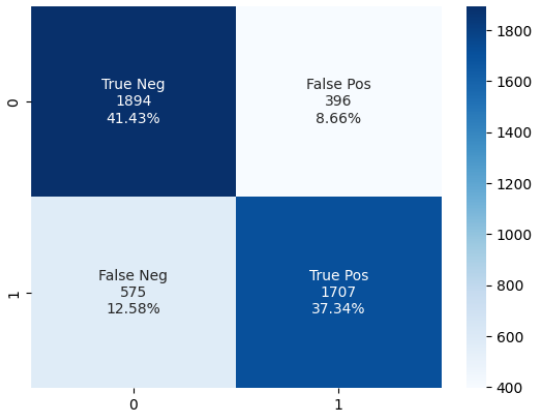


Fig 4.1 (a) Confusion Matrix for LR(Before Feature Selection)

The model had a TNR and TPR of 41.43% and 37.34% respectively. The FNR and FPR were found to be 12.58% and 8.66% respectively.

b. Support Vector Machine

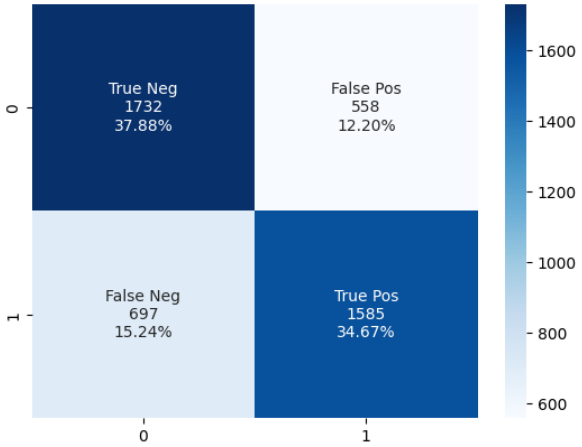


Fig 4.1 (b) Confusion Matrix for SVM (Before Feature Selection)

The model had a TNR and TPR of 37.88% and 34.67% respectively. The FNR and FPR were found to be 15.24% and 13.20% respectively.

c. Decision Trees

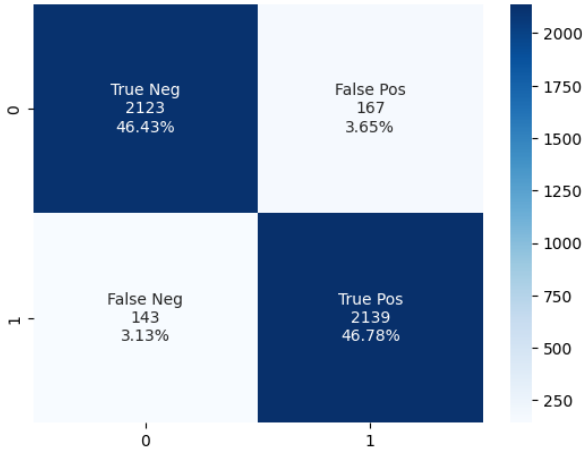


Fig 4.1 (c) Confusion Matrix for DT (Before Feature Selection)

The model had a TNR and TPR of 46.43% and 46.78% respectively. The FNR and FPR were found to be 3.13% and 3.65% respectively.

d. k-Nearest Neighbors

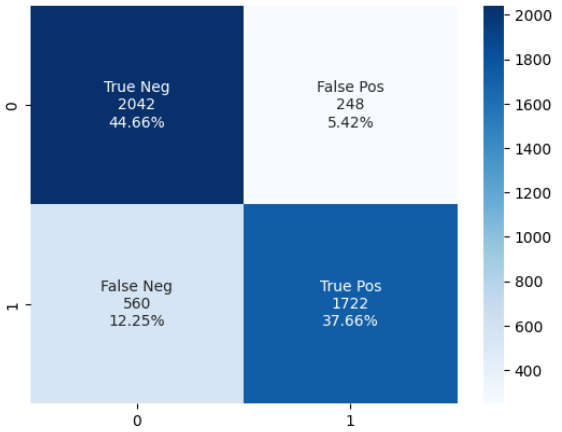


Fig 4.1. (d) Confusion Matrix for KNN (Before Feature Selection)

The model had a TNR and TPR of 44.66% and 37.66% respectively. The FNR and FPR were found to be 12.25.% and 5.42% respectively.

e. Random Forest

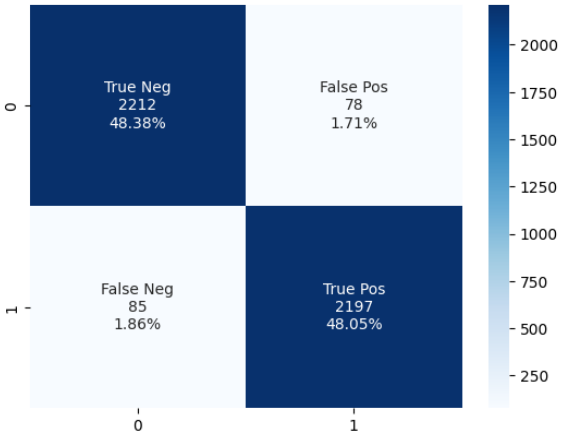


Fig 4.1. (e) Confusion Matrix for RF (Before Feature Selection)

The model had a TNR and TPR of 48.38% and 48.05% respectively. The FNR and FPR were found to be 1.86% and 1.71% respectively.

