Detection of Phishing URLs using Machine Learning

MSc Research Project
Cyber Security

Emin Varghese
Student ID: x23154250

School of Computing
National College of Ireland

Supervisor: Arghir Nicolae Moldovan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Emin Varghese		
Student ID:	X23154250		
Programme:	MSc. Cybersecurity	Year:	2024 - 25
Module:	MSc. Research Project		
Supervisor:	Arghir Nicolae Moldovan		
Submission Due Date:	12/12/2024		
Project Title:	Detection of Phishing URLs using Machine Learning		
Word Count:	6690 words excluding references	Page Count: 20	

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Emin Varghese

Date: 12/12/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection of Phishing URLs using Machine Learning

Emin Varghese
x23154250

Abstract

Phishing websites are an increasing cybersecurity threat which puts both customers and business at risk as they compromise sensitive data. As phishing acts as the initial step in cyberattacks, early detection of these sites is important. Even though traditional methods, which rely on manual feature engineering are effective, they often find it difficult to catch up with the fishing tactics. This research emphasis on using machine learning (ML) to facilitate and enhance the detection process, improving its adaptability to rising threats. By reviewing URL-based features and training models on a mixed dataset, this study aims to make an innovative system that can easily identify phishing websites and zero-day attacks, improving the security measures and the decreasing the risks related to phishing attacks. PHIUSIIL and Mendeley datasets were preprocessed and then analyzed under six machine learning algorithms. The work also emphasizes the effectiveness of incremental learning solutions, especially for updating data in real-time. This work focuses on the application of machine learning (ML) for the detection of phishing URL. Using PHIUSIIL dataset and Mendeley dataset, a comparative analysis of URLs and content of different models was performed. The results prove the effectiveness of incremental learning strategies and bring focus on the need for strong and easily extendable methods adapted to the current threats.

1 Introduction

1.1 Background

The digital evolution of the society driven by the accessibility and affordability of the internet which changed our engagement with essential services like banking, shopping, and education, meanwhile these advancements also opened the door to new vulnerabilities. The number of exploitations or cyber-attacks that occur on online platforms are increasing rapidly thereby compromising the sensitive or confidential data of groups or individuals. The phishing attack plays an important role among these attacks and stands out as a major concern. By creating fake URLs that look like real websites thereby tricking the users into unknowingly sharing personal or financial information.

Phishing attacks continue to be a persistent problem thereby costing individuals and companies billions annually. The recent surveys and reports show phishing scams were responsible for more than 75% of data breaches. To mitigate these issues several technologies and strategies have been developed. Heuristic methods look for unusual patterns in URLs, while machine learning models analyze large datasets to identify phishing trends. Blacklists and whitelists help by maintaining lists of harmful or trusted websites. Additionally, user training helps individuals to recognize suspicious emails and links, thereby adding an extra layer of security. Traditional rule-based systems often fail to address the evolving tactics of cybercriminals which highlights the need for advanced detection mechanisms(*Detecting Phishing Websites Using Machine Learning - Javatpoint,* n.d.).

1.2 Research Question

How can machine learning models effectively detect phishing URLs with high precision, scalability and adaptability to real-time threats?

1.3 Objectives

- Establish a strong machine learning framework that shall be used to detect the phishing URLs.
- Incorporating structural and content-based features has the potential of improving the results produced by the models.
- Find the pros and cons of one model over another to come up with the best model learning technique.
- Leverage structural and content-based feature extraction to enhance model accuracy.
- Compare the performance of various ML models to identify the most efficient approach.
- Ensure the scalability and adaptability of the detection system for real-world applications(Dutta, 2021).

1.4 Contributions

- The comparative evaluation of six ML algorithms on different datasets.
- Providing insights into incremental learning techniques for real-time adaptability.
- Implementation of comprehensive feature extraction, including URL, HTML, and derived features.

This research project focuses on detecting phishing URLs using machine learning, thereby implementing adaptability and pattern recognition capabilities of algorithms which incorporate similarity indices and incremental learning. The primary aim of this study is to address the challenges in phishing detection which includes scalability and real-time adaptability to new attack vectors. This approach also focuses on extracting and analyzing a wide range of features from URLs and the associated webpage content making it more adaptive and reliable. The project aims to contribute a robust solution against phishing threats in an evolving era of digital transformations(*Phishing URL Detection with ML. Phishing Is a Form of Fraud in Which… | by Ebubekir Büber | Towards Data Science*, n.d.).

The paper is structured as follows; Section 2 reviews literature to contextualize our study. Section 3 elaborates on data and methods. Section 4 discusses design specifications. Section 5 covers implementation. In Section 6, we evaluate the framework's performance. Finally, Section 7 concludes with findings and future directions.

2 Related Work

The detection of a phishing URL is important in the fight against cyber threats mainly because it employs deceptive URLs to capture information. This review aims to present an overview of the most recent developments, issues and best trends for efficiently detecting and mitigating efficient phishing URLs.

In this research, phishing is considered as a significant cyber threat where attackers exploit deceptive URLs to steal sensitive user information such as passwords, financial details, and account credentials. Arvind Prasad et al. (2023) address the limitations such as reliance on small datasets, static training, and third-party dependencies of existing phishing detection methods, thereby introducing a robust phishing URL detection framework to mitigate these gaps. This framework integrates a dataset of 235,000 entries, a URL Similarity Index for visual filtering and incremental learning algorithms (BernoulliNB, PassiveAggressive, SGD) for adaptability. PhiUSIIL framework achieves exceptional accuracy of 99.79% offering a scalable, versatile, and comprehensive solution against modern phishing threats(Prasad & Chandra, 2024).

