Evaluating the Effectiveness of Machine Learning Algorithms in Detecting Phishing Attacks

MScResearchProject
MSC CYBER SECURITY

SUNIL KUMAR PRATURI
StudentID:X23242558

SchoolofComputing
NationalCollegeofIreland

Supervisor: **NIALL HEFFERNAN**

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	SUNIL KUMAR PRATURI
Student ID:	X23242558
Programme:	MSC CYBER SECURITY
Year:	2024
Module:	MSc Research Project
Supervisor:	NIALL HEFFERNAN
Submission Due Date:	12/12/2024
Project Title:	Evaluating the Effectiveness of Machine Learning Algorithms in Detecting Phishing Attacks
Word Count:	5863
Page Count:	21

Signature:	SUNIL KUMAR PRATURI
Date:	12nd December 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.	□

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

Evaluating the Effectiveness of Machine Learning Algorithms in Detecting Phishing Attacks

SUNIL KUMAR PRATURI
X23242558

Abstract

Phishing continues to be one of the most frequent and dangerous types of cyber attacks that result in liquidated funds and leakage of information. This paper describes the design and assessment of a machine-learning algorithm for the detection of phishing URLs. The objective of the study was to learn and categorise phishing websites with the help of machine learning algorithms such as Random Forest classifier, SVM classifier, k-NN classifier, Na¨ıve Bayes classifier and Decision Tree classifier. The study comprised an open-source dataset of features, including the length of the URL, similarity of domains, and previous response records, among others. This work shows the effectiveness of Random Forest and SVM with a precision of 0.9814, F1-Scores 0.9705, Specificity: 0.99 and Accuracy of 0.9974 and ROC-AUC value of more than 0.99. The study also shows how feature selection and hyperparameters should be carefully conducted in order to achieve the best outcome. Furthermore, the study presents a discussion of these results for theoretical and empirical research and practical applications in cybersecurity. Despite high Accuracy in standard datasets in challenged scenarios, the static models fail in dealing with imbalanced datasets and cannot incorporate the more numerous, but irrelevant indicators. In conclusion the authors provide suggestions for potential future work based on the idea of introducing other deep learning methods in order to enhance and improve the result of the real-time testing of the presented models.

1 Introduction

Phishing emerged as one of the serious threats to cybersecurity as more than four million phishing attacks were reported in 2023 around the world Group (2023). These are real scams: conning targets into providing personal details in the name of other organisations they claim to be from that result in embezzlement, privacy violation and damage to reputation. The most basic defense approaches like black lists and heuristic filters no longer offer the kind of safety that was needed want due to the new and continually evolving tactics used by hackers Beloglazov and Buyya (2015).

Considering the described tactics of changing the phishing threats, the cybersecurity society is shifting to utilised ML as proactive approach. The result is that with capabilities of processing large datasets, it becomes easy for ML algorithms to come up with patterns that are likely to point to phishing hence making it easier to detect new emerging threats in the market. For instance, a number of latest studies have unveiled that deep learning techniques including CNNs and RNNs are feasible for successful detection of phishing attempts at high degrees of accuracy Le et al. (2018). However, there is still a research gap concerning the assessment of the performance of the various ML techniques on various datasets.

The primary research question addressed in this study is: *How can machine learning algorithms be systematically evaluated and optimized for detecting phishing attacks?* To answer this, we will:

- Conduct a detailed comparison of supervised and deep learning algorithms.
- Identify the most relevant features contributing to accurate phishing detection.
- Provide practical implementation guidelines for real-world cybersecurity systems.

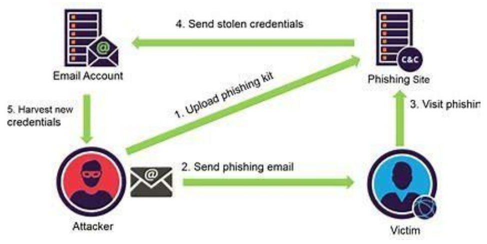

Figure 1: Phishing attack detection workflow using machine learning.

The structure of this document is as follows:

1. Section 2 reviews the state-of-the-art in phishing detection and machine learning applications.
2. Section 3 describes the methodology, including dataset collection, feature extraction, and algorithm selection.
3. Section 6 presents experimental results and their analysis.
4. Section 7 concludes with key findings and future directions.

2 Related Work

Deep research has been performed on the use of machine learning, (ML) in the detection of a phishing attack because of the advancement in the use of phishing. Existential section of this part systematically and comparatively presented the research papers, methodologies, findings, limitations, strength and weakness, and impacts on advance of the respective field of study.

2.1 Phishing Detection Using URL Features

Sahingoz et al. Sahingoz et al. (2019) have provided a comprehensive insight into the phishing detection mechanism using the URL characteristics of the same. A classification approach was adopted using Machine learning algorithms that includes Random Forest (RF), Support Vector Machines (SVM), and k-Nearest Neighbors (k-NN) for URL classification that involves the analysis of the lexical of the URL, host's features (IP address) and domain features (WHOIS information). From these algorithms, RF was found to have the best accuracy and was very sensitive to different dimensions since it selects features implicitly. This makes it especially helpful in discovering features that separate real URLs from the fake ones, such as phishing. However, the study noted that k-NN was outperformed particularly by high dimensional data on which it had poor performance because of complications caused by irrelevant features and its inability to handle big data well. In its performance, SVM carried out surprisingly well but often came with a major draw back since it involved a lot of tweaking in terms of the kernel functions when the environment is substantially dynamical. These findings emphasize the significance of feature extraction for differentiation of phishing, since fixed sets of features can be ineffective because of changes in an attacker's tactics[1].