f. Ensemble Stacking

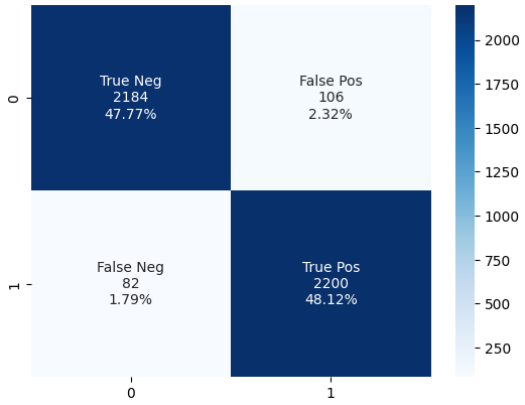


Fig 4.1. (f) Confusion Matrix for ES (Before Feature Selection)

The model had a TNR and TPR of 47.77% and 48.12% respectively. The FNR and FPR were found to be 1.79% and 2.32% respectively.

4.1.1.2 After Principal Component Analysis

The results obtained before as follows:

a. Logistic Regression:

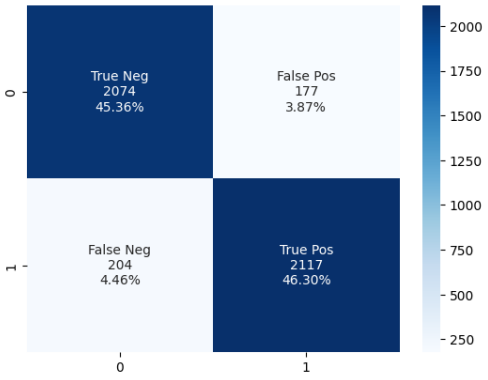


Fig 4.2 (a) Confusion Matrix for LR (After Feature Reduction)

The model had a TNR and TPR of 45.36% and 46.30% respectively. The FNR and FPR were found to be 4.46% and 3.87% respectively.

b. Support Vector Machine

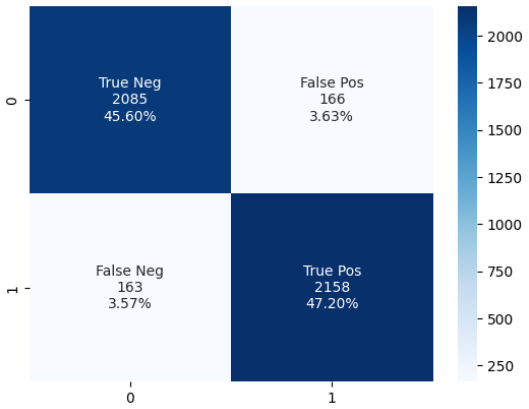


Fig 4.2 (b) Confusion Matrix for SVM (After Feature Reduction)

The model had a TNR and TPR of 45.60% and 47.20% respectively. The FNR and FPR were found to be 3.57% and 3.63% respectively.

c. Decision Trees

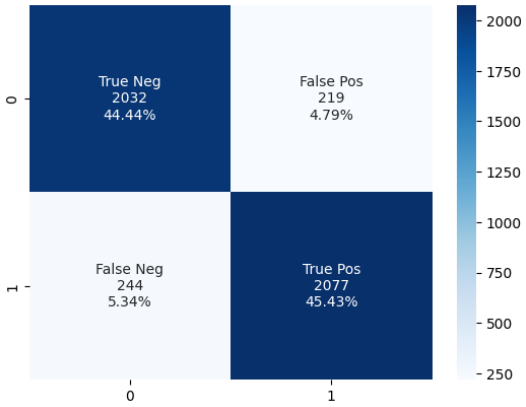


Fig 4.2 (c) Confusion Matrix for DT (After Feature Reduction)

The model had a TNR and TPR of 44.44% and 45.43% respectively. The FNR and FPR were found to be 5.34% and 4.79% respectively.

d. k-Nearest Neighbours

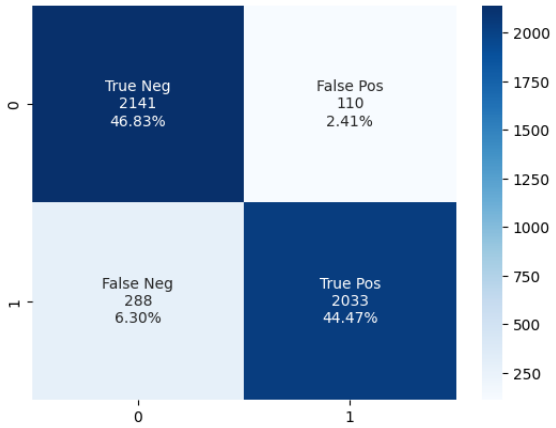


Fig 4.2 (d) Confusion Matrix for KNN (After Feature Reduction)

The model had a TNR and TPR of 46.83% and 44.47% respectively. The FNR and FPR were found to be 6.30% and 2.41% respectively.

e. Random Forest

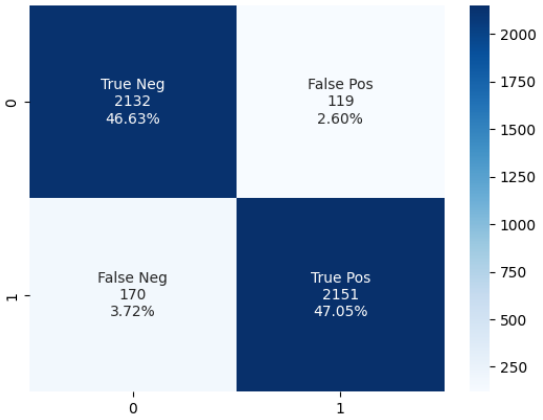


Fig 4.2 (e) Confusion Matrix for RF (After Feature Reduction)