Liaquathali, S. and Kadirvelu, V. (2024) discusses advancements in web security emphasizing on the usage of natural language processing and machine learning in identification of compromised web pages. Static based techniques lack effectiveness against sophisticated targets because these change in definition over time. NLP analyzes webpage content by leveraging techniques like tokenization and semantic analysis, meanwhile ML approaches such as SVM and artificial neural networks classify behaviors. Integration of text based, and structural based models are useful but there are issues such as scalability, adversarial method and interpretability of models. Future directions offer robust and scalable solutions to web content analysis(Liaquathali & Kadirvelu, 2024).

Hendaoui a et al. (2024) highlight the challenges of phishing detection, thereby emphasizing the limitations of traditional methods like heuristic analysis and machine learning. To this end, they introduce SENTINEY which is a new framework that combines Secure Multi-Party Computation (SMPC) with machine learning to overcome these challenges. SENTINEY optimizes its methods of detection according to the features of the attack and available resources with results of 99.4% accuracy. Through network virtualization, the system guarantees scalability and efficiency from the network, meanwhile maintaining the privacy of the data. SENTINEY is considered as a secure and private service in the context of new and developing types of phishing(Hendaoui & Hendaoui, 2024).

D. Jennifer Dsouza and A. P. Rodrigues and R. Fernandes conducted a comparative analysis on the detection of phishing techniques using artificial intelligence with offline, batch and incremental modes of evaluation. Their work focused equally on techniques of feature selection, heatmap correlation and random forest importance. The datasets from Mendeley and SMS spam datasets showed that the classifiers employing the adaptive random forest in the incremental mode have 97.1% accuracy, while deep learning models like the Keras Sequential one as high as 99.28%. They pointed out that incremental processing can be ideally suited for real-time applications because the computational cost is low(Dsouza et al., 2024).

K. S. Jishnu and B. Arthi also presented an effective phishing detection smart model using BERT model features with URL features extraction technique. It combines contextual understanding derived from BERT tokenization with structure URL features including token length, domain, and token containing special characters. To extract the input features, the dataset of 200,000 URLs is preprocessed and tokenized and by including those features authors got an accuracy of as high as 97.32%. Using BERT embedding and fine-tuning and integrating the dataset with URL features enhanced its performance in identifying phishing attempts. With this approach, excellent performance was achieved in comparison to the baseline methods thus making it easier to protect against threats such as phishing(Jishnu & Arthi, 2023a).

K. S. Jishnu and B. Arthi used RoBERTa for feature extraction and Long Short-Term Memory for classification of the proposed phishing detection system. Their method builds on RoBERTa's word-contextualization ability to capture semantic content of the URLs while using LSTM to capture temporal dependencies. For the experiments, a dataset containing 300000 URLs was used which included 150000 real URLs and 150000 phishing cases. The system's analysis reached a level of accuracy of 97.14% more efficiently in comparison with heuristic approaches. From key performance indicators perspective using precision, recall and F1-score for model evaluation, it proved that it has good results to classify legitimate links from phishing ones. Using RoBERTa for semantic analysis and LSTM for sequential learning, the paper presented an improved method in this area for real-world applications of the model. This integration improves the security measure by offering dependable safeguard against the attacks(Jishnu & Arthi, 2023b).

K. S. Jishnu and B. Arthi detected phishing URLs and reveal drawbacks of conventional machine learning approaches such as CatBoost and XGBoost which require feature engineering and have no retrain ability. CNN and LSTMs demonstrate enhanced accuracy but fail to address the real-time application of the model. Considering these gaps, the authors introduce a new Knowledge Distilled ELECTRA approach. This system integrates deep learning with real-time browser integration to predict with 99.74% accuracy and this is implemented on the fact that the authors update the algorithm with feedback from users in real-time, thereby preventing preset threats which makes it a benchmark in cybersecurity research(Jishnu & Arthi, 2024).

The paper by Vajrobol, Gupta and Gaurav presents a strategy for detecting phishing URLs using mutual information (MI) for feature selection and logistic regression for classification. Phishing attacks exploit deceptive URLs to obtain sensitive user information. The authors focus on identifying significant URL features through MI to improve model accuracy and efficiency. By combining MI's feature selection with logistic regression's predictive capabilities, their method achieves high detection rates and reduced false positives, offering an effective solution for real-time phishing detection in cybersecurity applications(Vajrobol et al., 2024).

Paper Title	Dataset Used	Preprocessing /Transformation	Feature Extraction	ML Algorithms Used	Evaluation Setting	Best Model Accuracy
PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning	PhiUSIIL phishing dataset (134,850 legitimate and 100,945 phishing URLs)	Continuous collection of URLs; parsing and cleaning of URL and HTML content, feature transformations such as CharContinuationRate and URLTitleMatchScore	Extracted URL features (e.g., TLD, subdomains) and HTML elements (e.g., title, redirects) with crafted patterns capturing phishing tendencies	Ensemble of BernoulliNB, PassiveAggressive, SGD (incremental learning algorithms)	Incremental learning per record; security profiles (Low, Best, High) with varying algorithm consensus for classification	Achieved up to 99.79% accuracy with pre-training approach on PhiUSIIL dataset
WCA: Integration of NLP and ML for Web Content Classification of Malicious Webpages	ISCX-URL2016, UNB, PhishTank; includes 5,530 benign URLs and 5,882 malicious URLs	Extraction and cleaning from HTML tags (<div>, <meta>, <para>); special character removal, lowercasing, stop-word removal, lemmatization; vectorization using Count, TF-IDF, Hashing	Tag-specific features using Count Vectorizer, TF-IDF, Hashing to capture semantic patterns in <div>, <meta>, <para> tags	Logistic Regression, Gaussian Naive Bayes, K-Nearest Neighbors, Decision Tree, Random Forest, Gradient Boosting, Extreme Gradient Boosting	Binary classification with accuracy, precision, recall, and F1-score; experiments on individual and combined tags across vectorizers	93.46% (Random Forest with Count Vectorizer)
SENTINEY: Securing ENcrypted mulTI-party computatIoN for Enhanced data privacY and phishing detection	Custom dataset created with 10,000,000 email messages (30% containing phishing URLs from OpenPhish and 70% legitimate messages)	Generated phishing and legitimate emails with common phishing terms; vectorization using TF-IDF; encrypted with Partially Homomorphic Encryption for secure processing	Enciphered TF-IDF vectors with clustering scores and similarity distances between encrypted patterns for phishing detection	Unsupervised models (Isolation Forest, K-means) for clustering/ anomaly detection; Random Forest, Multi-Layer Perceptron, Gradient Boosting	SMPC-based adaptive learning between unsupervised and string-matching models based on accuracy, volume of threats, and computational constraints	99.4% (Multi-Layer Perceptron with SMPC)
Multi-Modal Comparative Analysis on Execution of Phishing Detection Using AI	1. Mendley Phishing Dataset: 11,055 URLs with 32 features 2. UCI SMS Spam/Ham Dataset: 5,574 messages	For URL data: Checked data balance, normalized with standard scalar Removed irrelevant features using heat map correlation	Heat map correlation to remove redundant features Random Forest Feature	Random Forest, Extra Trees, Hoeffding Tree, Approximate Large Margin Algorithm	Offline Mode Batch Mode: PyTorch and PySpark Incremental	99.28%

