Phishing URLs Detection

L.O.K.I: Lookout Operatives Keeping the Internet
Debayan Datta
Anurag Joardar

May 1, 2024

Abstract

Phishing is a fraudulent act where attackers pretend as someone else via email or other channels to obtain sensitive information like login credentials or account details. Victims often receive messages that appear convincing enough to be from known contacts or organizations. These message contain malicious links to websites which persuade them into revealing personal and financial data. Phishing trick users into disclosing confidential information unwillingly, resulting in identity theft and financial fraud. In this study, feature extraction has been implemented to extract various insights of the data, and by further modeling and evaluation we can conclude that Random Forest gives the highest accuracy.

1 Introduction

1.1 What:

Phishing, an online threat, involves attackers impersonating trusted entities to obtain sensitive and confidential information from victims. There has been recent advancements in phishing detection by Machine Learning. This paper focuses on the development and comparison of several machine learning models aimed at enhancing phishing detection efficiency.

1.2 Why:

Phishing is a big problem for both people and companies, and we need strong ways to stop it. The usual methods aren't always good enough because phishing tactics keep changing. Machine learning can help by learning from past examples of phishing and spotting new ones. By trying out different machine learning methods, we can figure out which ones work best for spotting phishing attempts. This helps us build better defenses against online scams.

1.3 How:

The study utilizes a dataset having several URLs with their class. After getting the insights of this data through feature extraction, we provide standard comparisons of different models and also provide different evaluation metrics for better understanding. Some of these models include K-nearest neighbor (KNN), Naive Bayes Classifier (NBC), Support Vector Machines (SVMs), Decision Trees (DTs), and Random Forest (RF) techniques. Each machine learning model is trained and tested using appropriate techniques and parameters. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to assess the performance of each model.

2 Literature Survey:

- Kumi et al. [4] extracted eight features from web page content to a Classification Based on Association Rules Algorithm (CBA) to detect phishing web pages. These features include the number of special characters, sensitive words in a URL, and the entropy of the domain name. Evaluations of 700 phishing and 500 legitimate URLs yielded an accuracy of 95.8%.

- Yerima et al. [16] proposed a phishing detection approach that took 30 static features from a web page's URL and HTML content and applied them to 1D convolutional neural networks. Experiments on 6,157 legitimate and 4,898 phishing websites achieved an accuracy of 98.2% and an F1-score of 97.6%.

- Grega et al. [15] extracted 113 features from two dataset variations that consist of 58,645 and 88,647 websites labeled as legitimate or phishing and allow the researchers to train their classification models, build phishing detection systems, and mining association rules.

- Moghimi et al. [6] utilized 8 features in the CBA algorithm. The measure of the randomness factor in URLs emerged as the most significant feature and attained 95.8% accuracy.

- Chiew et al. [3] extracted 48 features were implemented on a Random Forest Classifier and attained an accuracy of 96.17%.

- Maroofi et al. [5] used 38 features from the URL were applied to a random forest classifier to detect compromised domains and attained an accuracy of 97%.

3 Proposed methodology

3.1 Feature Extraction

This phase of research aims to gather important information about the URL string. It entails a set of features with distinguishing characteristics used to specify different dataset categories. Since we wish to differentiate between Phishing and Non-Phishing URLs, we do this work using two key feature categories: lexical and web-scrapped features. These are covered in the next section.

3.1.1 Lexical Features

We used the fundamental features from the URL string which is further subdivided into 4 parts: Domain, Directory, File, Parameters. After implementing our program for extracting features from an URL and the four subgroups, below are the following lexical features shown in Tables 1 to 5

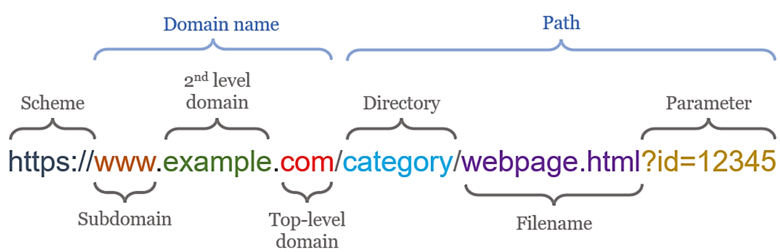

Figure 1: URL segregation

Table 1: URL Attributes Description

Attribute	Format	Description
quantity_dot_url	Number of ”.” signs	Numeric
quantity_hyphen_url	Number of ”-” signs	Numeric
quantity_underline_url	Number of ”_” signs	Numeric
quantity_slash_url	Number of ”/” signs	Numeric
quantity_questionmark_url	Number of ”?” signs	Numeric
quantity_equal_url	Number of ”=” signs	Numeric
quantity_at_url	Number of ”@” signs	Numeric
quantity_and_url	Number of ”&” signs	Numeric
quantity_exclamation_url	Number of ”!” signs	Numeric
quantity_space_url	Number of ” ” signs	Numeric
quantity_tilde_url	Number of ” ” signs	Numeric