The model had a TNR and TPR of 46.63% and 47.05% respectively. The FNR and FPR were found to be 3.72% and 2.60% respectively.

f. Ensemble Stacking

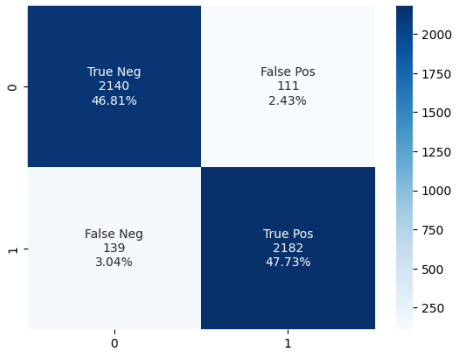


Fig 4.2 (f) Confusion Matrix for ES (After Feature Reduction)

The model had a TNR and TPR of 46.81% and 47.73% respectively. The FNR and FPR were found to be 3.04% and 2.43% respectively.

4.1.1.3 With Feature Selection (Information Gain)

The results obtained before as follows:

a. Logistic Regression :

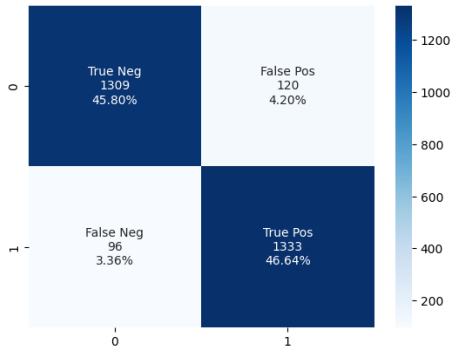


Fig 4.3 (a) Confusion Matrix for LR (After Feature Selection)

The model had a TNR and TPR of 45.80% and 46640% respectively. The FNR and FPR were found to be 3.36% and 4.20% respectively.

b. Support Vector Machine

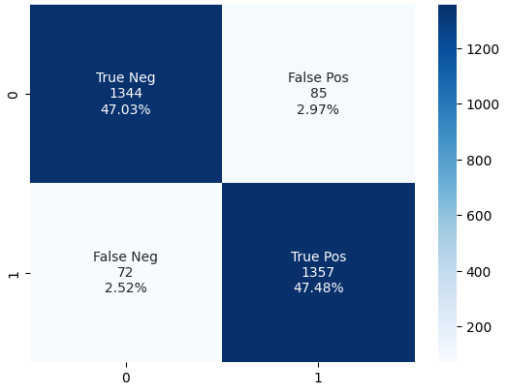


Fig 4.3. (b) Confusion Matrix for SVM (After Feature Selection)

The model had a TNR and TPR of 47.03% and 47.48% respectively. The FNR and FPR were found to be 2.52% and 2.97% respectively.

c. Decision Trees

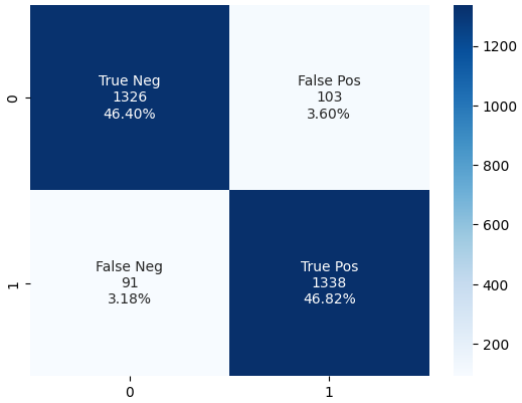


Fig 4.3. (c)Confusion Matrix for DT (After Feature Selection)

The model had a TNR and TPR of 46.40% and 46.82% respectively. The FNR and FPR were found to be 3.18% and 3.60% respectively.

d. k-Nearest Neighbors

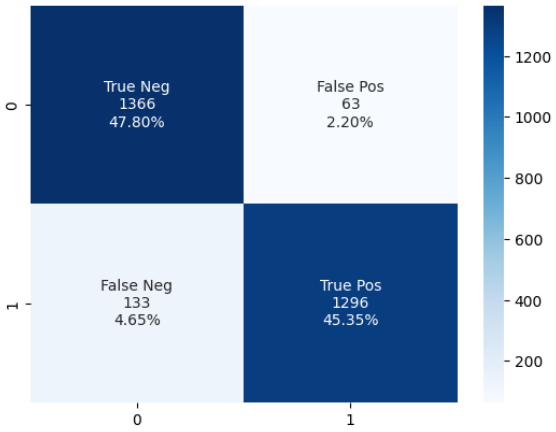


Fig 4.3 (d) Confusion Matrix for KNN (After Feature Selection)

The model had a TNR and TPR of 47.80% and 45.35% respectively. The FNR and FPR were found to be 4.65% and 2.20% respectively.

e. Random Forest

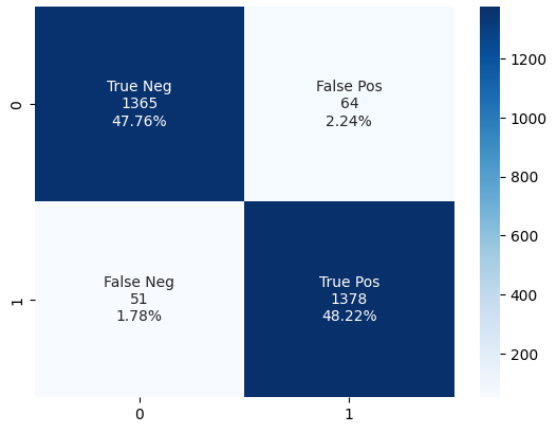


Fig 4.3 (e) Confusion Matrix for RF (After Feature Selection)