	labeled as spam or ham 3. Custom Phishing Dataset: 5,000 URLs with 64 features	For SMS data: Applied tokenization, removed stop words and punctuation, vectorized, applied TF-IDF weighting	Importance to retain significant attributes HTML, JavaScript, domain, and address bar-based features	(ALMA), AdaBoost, Gradient Boost, KNN, SVM, Logistic Regression, Naive Bayes, Decision Tree, Adaptive Random Forest	Mode: River Framework	
Enhanced Phishing URL Detection Using Leveraging BERT with Additional URL Feature Extraction	200,000 URLs collected from PhishTank, Kaggle, Majestic Million	Removal of duplicates, class balancing to maintain dataset integrity	URL features extracted: URL length, domain length, subdomain count, path length, query parameter count, presence of special characters, IP address presence, URL redirects	Fine-tuned BERT model with a classification layer, combined with extracted URL features	The dataset was split into training and testing subsets using an 80/20 train-test split.	The model achieved an impressive accuracy of 97.32% on the test set.
Phishing URL Detection by Leveraging RoBERTa for Feature Extraction and LSTM for Classification	300,000 URLs collected from Kaggle, Majestic Million, and PhishTank (150,000 phishing and 150,000 legitimate)	Tokenization using the RoBERTa tokenizer; padding and truncation to ensure uniform length URLs	RoBERTa embeddings for contextual information from URLs	RoBERTa for feature extraction combined with a two-layer LSTM for classification	80/20 train-test split	97.14%
Real-time Phishing URL Detection Framework Using Knowledge Distilled ELECTRA	450,176 URLs sourced from Kaggle, PhishStorm, PhishTank, and Majestic Million (345,738 legitimate and 104,438 phishing)	Elimination of duplicates, handling missing values, removal of irrelevant columns, label validation	Tokenization and input encoding using ELECTRA's tokenizer for feature representation	Knowledge Distilled ELECTRA model optimized for efficient URL classification	80% training, 10% validation, and 10% test split; batch size of 16, max sequence length of 512	99.74%

3 Research Methodology

Phishing URLs remains a significant cybersecurity threat which is constantly evolving every day. The effective detection of these threats requires a well-rounded and systematic approach along with technical accuracy and human insights. This framework suggests the steps to develop a machine learning based detection system that integrates data collection, preprocessing, feature extraction, model training, and evaluation. The main aim of this framework is to implement a robust detection model than can detect phishing URLs from the legitimate ones based on structural and content-based features(*KDD Process in Data Mining - GeeksforGeeks*, n.d.).

3.1 Data Selection

The data collection plays a crucial role in the development of a machine learning model based on the quality and diversity of data collected. For this research, the phishing URLs datasets were gathered from a few reliable online sources and data repositories such as Kaggle, UC Irvine Machine Learning Repository and Mendeley Data to ensure a comprehensive view of the current phishing tactics.

In this research, I employed two datasets for the development and training of machine learning model for detection of phishing URLs. The PHIUSIIL Phishing URL Dataset has a total of 134,850 safe and 100,945 phishing URLs with important features extracted from URL and webpage source code such as CharContinuationRate, URLTitleMatchScore, URLCharProb, TLDLegitimateProb which provides deeper insights into the characteristics of phishing URLs(*PhiUSIIL Phishing URL (Website) - UCI Machine Learning Repository*, n.d.). The Mendeley dataset is comprised of 450,176 URLs with 104,438 identified as phishing and 345,738 identified as legitimate, providing a large set of data for effective detection of the machine learning models(KAITHOLIKKAL & B, 2024). These datasets provide a strong foundation for phishing threat assessment.

3.2 Data Preprocessing

The data preprocessing deals with the conversion of the large volume of unstructured and raw data into usable data making it suitable for analysis and development of the machine learning model. The quality of data is ensured by handling missing values, consistency in the data and converting data into suitable formats for further analysis. Furthermore, data is divided into training, validation and testing to prevent overfitting and to ensure unbiased evaluation.

3.3 Feature Engineering

Feature extraction is important in the design of framework developed for the identification of phishing URLs. It involves three categories which includes URL features, HTML features, and derived features. Feature engineering aims at modifying the raw data to come up with better features for use within a model(*What Is Feature Extraction? - GeeksforGeeks*, n.d.).