Attribute	Format	Description
quantity_comma_url	Number of ”,” signs	Numeric
quantity_plus_url	Number of ”+” signs	Numeric
quantity_asterisk_url	Number of ”*” signs	Numeric
quantity_hashtag_url	Number of ”#” signs	Numeric
quantity_dollar_url	Number of ”$” signs	Numeric
quantity_percent_url	Number of ”%” signs	Numeric
quantity_tld_url	Top level domain character length	Numeric
length_url	Number of characters	Numeric
email_url	Is email present	Boolean

Table 2: Domain Attributes Description

Attribute	Format	Description
quantity_dot_domain	Number of ”.” signs	Numeric
quantity_hyphen_domain	Number of ”-” signs	Numeric
quantity_underline_domain	Number of ”_” signs	Numeric
quantity_slash_domain	Number of ”/” signs	Numeric
quantity_questionmark_domain	Number of ”?” signs	Numeric
quantity_equal_domain	Number of ”=” signs	Numeric
quantity_at_domain	Number of ”@” signs	Numeric
quantity_and_domain	Number of ”&” signs	Numeric
quantity_exclamation_domain	Number of ”!” signs	Numeric
quantity_space_domain	Number of ” ” signs	Numeric
quantity_tilde_domain	Number of ” ” signs	Numeric
quantity_comma_domain	Number of ”,” signs	Numeric
quantity_plus_domain	Number of ”+” signs	Numeric
quantity_asterisk_domain	Number of ”*” signs	Numeric
quantity_hashtag_domain	Number of ”#” signs	Numeric
quantity_dollar_domain	Number of ”$” signs	Numeric
quantity_percent_domain	Number of ”%” signs	Numeric
quantity_vowels_domain	Number of vowels	Numeric
domain_length	Number of domain characters	Numeric
domain_in_ip	URL domain in IP address format	Boolean
server_client_domain	”server” or ”client” in domain	Boolean

Table 3: Directory Attributes Description

Attribute	**Format**	**Description**
quantity_dot_directory	Number of ”.” signs	Numeric
quantity_hyphen_directory	Number of ”-” signs	Numeric
quantity_underline_directory	Number of ”_” signs	Numeric
quantity_slash_directory	Number of ”/” signs	Numeric
quantity_questionmark_directory	Number of ”?” signs	Numeric
quantity_equal_directory	Number of ”=” signs	Numeric
quantity_at_directory	Number of ”@” signs	Numeric
quantity_and_directory	Number of ”&” signs	Numeric
quantity_exclamation_directory	Number of ”!” signs	Numeric
quantity_space_directory	Number of ” ” signs	Numeric
quantity_tilde_directory	Number of ” ” signs	Numeric
quantity_comma_directory	Number of ”,” signs	Numeric
quantity_plus_directory	Number of ”+” signs	Numeric
quantity_asterisk_directory	Number of ”*” signs	Numeric
quantity_hashtag_directory	Number of ”#” signs	Numeric
quantity_dollar_directory	Number of ”$” signs	Numeric
quantity_percent_directory	Number of ”%” signs	Numeric
directory_length	Number of directory characters	Numeric

Table 4: File Attributes Description

Attribute	**Format**	**Description**
quantity_dot_file	Number of ”.” signs	Numeric
quantity_hyphen_file	Number of ”-” signs	Numeric
quantity_underline_file	Number of ”_” signs	Numeric
quantity_slash_file	Number of ”/” signs	Numeric
quantity_questionmark_file	Number of ”?” signs	Numeric
quantity_equal_file	Number of ”=” signs	Numeric
quantity_at_file	Number of ”@” signs	Numeric
quantity_and_file	Number of ”&” signs	Numeric
quantity_exclamation_file	Number of ”!” signs	Numeric
quantity_space_file	Number of ” ” signs	Numeric
quantity_tilde_file	Number of ” ” signs	Numeric
quantity_comma_file	Number of ”,” signs	Numeric
quantity_plus_file	Number of ”+” signs	Numeric