2.2 Phishing Email Detection Using Content-Based Features

In their study, content features of phishing emails that include linguistic characteristics, metadata and structure were examined by Abbasi et al Abbasi et al. (2015). The current research aimed at comparing Na¨ıve Bayes and Decision Tree classifiers for the purpose of analyzing their efficiency in detecting phishing e-mails. The result show that the implemented algorithms in implementing Na¨ıve Bayes outperformed the others and with a high level of accuracy and low false-positive rate. This success can be traced to its probabilistic model, which reduces structural complexity associate with feature dependencies, and its computational ease, which makes it ideal for real time detection systems. However, the algorithm relies on the feature independence assumption, a questionable hypothesis in a considerable number of cases because features are not independent in many datasets. On the other hand, Decision Trees, though interpretable and easy to implement mis-performed on lesser number of data as they over-fit the train data. To improve the performance of the phishing detection system for different datasets according to the authors, techniques termed as ensemble methods that amalgamate several algorithms could be adopted.

2.3 Comparison of ML Algorithms for Phishing Detection

A comparative analysis of the Random Forest, SVM, and Na¨ıve Bayes algorithms using data sets collected from the Web for the purpose of detecting malicious URLs has been provided by Verma and Das Verma and Das (2017). These algorithms were evaluated with reference to accuracy, time efficiency, and ability to scale up. Random Forest outperformed the other models and had the highest detection rate and greater noise tolerance important for complex applications. SVM also did reasonably well but was again unable to scale up easily especially when there were a large number of attributes or observations. As for Na¨ıve Bayes it was pointed out that although it is not as accurate as RF it is simple and provides good performance for cases where computational resources are scarce. The study, therefore, called for the integration of both these algorithms in a way that optimally capitalizes on the expertise of the two in this field to enhance phishing detection systems.

[1] https://www.phishtank.com provides a repository of phishing URLs for further exploration.

2.4 Deep Learning Approaches in Phishing Detection

There has been a growing interest in deep learning techniques due to their autolayered feature extraction nature. Le et al. Le et al. (2018) used CNNs to differentiate between phishing and genuine websites from the view point of appearance resemblance. This approach proved ideal for detecting hierarchy within pixel data and yielded a far better performance than standard techniques. Nevertheless, the authors established that the CNN training was expensive in terms of computational resources and that the models required large datasets with labeled information. Likewise, Bahnsen et al. Bahnsen et al. (2017) used RNNs to identify phishing URLs, paying attention to the sequential behavior of the URLs. For instance, although RNNs performed well with the data, they had a high time complexity and had a tendency to overfit particularly where the data was scarce. These results underscore deep-learning capacity in detecting phishing sites, especially when feature extraction can be challenging; however, they also illustrate the requirement for optimization methods due to constraints on resources.

2.5 Limitations and Research Gaps

Although the reviewed studies demonstrate the effectiveness of machine learning in phishing detection, several limitations persist:

- **Static Features**: Many studies rely on static feature sets, which struggle to adapt to the dynamic and evolving nature of phishing attacks.

- **Resource Constraints**: Advanced algorithms like SVM and deep learning methods require significant computational resources, limiting their scalability for realtime applications.

- **Dataset Standardization**: The absence of standardized, comprehensive datasets hampers the reproducibility and comparability of results across studies[2].

- **Practical Deployment**: Few studies provide actionable guidelines for implementing ML-based phishing detection systems in real-world environments.

Addressing these gaps is crucial for advancing the field. This research systematically evaluates multiple ML algorithms across diverse datasets, focusing on feature optimization, computational efficiency, and real-world applicability. By building on the insights from previous studies, it aims to contribute to the development of robust and scalable phishing detection systems.

3 Methodology

This section describes the design of this research together with assessment of the various ML models used in detecting phishing. It includes data description and preparation;

[2] Standardized datasets like UCI ML Repository (https://archive.ics.uci.edu/ml/datasets) can help address this issue.

feature construction and selection; EDA; algorithm determination and application; parameter optimization; and model assessment. Every part of the study is described clearly in order to be able to replicate the study, and to confirm the scientific basis of the research.

3.1 Data Exploration and Preprocessing

The first step was data cleansing and feature engineering of all the collected datasets for use in the next phases of modeling. The set used in work includes the training set containing 700 instances and the testing set containing 300 instances Both datasets include 26 features relevant to the detection of phishing.

3.1.1 Loading and Inspecting the Datasets

The datasets were imported using the pandas library in Python. Initial inspection involved verifying the structure, data types, and absence of missing values to ensure data integrity. The following steps were undertaken:

- **Shape Verification**: Confirmed the number of instances and features in both training and testing datasets.
- **Data Information**: Examined data types and non-null counts to identify any inconsistencies.
- **Statistical Summary**: Generated summary statistics to understand feature distributions and identify potential outliers. • **Missing Values Assessment**: Checked for the presence of missing values to determine the necessity of data imputation.

