Phishing Detection Model - Machine Learning: A Comparison among Logistic Regression, Random Forest, Naive Bayes, and SVM Algorithms

Diana Nicole D. Danga
College of Computing and Information Technologies
National University - Manila
Manila, Philippines
dangadd@students.national-u.edu.ph

John Efren V. Gannaban
College of Computing and Information Technologies
National University - Manila
Manila, Philippines
gannabanjv@students.national-u.edu.ph

Jascent Pearl G. Navarro
College of Computing and Information Technologies
National University - Manila
Manila, Philippines
navarrojg@students.national-u.edu.ph

***Abstract*—This paper evaluates machine learning algorithms—Logistic Regression, Random Forest, SVM, and Naive Bayes—for phishing detection using a URL-based dataset. Preprocessing addressed class imbalance and redundant features to enhance model performance. Logistic Regression and SVM emerged as top performers with 99% accuracy, while Random Forest achieved 98%, potentially affected by noise sensitivity in high-dimensional data. Naive Bayes lagged due to feature independence assumptions. Results indicate that Logistic Regression and SVM are optimal for URL-based phishing detection, with Random Forest as a viable alternative.**

***Index Terms*—machine learning, classification, URL, phishing, Logistic Regression, Random Forest, Naive Bayes, SVM.**

I. Introduction

Phishing is a cyber-crime, specifically "a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials" (APWG, 2018, p. 1). It remains a major threat in the digital age, as many internet users fall prey to it. This form of social engineering involves attackers, or phishers, deceiving users into disclosing sensitive information by impersonating reputable or well-known organizations in an automated fashion, fostering trust that prompts victims to reveal private details (Jakobsson and Myers, 2006). In phishing schemes, phishers often use tactics like redirecting users to malicious sites via embedded links in emails (Gupta et al., 2015). Beyond email, attackers may also utilize channels such as Voice over IP (VoIP), SMS, and instant messaging (IM) to carry out their attacks (Gupta et al., 2015). Additionally, phishers have shifted from indiscriminate mass-email campaigns to a more targeted approach, known as "spear-phishing," in which emails are crafted to deceive specific individuals [1].

The most immediate and direct consequence of phishing attacks is financial loss. For individuals, this often involves drained savings, fraudulent transactions, or identity theft. For businesses, it can lead to stolen funds, ransom payments, damaged reputation, or the loss of trade secrets, potentially causing a competitive setback. In 2020, the FBI reported that phishing-related losses surpassed $4.2 billion, highlighting the severe financial toll of these attacks [2].

Building on the information above, the study aims to develop a machine learning model designed to detect and classify phishing websites based on their URLs. The group selected this idea to further showcase the capacity of machine learning in cyberspace, offering valuable insights and practical solutions that contribute meaningfully to the existing body of knowledge. The paper will compare multiple algorithms to identify the best-performing model for classifying phishing sites accurately. This straightforward approach aims to curb phishing attempts that rely on website impersonation, offering added protection for general users and those who are less tech-savvy.

II. Review of Related Literatures

A. *Overview of key concepts and background information.*

Logistic Regression

The logistic regression as a general statistical model was originally developed and popularized primarily by **Joseph Berkson**, beginning in Berkson (1944), where he coined "logit" [3]. Logistic Regression (LR) is one of the most important statistical and data mining techniques employed by statisticians and researchers for the analysis and classification of binary and proportional response data sets. Some of the main advantages of LR are that it can naturally provide probabilities and extend to multi-class classification problems.

A key advantage of logistic regression (LR) is that its analysis methods often mirror those in linear regression. Additionally, various unconstrained optimization techniques can be applied to LR. Komarek and Moore demonstrated that Truncated-Regularized Iteratively Re-Weighted Least Squares (TR-IRLS) can be used in LR to classify large datasets, even outperforming Support Vector Machines (SVM) in certain cases [4].

Random Forest

Leo Breiman introduced Random Forests, building on earlier work by Amit and Geman. Although initially subtle, Random Forests build on Breiman's concept of bagging and were developed as a competitor to boosting. They support both categorical and continuous response variables, known as "classification" and "regression," respectively, and can handle both categorical and continuous predictor variables. From a computational perspective, Random Forests are efficient because they support both regression and multiclass classification, are relatively fast to train and predict, require minimal tuning, offer a built-in generalization error estimate, work well with high-dimensional data, and can be parallelized. Statistically, they are valuable due to features like variable importance measures, differential class weighting, missing value imputation, and visualization capabilities [5].