Attribute	Format	Description
quantity_asterisk_file	Number of ”*” signs	Numeric
quantity_hashtag_file	Number of ”#” signs	Numeric
quantity_dollar_file	Number of ”$” signs	Numeric
quantity_percent_file	Number of ”%” signs	Numeric
file_length	Number of file name characters	Numeric

Table 5: Parameters Attributes Description

Attribute	Format	Description
quantity_dot_params	Number of ”.” signs	Numeric
quantity_hyphen_params	Number of ”-” signs	Numeric
quantity_underline_params	Number of ”_” signs	Numeric
quantity_slash_params	Number of ”/” signs	Numeric
quantity_questionmark_params	Number of ”?” signs	Numeric
quantity_equal_params	Number of ”=” signs	Numeric
quantity_at_params	Number of ”@” signs	Numeric
quantity_and_params	Number of ”&” signs	Numeric
quantity_exclamation_params	Number of ”!” signs	Numeric
quantity_space_params	Number of ” ” signs	Numeric
quantity_tilde_params	Number of ” ” signs	Numeric
quantity_comma_params	Number of ”,” signs	Numeric
quantity_plus_params	Number of ”+” signs	Numeric
quantity_asterisk_params	Number of ”*” signs	Numeric
quantity_hashtag_params	Number of ”#” signs	Numeric
quantity_dollar_params	Number of ”$” signs	Numeric
quantity_percent_params	Number of ”%” signs	Numeric
params_length	Number of parameters characters	Numeric
tld_present_params	TLD present in parameters	Boolean
quantity_params	Number of parameters	Numeric

3.1.2 Web-Scrapped Features

Certain features of an URL have come out to be crucial for classification which does not depend on the lexical features of the URL so they have been extracted by accessing the web for each URL. Below are such features:

Table 6: External Attributes Description

Attribute	Format	Description
time_response	Domain lookup time response	Numeric
time_domain_activation	Domain activation time (in days)	Numeric
time_domain_expiration	Domain expiration time (in days)	Numeric
quantity_redirects	Number of redirects	Numeric
url_shortened	Is URL shortened	Boolean

3.2 Modelling

A total of 100 features for 350000 URLs were obtained after the feature extraction process which acts as our primary dataset. For model building, we divided the dataset into 75/25 ratio for train and test data.

We use a number of ML models from Scikit-Learn Library for the purpose of modelling to predict if an URL is Phishing or Non-Phishing.

3.2.1 K-Nearest Neighbours Algorithm

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

3.2.2 Naïve Bayes algorithm

Naïve Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

3.2.3 Logistic Regression

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative distribution function of logistic distribution.

3.2.4 Decision Trees

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogeneous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

3.2.5 Random Forests

Random forests create decision trees on randomly selected data samples, get predictions from each tree, and select the best solution by means of voting. It also provides a pretty good indicator of the feature importance.

3.2.6 Adaboost

AdaBoost works by choosing a base algorithm (e.g. decision trees) and iteratively improving it by accounting for the incorrectly classified examples in the training set.

3.2.7 XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.), artificial neural networks tend to outperform all other algorithms or frameworks.

3.2.8 Support Vector Machines

SVM is a binary classification algorithm. Given a set of points of 2 types in N-dimensional place, SVM generates a (N-1) dimensional hyperplane to separate those points into 2 groups. Say you have some points of 2 types on a paper which are linearly separable. SVM will find a straight line which separates those points into 2 types and situated as far as possible from all those points.

3.3 Performance Metrics

Confusion Matrix

A confusion matrix is a table used in classification to present the performance of a classification algorithm. It displays the number of true positives, true negatives, false positives, and false negatives.

Accuracy

Accuracy is a measure of the overall correctness of the model. It is the ratio of the number of correct predictions to the total number of predictions. Mathematically, it can be represented as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It measures the accuracy of positive predictions. Mathematically, it can be represented as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall (Sensitivity)

Recall is the ratio of correctly predicted positive observations to the all observations in actual class. It measures the ability of the model to correctly identify positive instances. Mathematically, it can be represented as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1 Score

The F1 score is the harmonic mean of precision and recall. It considers both false positives and false negatives. It is a useful metric when the classes are imbalanced. Mathematically, it can be represented as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.4 Model Deployment

After completing the modeling phase and creating a pickle file, we proceeded to deploy our model for real-time usage. Leveraging the Streamlit framework, we developed an application that assesses whether an input URL is potentially phishing or not. By running this application locally, users can conveniently access its functionality.