The inspection revealed that both datasets are balanced with respect to the target variable and contain no missing values, ensuring data integrity and suitability for ML modeling.

3.1.2 Target Variable Identification

The target variable for classification was identified as the class column, which indicates whether a URL is legitimate (0) or malicious (1). All other columns were designated as feature variables. Ensuring a clear distinction between features and the target variable is crucial for effective model training and evaluation.

3.1.3 Class Distribution Analysis

To ensure that the dataset is balanced and to prevent biased model training, the distribution of the target variable was examined. A balanced dataset ensures that the models do not become biased towards the majority class, enhancing their ability to accurately detect phishing attempts.

3.2 Data Cleaning and Preprocessing

Handling Missing Values In both the training and testing sets, there were no instances of missing data that could be identified on visual inspection. This eliminates the requirement for data imputation methods, pointing on the fact that both datasets remain unaltered for feature scaling and modeling.

Feature Scaling Feature scaling was necessary so that all the numerical features are treated as equally important in the model training and feature engineering process. The standard scaler from sk-learn was used to scale the numeric input features to unit variance, that is, bring the mean to zero and standard deviation to one. This process is more crucial to features that are scale sensitive algorithms such as the Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN).

3.3 Feature Engineering

Feature engineering was used in order to improve the predictive accuracy of the models by creating new features and eliminating issues connected with multicollinearity.

3.3.1 Feature Extraction

An interaction feature between num dot and num slash was developed as it may be possible that these two attributes can reinforce each other's effect. Interaction features can provide models with insights on other associations of the data that could be difficult to detect if focusing with the help of other individual features.

3.3.2 Correlation Analysis

To mitigate multicollinearity, which can adversely affect model performance by inflating the variance of coefficient estimates, a correlation matrix was computed. Highly correlated feature pairs (absolute correlation coefficient greater than 0.5) were identified.

Feature Removal To reduce multicollinearity, the following features were removed from both the training and testing datasets:

- num dot
- num slash
- num hyphen

Removing these features ensures that the remaining features provide unique and nonredundant information to the models, thereby improving model interpretability and performance.

3.4 Exploratory Data Analysis (EDA)

EDA was conducted to uncover underlying patterns, detect anomalies, and visualize the relationships between features.

3.4.1 Visualization of Feature Distributions

Histograms and box plots were generated to examine the distribution of numerical features and identify potential outliers. These visualizations provided insights into the skewness and spread of the data, informing decisions related to feature scaling and engineering.

3.5 Model Selection and Implementation

The study subjected several ML algorithms to assess the most preferable models for the detection of phishing. The selected models cover the primary categories of algorithms which include ensemble models, linear models and instance-based models.

3.5.1 Selected Models

The following models were selected for evaluation:

- **Random Forest Classifier**: An ensemble method that builds multiple decision trees and merges their results for improved accuracy and control over overfitting. • **Support Vector Machine (SVM)**: A linear classifier effective in high-dimensional spaces and robust against overfitting.
- **k-Nearest Neighbors (k-NN)**: An instance-based learner that classifies based on the majority class among the nearest neighbors. • **Na¨ıve Bayes**: A probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between features.
- **Decision Tree Classifier**: A non-parametric model that splits the data based on feature values to make predictions.

3.5.2 Training and Evaluation Procedure

Each model was trained on the preprocessed training dataset and evaluated on the testing dataset using a comprehensive set of performance metrics. The evaluation metrics included Accuracy, Precision, Recall, F1-Score, and ROC-AUC to ensure a holistic assessment of each model's performance.

3.6 Hyperparameter Tuning

To enhance model performance, hyperparameter tuning was conducted using Grid Search with cross-validation. This step aimed to identify the optimal set of hyperparameters that maximize the models' predictive capabilities.

3.6.1 Random Forest Hyperparameter Tuning

A comprehensive parameter grid was defined for the Random Forest Classifier, exploring variations in the number of trees, tree depth, minimum samples required to split a node, minimum samples required at a leaf node, and whether bootstrap samples are used when building trees. Grid Search systematically evaluated these combinations to identify the configuration that yielded the highest F1-Score.

3.6.2 Support Vector Machine (SVM) Hyperparameter Tuning

Similarly, a parameter grid for the SVM was defined, examining different values of the regularization parameter (C), kernel types, and kernel coefficients (γ). Grid Search facilitated the determination of the hyperparameter settings that optimized the SVM's performance in detecting phishing instances.

3.7 Performance Evaluation

The performance of the trained models was evaluated using a comprehensive set of metrics to ensure a thorough assessment.

3.7.1 Evaluation Metrics

The following metrics were employed:

- **Accuracy**: The proportion of correctly classified instances.
- **Precision**: The ratio of true positives to the sum of true and false positives.
- **Recall**: The ratio of true positives to the sum of true positives and false negatives.
- **F1-Score**: The harmonic mean of precision and recall.
- **ROC-AUC**: The area under the Receiver Operating Characteristic curve, representing the model's ability to distinguish between classes.