Category	Features
URL-based	URLLength, URLSimilarityIndex, CharContinuationRate, URLCharProb, ObfuscationRatio, LetterRatioInURL, DegitRatioInURL, SpacialCharRatioInURL, NoOfEqualsInURL, NoOfQMarkInURL, NoOfAmpersandInURL, IsHTTPS
Domain-based	Domain, DomainLength, IsDomainIP, TLD, TLDLength, TLDLegitimateProb, NoOfSubDomain
Obfuscation-based	HasObfuscation, NoOfObfuscatedChar, ObfuscationRatio
Character counts	NoOfLettersInURL, NoOfDegitsInURL, NoOfOtherSpecialCharsInURL
HTML-based	LineOfCode, LargestLineLength, HasTitle, DomainTitleMatchScore, URLTitleMatchScore, HasFavicon, IsResponsive, HasDescription, NoOfPopup, NoOfiFrame, HasExternalFormSubmit, HasSocialNet, HasSubmitButton, HasHiddenFields, HasPasswordField
Keyword-based	Bank, Pay, Crypto
Content-based	HasCopyrightInfo, NoOfImage, NoOfCSS, NoOfJS, NoOfSelfRef, NoOfEmptyRef, NoOfExternalRef
Labels	Label

Table 3.3.1 : Extracted Features

3.4 Model Development

The fundamental concept of this methodology involves developing and training machine learning models. Machine learning models like naive bayes classifiers such as gaussian naïve bayes, multinomial naive bayes and Bernoulli naive bayes and linear classifiers including passive aggressive classifier and perceptron are taken into consideration based on the requirements of the problem. Model performance can be optimized by hyperparameter tuning conducted via grid or random search. Ensemble approaches are used to increase the model validation by integrating the strength of multiple algorithms(*Naive Bayes Classifiers - GeeksforGeeks,* n.d.).

Gaussian Navie Bayes Classifier

In gaussian naive bayes classifiers, the features are distributed in a normal fashion. It is most effective when applied to continuous data. The model is defined using *sklearn.naive_bayes* with the partial fit method enabling incremental training on subsets of the data. Updating the parameters row by row in the training dataset making it suitable for large scale or streaming based datasets. From the results obtained, the above measures of accuracy, precision, recall, F1 score and CPU time for learning and testing are obtained offering a comprehensive ability of the model to classify the data(*Understanding Gaussian Classifier | by Rina Buoy | The Startup | Medium*, n.d.).

Multinomial Navie Bayes Classifier

The multinomial naive bayes classifiers is particularly useful for datasets that contain discrete count data. This model is done using *sklearn.naive_bayes* along with the usage of *partial_fit* method. The model training process iterates over rows of the training data, thereby incrementally updating the probabilities of the data. The classifiers performance is analysed based in training and testing datasets which enhances the ability to adapt to the specific data distribution(*Multinomial Naive Bayes - GeeksforGeeks*, n.d.).

Bernoulli Navie Bayes Classifier

The bernoulli naive bayes is appropriate for binary data where the features are the existence or non-existence of specific characteristics (For example, word in a document). The implementation uses *sklearn.naive_bayes* and *partial_fit* method with an argument for incremental learning which is set to false by default. The training and prediction performance of bernoulli naive bayes classifiers shows satisfactory results, thereby highlighting high accuracy and computational efficiency specially for datasets with binary features(*Bernoulli Naive Bayes*, n.d.).

Passive Aggressive Classifier

The passive aggressive classifier is a linear model developed for online learning. This approach makes it very useful for large datasets as well as problems which require real time classifications(*Build a Passive Aggressive Classifier Model & Deploy It as WebApp. | by Gurmanjot Singh Cheema | Medium*, n.d.). The classifier applies *sklearn.linear_model*, which is a versatile implementation including classification and regression. Classification is performed using the hinge loss on the hyperplane, the training process proceeds through each element of the dataset invoking adjustments to the hyperplane. The incremental nature of the model fastens updates and efficiency of handling streaming data(*Passive Aggressive Classifiers - GeeksforGeeks*, n.d.).

Perceptron

This is a simple linear classifier implemented for using step function to make predictions. It works by taking, input features, multiplying these features by weight, sum the weights and then passes the features through activation function which is usually a step function to produce 0/1. In this context, perceptron may be employed to distinguish between phishing (labelled as 1) and safe (marked by 0) addresses. It also fits the model which performs prediction using the training and testing datasets and provides a metrics of accuracy, precision, recall, and F1-score. A leak accuracy and F1-score mean a better performance, but low precision or recall rate can be problematic. Accuracy should not be the only consideration when choosing our models, efficiency should also be considered. The Perceptron is powerful for linearly separable datasets and serves as a baseline for more complex models(*Perceptrons*, n.d.).

Stochastic Gradient Descent Classifier (SGD Classifier)

The Stochastic Gradient Descent (SGD) Classifier is used for the identification of the phishing URLs. This linear model gradually enhances weights learnt from data to reduce errors in prediction. The code uses *partial_fit* method to retrain the SGD Classifier, as this helps to adapt to new data. This classifier generates labels for both the training set and a testing set after being trained. The performance of the model is evaluated by comparing the results of predicted labels and the actual ones are measured with the help of accuracy, precision, recall and F1-score(*Stochastic Gradient Descent Classifier - GeeksforGeeks,* n.d.). Evaluation of these measures facilitates determination of the performance of SGD Classifier. The low value of precision points out to the fact that there could be several false positives. The low recall value implies that there could be several phishing URLs which could have been missed out.

3.5 Implementation Tools

Flexibility is achieved through proper and effective use of programming frameworks and tools. The packages like NumPy, Pandas can be used for data manipulation and data exploration and easy to use machine learning package is scikit-learn. Data exploration and result presentation software are Matplotlib and Seaborn. Google Colab are the main resource to work further and implement changes since they allow for including code, output and comments. Functionalities like version control systems such as git make sure that different datasets are reproducible throughout the different stages of application development and involves different developers.

3.6 Model Evaluation

The evaluation stage is crucial to assess the model's accuracy, reliability and suitability for the task. To quantify the performance common metrics such as accuracy, precision, recall, F1-score and ROC-AUC are utilised. Cross validation secures the model by dividing the dataset into multiple subsets to unseen data for training and testing. To enable targeted refinements diagnostic tool like confusion matrices are used which differentiates insights into false positives and false negatives. This meticulous assessment ensures the model is not only accurate but also interpretable and aligned with the problem's requirements.