To extend its accessibility beyond local environments, we employed ngrok, a tool that generates a temporary public URL for accessing our locally hosted Streamlit app. This URL is obtained by activating an authtoken, allowing anyone with the link to utilize the application until the token remains active. This seamless deployment approach ensures broader accessibility and usability of our phishing URL detection system.

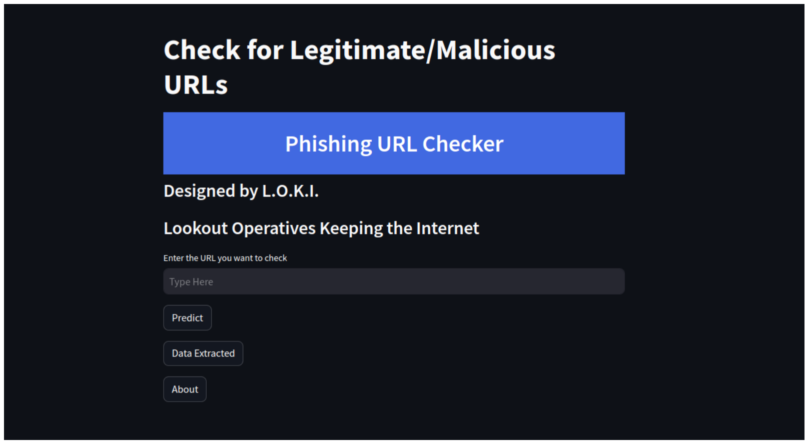

Figure 2: Deployed Model

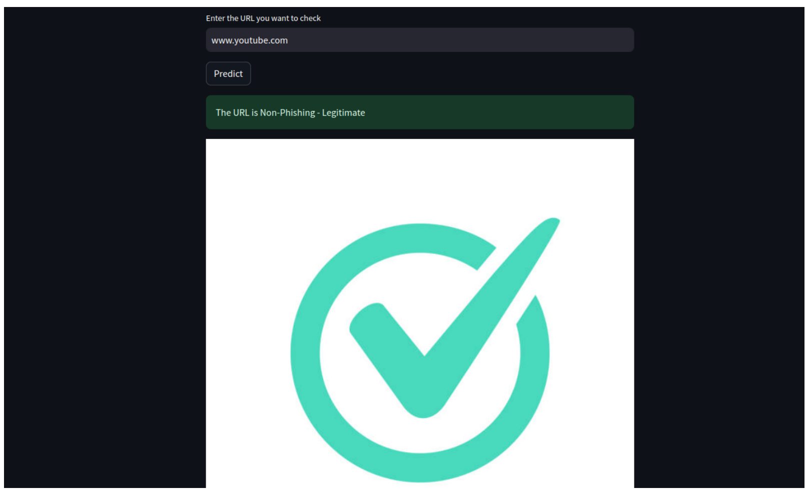

Figure 3: Demo of model

4 Experimental Result

4.1 Dataset

URLs of Phishing and Non-Phishing were obtained from a website named Gram Beddings under Hacettepe University, Turkey. They have collected the data by web-crawling through a long term between May 2019 to June 2021. In this way, they avoided to include duplicate URLs. During their data collection, the time in between each URL collection session was kept as one week to avoid repeating records. Moreover, they have designed and implemented their custom crawler to select and filter legitimate URLs to create a realistic sampling. Source: https://web.cs.hacettepe.edu.tr/ selman/grambeddings-dataset/

4.2 Experimental settings

From the source dataset we have extracted 101 features mentioned in the Proposed Methodology. Our further work on modelling has been done on these 101 features.