The model had a TNR and TPR of 47.76% and 48.22% respectively. The FNR and FPR were found to be 1.78% and 2.24% respectively.

f. Ensemble Stacking

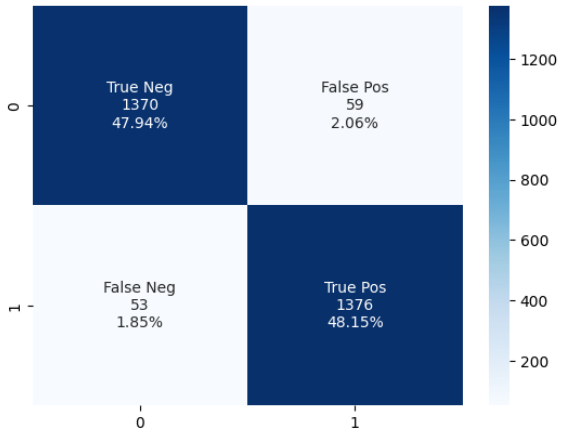


Fig 4.3 (f) Confusion Matrix for ES (After Feature Selection)

The model had a TNR and TPR of 47.94% and 48.15% respectively. The FNR and FPR were found to be 1.85% and 2.06% respectively.

4.1.2 Overall Result

This section provides an overall analysis of experimental results along with their visualization (Confusion Matrix). All the six classifiers are made to classify phishing between phishing URLs and safe URLs by using all the eighty-seven features.

Table 4.1 shows the training time, testing time, accuracy, recall, precision and F1 score of each classifier. It shows that RF supplies the highest accuracy (96.43%) followed by ES (95.89%), DT (93.21%), KNN (82.33%) and LR (78.76%). SVM supplies the least accuracy of 72.55%.

Figure 4.4 can show us the comparison of all the classifiers on the basis of accuracy, recall, precision, MCC and F1-score.

Table 4.1: Classification Results before Feature Selection (with 87 features)

Classifier	Train time (s)	Test time (s)	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	MCC
LR	0.106	0.006	78.76	74.80	0.81	0.78	0.576
SVM	4.801	2.196	72.55	0.69	0.74	71.63	0.452
DT	0.108	0.006	93.21	93.73	92.75	93.24	0.594
KNN	0.012	0.419	82.33	75.46	87.41	80.99	0.653
RF	0.965	0.076	96.43	96.28	96.57	96.42	0.929
ES	6.766	0.460	95.89	96.41	95.40	95.90	0.918

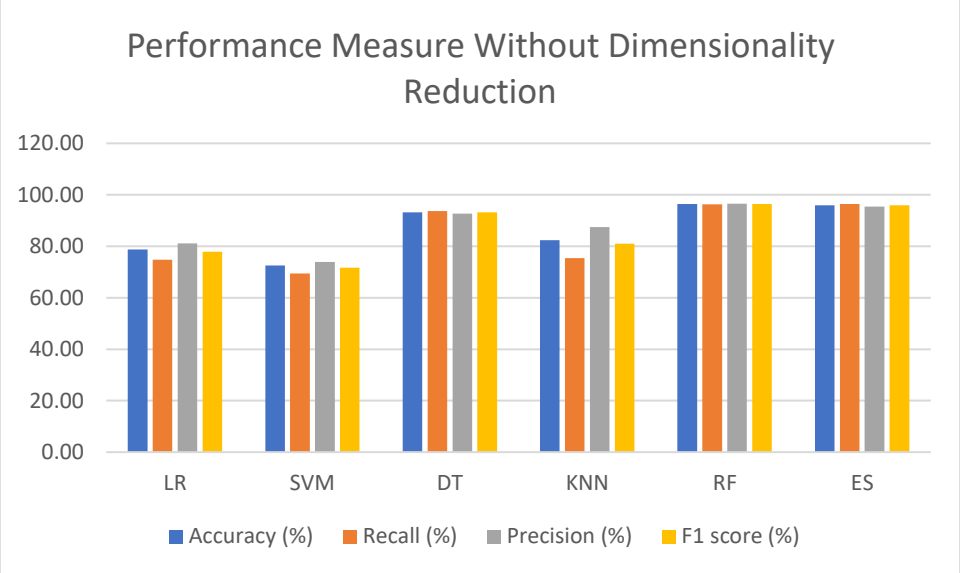


Fig. 4.4 Comparison of the Models based on Evaluation Metrics before applying Feature Selection/Reduction

Next, we selected the top 21 components using PCA methods and then we used the same classification algorithms to classify between phishing and safe URLs. Table 4.2 shows the training time, testing time, accuracy, recall, precision and F1 score of each classifier after performing feature reduction. Out of six

machine learning models built using the 21 components after feature reduction, we found that highest accuracy is given by the RF classifier with 94.68% and then followed by ES(93.53%), SVM (92.80%), LR (91.67%), and KNN (91.29%). The least accuracy is given by the DT classifier with an accuracy of 91.67 %. Figure 4.5 can show us the visualization of all the models based on accuracy, recall, precision, MCC and F1-score.