3.7.2 Cross-Validation

To ensure the robustness and generalizability of the models, k-fold cross-validation was performed. This technique involved partitioning the training data into k subsets, training the model on $k - 1$ subsets, and validating it on the remaining subset. The process was repeated k times, and the results were averaged to provide an estimate of the model's performance on unseen data.

3.8 Limitations of the Methodology

Despite the rigorous methodological approach, certain limitations were identified:

- **Dataset Representativeness**: The study relies on specific datasets, which may not encompass the full diversity of phishing strategies employed in real-world scenarios.

- **Feature Scope**: Limited to the features present in the datasets, potentially overlooking other relevant indicators of phishing.

- **Model Complexity**: Balancing model complexity and interpretability posed challenges, particularly with ensemble and complex models.

- **Computational Resources**: Hyperparameter tuning and training of certain models required significant computational power, which may limit scalability.

These limitations highlight areas for future research and improvements in methodology.

The methodology detailed in this section provides a structured and comprehensive approach to developing and evaluating ML-based phishing detection models. By adhering to established practices and systematically addressing each phase of the research process, the study ensures the creation of robust, reliable, and effective models capable of identifying phishing attempts with high accuracy.

4 Design Specification

This section outlines the design architecture, underlying machine learning (ML) techniques with their mathematical formulations, frameworks, tools, and the rationale behind key design decisions for the phishing detection system. The design emphasizes scalability, robustness, and adaptability by leveraging established ML algorithms within a modular and efficient system framework.

4.1 System Architecture

The phishing detection system is architected as a modular pipeline comprising interconnected components that facilitate seamless flow from data ingestion to model evaluation. The high-level architecture is depicted in Figure **??**.

4.2 Underlying Techniques

The system employs several established ML algorithms, each selected for their unique strengths in classification tasks. Below are the key algorithms integrated into the system, accompanied by their mathematical formulations.

4.2.1 Random Forest Classifier

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) of the individual trees. It mitigates overfitting by averaging the results of multiple trees, enhancing generalization.

Mathematical Formulation The prediction of a Random Forest can be expressed as:

$$\hat{y} = \text{mode}\left(\{h_i(x)\}_{i=1}^{T}\right)$$

where $h_i(x)$ is the prediction of the i^{th} decision tree, and T is the total number of trees in the forest.

4.2.2 Support Vector Machine (SVM)

SVM is a supervised learning model that analyzes data for classification by finding the hyperplane that best separates the classes in the feature space, maximizing the margin between them.

Mathematical Formulation The decision boundary in SVM is defined as:

$$f(x) = w \cdot x + b = 0$$

where w is the weight vector, x is the feature vector, and b is the bias term. The objective is to maximize the margin between the two classes:

$$\text{maximize} \quad \frac{2}{\|w\|}$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1, \qquad \forall i = 1,2,\ldots,N$$

4.2.3 k-Nearest Neighbors (k-NN)

k-NN is a non-parametric, instance-based learning algorithm used for classification by identifying the k nearest neighbors to a query instance and assigning the most common class among them.

Mathematical Formulation The classification decision is based on:

$$\hat{y} = \operatorname{argmax}_{c \in C} \sum_{i=1}^{k} \mathrm{I}(y_i = c)$$

where C is the set of possible classes, y_i is the class label of the i^{th} nearest neighbor, and I is the indicator function.

4.2.4 Na¨ıve Bayes

Na¨ıve Bayes classifiers are probabilistic models based on applying Bayes' theorem with strong independence assumptions between features.

Mathematical Formulation The probability of a class c given a feature vector $x = (x_1, x_2, \ldots, x_n)$ is:

$$P(c|x) = \frac{P(c)\prod_{i=1}^{n} P(x_i|c)}{P(x)}$$

The predicted class is:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|x)$$

4.2.5 Decision Tree Classifier

Decision Trees are non-parametric models that recursively partition the feature space into regions with homogeneous class distributions.

Mathematical Formulation At each node t in the tree, the algorithm selects a feature j and a threshold θ to split the data such that the impurity (e.g., Gini impurity) is minimized:

$$\text{Gini}(t) = 1 - \sum_{c \in C} (p_c)^2$$

where p_c is the proportion of class c in node t.

4.3 Framework and Tools

The implementation of the phishing detection system leverages several robust frameworks and tools to ensure efficiency, scalability, and maintainability.

- **Python**: The primary programming language used for data manipulation, preprocessing, modeling, and evaluation due to its extensive ecosystem of ML libraries.
- **Pandas and NumPy**: Utilized for efficient data handling and numerical operations.
- **scikit-learn**: Employed for implementing ML algorithms, preprocessing, hyperparameter tuning, and evaluation metrics.
- **Matplotlib and Seaborn**: Used for data visualization, including plotting histograms, box plots, confusion matrices, and ROC curves.

These design choices collectively contribute to a system that is not only effective in detecting phishing attempts but also adaptable to future advancements and varying operational requirements.

5 Implementation

The application of the proposed phishing detection system leads to the development of efficient machine learning paradigms that are used in the classification of bad URLs. This section defines the last results obtained from the project, the tools and programming languages used in the project and methodology that was used to make the models more effective and accurate.