SVM

Support Vector Machines (SVM), introduced by Vapnik, are kernel-based machine learning models designed for classification and regression tasks. SVM's strong generalization ability, optimal solutions, and discriminative power have made it popular across data mining, pattern recognition, and machine learning communities in recent years. Known for its effectiveness in binary classification, SVM has proven superior to many other supervised learning approaches. Due to its solid theoretical foundation and robust generalization, SVM has become one of the most widely used classification methods. SVM determines decision functions by maximizing the separation (margin) between classes within a high-dimensional feature space, reducing training classification errors and enhancing generalization. SVM's unique advantage lies in its ability to identify a subset of support vectors from the training data, often only a small fraction of the dataset, which effectively represents the classification task at hand [6].

Naïve Bayes

Bayesian classification, a supervised learning method and statistical approach for classification, is based on an underlying probabilistic model that captures uncertainty by determining the probabilities of outcomes. Named after Thomas Bayes (1702-1761), who proposed Bayes' Theorem, Bayesian classification facilitates the combination of prior knowledge with observed data, offering valuable insights for understanding and evaluating various learning algorithms. It calculates explicit probabilities for hypotheses and demonstrates robustness against noise in input data. Naïve Bayes is a straightforward learning algorithm that applies Bayes' rule alongside a strong assumption of conditional independence among attributes given the class. Although this independence assumption is frequently violated in real-world applications, Naïve Bayes often achieves competitive classification accuracy. Its computational efficiency and other desirable features contribute to its widespread use in practice [7].

Machine learning, one of the fastest-growing fields in computer science, has far-reaching applications and focuses on the automated detection of meaningful data patterns. These tools enable programs to learn and adapt independently. Now a cornerstone of Information Technology, machine learning plays an increasingly central—if often unseen—role in daily life. Machine learning algorithms are categorized into a taxonomy based on the intended outcomes. In supervised learning, the goal is to create a function that maps inputs to specified outputs. Supervised learning is fairly common in classification problems because the goal is often to get the computer to learn a classification system that we have created. According to [8], the supervised machine learning algorithms which deals more with classification includes the following: Linear Classifiers, Logistic Regression, Naïve Bayes Classifier, Perceptron, Support Vector Machine; Quadratic Classifiers, K-Means Clustering, Boosting, Decision Tree, Random Forest (RF); Neural networks, Bayesian Networks and so on [9].

Machine learning (ML) is poised to play a pivotal role in the evolution of phishing detection and prevention. As phishing attacks continue to grow in sophistication and volume, traditional methods like blacklisting and whitelisting are proving inadequate. ML algorithms offer the potential for advanced, automated monitoring systems that can identify and respond to phishing threats in real time. Researchers have identified distinctive features that differentiate phishing from legitimate websites, indicating that effective feature engineering can enhance ML detection capabilities. However, there remains a gap in studies focused on applying these methods to real-world production datasets [10].

B. Review of other relevant research papers

Purbay et al. utilized a UCI Machine Learning Repository dataset with various training and testing splits alongside several supervised ML algorithms, achieving a peak accuracy of 96.92% with Random Forest Classification by incorporating lexical features, Whois properties, PageRank, Traffic Rank details, and page importance at a 0.2 train/test split ratio [11].

Wang et al. introduced a phishing website detection method named PDRCNN, focusing solely on URL features without relying on third-party services or website content. Their approach combined Long-Short Term Memory (LSTM) and Convolutional Neural Network (CNN) models on a dataset of

500,000 URLs, yielding a classification accuracy of 97% [12].

Shirazi et al. achieved 97% accuracy using just seven features on a dataset comprising 1,000 legitimate URLs from Alexa, 1,000 phishing URLs from PhishTank, and 2,013 phishing URLs from OpenPhish, utilizing Gradient Boosting while noting model biases related to feature selection, such as URL length [13].

Guo et al. explored Heterogeneous Information Networks (HIN) derived from link relationships among websites, finding that phishing sites typically contain foreign links compared to legitimate ones. Their model evaluated approximately 20,556 phishing URLs and 20,949 legitimate URLs from Phishpedia and OpenPhish [14].

Das et al. constructed a hybrid feature set using 15 URL-based and 10 hyperlink-based features from their custom dataset, which included URL length, domain, and presence of an IP address, alongside hyperlink ratios. They reported classification accuracies of 84.67% for hyperlink-based features, 98.42% for URL-based features, and 99.17% for hybrid features using XGBoost [15].