We choose k=7 for KNN since it gives the highest accuracy

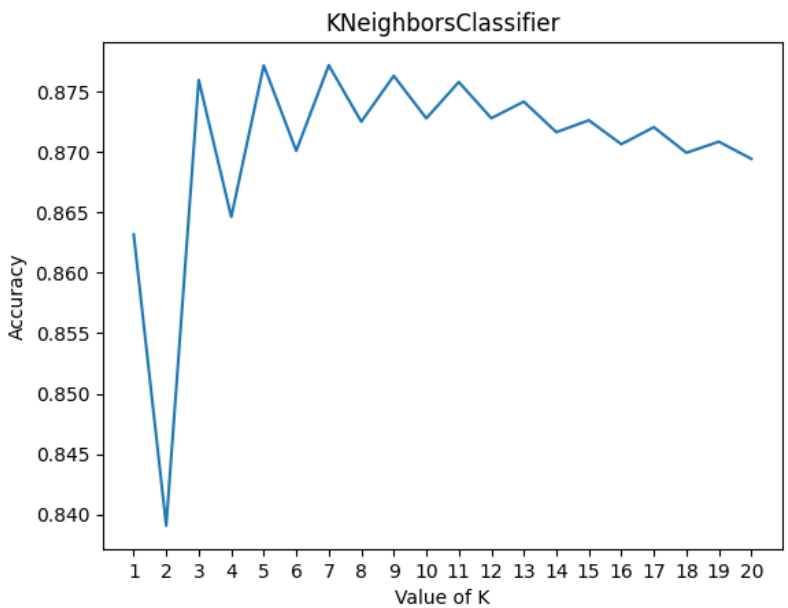

Figure 4: Comaprison of various values of k for KNN

Below are the Confusion Matrices for various Models

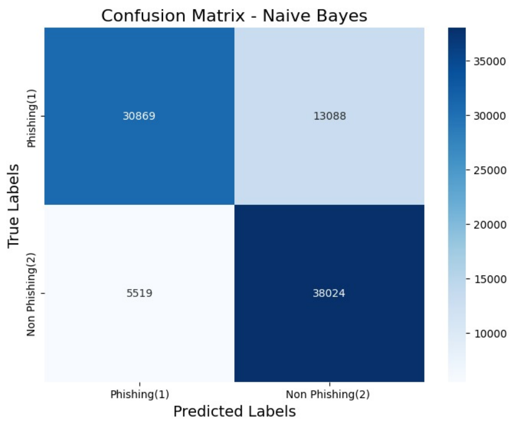

Figure 5: Naive Bayes

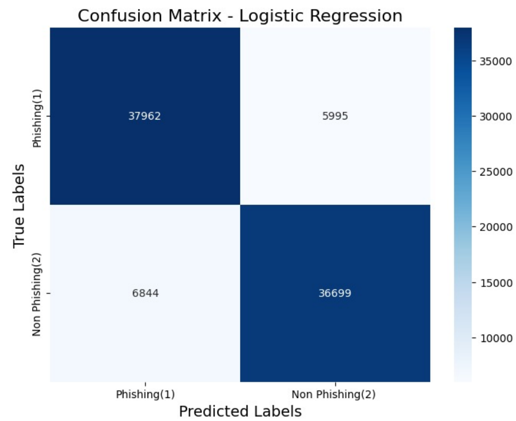

Figure 6: Logistic Regression

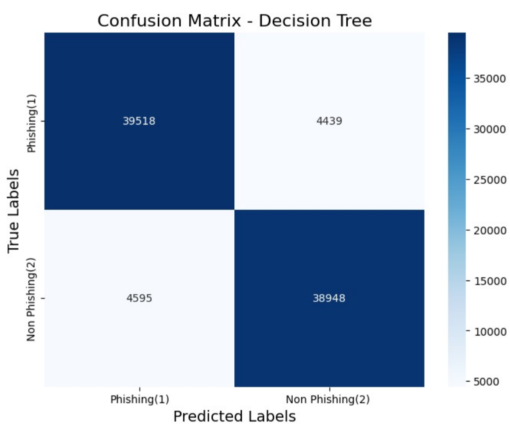

Figure 7: Decision Tree

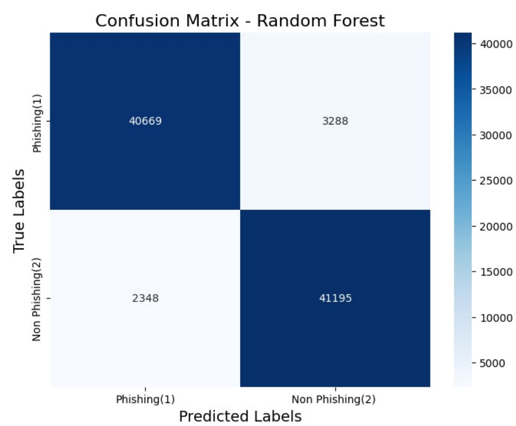

Figure 8: Random Forest

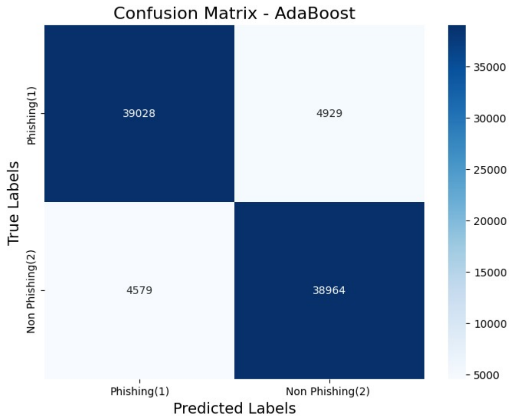

Figure 9: AdaBoost

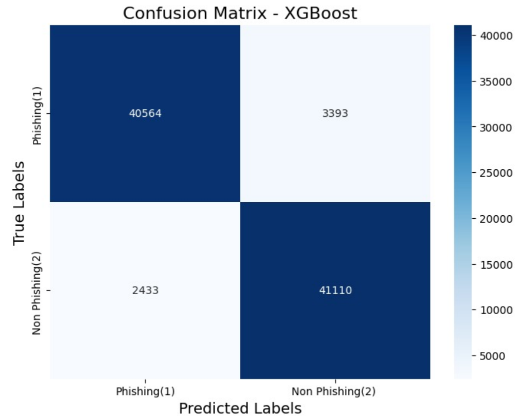

Figure 10: XGBoost

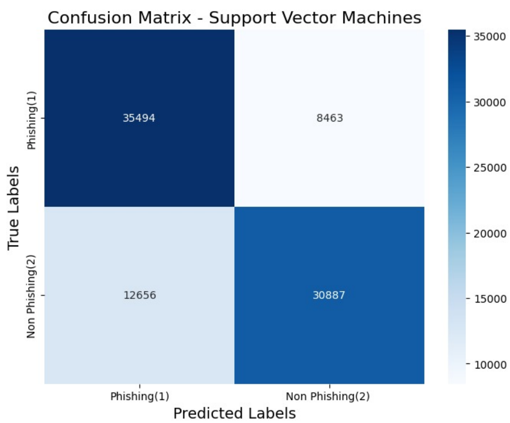

Figure 11: Support Vector Machine

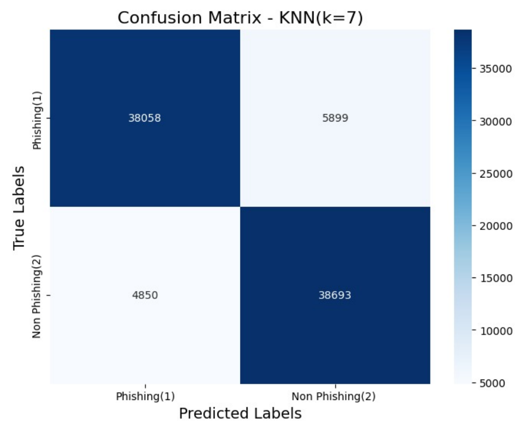

Figure 12: KNN (k=7)

Below is table consisting of names of the model and their performances: -

Table 7: Performance Metrics for Various Models

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Naive Bayes	0.787348571	0.848329119	0.702254476	0.768411227
Logistic Regression	0.853268571	0.8472526	0.863616716	0.855356398
Decision Tree	0.896754286	0.895835695	0.899014946	0.897422505
Random Forest	0.935588571	0.945416928	0.925199627	0.935199025
AdaBoost	0.891337143	0.894993923	0.887867689	0.891416564
XGBoost	0.933417143	0.923757949	0.944124199	0.933830043
Support Vector Machine	0.75864	0.737154725	0.807470938	0.770712324
KNN (k=7)	0.877154286	0.886967465	0.865800669	0.87625626