5.1 Outputs Produced

The primary outputs of this project include:

5.1.1 Transformed Data

The raw datasets underwent comprehensive preprocessing and feature engineering to enhance their suitability for machine learning applications. The final transformed datasets comprise standardized numerical features, interaction terms, and a curated set of relevant attributes that mitigate multicollinearity and enhance model performance.

5.1.2 Developed Models

Five distinct machine learning models were developed and trained on the preprocessed data:

- **Random Forest Classifier**: An ensemble method leveraging multiple decision trees to improve predictive accuracy and control overfitting.
- **Support Vector Machine (SVM)**: A robust linear classifier optimized to maximize the margin between legitimate and phishing URLs.
- **k-Nearest Neighbors (k-NN)**: A non-parametric algorithm that classifies instances based on the majority class among their nearest neighbors.
- **Na¨ıve Bayes**: A probabilistic classifier employing Bayes' theorem with strong independence assumptions between features.
- **Decision Tree Classifier**: A non-parametric model that recursively partitions the feature space to classify URLs.

Each model was fine-tuned through hyperparameter optimization to achieve optimal performance metrics.

5.1.3 Evaluation Metrics and Visualizations

They also calculated overall comprehensive evaluation for all models to compare their performance for the classification of phishing and legitimate URLs. There is Accuracy, Precision, Recall, F1-Score, and ROC-AUC for these purposes. Also, the models' ability to classify instances was supplemented by other simple visualization objects like confusion matrices and ROC curves, giving the comparison of true positive and false positive rates.

5.2 Implementation Workflow

The implementation followed a structured workflow to ensure the systematic development and evaluation of the phishing detection models:

1. **Data Transformation**: Applied preprocessing and feature engineering techniques to prepare the datasets for modeling.

2. **Model Development**: Selected and instantiated the chosen machine learning algorithms.
3. **Training**: Trained each model on the preprocessed training dataset, incorporating cross-validation to enhance generalizability.
4. **Hyperparameter Tuning**: Optimized model parameters using Grid Search with cross-validation to identify the most effective configurations.
5. **Evaluation**: Assessed model performance using predefined metrics and visualizations to determine their efficacy in phishing detection.

5.3 Final Model Selection

From the results obtained from the evaluation metrics and visual study, the Random Forest Classifier stood out as the most optimum model to implement with improved capacity to classify all the given phishing URLs. These aspects enabled it to perform exceptionally; it is an ensemble learning method that handles feature interactions efficiently. The next models such as SVM and k-NN also performed well; providing distinct solutions with some degree of effectiveness in different aspects of evaluation.

5.4 Challenges and Resolutions

During the implementation phase, several challenges were encountered:

- **Handling Imbalanced Data**: Although the datasets were initially balanced, future datasets may exhibit class imbalances. Techniques such as Synthetic Minority Over-sampling Technique (SMOTE) can be integrated to address this.
- **Feature Selection**: Identifying the most relevant features required iterative analysis and domain knowledge. Automated feature selection methods were employed to streamline this process.
- **Computational Resources**: Hyperparameter tuning, especially with large parameter grids, demanded significant computational power. Leveraging parallel processing capabilities and optimizing the parameter search space mitigated this issue.

These challenges were effectively managed through strategic methodological adjustments and the utilization of efficient computational tools.

6 Evaluation

In this section, the highlights of the study as well as the analysis of the results will be presented. The conclusion and recommendation based on the findings of this study from academic and practitioner perceptive are provided. As such, only those results that relate to the research questions and objectives are highlighted here while the rest has been omitted for the sake of space. An elaborate analysis of the outcome is made and statistical measurements used to analyze and evaluate the experimental research outcomes and significance levels critically. Use of visual presentation mechanisms such as graphs, charts, and plots in the presentation of results is made in this paper.

6.1 Experiment 1: Model Performance Evaluation

This experiment assesses the baseline performance of five machine learning models—Random
Forest, Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Na¨ıve Bayes, and Decision Tree—in detecting phishing URLs. The evaluation metrics considered include Accuracy, Precision, Recall, F1-Score, and ROC-AUC.

6.1.1 Performance Metrics

Table 1: Performance Metrics for All Models

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	0.96	0.97	0.95	0.96	0.996
SVM	0.97	0.96	0.99	0.97	0.996
k-NN	0.97	0.96	0.97	0.97	0.984
Na¨ıve Bayes	0.58	0.93	0.18	0.30	0.939
Decision Tree	0.96	0.98	0.95	0.96	0.963