Aljofey et al. implemented a feature set limited to URLs with a maximum length of 200 characters, 13 hyperlinks, and textual content, excluding third-party features. By applying Term Frequency-Inverse Document Frequency (TF-IDF) in their preprocessing, they generated 20,332 features and achieved an accuracy of 96.76% using XGBoost on a dataset of 32,972 legitimate and 27,280 phishing websites [16].

C. *Current State of the Art*

Traditional methods like blacklist-based filtering and heuristic analysis are simple but limited by their reliance on outdated information and difficulty in adapting to new phishing [17]. Machine learning offers more robust solutions, with traditional algorithms like Support Vector Machines (SVMs) and Random Forests proving effective in smaller datasets [18]. Specifically, simple algorithms like Naive Bayes, Logistic Regression, SVM, and Random Forest have been widely used for phishing detection due to their interpretability and efficiency. Naive Bayes, while making a strong independence assumption, can be effective in text-based features like email content [19]. Logistic Regression provides probabilistic predictions and is suitable for understanding the impact of different features (18). SVM excels in high-dimensional spaces and can handle complex decision boundaries [20]. Random Forest, an ensemble method, can improve accuracy and reduce overfitting by combining multiple decision trees [18].

However, these algorithms may struggle with complex patterns and require careful feature engineering. While they offer a solid foundation for phishing detection, their performance can be surpassed by more advanced deep learning techniques, especially when dealing with large and diverse datasets. Deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), remain and have shown superior performance in detecting complex phishing patterns [20]. However, while these methods offer high accuracy, they require substantial computational resources and large datasets.

D. *Prior Attempts to Solve the Same Problem*

Purbay et al. (2022) - This study utilized a dataset from the UCI Machine Learning Repository and various supervised machine learning algorithms, achieving a peak classification accuracy of 96.92% using a combination of lexical features and Random Forest Classification (RFC) with a train/test split ratio of 0.2. However, while the accuracy was commendable, the study did not explore the generalizability of the model across different datasets or real-world scenarios (2021)** - The researchers introduced PDRCNN, a phishing detection method based solely on URL features, leveraging Long-Short Term Memory (LSTM) and Convolutional Neural Networks (CNN). They reported a classification accuracy of 97% on a dataset of 500,000 URLs. Despite this high accuracy, the reliance on URL features alone may limit the model's effectiveness in detecting phishing attempts that do not exhibit obvious URL manipulations [11].

Shirazi seven features, **Shirazi et al (2018)**. achieved a classification accuracy of 97% on a dataset comprised of both legitimate and phishing URLs. Their findings highlighted potential model bias related to URL length, but they did not thoroughly investigate the impact of feature selection on overall model performance, which could leave room for further improvement [13].

Guo et al. (2021) - This research f (HIN) derived from website link relationships, identifying that phishing sites often feature foreign links. They utilized around 14 features for classification but did not fully address how their model adapts to the evolving tactics of phishing schemes [14].

Das et al. (2021) - By constructing a hybrid feature set with both URL-baseds et al. achieved a classification accuracy of up to 99.17% using XGBoost. While their method was robust, it raised concerns about scalability and adaptability to various phishing tactics over time [15].

Aljofey et al. (2021) - Using a dataset with 32,972 legitimate and 27,280 phishing websites, Aljofey et al. of 96.76% through a combination of URL features and TF-IDF processing. However, their approach may be limited by the fixed URL length used, potentially missing out on varied phishing tactics [16].

Many researchers have reported high classification accuracies in their phishing detection models, highlighting notable

successes in the field. However, these advancements come with several shortcomings. One major challenge is the limited scope of many studies, which often focus primarily on specific features, such as URLs. While the current study also concentrates on dissecting URL-based characteristics, it aims to deepen the analysis by examining important components that contribute to a URL's effectiveness in phishing detection. This approach addresses the complexities of modern phishing tactics that can manipulate these components. Furthermore, few existing models have been tested across diverse datasets, raising concerns about their generalizability in real-world scenarios. The dynamic nature of phishing tactics also presents a challenge, as many existing models may struggle to adapt quickly enough to identify new threats effectively. This research seeks to address these unresolved issues by enhancing URL analysis and incorporating broader methodologies to improve detection capabilities.

