Comparative Study of Machine Learning Algorithms for Phishing Website Detection

Kamal Omari
Department of Mathematics & Computer Science-Faculty of Sciences Ben M'sik,
University Hassan II. Casablanca, Morocco

***Abstract*—Phishing, a prevalent online threat where attackers impersonate legitimate organizations to obtain sensitive information from victims, poses a significant cybersecurity challenge. Recent advancements in phishing detection, particularly machine learning-based methods, have shown promising results in countering these malicious attacks. In this study, we developed and compared seven machine learning models, namely Logistic Regression (LR), k-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), and Gradient Boosting, to assess their efficiency in detecting phishing domains. Employing the UCI phishing domains dataset as a benchmark, we rigorously evaluated the performance of these models. Our findings indicate that the Gradient Boosting-based model, in conjunction with the Random Forest, exhibits superior performance compared to the other techniques and aligns with existing solutions in the literature. Consequently, it emerges as the most accurate and effective approach for detecting phishing domains.**

Keywords—Phishing detection; cybersecurity; machine learning; Gradient Boosting; Random Forest

I. INTRODUCTION

Phishing, a widespread and dangerous cyber-attack method, continues to pose significant threats in today's digital world. With the increasing reliance on online platforms, for various activities such as business, transactions and healthcare services, the risk of falling victim to phishing attacks has escalated [1]. Phishing attacks involve the deceptive acquisition of personal and sensitive information through a combination of technical deception and social engineering tactics [2, 3]. These attacks often utilize fraudulent emails or messages that appear to originate from reputable entities, tricking unsuspecting users into sharing their confidential data [2].

Despite advancements in cybersecurity that have greatly improved malware detection and reduced the presence of malware-hosting websites, combating phishing attacks remains challenging due to their social engineering nature [1, 4]. Phishing domains, in particular, exploit users' trust by directing them to counterfeit websites that closely resemble legitimate ones, leading to the compromise of sensitive information [5]. Falling victim to phishing attacks can have severe consequences, including identity theft, financial fraud, and reputational damage [1].

To address the persistent threat of phishing attacks, robust cybersecurity measures are required, and artificial intelligence (AI) has emerged as a promising approach [6]. Machine learning (ML) algorithms, a subset of AI, offer the potential to detect and classify phishing attacks by analyzing patterns and indicators of fraudulent activity based on historical data [6]. By leveraging ML models, it becomes possible to enhance detection capabilities and accurately predict whether a webpage is a phishing site or legitimate [6].

The objective of this research paper is to compare the effectiveness of ML classification models in detecting phishing domains. By identifying the most accurate ML model among the considered algorithms, the aim is to enhance detection capabilities and mitigate the risks associated with visiting phishing websites, ultimately restoring consumer trust.

The rest of this paper is organized as follows: Section II offers insights into the algorithms used. In Section III, a review of the latest research on phishing attacks is presented. Section IV outlines the methodology employed in this study. The experimental results of our comparative study are presented and discussed in Section V. Finally, Section VI concludes the paper by summarizing the key findings and proposing avenues for future research.

II. BACKGROUND

Various machine-learning classification methods have demonstrated their effectiveness in detecting phishing domains. Some of the prominent techniques encompass:

A. *Logistic Regression*

Logistic regression is a prevalent statistical model utilized for binary classification tasks [7]. As a supervised learning algorithm, it predicts the probability of an instance belonging to a particular class. In logistic regression, the dependent variable is binary or categorical, while the independent variables can be continuous or categorical.

The primary objective of logistic regression is to determine the best-fitting logistic function that establishes the relationship between the independent variables and the probability of the binary outcome. This logistic function, also known as the sigmoid function, maps any real-valued number to a value between 0 and 1 (see Fig. 1) [8]. The resulting probability estimate is then utilized to classify the instances into their respective classes.

The logistic regression model estimates its parameters through maximum likelihood estimation, involving the optimization of the log-likelihood function [7]. Various optimization algorithms, such as gradient descent, are commonly employed to minimize the cost function and obtain the optimal parameter values.