4 Design Specification

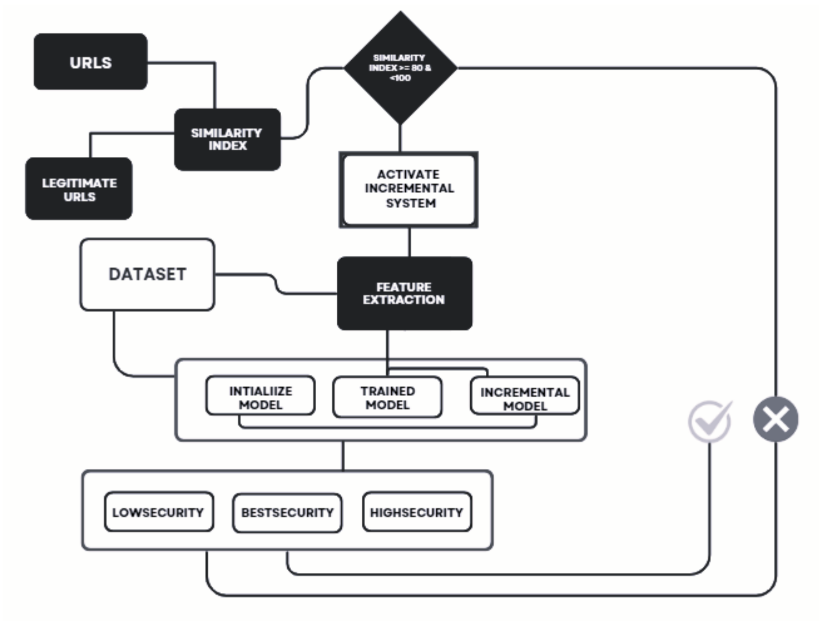

Figure 1 : Workflow Diagram

Phishing detection involves assessment of several attributes of a webpage with a view of identifying various threats. These characteristics are further divided into HTML-based characteristics, content characteristics, behaviour characteristics, link and navigation characteristics, media characteristics and derived characteristics.

4.1 HTML Features

LargestLineLenght : This feature is used to measure the length of the longest line of the code. These longer lines indicate hidden malicious content.

HasTitle : HasTitle is used to identify the presence of title tag in the code, the legitimate websites contain a descriptive title, whereas phishing websites lacks a good title to avoid detection.

HasFavicon : This feature detects a website logo of a webpage. Most of the legitimate webpage have a favicon tag which enhances brand identity. A webpage with no favicon tag can be categorized as a phishing URL.

IsResponsive : Used to evaluate responsiveness of webpage design to various devices. Most legal sites use responsive designs while phishing sites are not much responsive to devices.

NoOfURLRedirects : The features count the number of URL redirects embedded in the HTML code. Multiple redirects within HTML or JavaScript can lead users to unexpected phishing pages.

4.2 Content Features

HasDescription : This feature checks for a meta description tag in the HTML code. Legitimates websites use descriptions for SEO and user information. When a webpage does not have meta description tag, it may be interpreted as a false intent of the page.

HasCopyrightInfo : This concludes whether the webpage has copyright information included. Original websites sometimes contain information on copyrights that are usually missing in phishing sites.

HasSocialNet : This feature detects the presence of social media profile links in the webpage. Official social media accounts are supposed to be linked on genuine sites, but in phishing sites there won't be any social media accounts.

4.3 Interactive and Behavioral Features

NoOfPopup : Counts the number of pop-up windows generated by the webpage. Too many pop-ups might disrupt users' attention and attempt to receive personal details from the user.

NoOfiFrame : This feature counts the number of <iframe> tags embedded in the HTML code. These iframe elements are utilized to inject malicious content into the webpage, the often usage of iframe elements detects the presence of phishing URLs.

HasExternalFormSubmit : This feature discovers the HTML forms which submit data to external links. Links that direct the form with data to other URLs are usually a form of phishing.

HasHiddenFields : The websites with hidden fields embedded in the form fields are detected using this feature. The attacks can capture sensitive user information without any consent of the users.

4.4 Link and Navigation Features

NoOfSelfRef, NoOfEmptyRef, NoOfExternalRef : These features counts the hyperlinks and navigation patterns in the webpages. The excessive usage of such links detects phishing intentions. Phishing websites navigates to the same page, empty links and external links to steal sensitive information.

Bank, Pay, Crypto Keywords : These keywords are used to surf words related to financial transactions, phishing websites can attempt to collect confidential information using these keywords.

4.5 Media Features

NoOfImage : This feature counts the number of images on the webpage. Phishing sites may use numerous images with the aim of tricking the users into accessing the sites.

NoOfJS : Counts the number of JavaScript elements embedded in the webpage. Phishing sites often use JavaScript for misleading elements like pop-ups and redirects to other websites, thereby stealing personal information. The high usage of JavaScript's in a webpage increase suspicion on that website.

4.6 Derived Features

CharContinuationRate : The feature computes the longest contiguous sequences of similar characters in a URL. The genuine sites usually have easily recognizable and organized patterns of character, while their corresponding phishing sites have intricated and often senseless strings. A significantly lower CharContinuationRate means that the sequential or random character sequences are the typical characteristic of phishing URLs.

URLTitleMatchScore : This feature measures the level of similarity between a webpages, URL and the page title. A low score is the sign of mismatching, and this is observed in phishing sites. Sometimes, the title of the phishing URLs is manipulated to make the user think it is safe and in fact the address may have small changes or actual misspelling.