III. Methodology

The development is divided into four stages: data collection, where the dataset containing URL attributes is gathered; data pre-processing, where the data is cleaned and transformed for analysis; modeling, where Logistic Regression, Random Forest, Naive Bayes, and Support Vector Machine (SVM) algorithms are trained to learn patterns in the data; and evaluation, where the performance of the trained models is assessed using accuracy, precision, recall, and F1 score on the test data.

A. *Data Collection*

The PhiUSIIL Phishing URL Dataset is an open-source collection on Kaggle, containing 235,795 entries used for detecting phishing attempts, with 134,850 legitimate URLs ("1") and 100,945 phishing URLs ("0"). This dataset, prepared by Kaggle Pro LLC, is structured for machine learning classification tasks and includes both numerical and categorical features extracted from webpage source code and URLs, such as URL length, domain attributes, HTTPS presence, and various URL-based similarity scores. It is a valuable resource for building phishing detection models, offering pre-labeled data that requires no additional analysis.

B. *Data Pre-Processing*

Data Loading & Cleaning:

The preprocessing phase began with loading the PhiUSIIL Phishing URL Dataset and inspecting it for any inconsistencies. Duplicate entries were reviewed and null values were checked. Irrelevant columns, which contained string data, were eliminated to streamline the dataset and focus on the relevant features.

Feature Engineering:

In this step, the TLD (Top-Level Domain) column was transformed by consolidating it to include only the 20 most frequent TLDs, categorizing all others as 'others.' These categories were then mapped to specific integers through the use of ordinal encoding. This simplification aimed to enhance the model's ability to identify patterns within the data.

Feature Selection:

A correlation heatmap was generated to visualize the relationships among the remaining features. This visualization assisted in identifying highly correlated variables to the label. Mean Absolute Correlation was also computed. Additionally, the Variance Inflation Factor (VIF) was calculated for the features to detect multicollinearity. Features with high VIF values were removed to prevent redundancy and ensure model stability.

Class Balancing:

To tackle the issue of class imbalance between phishing and legitimate URLs, the Synthetic Minority Oversampling Technique (SMOTE) was applied. This technique helped create a balanced distribution of classes.

Scaling:

Features are scaled using StandardScaler to normalize the data, ensuring all features contributed equally.

C. *Experimental Setup*

Tools and Frameworks Used:

The experimentation was conducted on Google Colab that runs on Python programming language, along with several key libraries and frameworks for data manipulation and machine learning. The following tools were utilized:

- Pandas (version 2.2.2)
- NumPy (version 1.26.4)
- Seaborn (version 0.13.2)
- Matplotlib (version 3.7.1)
- scikit-learn (version 1.5.2)
- imbalanced-learn (version 0.12.4)

The data was organized into training and test sets, in which 80% of the data (188,636 entries) was used for training the models while 20% of the data (47,159 entries) was used for evaluating the model performance.

Hyperparameters Used:

The hyperparameters utilized in this study varied across the models. The Logistic Regression model had its regularization strength C set to the default value of 1.0. For the Random Forest Classifier, the hyperparameters included n_estimators=100 (indicating the number of trees), max_depth=4 (to mitigate overfitting), and random_state=42 (ensuring reproducibility). Both the Gaussian Naive Bayes model and the Support Vector Classifier (SVC) were implemented with their default configurations, without any specific hyperparameter adjustments. All models employed cv=10 for cross-validation to ensure robust evaluation.

D. *Algorithm*

Logistic Regression

Logistic Regression is a statistical model used for binary classification tasks. It estimates the probability of a binary outcome based on one or more predictor variables.

This algorithm was chosen for its simplicity and interpretability, which is crucial when working with a dataset containing various numerical features derived from URL characteristics. Logistic Regression provides insights into how each feature contributes to the likelihood of an outcome, which helps in understanding the relationship between the features. Its efficiency in training and low computational requirements make it suitable for large datasets. Equation 1 shows the formula for Logistic Regression.

$$p = \frac{e^{m(x)+b}}{1 + e^{m(x)+b}}$$

Equation 1. *Logistic Regression*

Where,

- x is the value of the independent variable.
- m is the slope/ coefficient of the line.
- b is the y-intercept.

Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes for classification tasks. This algorithm was selected for its robustness and ability to handle high-dimensional data without overfitting. Given the diverse set of features, Random Forest can effectively capture complex interactions among features. Furthermore, it performs well with imbalanced datasets, ensuring reliable performance across different classes. Equation 2 and 3 shows the formula for Random Forest.