Table 4.2: Classification Results after Feature Reduction using PCA (with 21 Components)

Classifier	Train time (s)	Test time (s)	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	MCC
LR	0.031	0.003	91.67	91.21	92.28	91.74	0.833
SVM	1.041	0.305	92.80	92.98	92.86	92.92	0.856
DT	0.286	0.002	89.87	89.49	90.47	89.97	0.797
KNN	0.001	0.367	91.29	87.59	94.87	91.08	0.828
RF	2.777	0.076	94.68	94.68	96.76	95.71	0.894
ES	18.170	0.855	93.53	94.01	95.16	94.58	0.890

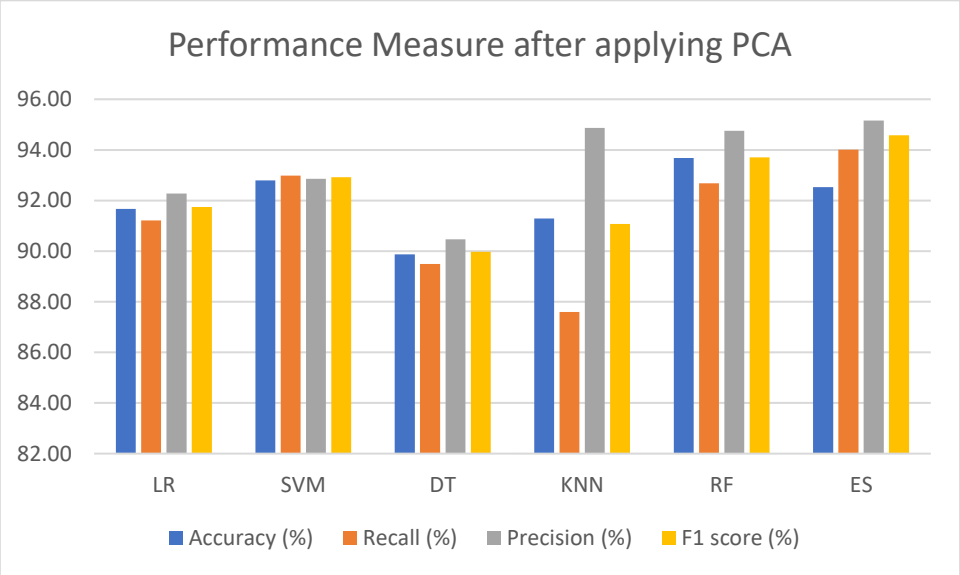


Fig. 4.5 Comparison of Models based on Evaluation Metrics after Feature Reduction(PCA)

Finally, we selected the top 22 features from the already existing 87 features. For the feature selection we used Information Gain or Mutual Information. We calculated the dependence between two variables and calculated those features that have the largest mi-scores.

We again used the same six machine learning models to classify the Phishing urls and the safe URLs. Table 4.3 shows the training time, testing time, accuracy, recall, precision and F1 score of each classifier after performing feature selection. We again notice that ES provides the highest accuracy with 96.08 % followed by RF (95.98%), SVM (94.51%), DT (93.21%), and KNN (93.14%). In this method LR gives the minimum accuracy of 92.44%. Figure 4.6 can show us the comparison of all the classifiers on the basis of accuracy, recall, precision, MCC and F1-score.

Table 4.3: Classification Results after Feature Selection using Information Gain/Mutual Information (with 22 features)

Classifier	**Train time (s)**	**Test time (s)**	**Accuracy (%)**	**Recall (%)**	**Precision (%)**	**F1 score (%)**	**MCC**
LR	0.078	0.013	92.44	93.29	91.74	92.51	0.849
SVM	1.723	0.208	94.51	94.96	94.11	94.53	0.89
DT	0.081	0.007	93.21	93.63	93.85	93.24	0.864
KNN	0.004	0.164	93.14	90.69	95.36	92.97	0.864
RF	1.115	0.049	95.98	96.47	95.56	95.99	0.919
ES	15.598	0.257	96.08	96.29	95.88	96.10	0.923

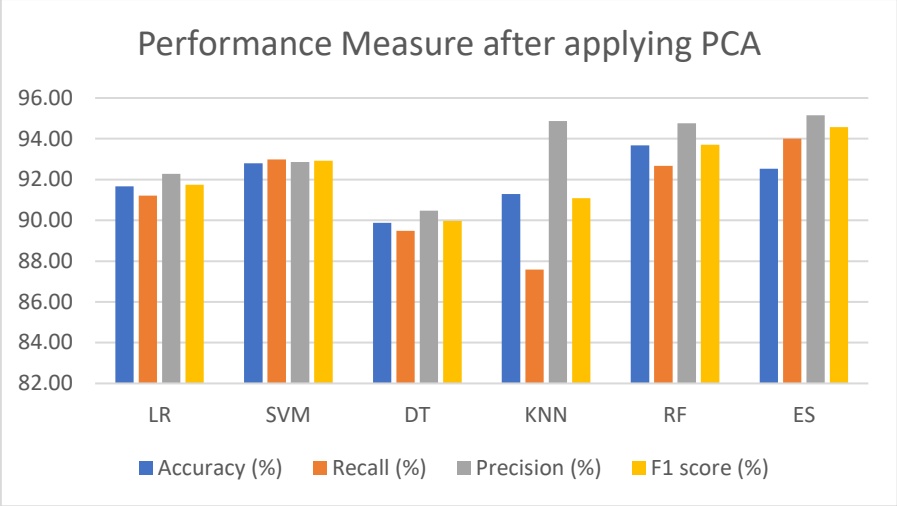


Fig. 4.6 Comparison of Models based on Evaluation Metrics after Feature Selection (Mutual Information)

From the above tables, we can conclude that with the decreasing number of features the training time and testing time of each model reduces gradually. However, the time difference for Ensemble Stacking seems to be non-uniform. This is mainly because we selected the top performing model and at each stage the top performing model is different and hence, we notice non-uniformity. Also, we see some slight non-uniformity in the time difference in each method and it can be due to internet speed that often changes every second.