URLCharProb : The URLCharProb feature examines the character distributions of URLs and the character distributions of legitimate sites are then compared. The URL characters are common in phishing URLs and do not appear in genuine URLs. Phishing attempts tend to use the numbers 0 to 9 and the lower-case letters a, b, c and d occasionally but a higher frequency of the remaining characters especially 'q', 'v', 'x' and 'z' is normally associated with a phishing attack.

TLDLegitimateProb : TLDLegitimateProb determines the probabilities that a given URL's TLD is legitimate. Phishing sites creates new domains, while normal sites domain extension of the website usually ends in ".com," ".org," or country-specific extensions. A lower value of TLDLegitimateProb means the character='/www.' present in the URL may be using the TLD which is mostly used by phishing URLs.

5 Implementation

5.1 Dataset and Feature Extraction

The dataset containing 450,176 links of which 104,438 was labelled as phishing and 345,738 as non-phishing was first pre-processed before passing through the systematic designed pipeline to create a feature containing 54 features for the detection of phishing. The pipeline started with data loading that checks compatibility and checks for consistency in the extraction of features. Python functions were used to extract characteristics at the URL level such as the URL length and the number of special characters, the characteristics of the domain level such as the length of the domain and characteristics of the obfuscation. These various functions were applied with great efficiency in this process of large-scale record indexing because the process was fully automated by making it possible to apply these functions over all the records uniformly. The newly extracted dataset namely, Extracted_Phishing_URL_Features.csv is made suitable for all the requirements of the machine learning model. Through utilising domain knowledge and adopting structured data processing, the pipeline effectively captured features suitable for phishing detection while effectively responding to several significant cybersecurity issues accurately and flexibly.

5.2 Libraries

These libraries are essential for data preprocessing, machine learning, feature extraction, and utility operations in each part of the code(*Libraries in Python - GeeksforGeeks,* n.d.).

- **numpy** : A fundamental package for scientific computing in Python. It has the capability for working on large multi-dimensional array and matrices and has included a set of mathematical function this can operate on these structures.

- **pandas** : A remarkable library for manipulation and analysis of data. It contains functionalities for handling structured data through DataFrames, it contains tools to manipulate, clean and analyse large datasets.

- **re** : A library for working with regular expressions. It is used for pattern matching and document retrieval.

- **urllib.parse** : A module for navigating to different parts of an URL including scheme, netloc and path. A method applicable for extracting features from data in URLs.

- **random** : A generator of random numbers and sequences that can be used in cases such as random sampling and shuffling.

- **string** : Offers a range of constants and functions that work with text with sets of values (characters: letters, digits)

- **sklearn.model_selection** : Functions for splitting the data into training set and test set (like split in train_test_split) and for using the cross validation tool (cross_val_score).

- **sklearn.metrics** : A module for evaluating the performance of classification models such as accuracy, precision, recall and F1 score performance.

- **sklearn.preprocessing** : Functionality that allows scaling the data, encoding categories and normalizing the features.

- **matplotlib.pyplot** : A plotting library that allows designing static, animated, and interactive in python-based visualization. Most often applied in usage when using specific types of charts such as line, histogram and scatter charts.

- **os** : Contains functions to work with objects of the operating system, including files and directories.

- **time** : A library for time metrics used to measure the execution time of tasks or implementing delays.

- **warnings** : Utilized for controlling and filtering warning messages which help to resolve non-critical problems occurring during code execution.

- **pandas.concat** : A specific function in pandas that is used in operations of concatenating several DataFrames or series at a particular axis(*Best Python Libraries for Machine Learning - GeeksforGeeks*, n.d.).

6 Evaluation

6.1 Performance Metrics

The train_test_split is used on the dataset to create train and test split. This is a standard procedure to check how well a created model is going to generalize on new unseen data. The classification models that include Gaussian, Multinomial, Bernoulli, Passive Aggressive, Perceptron and SGD are used with the training data set. For some models, the partial fit method is used which allows incremental learning. For each model, the performance metrics are calculated on both testing and training data. Performance of all the models is stored in pandas DataFrame named PERFORMANCE. The performance metrics covers model, accuracy, precision, recall, f1score, training time and prediction time.

Machine Learning Models	**Accuracy**	**Precision**	**Recall**	**F1Score**	**TrainingTime**	**PredictionTime**
GaussianNB_Training	0.572048	0.572048	1.000000	0.727774	207	0
GaussianNB_Prediction	0.571538	0.571538	1.000000	0.727361	207	0
MultinomialNB_Training	0.689227	0.652747	0.975906	0.782265	240	0
MultinomialNB_Prediction	0.690694	0.653585	0.976255	0.782980	240	0
BernoulliNB_Training	0.986762	0.979267	0.997988	0.988539	269	0
BernoulliNB_Prediction	0.986500	0.978752	0.998046	0.988305	269	0
PassiveAggressive_Training	0.993954	0.993857	0.995584	0.994720	186	0
PassiveAggressive_Prediction	0.994162	0.994244	0.995548	0.994896	186	0
Perceptron_Training	0.941559	0.908479	0.998422	0.951329	256	0
Perceptron_Prediction	0.943638	0.911218	0.998689	0.952951	256	0
SGDClassifier_Training	0.944328	0.913169	0.997532	0.953488	191	0
SGDClassifier_Prediction	0.946578	0.916044	0.997997	0.955266	191	0

Table 6.1.1 : Performance Metrics of the PhiUSIIL phishing dataset (134,850 legitimate and 100,945 phishing URLs)