Classification (majority voting):

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), \ldots, T_n(x)\}$$

Equation 2. *Classification*

Where,

Ti(x)T_i(x)Ti(x) is the prediction of the iii-th tree for an input x.

Regression (average):

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} T_i(x)$$

Equation 3. *Regression*

Where,

Ti(x)T_i(x)Ti(x) is the prediction of the iii-th tree, and nnn is the number of trees in the forest.

Naive Bayes (Gaussian)

Gaussian Naive Bayes is a probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features, which are assumed to follow a Gaussian distribution. Despite its simplistic assumptions, Gaussian Naive Bayes is highly efficient and performs well with large datasets, making it a good choice for rapid prototyping. Its speed in both training and prediction phases is advantageous, particularly when iterating through model development. Equation 4 shows the formula for Naive Bayes [22].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Equation 4. *Naive Bayes*

Where,

- P(A—B) is Posterior probability: Probability of hypothesis A on the observed event B.
- P(B—A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.
- P(A) is Prior Probability: Probability of hypothesis before observing the evidence.
- P(B) is Marginal Probability: Probability of Evidence.

Support Vector Machine (SVM)

The Support Vector Classifier is a supervised learning algorithm that finds the hyperplane that best separates different classes in the feature space, allowing for both linear and non-linear classification through the use of kernel functions.

SVC was selected for its capability to handle high-dimensional data effectively, making it well-suited for datasets with numerous features. Its flexibility in using different kernel functions allows for modeling complex relationships in the data, which is valuable when dealing with the various attributes of URLs. SVC is particularly effective in cases where there is a clear margin of separation between classes, which could be applicable given the diverse features of the chosen dataset. Additionally, SVC is robust against overfitting, especially in high-dimensional spaces, making it a suitable choice for this research. Equation 5 and 6 shows the formula for SVM.

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \cdots + \beta_n * x_n = 0$$

Equation 5. *Hyperplane Representation*

Where,

- is each beta is a parameter for one of the many dimensions we have in our space.
- 0 is the intercept
- 1 is the first axis, and so on.
- X is a point X that satisfies the above equation then it means the point is on the hyperplane

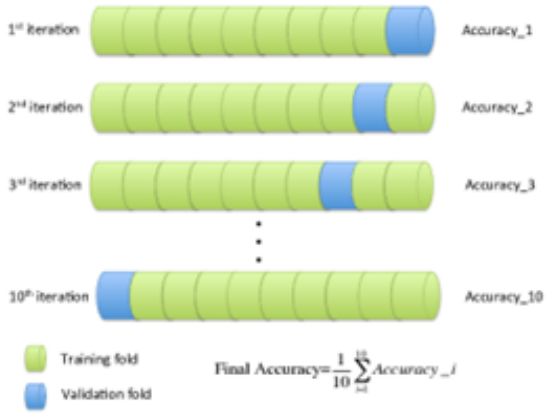
The equation on Equation 6 is the equation of a hyperplane in a two-dimensional space, which is a line.

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 = 0$$

Equation 6. *Hyperplane in a two-dimensional space*

E. Training Procedure

10-fold cross-validation was used as the main strategy to validate model performance on different subsets of the training data. This method divides the data into 10 equal parts, with each fold serving as a validation set once, while the remaining nine folds are used for training in each iteration. This rotation ensures that every observation is used for validation exactly once, minimizing performance estimation bias and reducing the risk of overfitting by allowing the model to learn from varied data configurations.



For each model, mean accuracy and standard deviation were calculated across the 10 folds to provide an averaged, stable performance metric, helping ensure that each model performs consistently across diverse samples rather than relying on a single train-test split.

F. Evaluation Metrics

The following metrics were used to evaluate the performance of the models:

1. Accuracy: The ratio of correctly predicted instances to the total instances, indicating overall model performance.

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

Where,

- TN = True Negative
- TP = True Positive
- FP = False Positive
- FN = False Negative

2. Precision: The ratio of true positive predictions to the total positive predictions, reflecting the model's ability to avoid false positives.

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

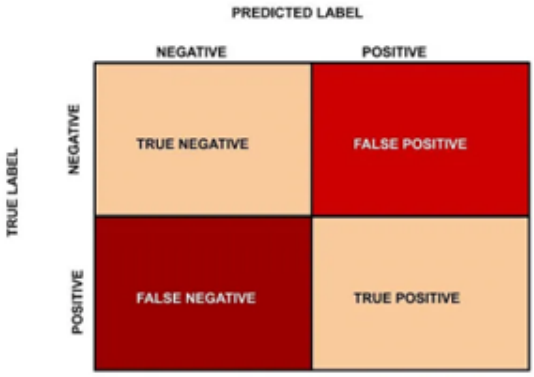
3. Recall: (Sensitivity): The ratio of true positive predictions to the total actual positives, assessing the model's ability to capture all relevant instances.

$$Recall = \frac{TP}{TP + FN}$$

4. F1 Score: The harmonic mean of precision and recall, providing a balance between the two and useful in scenarios where class distribution is imbalanced [23].