4.3 Experimental results and comparison with the state-of-the-art methods

Table 8: Machine learning based methods using automatic feature selection

PAPER	**METHOD**	**DATASET**	**PERFORMANCE**
Bahnsen et al. (2017)	An LSTM network was employed to embed raw URLs.	2 million phishing and legitimate URL.	98.7% accuracy
Wei et al. (2019)	A word embedding of raw URLs was applied on a Convolutional Neural Network.	999,996 legitimate URLs and 523,970 phishing URLs.	86.6% accuracy
Ozcan et al. (2021)	The study proposed a hybrid deep learning model that combines NLP features with character embedding prior to applying them to a DNN+LSTM model.	37385 phishing URLs and 36,400 legitimate URLs.	98.79% accuracy
Zhang et al. (2021)	MultiPhish was developed, incorporating a VAE to fuse the text, image, and URL features.	5887 phishing instances and 4259 legitimate instances.	97.79% accuracy

Table 9: Machine Learning based methods using manual feature engineering

PAPER	METHOD	DATASET	PERFORMANCE
Kumi et al. (2021)	The authors utilized 8 features in the CBA algorithm. The measure of the randomness factor in URLs emerged as the most significant feature.	700 phishing and 500 legitimate websites.	95.8% accuracy
Moghimi and Varjani (2016)	The authors applied 9 features to the SVM algorithm.	1448 phishing and 686 legitimate websites.	99.14% accuracy
Singh et al. (2015)	15 features from the URL, HTML and networks were implemented on Adaline with SVM.	79 phishing and 179 legitimate URLs.	99.1% accuracy
Mohammad et al.(2012b)	17 features were implemented on a self-learning neural network.	600 legitimate websites and 800 phishing websites.	92.18% accuracy
Chiew et al. (2019)	48 features were implemented on a Random Forest Classifier.	2456 legitimate and phishing instances.	96.17% accuracy
Rendall et al. (2020)	Used a multi-layered approach implemented on supervised machine learning algorithms.	17,244 legitimate and 7970 phishing domains.	89% accuracy on the SVM algorithm.
Yerima and Alzaylaee (2020)	In this study, 30 static features from the URL and HTML content of the web page were applied to a 1D convolutional neural network.	6,157 legitimate and 4,898 phishing websites.	98.2% accuracy
Maroofi et al. (2020)	38 features from the URL were applied to a random forest classifier to detect compromised domains.	41,002 URLs.	97% accuracy.
Aljofey et al. (2022)	Character-level Term Frequency-Inverse Document Frequency (TF-IDF) features were extracted from noisy parts of HTML and plaintext.	32,972 benign web pages and 27,280 phishing web pages.	96.76% accuracy

4.4 Time Complexity

Table 10: Time Complexity for Training and Prediction

Model	for Training	for Prediction
Naive Bayes	$O(n \times m)$	$O(m \times k)$
Logistic Regression	$O(k \times n \times m)$	$O(m)$
Decision Tree	$O(n \times m \times \log(n))$	$O(\log(n))$
Random Forest	$O(n \times m \times \log(n))$	$O(k \times \log(n))$
AdaBoost	$O(k \times m \times n)$	$O(k \times m)$
XGBoost	$O(n \times m \times \log(n)) - O(n \times m \times \sqrt{n})$	$O(m \times \log(n))$
Support Vector Machine	$O(n^2 \times m) - O(n^3 \times m)$	$O(m)$
KNN (k=7)	$O(1)$	$O(n \times m)$

Table 11: Abbreviations

Naive Bayes	n = no of samples, m = no of features, k = number of classes
Logistic Regression	n = no of samples, m = no of features, k = number of iterations
Decision Tree	n = no of samples, m = no of features
Random Forest	n = no of samples, m = no of features, k = number of trees
AdaBoost	n = no of samples, m = no of features, k = number of iterations
XGBoost	n = no of samples, m = no of features
Support Vector Machine	n = no of samples, m = no of features
KNN (k=7)	n = no of samples, m = no of features

5 Summary

This project tackles the urgent issue of phishing URL detection using machine learning models, recognizing the pervasive threat posed by cybercriminals in our digital landscape. It aims to provide users with real-time protection by leveraging advanced algorithms to identify potentially malicious URLs. Through rigorous dataset collection, feature extraction, and model training, the project equips users with proactive defence mechanisms against evolving cyber threats. By addressing this critical need, it contributes to fostering a safer online environment for individuals and organizations.

5.1 Some Key Points

1. **Necessity of the Project**: Phishing attacks remain a pervasive threat in our digital ecosystem, jeopardizing sensitive information and undermining trust in online interactions. With URLs being integral to our online experience, users are constantly exposed to potential phishing attempts. Hence, a robust solution for swiftly detecting phishing URLs is imperative to safeguard users and their data.