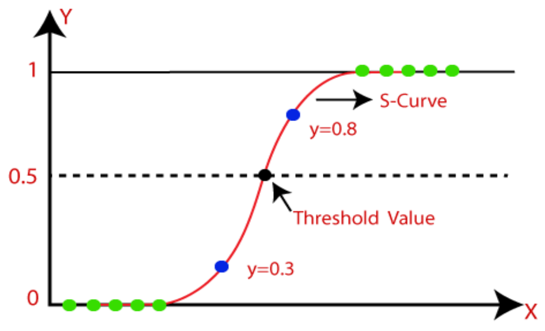

Fig. 1. Logistic regression algorithm [9].

One of the key advantages of logistic regression is its interpretability. The coefficients of the independent variables offer valuable insights into the influence and direction of each variable on the probability of the outcome [10]. Furthermore, logistic regression can effectively handle both linear and nonlinear relationships between the independent variables and the log-odds of the outcome.

B. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) stands as a widely employed non-parametric supervised learning algorithm for classification tasks [11]. Its simplicity, combined with its effectiveness, allows it to predict the class of an instance based on the classes of its nearest neighbors in the feature space.

In the KNN algorithm, the parameter K represents the number of nearest neighbors considered when making a prediction. To classify a new instance, KNN calculates the distances between the instance and all the training instances in the feature space (see Fig. 2). Subsequently, it identifies the K nearest neighbors based on a distance metric, such as Euclidean distance or Manhattan distance [12].

Upon identifying the K nearest neighbors, the majority class among them is assigned to the new instance. In cases of ties, a voting mechanism can resolve the class assignment. One remarkable feature of KNN is that it is a lazy learner, as it performs classification at runtime without requiring an explicit training phase [13].

KNN exhibits versatility, as it adeptly handles both binary and multi-class classification problems. Additionally, it proves to be robust in capturing complex decision boundaries and coping with noisy data [14]. Nonetheless, it is essential to carefully consider the selection of K and the distance metric, as these choices significantly impact the algorithm's performance.

C. Support Vector Machine

Support Vector Machine (SVM) stands as a potent supervised learning algorithm extensively employed for classification and regression tasks [16]. The fundamental objective of SVM is to identify an optimal hyperplane that effectively segregates data points into distinct classes within the feature space (see Fig. 3).

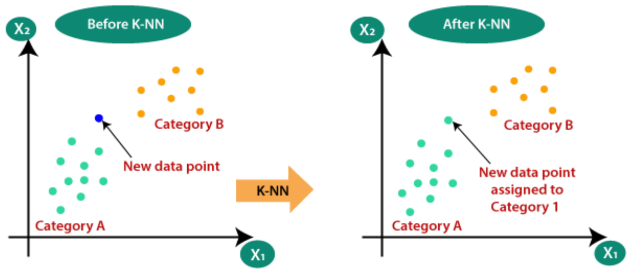

Fig. 2. K-Nearest neighbors' algorithm [15].

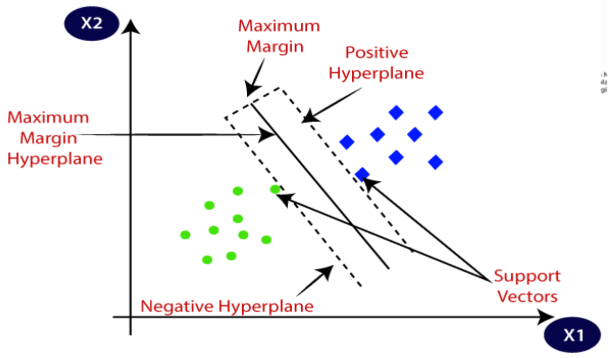

Fig. 3. Support vector machine algorithm [17].

In the SVM approach, the algorithm maps input data into a higher-dimensional feature space using a kernel function, which could be linear, polynomial, or radial basis function (RBF) kernel [18]. By transforming the data in this manner, SVM can discover a hyperplane that maximizes the margin between classes, thereby enhancing its generalization capability.