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

5. Confusion Matrix: A table used to visualize the performance of the model, showing the counts of true positives, true negatives, false positives, and false negatives.



True Positives (TP): The number of legitimate websites (1) correctly classified as legitimate (1). This indicates successful identification of safe websites.
True Negatives (TN): The number of phishing websites (0) correctly classified as phishing (0). This shows that the model effectively identified malicious URLs.
False Positives (FP): The number of phishing websites (0) incorrectly classified as legitimate (1). This means the model failed to identify these URLs as phishing threats.
False Negatives (FN): The number of legitimate websites (1)

incorrectly classified as phishing (0). This indicates a failure to correctly identify safe URLs.

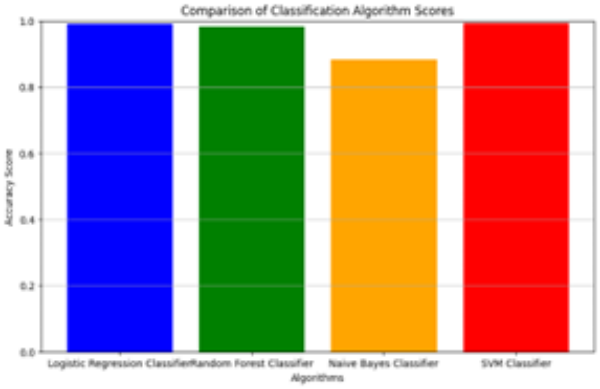
These metrics were used because they provide a nuanced understanding of the model's performance, particularly in the presence of class imbalance. They are standard in the machine learning field and effectively measure specific aspects of model performance. For instance, precision and recall help address the challenges posed by imbalanced classes, ensuring that the model not only predicts accurately but also captures as many relevant instances as possible [21].

G. Baselines and Comparative Models

Four algorithms, specifically Logistic Regression, Random Forest, Naive Bayes, and Support Vector Machine, were trained and tested to identify the most effective model for classifying legitimate and phishing websites based on URL attributes. Each algorithm was evaluated using performance metrics to determine the best fit for the data, rather than relying on established baseline models. This approach allows for a comprehensive assessment of the algorithms' capabilities in addressing the classification task.

IV. Result and Discussion

Bar Chart I. *Comparison of Classification Algorithms Scores*



The chart comparing classification algorithms for distinguishing legitimate (labeled as 1) and phishing websites (labeled as 0) based on URL attributes reveals that Logistic Regression, Random Forest, and Support Vector Machine (SVM) achieved high accuracy scores, all above 90%, as illustrated in the bar chart above. Among these models, Logistic Regression and SVM performed the best, both achieving a remarkable accuracy score of 99%, slightly outperforming Random Forest, which scored 98%. In contrast, Naive Bayes had a lower accuracy score of 88%. These results suggest that Logistic Regression and SVM are the most effective algorithms for this classification task.

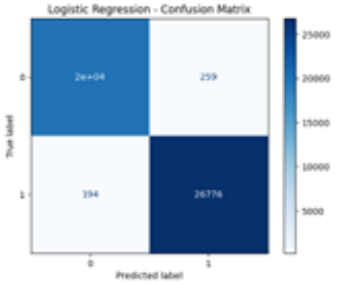
Table I. *Evaluation Metrics Scores*

	Accuracy	Precision	Recall	F1 Score
Logistic Regression				
Training	0.99	0.99	0.99	0.99
Test	0.99	0.99	0.99	0.99
Random Forest				
Training	0.98	0.99	0.98	0.98
Test	0.98	0.99	0.98	0.98
Naïve Bayes				
Training	0.88	0.81	1	0.89
Test	0.89	0.84	1	0.92
SVM				
Training	0.99	0.99	0.99	0.99
Test	0.99	0.99	0.99	0.99