2. **Essential Elements**:

    - **Dataset Collection**: Acquiring a dataset containing URLs labelled as phishing or non-phishing is foundational. This dataset serves as the basis for training machine learning models.
    - **Feature Extraction**: Extracting relevant features from URLs is crucial. In this project, 100 lexical and web-scraped features were extracted to capture diverse characteristics that distinguish phishing from legitimate URLs.
    - **Model Selection and Training**: Employing various machine learning models to identify the most effective one for the task at hand. This involves training and fine-tuning models using the extracted features and the labelled dataset.
    - **Deployment on App**: Developing an application interface where users can input URLs and receive instant feedback on whether they are potentially phishing or not. This requires integrating the trained model into the app's backend.

3. **Future Implications**:

    - **Enhanced Security**: By deploying an efficient phishing URL detection system, users can navigate the internet with greater confidence, knowing that they have a tool to identify potential threats.
    - **User Awareness**: This project also contributes to raising awareness about phishing attacks and the importance of scrutinizing URLs before interacting with them, thereby empowering users to make informed decisions.
    - **Adaptability and Scalability**: As the digital landscape evolves and new phishing techniques emerge, machine learning models can be continuously updated and improved to adapt to these changes, ensuring ongoing protection for users.
    - **Potential Integration**: The methodologies and insights gained from this project can potentially be integrated into larger cybersecurity frameworks, contributing to a more comprehensive defence against online threats.

Overall, this project serves as a proactive measure to mitigate the risks associated with phishing attacks, ultimately fostering a safer online environment for users worldwide.

References

[1] Ali Aljofey, Qingshan Jiang, Abdur Rasool, Hui Chen, Wenyin Liu, Qiang Qu, and Yang Wang. An effective detection approach for phishing websites using url and html features. *Scientific Reports*, 12(1):8842, 2022.

[2] Alejandro Correa Bahnsen, Eduardo Contreras Bohorquez, Sergio Villegas, Javier Vargas, and Fabio A. González. Classifying phishing urls using recurrent neural networks. In *2017 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–8, 2017.

[3] Kang Leng Chiew, Choon Lin Tan, KokSheik Wong, Kelvin S.C. Yong, and Wei King Tiong. A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. *Information Sciences*, 484:153–166, 2019.

[4] Sandra Kumi, ChaeHo Lim, and Sang-Gon Lee. Malicious url detection based on associative classification. *Entropy*, 23(2), 2021.

[5] Sourena Maroofi, Maciej Korczyński, Cristian Hesselman, Benoît Ampeau, and Andrzej Duda. Comar: Classification of compromised versus maliciously registered domains. In *2020 IEEE European Symposium on Security and Privacy (EuroSP)*, pages 607–623, 2020.

[6] Mahmood Moghimi and Ali Yazdian Varjani. New rule-based phishing detection method. *Expert Systems with Applications*, 53:231–242, 2016.

[7] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25:443–458, 2014.

[8] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 3rd edition, 2012.

[9] Chidimma Opara, Bo Wei, and Yingke Chen. Htmlphish: Enabling phishing web page detection by applying deep learning techniques on html analysis. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[10] Alper Ozcan, Cagatay Catal, Emrah Donmez, and Behcet Senturk. A hybrid dnn–lstm model for detecting phishing urls. *Neural Computing and Applications*, pages 1–17, 2023.

[11] Kieran Rendall, Antonia Nisioti, and Alexios Mylonas. Towards a multi-layered phishing detection. *Sensors*, 20(16), 2020.

[12] David G. Stork Richard O.Duda, Peter E. Hart. *Pattern Classification*. John Wiley Sons, 2nd edition, 2000.

[13] Priyanka Singh, Yogendra P.S. Maravi, and Sanjeev Sharma. Phishing websites detection through supervised learning networks. In *2015 International Conference on Computing and Communications Technologies (ICCCT)*, pages 61–65, 2015.

[14] Sami Smadi, Nauman Aslam, and Li Zhang. Detection of online phishing email using dynamic evolving neural network based on reinforcement learning. *Decision Support Systems*, 107:88–102, 2018.

[15] Grega Vrbančič, Iztok Fister, and Vili Podgorelec. Datasets for phishing websites detection. *Data in Brief*, 33:106438, 2020.

[16] Suleiman Y. Yerima and Mohammed K. Alzaylaee. High accuracy phishing detection based on convolutional neural networks. In *2020 3rd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–6, 2020.