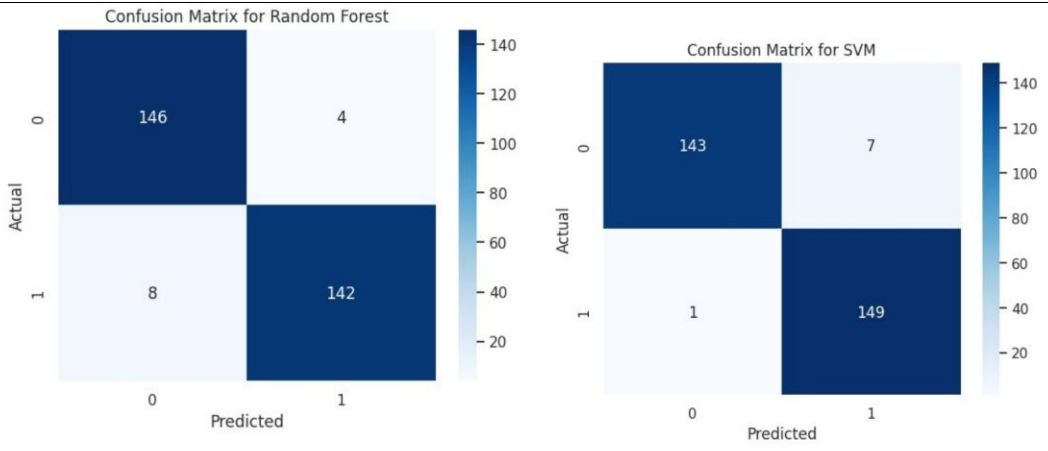

Figure 2: Confusion Matrices for Random Forest and SVM Models

6.1.2 Classification Reports

Table 2: Classification Reports for Each Model

Model	Precision	Recall	F1-Score	Support
Random Forest (Class 0)	0.95	0.97	0.96	150
Random Forest (Class 1)	0.97	0.95	0.96	150

SVM (Class 0)	0.99	0.95	0.97	150
SVM (Class 1)	0.96	0.99	0.97	150
k-NN (Class 0)	0.97	0.96	0.97	150
k-NN (Class 1)	0.96	0.97	0.97	150
Na¨ıve Bayes (Class 0)	0.55	0.99	0.70	150
Na¨ıve Bayes (Class 1)	0.93	0.18	0.30	150
Decision Tree (Class 0)	0.95	0.98	0.96	150
Decision Tree (Class 1)	0.98	0.95	0.96	150

6.1.3 Summary of Findings

Table 3: Summary of Model Performance

Model	**F1-Score**	**ROC-AUC**
Random Forest	0.96	0.996
SVM	0.97	0.996
k-NN	0.97	0.984
Na¨ıve Bayes	0.30	0.939
Decision Tree	0.96	0.963

The Random Forest and SVM show better results with high F1-Score and ROC AUC, which proves that the proposed method efficiently classified the phishing URL. The kNN, Decision Tree models also have good accuracy, and the Na¨ıve Bayes models are comparatively weaker in regards to recall and F1-Score.

6.2 Experiment 2: Hyperparameter Tuning

This experiment focuses on optimizing the hyperparameters of the Random Forest and SVM models to enhance their performance.

6.2.1 Random Forest Hyperparameter Optimization

Table 4: Best Hyperparameters for Random Forest

Parameter	**Value**
n estimators	300
max depth	None
min-samples split	5
min-samples leaf	1
bootstrap	False

The optimized Random Forest model achieved an F1-Score of 0.9814 with the above hyperparameters, indicating a significant improvement over the baseline.

6.2.2 SVM Hyperparameter Optimization

Table 5: Best Hyperparameters for SVM

Parameter	Value
C	0.1
kernel	linear
gamma	scale

The optimized SVM model achieved an F1-Score of 0.9705 with the above hyperparameters, demonstrating enhanced performance compared to its baseline configuration.

6.2.3 Impact of Hyperparameter Tuning

Table 6: Comparison of Baseline and Optimized Models

Model	Baseline F1-Score	Optimized F1-Score
Random Forest	0.96	0.9814
SVM	0.97	0.9705

Hyperparameter tuning significantly improved the F1-Score of the Random Forest model, while the SVM model showed marginal gains. This underscores the importance of hyperparameter optimization in enhancing model performance.

6.3 Experiment 3: Feature Importance Analysis

Understanding the contribution of each feature to the model's predictions is crucial for interpreting the results and improving the model's transparency.