A key aspect of SVM is to identify the hyperplane that not only separates classes but also maximizes the distance to the nearest data points, known as support vectors. This property renders SVM robust to outliers and allows it to accommodate non-linear decision boundaries through various kernel functions.

SVM is versatile, accommodating both binary and multi-class classification tasks. For binary classification, SVM endeavors to locate a decision boundary that effectively distinguishes between the two classes. In multi-class scenarios, SVM can be extended using approaches such as one-vs-one or one-vs-rest [19].

D. Naive Bayes

The Naive Bayes classifier stands as a well-known machine learning algorithm based on the application of Bayes' theorem, assuming feature independence [20]. It finds wide application in classification tasks, particularly in natural language processing and text mining.

The Naive Bayes classifier computes the probability of each class given a set of input features and selects the class with the highest probability as the predicted outcome. Its strength lies in its simplicity, efficiency, and ability to handle high-dimensional data effectively (see Fig. 4).

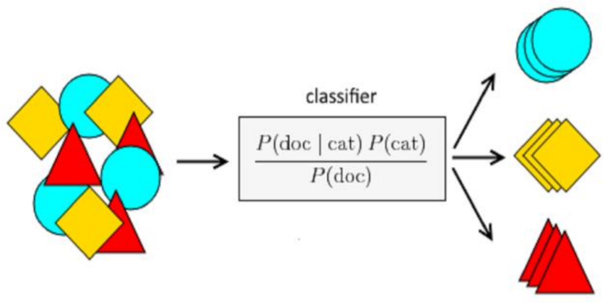
classifier
$\frac{P(\text{doc} \mid \text{cat})\, P(\text{cat})}{P(\text{doc})}$

Fig. 4. Naive Bayes algorithm [21].

The algorithm is termed 'naive' due to its assumption of feature conditional independence given the class. This simplifying assumption enables efficient estimation of class probabilities by multiplying individual feature probabilities. While this assumption may not hold true in real-world scenarios, Naive Bayes often performs well and yields reliable results.

Various variations of the Naive Bayes classifier exist, such as Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes, each suited for different data types and feature distributions [22].

Despite its simplicity, the Naive Bayes classifier has shown competitive performance compared to more complex algorithms. However, it is essential to acknowledge that Naive Bayes assumes feature independence, which may not always hold in practical scenarios.

E. Decision Trees

Decision Trees are a well-established machine-learning algorithm utilized for classification and regression tasks [23].

These models are renowned for their intuitive and interpretable nature, constructing a tree-like flowchart based on dataset features. The Decision Tree algorithm recursively partitions the dataset, creating a tree-like structure (see Fig. 5), with internal nodes representing features and branches denoting possible feature values, leading to leaf nodes as final predicted outcomes or classes.

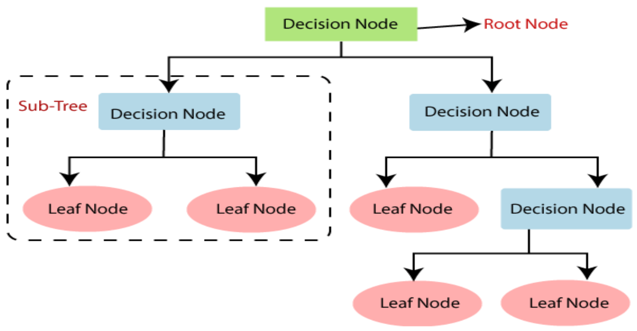

Fig. 5. Decision tree algorithm [24].

The construction involves selecting the best feature to split the data at each node based on specific criteria like information gain or Gini impurity, aiming for maximized homogeneity within subsets. Decision Trees handle both categorical and numerical features, learning complex decision boundaries and effectively managing missing values and outliers [25].

Their applications extend to feature selection and identifying crucial features for decision-making. However, overfitting risks exist, especially with excessively deep or complex trees, mitigated by techniques like pruning and maximum tree depth setting [26].