Machine Learning Models	Accuracy	Precision	Recall	F1Score	TrainingTime	PredictionTime
GaussianNB_Training	0.000104	0.000000	0.000000	0.000000	66	0
GaussianNB_Prediction	0.000061	0.000000	0.000000	0.000000	66	0
MultinomialNB_Training	0.814879	0.999968	0.814885	0.897989	54	0
MultinomialNB_Prediction	0.817176	0.999926	0.817225	0.899391	54	0
BernoulliNB_Training	0.999739	0.999896	0.999844	0.999870	59	0
BernoulliNB_Prediction	0.999696	0.999939	0.999757	0.999848	59	0
PassiveAggressive_Training	0.999896	0.999896	1.000000	0.999948	42	0
PassiveAggressive_Prediction	0.999939	0.999939	1.000000	0.999970	42	0
Perceptron_Training	0.999896	0.999896	1.000000	0.999948	54	0
Perceptron_Prediction	0.999939	0.999939	1.000000	0.999970	54	0
SGDClassifier_Training	0.999896	0.999896	1.000000	0.999948	42	0
SGDClassifier_Prediction	0.999939	0.999939	1.000000	0.999970	42	0

Table 6.1.2 : Performance Metrics of Mendeley dataset (104,438 phishing and 345,738 legitimate)

The PhiUSIIL dataset achieves moderate to high model performance, the PassiveAggressive, BernoulliNB, Perceptron and SGDClassifier models obtained an accuracy level above 94% and all the models obtained a level slightly less than one for Precision, Recall and F1 Score. GaussianNB is not very accurate which achieves only 57% on accuracy together with a low F1 score of 0.72. In case of the Mendeley dataset, the performance improved significantly. Based on the results, the models such as PassiveAggressive, BernoulliNB, Perceptron, and SGDClassifier give very high predictability nearly 99.99% on all forms of metrics. GaussianNB performs even worse with any kind of accuracy, precision, recall, as well as F1 scores coming close to zero. Training times are notably faster than that of the PhiUSIIL dataset. The PassiveAggressive classifier is the fastest for both datasets with also good accuracy.

6.2 Metrics DataFrame

The Metrics DataFrame evaluates voting strategies using key metrics including accuracy measures the level of correctness while on the other hand, precision captures the positive prediction. While sensitivity is estimator of how many real phishing URLs the program can identify and the F1 Score provides balance. MCC measures the performance in all aspects, while OneTime and IncTime defines the impact of training.

Mode	Accuracy	Precision	Sensitivity	F1Score	MCC	OneTime	IncTime
LowSecurity	0.984690	0.974046	1.000000	0.986852	0.969017	39	151
BestSecurity	0.998176	0.998156	0.998671	0.998413	0.996270	39	151
HighSecurity	0.992621	1.000000	0.987157	0.993537	0.985052	39	151

Table 6.2.1 : Metrics DataFrame of the PhiUSIIL phishing dataset (134,850 legitimate and 100,945 phishing URLs)

Mode	Accuracy	Precision	Sensitivity	F1Score	MCC	OneTime	IncTime
LowSecurity	0.999818	0.999818	1.000000	0.999909	0.000000	0	28
BestSecurity	0.999818	0.999818	1.000000	0.999909	0.000000	0	28
HighSecurity	0.909506	1.000000	0.909489	0.952599	0.042778	0	28

Table 6.2.2 : Metrics DataFrame of Mendeley dataset (104,438 phishing and 345,738 legitimate)

The model evaluates three security measures, namely Low Security, Best Security and High Security on two datasets. PhiUSIIL phishing dataset shows that the best security achieves the best accuracy of 0.998176, precision of 0.998156, sensitivity of 0.998671 and f1-score of 0.998413. The Mendeley dataset sees best security achieving perfect scores in accuracy, precision, sensitivity, and f1 score, while high security exhibits reduced performance with accuracy of 0.909506 and MCC of 0.042778. In both datasets one can notice that the profile of Best Security is shown as the most trustworthy one.

6.3 Confusion Matrix (CM) DataFrame

The Confusion Matrix (CM) DataFrame highlights prediction outcomes such as True Positive, True Negative, False Positive and False Negative. It identifies the frequency of errors and directs changes in the model.

Mode	TP	TN	FP	FN
LowSecurity	13548	9671	361	0
BestSecurity	13530	10007	25	18
HighSecurity	13374	10032	0	174

Table 6.3.1 : Confusion Matrix of PhiUSIIL phishing dataset (134,850 legitimate and 100,945 phishing URLs)

Mode	TP	TN	FP	FN
LowSecurity	5480	0	1	0
BestSecurity	5480	0	1	0
HighSecurity	4984	1	0	496

Table 6.3.2 : Confusion Matrix of Mendeley dataset (104,438 phishing and 345,738 legitimate)

The evaluations analyse the True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN) of three security profiles. In PhiUSIIL phishing dataset, I observe that High Security has both the highest true positive score of (13,374) and true negative score of (10,032) but has a very high false negative score of 174. The Best Security has a low FP and FN rate of 25 and 18 respectively which makes it the best performing model. In Mendeley dataset, the Best Security predominates with TP=5480, TN=0, FP=1, FN=0. The High Security again has the highest FN of 496 in Mendeley dataset and again underperforms the Low Security level. In general, observed that Best Security provides near best performance in both datasets and focuses on stability.

6.4 Discussion

The evaluation raised the methods which describes the effectiveness of various classification models for two datasets, PhiUSIIL and Mendeley, along with several evaluation metrics. It is noticed that the models like PassiveAggressive, BernoulliNB, Perceptron and SGDClassifier are quite efficient enough on both datasets where accuracy and F1- scores are higher than 94% for PhiUSIIL and nearly 80% for Mendeley. In Mendeley, GaussianNB performed even worse, barely providing all values are close to zero. The results of Security architecture analysis indicated that the Best Security evenly outperforms other profiles in terms of stability and reliability in two datasets in terms of precision, sensitivity, and F1 score with less false positive and false negative value. Overall, Best Security emerges as the most trustworthy and stable profile for phishing detection.