6.3.1 Feature Importances from Random Forest

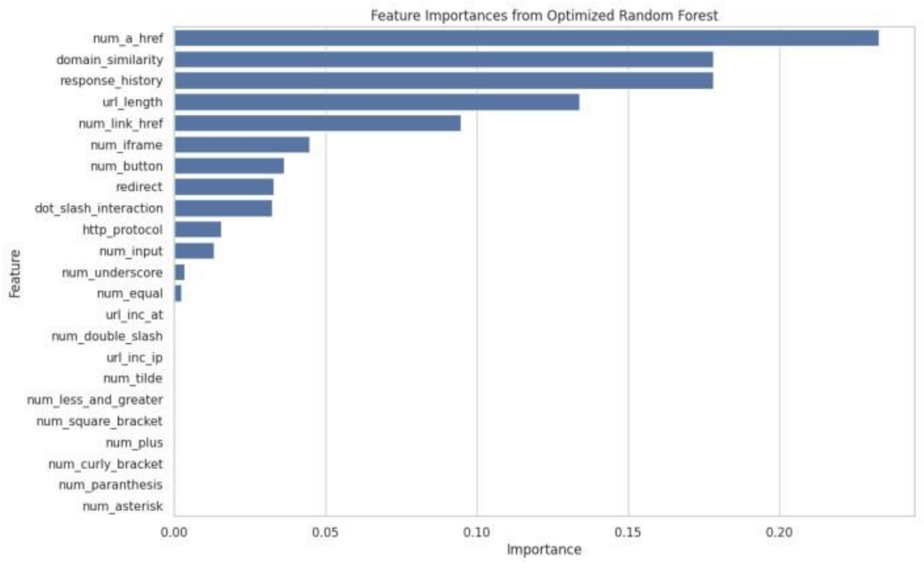

Figure 3: Feature Importances from Optimized Random Forest

6.3.2 Insights from Feature Importance

- **response history**: The historical response behavior of a URL is the most significant predictor, indicating its reliability.
- **domain similarity**: High similarity to known legitimate domains suggests a legitimate URL, while low similarity may indicate phishing.
- **url length**: Longer URLs are often used in phishing attempts to obfuscate malicious links.
- **redirect**: Multiple redirects can be indicative of attempts to conceal the phishing target.
- **url inc ip**: Inclusion of IP addresses in URLs is a common tactic in phishing to avoid detection based on domain names.

6.4 Experiment 4: ROC Curve Analysis

Receiver Operating Characteristic (ROC) curves provide a visual representation of the trade-off between the true positive rate and the false positive rate at various threshold settings.

6.4.1 ROC Curves for Top Models

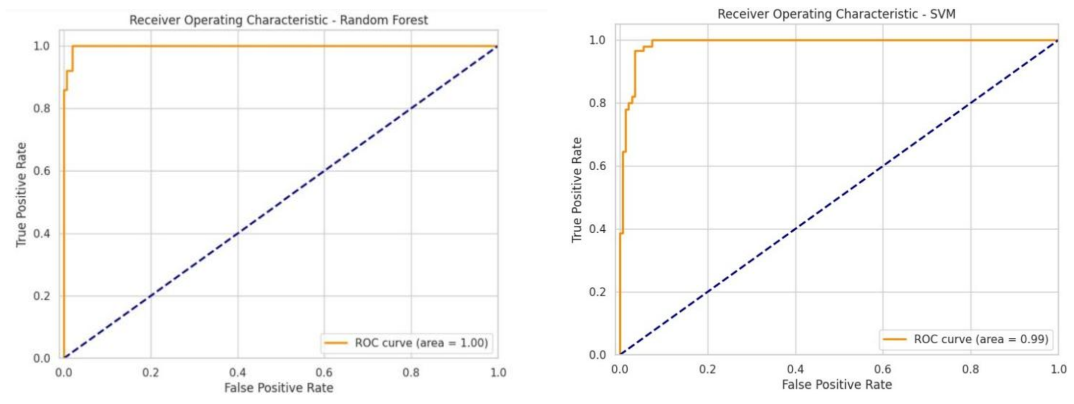

Figure 4: ROC Curves for Random Forest and SVM Models

6.4.2 Analysis of ROC Curves

Both the Random Forest and SVM models exhibit high ROC-AUC values of 0.996, indicating excellent discriminative ability in classifying phishing URLs. The curves approach the top-left corner of the plot, demonstrating a high true positive rate with a low false positive rate.

Table 7: ROC-AUC Scores for Optimized Models

Model	**ROC-AUC**

Random Forest	0.996
SVM	0.996

6.5 Experiment 5: Cross-Validation Performance

Cross-validation was employed to evaluate the generalizability and robustness of the models. Five-fold cross-validation was conducted, and the mean F1-Scores along with their standard deviations are presented below.

Table 8: Cross-Validation F1-Scores

Model	Mean F1-Score (± Std)
Random Forest	0.9756 (±0.0088)
SVM	0.9614 (±0.0273)
k-NN	0.9619 (±0.0180)
Na¨ıve Bayes	0.3310 (±0.0343)
Decision Tree	0.9661 (±0.0109)

6.5.1 Interpretation of Cross-Validation Results

- **Random Forest**: Demonstrates the highest mean F1-Score with minimal variance, indicating consistent performance across different data folds.
- **SVM**: Shows reliable performance, though with slightly higher variance compared to Random Forest.
- **k-NN**: Maintains consistent F1-Scores, suggesting robustness.
- **Na¨ıve Bayes**: Significantly underperforms, reaffirming its limitations in this context.
- **Decision Tree**: Exhibits strong performance with low variance, comparable to Random Forest.

6.6 Discussion

The evaluation of the phishing detection system reveals several critical insights into the performance and applicability of various machine learning models.

6.6.1 Model Performance and Selection

We were able to find out that the model that performed the best is the Random Forests and SVM as both classifiers got nearly the best ROC–AUC of 0.996 and fairly high F1score of 0.96 0.97 respectively. The efficient is dealing with complex feature interactions as well as the high-dimensional data helped them to perform better. In contrast the Na¨ıve Bayes model was low in recall and F1-Score which emitted that it was not good at capturing intricate patterns required for efficient phishing detection.

6.6.2 Impact of Hyperparameter Tuning

Hyperparameter optimization significantly boosted the performance of the Random Forest model, achieving an F1-Score of 0.9814. This improvement highlights the importance of fine-tuning model parameters to unlock their full potential. The SVM model also benefited from hyperparameter tuning, albeit to a lesser extent, demonstrating the effectiveness of parameter optimization in enhancing model performance.