We noticed that the Random Forest classifier was the best classifier for this task, hence we took all three models from our project and implemented K-fold Cross Validation on them. We set the value of K to 10 and the results are given in Table 4.4 and Figure 4.7.

Table 4.4: Performance Measure of the three RF models after applying K-fold Cross Validation

Classifier	Train time (s)	Test time (s)	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	MCC
RF	15.471	0.081	99.61	99.65	99.56	99.61	0.992
RF (FS)	14.489	0.053	99.58	99.65	99.51	99.58	0.991
RF (PCA)	42.220	0.072	99.37	99.09	99.65	99.37	0.987

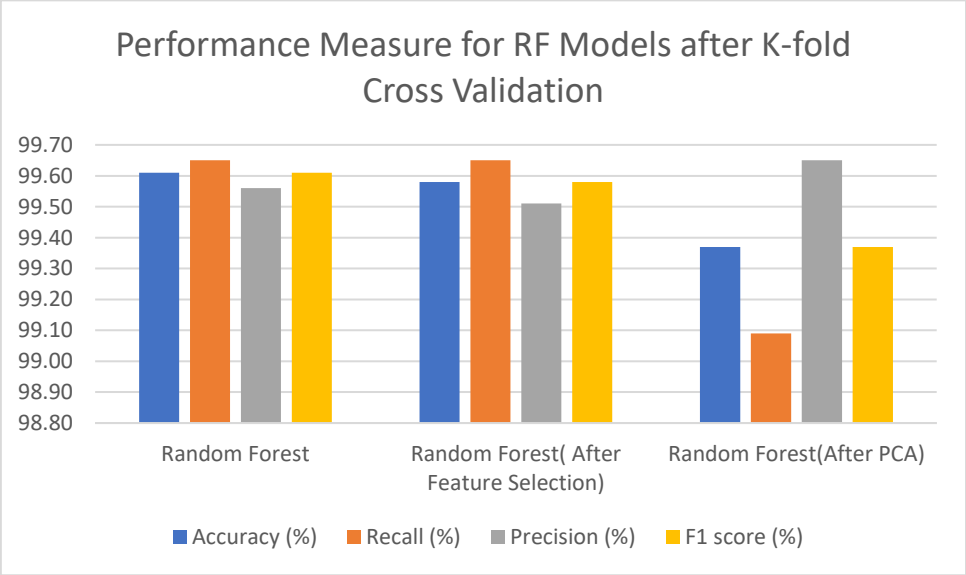


Fig. 4.7 Performance Measures of RF models after K-fold Cross Validation

As seen in the table and graph, the Random Forest model without any dimensionality reduction has the highest accuracy, 99.61% but not a huge difference from the other two RF models that both have accuracy scores above 99.37%. With the results, we decided to use the cross validated RF model without any dimensionality reduction for our Chrome Extension.

4.1.3 Use Case – Chrome Extension for Detection of Phishing URLs

The Chrome Extension is also one of the results (product achieved) from this project, which can be used practically in many circumstances. The extension was built for the purpose of identifying URLs as either phishing or legitimate.

The Chrome Extension displays an alert pop up box in case a user visits a suspected phishing URL and also mentions the URL as unsafe with a red cross graphic. when the user visits a legitimate URL, the extension displays that the URL is safe with a green tick graphic.

The snippets are shared in the following figures Fig. 4.8 and Fig. 4.9:

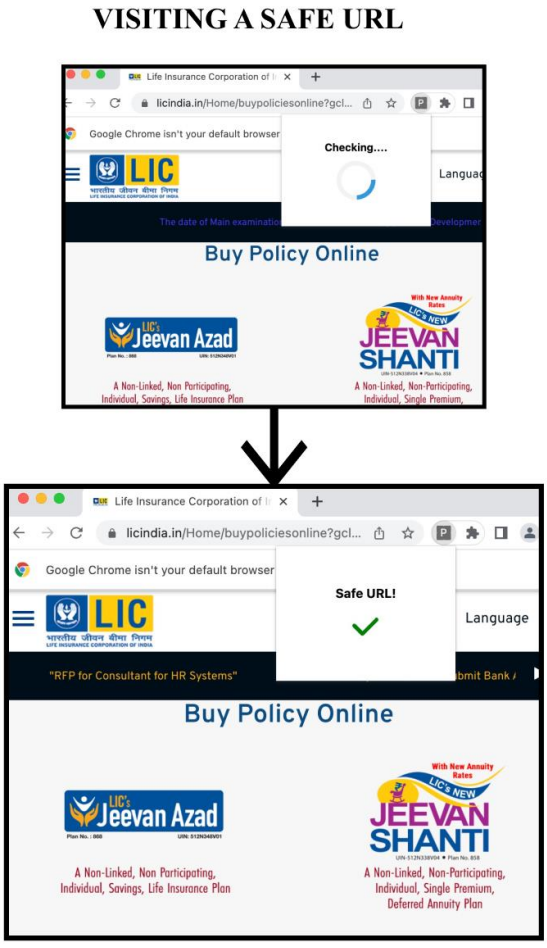


Fig. 4.8 The Chrome Extension When Visiting a Safe URL

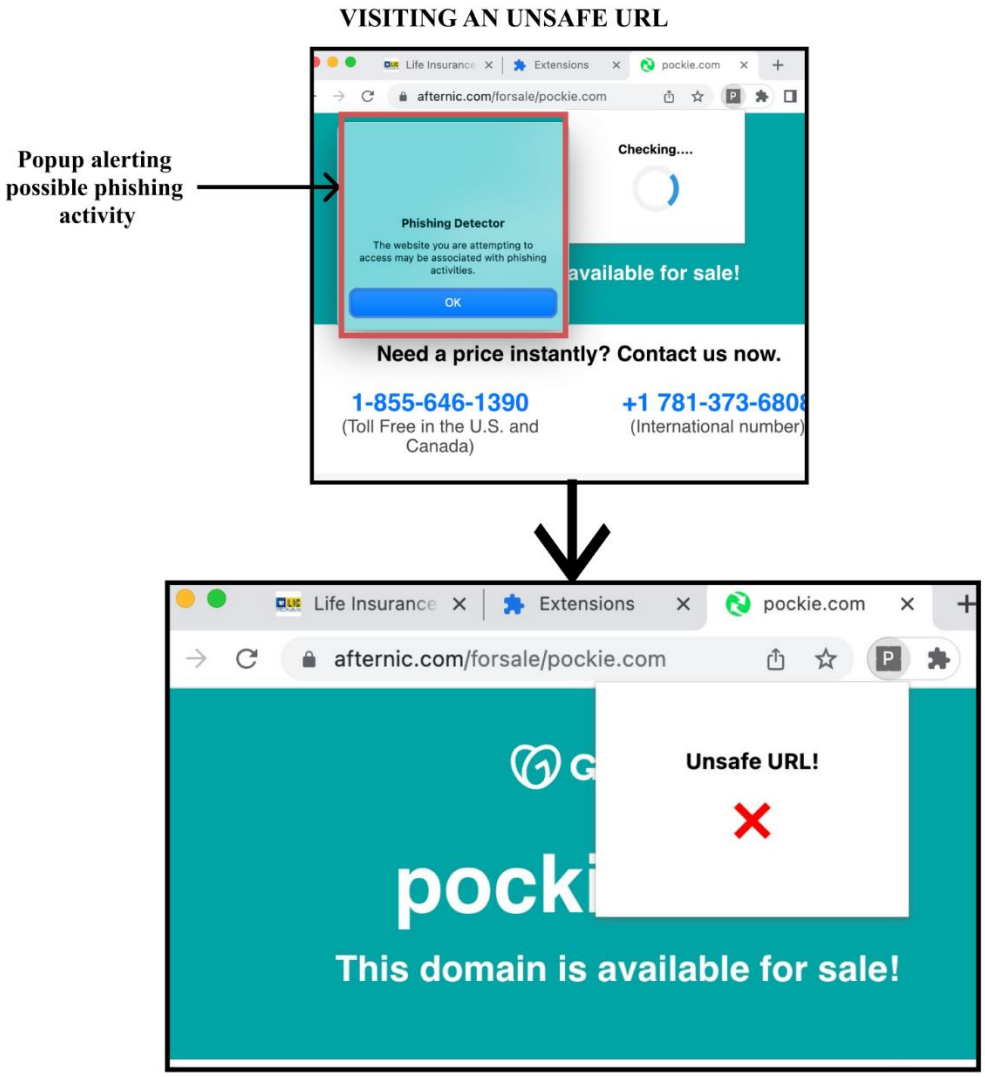


Fig. 4.9 The Chrome Extension When Visiting an Unsafe URL

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

Through this project, we were able to build an efficient system that can classify phishing and legitimate URLs. Dimensionality reduction methods were implemented along with different machine learning algorithms to improve the accuracy of the model. The performance of six classifiers built without feature selection, with feature reduction (PCA) and with feature selection (Information Gain/ Mutual Information Gain) were compared side by side. The results after the experiments showed that the overall accuracy of the model improved upon performing dimensionality reduction. We also noticed some slight changes in the build time when using all the features and when performing dimensionality reduction. However, since the model was built on Google Colab, which runs in the cloud, the build time was highly affected by the internet speed.

Furthermore, we developed a Chrome extension that uses the best machine learning model to classify URLs in real-time. The Chrome extension allows users to check whether a website is safe to use or not by simply clicking on the extension icon. We integrated the best model with the Flask framework and hosted it on Railway.app, a cloud-based platform for easy deployment. The use of Flask allowed us to create an API endpoint that the Chrome extension could communicate with. The extension was designed using manifest 3, background.js, popup.html, and internal CSS in the popup.html file. The extension successfully classified URLs with high accuracy, making it a useful tool for users to detect phishing websites.

5.2 FUTURE SCOPE

Regarding future scope, our product can be improved by adding a feature that directly detects an email for phishing by checking the email content, through reading hyperlinks. While phishing emails are very common, it is also the method that works the most for cybercriminals, hence the need for a detection tool in the inbox of every email service is proving to be essential. Additionally, the extension can be integrated with popular email clients to supply real-time scanning of links in emails.

There is also a room for improvement in our model's performance. While the accuracy is at 99.61% at its current state, it can still be improved using different methods or modifying parameters that can help the model achieve higher performance levels and lower testing and training times.