7 Conclusion and Future Work

The detection of phishing URLs continues to be an open problem area in cybersecurity because hackers are constantly getting smarter. This research endeavour was able to assess the applicability of machine learning models to this challenge with the help of broad datasets and varying feature types such as URLs, domains, and HTML properties. The current classification methods were implemented and tested along with key machine learning models like Gaussian Naive Bayes, Passive Aggressive Classifier, Stochastic Gradient Descent Classifier and so on to make the models more stable and responsive to emerging phishing threats. The results indicate that the proposed models are worthy of consideration and the accuracy, precision, recall, or F1 score of the two proposed models is relatively ideal. These outcomes indicate that machine learning is feasible and efficient scalable solution for phishing detection(Dutta, 2021).

The main target of detection of phishing URLs using machine learning in the future should be the enhancement of realm time adaptability and robustness against evolving cyber threats. One of the promising ways is the combination of continuous learning mechanism like incremental or online learning models to enable system to spontaneously adapt to new and sophisticated phishing tactics without extensive retraining. Moreover, addressing adversarial attacks is very important, that is the resilience of detection frameworks can be strengthened by incorporating adversarial training and model interpretability. Comprehensive understanding of phishing can be improved by using user interaction patterns and network traffic analysis which is the behavioural and contextual data. Another important aspect is the developing lightweight browser extensions or plugins that support trained models to provide real time protection to users. One of the main challenges is scalability and to handle a large scale and high traffic environments efficiently. The future research should produce cloud based or distributed architects. Finally, combining machine learning approaches with traditional heuristic or rule-based techniques could result in a hybrid system capable of providing robust and versatile defences. These enhancements can significantly improve the effectiveness and usability of phishing detection solutions in real-world circumstances.

References

Bernoulli Naive Bayes. (n.d.). Retrieved 12 December 2024, from https://iq.opengenus.org/bernoulli-naive-bayes/

Best Python libraries for Machine Learning - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/

Build a Passive Aggressive Classifier Model & Deploy it as WebApp. | by Gurmanjot Singh Cheema | Medium. (n.d.). Retrieved 12 December 2024, from https://medium.com/@gurmanjotsingh97/build-a-passive-aggressive-classifier-model-deploy-it-as-webapp-835f9a3094d5

Detecting Phishing Websites using Machine Learning - Javatpoint. (n.d.). Retrieved 12 December 2024, from https://www.javatpoint.com/detecting-phishing-websites-using-machine-learning

Dsouza, D. J., Rodrigues, A. P., & Fernandes, R. (2024). Multi-modal Comparative Analysis on Execution of Phishing Detection using Artificial Intelligence. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3491429

Dutta, A. K. (2021). Detecting phishing websites using machine learning technique. *PLoS ONE*, *16*(10), e0258361. https://doi.org/10.1371/JOURNAL.PONE.0258361

Hendaoui, F., & Hendaoui, S. (2024). SENTINEY: Securing ENcrypted mulTI-party computatIoN for Enhanced data privacY and phishing detection. *Expert Systems with Applications*, *256*, 124896. https://doi.org/10.1016/J.ESWA.2024.124896

Jishnu, K. S., & Arthi, B. (2023a). Enhanced Phishing URL Detection Using Leveraging BERT with Additional URL Feature Extraction. *Proceedings of the 5th International Conference on Inventive Research in Computing Applications, ICIRCA 2023*, 1745–1750. https://doi.org/10.1109/ICIRCA57980.2023.10220647

Jishnu, K. S., & Arthi, B. (2023b). Phishing URL detection by leveraging RoBERTa for feature extraction and LSTM for classification. *Proceedings of the 2023 2nd International Conference on Augmented Intelligence and Sustainable Systems, ICAISS 2023*, 972–977. https://doi.org/10.1109/ICAISS58487.2023.10250684

Jishnu, K. S., & Arthi, B. (2024). Real-time phishing URL detection framework using knowledge distilled ELECTRA. *Automatika*, *65*(4), 1621–1639. https://doi.org/10.1080/00051144.2024.2415797

KAITHOLIKKAL, J. K. S., & B, A. (2024). *Phishing URL dataset*. *1*. https://doi.org/10.17632/VFSZBJ9B36.1

KDD Process in Data Mining - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/kdd-process-in-data-mining/

Liaquathali, S., & Kadirvelu, V. (2024). WCA: Integration of Natural Language Processing Methods and Machine Learning Model for Effective Analysis of Web Content to Classify Malicious Webpages. *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *47*(1), 105–122. https://doi.org/10.37934/ARASET.47.1.105122

Libraries in Python - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/libraries-in-python/

Multinomial Naive Bayes - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/multinomial-naive-bayes/

Naive Bayes Classifiers - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/naive-bayes-classifiers/

Passive Aggressive Classifiers - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/passive-aggressive-classifiers/

Perceptrons. (n.d.). Retrieved 12 December 2024, from https://www.w3schools.com/ai/ai_perceptrons.asp

Phishing URL Detection with ML. Phishing is a form of fraud in which… | by Ebubekir Büber | Towards Data Science. (n.d.). Retrieved 12 December 2024, from https://towardsdatascience.com/phishing-domain-detection-with-ml-5be9c99293e5

PhiUSIIL Phishing URL (Website) - UCI Machine Learning Repository. (n.d.). Retrieved 12 December 2024, from https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset

Prasad, A., & Chandra, S. (2024). PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Computers & Security*, *136*, 103545. https://doi.org/10.1016/J.COSE.2023.103545

Stochastic Gradient Descent Classifier - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/stochastic-gradient-descent-classifier/

Understanding Gaussian Classifier | by Rina Buoy | The Startup | Medium. (n.d.). Retrieved 12 December 2024, from https://medium.com/swlh/understanding-gaussian-classifier-6c9f3452358f

Vajrobol, V., Gupta, B. B., & Gaurav, A. (2024). Mutual information based logistic regression for phishing URL detection. *Cyber Security and Applications*, *2*, 100044. https://doi.org/10.1016/J.CSA.2024.100044

What is Feature Extraction? - GeeksforGeeks. (n.d.). Retrieved 12 December 2024, from https://www.geeksforgeeks.org/what-is-feature-extraction/