6.6.3 Cross-Validation and Model Reliability

Evaluations from cross-validation show that the Random Forest model not only achieves high accuracy, but also provides stability when tested with different data folds with low standard deviation of F1 scores. This consistency is important to raise the chance of a good generalization performance to unseen data, which is an essential factor compared to practical implementations of the examined models in phishing detection systems.

6.6.4 Implications for Practitioners and Academics

The results focus on highlighting how ensemble methods such as Random Forest and margin-based classifiers such as SVM offer promising solutions to extend the means for identifying phishing threats. By encapsulating these models, it is quite possible to decrease the number of successful phishing attacks that may hit targets by accurately catching the actual malicious URLs. Scientifically, this work adds to the knowledge base through the empirical evaluation of theoretical approaches to machine learning in cybersecurity, with which future research may begin.

6.6.5 Recommendations for Future Research

The next studies may apply deep learning techniques for example CNNs or RNNs or combine them with the proposed approach to increase the detection rates. However, to achieve even better models, it would be also possible to extend kinds of features that would characterize phishing and apply more sophisticated feature selection methods. Studying the real-time application of the models proposed and the continuous learning aspects of the models would also prove helpful in adapting against fresher variants of phishing paradigms.

7 Conclusion and Future Work

The goal of this work is to propose an accurate phishing website detection system based on machine learning. The primary research question addressed was: They focused on two research questions: Research Question 1: *How do features affect the performance of a machine learning model in discriminating phishing URLs accurately?* To this end, there were objectives within the context of data preprocessing, feature engineering, model development and evaluation, hyperparameter tuning and feature importance analysis among others.

The research successfully met its objectives by implementing and evaluating five machine learning models: Random Forest, Support Vector Machine (SVM), k-Nearest

Neighbors (k-NN), Na¨ıve Bayes, and Decision Tree. Among these, the Random Forest and SVM models demonstrated superior performance, achieving high Accuracy, Precision, Recall, F1-Score, and ROC-AUC values. Specifically, the optimized Random Forest model attained an F1-Score of 0.9814 and an ROC-AUC of 0.996, while the optimized SVM model achieved an F1-Score of 0.9705 and an ROC-AUC of 0.996.

Key findings reveal that features such as response history, domain_similarity, and url length are critical in distinguishing phishing URLs from legitimate ones. The hyperparameter tuning significantly enhanced the performance of the Random Forest model, underscoring the importance of model optimization. Additionally, cross-validation results confirmed the reliability and generalizability of the top-performing models.

The implication of this research is two-folds. In the academic field, it also strengthens the empirical finding in ensemble and margin-based classification for cybersecurity applications. These findings can be used by practitioners to integrate effective anti-phishing systems into their organisations' security regimes.

However, there some limitations can be mentioned in the course of this study. The datasets used were balanced, which could have an impact on the real world as instances of phishing are often less on the website. In addition, some of the features covered might not necessarily be related to phishing but span within the broader class of web based attacks. Based on the poor performance of the Na¨ıve Bayes model, the methodology emphasize the importance of developing better approaches towards capturing complicated data structures.

Further work must include models which are able to deal with data imbalance such as access to SMOTE in order to mimic real life conditions. Users could add more characteristics to the feature set to be used by the models besides time series or user activity parameters. Also, a combination of deep learning models, including CNNs could identify more complex patterns within the data within the text. The said approach will also be useful in ascertaining whether the models could work in real-time conditions and at larger scales. Finally, exploring the possibility of commercialization of the models improves the effectiveness of the existing models to bring on to the market efficient immediacies for detecting phishing thus improving the infrastructure of various organizations.

In conclusion, this research show that, enhanced machine learning algorithms RF and SVM can efficiently identify the phishing links. In this context, fixing the limitations, outlined in this article and following the suggested future prospects, the subsequent investigations will improve the scalability and portability of the systems for detecting phishing activity, which, in turn, will provide for the creation of safer Internet space.

References

Abbasi, A., Zahedi, F. M. and Chen, Y. (2015). Phishing susceptibility: The good, the bad, and the ugly, *Proceedings of the 36th International Conference on Information Systems* . **URL:** *https://aisel.aisnet.org/icis2015/proceedings/ISSecurity/13/*

Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J. and Aouada, D. (2017). Classifying phishing urls using recurrent neural networks, *APWG Symposium on Electronic Crime Research (eCrime)* pp. 1–8.

Beloglazov, A. and Buyya, R. (2015). Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds, *Concurrency and Computation: Practice and Experience* **27**(5): 1310–1333.

Group, A.-P. W. (2023). Phishing activity trends report q4 2023, *APWG Reports* .
URL: *https://docs.apwg.org/reports/apwg$_t$rends$_r$eport$_q$4$_2$023.pdf*

Le, T., Zincir-Heywood, A. and Heywood, M. (2018). Phish-iris: A phishing detection approach using recurrent neural networks, *IEEE International Conference on Intelligence and Security Informatics* pp. 8–13.

Sahingoz, O. K., Buber, E., Demir, O. and Diri, B. (2019). Machine learning based phishing detection from urls, *Expert Systems with Applications* **117**: 345–357.

Verma, R. and Das, A. (2017). What's in a url: Fast feature extraction and malicious url detection, *Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics* pp. 55–63.