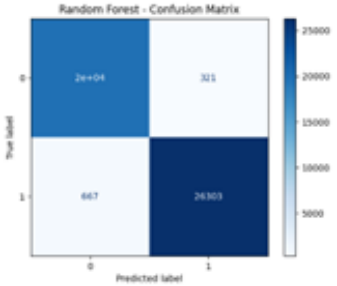
Table I shows the evaluation of all algorithms based on accuracy, precision, recall, and F1 score to comprehensively gauge their performance. Logistic Regression and SVM achieved the highest accuracy across both training and test data, with scores of 0.99 across all metrics, indicating these models excelled in distinguishing between legitimate and phishing URLs. Random Forest also demonstrated strong performance, albeit with slightly lower accuracy and recall compared to Logistic Regression and SVM. Although Naive Bayes had the lowest accuracy, it demonstrated a perfect recall score, indicating it identified almost all legitimate websites correctly but with lower precision due to a higher number of false positives.

Matrix 1-4. *Confusion Matrices for all Algorithms*

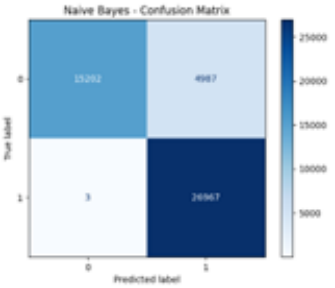
The confusion matrices (Figures 1-4) provide further insights into the classification performance:



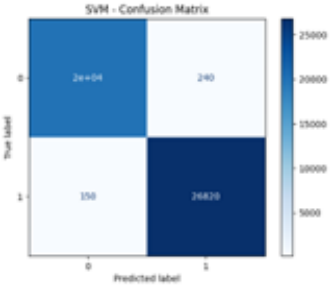
Logistic Regression exhibited minimal false positives (259) and false negatives (194), as shown in its confusion matrix. This aligns with its high precision, recall, and F1 scores (all 0.99). The low number of misclassifications underscores its effectiveness in distinguishing between legitimate and phishing URLs, indicating it is well-suited for this task.



Random Forest experienced a slightly higher number of false positives (321) and false negatives (667) compared to Logistic Regression and SVM, leading to slightly lower accuracy and recall. Nevertheless, Random Forest still performed admirably, with strong precision and recall reflected in its overall high scores.



Naive Bayes had a larger number of false positives (4,987) compared to other models, indicating that many phishing websites were incorrectly classified as legitimate, which impacted its precision. Despite this, it achieved a perfect recall score, meaning it successfully identified almost all actual legitimate websites. This trade-off between precision and recall suggests that Naive Bayes tended to classify more websites as legitimate, leading to more false positives while maintaining a perfect identification rate for actual phishing threats.



Similar to Logistic Regression, SVM also performed exceptionally well, with only 240 false positives and 150 false negatives. These minimal misclassifications reinforce its high accuracy, precision, recall, and F1 scores (all 0.99). SVM's performance is comparable to Logistic Regression, suggesting that both models handle URL-based features effectively and can reliably differentiate between phishing and legitimate websites.

These results indicate that Logistic Regression and SVM were the most effective algorithms for distinguishing between legitimate and phishing websites, achieving the highest scores in accuracy, precision, recall, and F1 metrics.

It is important to note that feature correlation was assessed to ensure independence, particularly for Naive Bayes, which relies on this assumption. By calculating the Mean Absolute Correlation, it was confirmed that interdependencies were minimized after removing irrelevant or redundant features. This preprocessing step supports the assumption of feature independence for Naive Bayes, although it still showed limitations in precision.

V. Conclusion

This paper seeks to investigate algorithms for building a machine learning model to detect phishing. It evaluates and compares Logistic Regression, Random Forest, SVM, and Naive Bayes algorithms, specifically applied to a URL-based dataset of phishing websites.

With the applied preprocessing steps, including addressing class imbalance and eliminating redundant features that could lead to overfitting, we can conclude that for this classification task, Logistic Regression and SVM are the best-performing algorithms with scores of 0.99% across all metrics. Random Forest, despite its strong performance achieved a slightly lower score of 98%, falling behind Logistic Regression and SVM, indicating that its ensemble nature, while generally robust, could be sensitive to noise in a high-dimensional data. This sensitivity might be the cause of slightly more misclassifications compared to the other top-performing algorithms. Conversely, Naive Bayes' lower performance suggests that, despite satisfying the conditional independence between features, it may not be a reliable choice for this problem, potentially due to limitations in feature selection or its inherent assumptions about feature distributions.