The Chrome Extension can also be improved by adding a better User Interface and implementing some extra features like the ability to scan a URL for phishing manually by the users. The extension can also be improved by making it more responsive and reducing loading times.

With the evolving techniques for phishing, we must be cautious about new trends in phishing URLs. Hence, it is important to keep updating the database of phishing URLs and add more data points to the dataset, which we are currently working on.

REFERENCES

[1] S. Tiwari, "Web page Phishing Detection Dataset", Kaggle, [Online]. Available: https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset/download?datasetVersionNumber=2

[2] IRONSCALES releases Findings from State of Cybersecurity Survey, https://ironscales.com/blog/ironscales-releases-findings-from-state-of-cybersecurity-survey/

[3] Gone Phishing Tournament Report 2020, https://terranovasecurity.com/2020-gpt-report/?utrm_campaign=En_GPTReport2020&utm_medium=Google&utm_source=Ads&utm_content=NewAd3

[4] ESET Threat Report 2020, https://www.welivesecurity.com/wp-content/uploads/2020/10/ESET_Threat_Report_Q32020.pdf

[5] KnowBe4 TOP-CLICKED PHISHING TESTS, https://blog.knowbe4.com/q3-2021-top-clicked-phishing-report-infographic-with-global-data

[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.

[7] R. S. Rao, and A. R. Pais, "Detecting Phishing Websites using Automation of Human Behavior." In Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security, ACM, 1994, pp. 33-42.

[8] A.K Jain, & B.B Gupta, "Towards Detection of Phishing Websites on Client-Side using Machine Learning Based Approach." Telecommunication Systems, vol. 67, no. 2, pp. 687-700, 2018.

[9] E. Gandotra, D. Gupta, "An Efficient Approach for Phishing Detection using Machine Learning" in Multimedia Security, Springer, 2021, pp. 239-253

[10] E. Gandotra, D. Gupta, "Improving Spoofed Website Detection Using Machine Learning", Cybernetics and Systems: An International Journal, vol. 52, no.2, pp. 169-190, Oct. 2020

[11] M. Almseidin, A. A. Zuraiq, M. Al-kasassbeh, & N. Alnidami, "Phishing Detection Based on Machine Learning and Feature Selection Methods." iJIM, pp. 171-183, 2019.

[12] O. K. Sahingoz, E. Buber, O. Demir, and B. Diri," Machine Learning Based Phishing Detection from URLs", Expert Systems with Applications, vol. 117, pp.345–57, 2019, doi: 10.1016/j.eswa.2018.09.029.

[13] A. Zamir, H. U. Khan, T. Iqbal, N. Yousaf, F. Aslam, A. Anjum, & M. Hamdani, "Phishing Web Site Detection using Diverse Machine Learning Algorithms." The Electronic Library, 2020

[14] A. F. Hossain, A. K. Das, M. A. M. A. Haque and M. M. Hasan, "Phishing Detection Using Machine Learning: A Comparative Study," in IEEE Access, vol. 8, pp. 131078-131089, 2020.

[15] R. Khanna and S. Gupta, "Phishing Detection Techniques: A Survey," in Proceedings of the 2021 IEEE International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 146-151.

[16] M. R. Islam, M. A. Khan, and J. H. Park, "An Ensemble Learning Approach to Phishing Detection," ACM Transactions on Internet Technology, vol. 18, no. 3, pp. 1-26, May 2018. DOI: 10.1145/3182376

[17] S. Roy and S. K. Singh, "A Comparative Study of Phishing Detection Using Machine Learning Algorithms," in IEEE Access, vol. 8, pp. 147583-147596, 2020.

[18] C. Zhao and Y. Liu, "A Phishing Website Detection Method Based on Feature Selection and Ensemble Learning," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), 2020, pp. 432-441, doi: 10.1109/QRS51294.2020.00060.

[19] S. Yadav, A. Jain, and R. Jain, "A Novel Approach for Phishing Detection using Machine Learning Techniques," Journal of Information Security

and Applications, vol. 57, pp. 102852, 2021. doi: 10.1016/j.jisa.2020.102852

[20] A. Singh, A. Singh, and V. Tyagi, "Phishing Detection using Machine Learning Techniques: A Review," Journal of Intelligent and Fuzzy Systems, vol. 39, no. 2, pp. 1955-1974, 2020. doi: 10.3233/JIFS-191542

[21] Y. Yang, Y. Li, and F. Wang, "Phishing Website Detection using Machine Learning Techniques and Website Features," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 1, pp. 1311-1324, 2021. doi: 10.1007/s12652-020-02268-8

[22] H. K. Dam, K. T. Tran, and H. T. Nguyen, "Phishing Detection Using Machine Learning Techniques," ACM Transactions on Information and System Security, vol. 21, no. 4, pp. 1-23, Aug. 2018. DOI: 10.1145/3239747

[23] A. Kumar, A. Kumar, and R. Kumar, "Machine Learning Techniques for Phishing Detection," ACM Transactions on Internet Technology, vol. 19, no. 3, pp. 1-24, Aug. 2019. DOI: 10.1145/3337005

TMP02

ORIGINALITY REPORT

8%	4%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Middlesex University Student Paper	<1%
2	doctorpenguin.com Internet Source	<1%
3	link.springer.com Internet Source	<1%
4	web.archive.org Internet Source	<1%
5	Submitted to Monash University Student Paper	<1%
6	Submitted to Liverpool John Moores University Student Paper	<1%
7	"Soft Computing for Security Applications", Springer Science and Business Media LLC, 2022 Publication	<1%
8	expertinsights.com Internet Source	<1%