F. Random Forest

Random Forest stands as a widely used ensemble learning algorithm renowned for its ability to combine multiple decision trees to make accurate predictions [27]. Its versatility and robustness make it well-suited for a wide range of classification and regression tasks.

The essence of Random Forest lies in building an ensemble of decision trees, each trained on a different subset of the training data through bootstrapping [27]. Moreover, at each node of the tree, only a random subset of features is considered for splitting, introducing an element of randomness that helps mitigate overfitting and enhances the model's generalization ability (see Fig. 6).

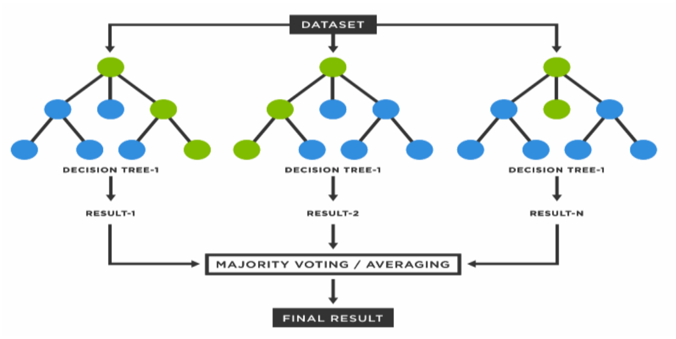

Fig. 6. Random forest algorithm [28].

To produce the final prediction, Random Forest aggregates the predictions of all individual trees either through voting (for classification) or averaging (for regression). This ensemble approach effectively reduces variance and improves the overall performance of the model.

The advantages of Random Forest are numerous, encompassing its capacity to handle high-dimensional data, automatically select important features, and remain robust in the presence of outliers and noisy data [29]. Additionally, it facilitates the estimation of feature importance, offering valuable insights into the underlying data.

However, it is essential to acknowledge that Random Forest may suffer from high computational complexity, and its results may not be as interpretable as those of single decision trees. To optimize its performance, hyperparameter tuning, including adjusting the number of trees and the maximum depth of each tree, becomes necessary [27].

G. Gradient Boost

Gradient Boost is a highly popular and effective machine-learning algorithm widely employed in various domains due to its ability to combine weak prediction models and create a robust predictive model [30]. It belongs to the category of boosting algorithms, which iteratively train new models to correct the errors made by previous models.

In Gradient Boost, weak models, typically decision trees, are trained in a stage-wise manner. At each stage, the model is trained on the data with a modified version of the target variable, representing the residuals or errors of the previous models. This process focuses on minimizing the errors made by the ensemble of models (see Fig. 7).

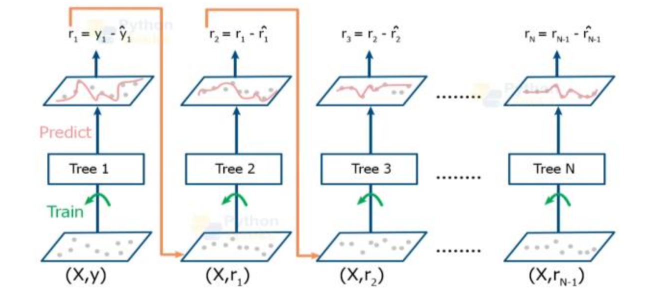

Fig. 7. Gradient boost algorithm [31].

The training process involves optimizing a loss function, such as mean squared error for regression or log loss for classification, to determine the optimal weights and parameters of the weak models. The final model is a combination of these weak models, and their predictions are weighted based on their performance on the training data.

One of the key strengths of Gradient Boost is its ability to handle complex datasets and capture non-linear relationships, leading to accurate predictions [30].

However, it is essential to be cautious about overfitting when using Gradient Boost. Controlling model complexity through regularization techniques, such as shrinkage and subsampling, can help mitigate this issue. Moreover, due to its iterative nature, Gradient Boost can be computationally expensive, necessitating careful tuning of hyperparameters, such as learning rate, tree depth, and number of iterations, to achieve optimal performance [32].