References

[1] Alkhalil, Z., Hewage, C., Nawaf, L., & Khan, I. (2021). Phishing Attacks: a recent comprehensive study and a new anatomy. *Frontiers in Computer Science*, *3*. https://doi.org/10.3389/fcomp.2021.563060

[2] Zahra, Syeda & Abbasi, Muhammad Noman & Arshad, Ali & Riaz, Saman & Ahmed, Waqas. (2023). Phishing Attack, Its Detections and Prevention Techniques. International Journal of Wireless Information Networks. 12. 13-25. 10.37591/IJWSN.

[3] Wikipedia contributors. (2024, October 15). *Logistic regression*. Wikipedia. https://en.wikipedia.org/wiki/Logistic_regression

[4] Maalouf, Maher. (2011). Logistic regression in data analysis: An overview. International Journal of Data Analysis Techniques and Strategies. 3. 281-299. 10.1504/IJDATS.2011.041335.

[5] Cutler, Adele & Cutler, David & Stevens, John. (2011). Random Forests. 10.1007/978-1-4419-9326-7_5.

[6] Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., & Lopez, A. (2020). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, *408*, 189–215. https://doi.org/10.1016/j.neucom.2019.10.118

[7] Webb, Geoffrey. (2016). Naïve Bayes. 10.1007/978-1-4899-7502-7_581-1.

[8] Taiwo, O. A. (2010). Types of Machine Learning Algorithms, New Advances in Machine Learning, Yagang Zhang (Ed.), ISBN: 978-953-307-034-6, InTech, University of Portsmouth United Kingdom. Pp 3 – 31.

[9] Akinsola, J E T. (2017). Supervised Machine Learning Algorithms: Classification and Comparison. International Journal of Computer Trends and Technology (IJCTT). 48. 128 - 138. 10.14445/22312803/IJCTT-V48P126.

[10] Ghalechyan, H., Israyelyan, E., Arakelyan, A. *et al.* Phishing URL detection with neural networks: an empirical study. *Sci Rep* 14, 25134 (2024). https://doi.org/10.1038/s41598-024-74725-6

[11] Purbay, M. & Kumar, D. Split behavior of supervised machine learning algorithms for phishing url detection. In *Advances in VLSI, Communication, and Signal Processing* (eds Harvey, D. et al.) 497–505 (Springer Singapore, 2021).

[12] Wang, W., Zhang, F., Luo, X. & Zhang, S. PDRCNN: precise phishing detection with recurrent convolutional neural networks. *Security and Communication Networks* **2019** (2019).
[13] Shirazi, H., Bezawada, B. & Ray, I. "Kn0w Thy Doma1n Name" unbiased phishing detection using domain name based features. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies* 69–75 (2018).
[14] Guo, B. et al. HinPhish: An effective phishing detection approach based on heterogeneous information networks. *Appl. Sci.* **11**, 9733 (2021).
[15] Das Guptta, S., Shahriar, K. T., Alqahtani, H., Alsalman, D. & Sarker, I. H. Modeling hybrid feature-based phishing websites detection using machine learning techniques. *Ann. Data Sci.* 1–26 (2022).
[16] Aljofey, A. et al. An effective detection approach for phishing websites using URL and HTML features. *Sci. Rep.* **12**, 1–19 (2022).
[17] Yang, J., Wang, X., & Song, M. (2018). A Survey of Phishing Detection Techniques. IEEE Access, 6, 32143-32156.
[18] Phua, C.-K., Lee, V. W.-S., & Smith, K. A. (2010). A Comprehensive Survey of Data Mining-Based Techniques for Intrusion Detection Systems. ACM Computing Surveys, 43(1), 1-41.
[19] Al Fayoumi, Q. A., Alqasass, A. E., Aljundi, I., & Abu Al-Haija, Q. (2019). Email phishing detection based on naïve Bayes, Random Forests, and SVM classifications: A comparative study. ResearchGate.
[20] Zhang, Y., Li, Z., & Li, H. (2021). A Survey on Deep Learning for Phishing Website Detection. IEEE Access, 9, 155116-155132.
[21] Analytics Vidhya. (n.d.). *Confusion matrix, accuracy, precision, recall & F1 score*. Medium. https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd
[22] Javatpoint. (n.d.). *Machine learning - Naive Bayes classifier*. https://www.javatpoint.com/machine-learning-naive-bayes-classifier
[23] K-Fold Cross Validation: https://www.researchgate.net/figure/fold-cross-validation_fig4_321814315