III. Related Work

URL verification is crucial for protecting users from phishing attacks, an often overlooked vulnerability [33]. However, traditional phishing detection methods exhibit limited accuracy, detecting only around 20% of attempts [33]. To overcome this, machine learning (ML) techniques have shown promise, though challenges arise with large databases and time-consuming processes [34]. Additionally, heuristics-based approaches suffer from significant false-positive rates [34]. Previous research focuses on improving anti-phishing models through feature reduction and ensemble methods [14].

Phishing URL detection is commonly treated as a classification problem using ML algorithms [35]. Constructing an ML-based detection model requires relevant properties to distinguish phishing from legitimate websites [35]. Robust ML approaches have demonstrated high detection accuracy [35], with various feature selection strategies employed to reduce feature numbers [35].

Common classifiers like Decision Trees (DT), C4.5, k-Nearest Neighbors (k-NN), and Support Vector Machines (SVM) are widely used in phishing detection research due to their accuracy and efficiency [36]. However, deep learning models face challenges such as manual parameter adjustment, lengthy training periods, and suboptimal detection accuracy [37]. Researchers emphasize the significance of ensemble learning techniques, feature selection, and reduction to address these issues [38]. Different classifiers, including Naive Bayes (NB) and SVM, have been explored [39]. Random Forest (RF) has also been successful in distinguishing phishing attacks from normal websites, with Subasi et al. [40] achieving an exceptional classification performance of 97.36% using the random forest classifier. Feature selection has been a focus in another study, with characteristics grouped to identify the most effective ones for accurate phishing attack detection [41].

In the field of phishing website detection, Patil et al. [42] proposed three strategies involving URL attribute assessment, validation based on hosting and management, and visual appearance-based analysis. They comprehensively assessed various aspects of URLs and websites using ML methodologies and algorithms.

Joshi et al. [43] conducted research on phishing attack prediction, utilizing a binary classifier based on the RF algorithm and a feature selection algorithm called relief, using data from the Mendeley domain for feature selection and training the RF algorithm.

Ubing et al. [44] explored ensemble learning strategies like bagging, boosting, and stacking to achieve high accuracy in phishing detection by integrating decision tree classifiers.

Similarly, Alsariera et al. [45] investigated phishing website detection using the "Forest by Penalizing Attributes" (FPA) algorithm and its enhanced variations, employing ensemble learning strategies like bagging, boosting, and stacking.

Pandey et al. [46] contributed to the field with a novel hybrid model combining Random Forest and Support Vector Machine (SVM) techniques for detecting phishing on websites. Their experimental results demonstrated an impressive accuracy of 94%, outperforming traditional ML algorithms SVM (90%) and Random Forest (92.96%), highlighting the superior performance of the hybrid model in classifying phishing attacks.

Furthermore, Lakshmi et al. [47] introduced an innovative approach for detecting phishing websites, analyzing hyperlinks in the HTML source code. They constructed a feature vector with 30 parameters to train a supervised DNN model with an Adam optimizer. The model demonstrated exceptional performance, outperforming traditional ML algorithms with a remarkable accuracy rate of 96%.

Table I displays a concise overview of machine learning approaches employed in phishing website detection.

TABLE I. COMPARATIVE ANALYSIS OF RECENT MACHINE LEARNING TECHNIQUES FOR PHISHING DETECTION

Model	Dataset	Algorithm	Accuracy
Subasi et al. [40]	website	RF, KNN, SVM, ANN, RF, C4.5, CART, NB	97.36% 97.18% 97.17% 96.91% 96.79% 95.88% 95.79% 92.98%
Patil et al.[42]	URLs	LR, DT, RF	96.23% 96.23% 96.58%
Joshi et al.[43]	Websites	RF	97.63%
Ubing et al.[44]	UCI	Ensemble bagging, boosting, stacking	95.40%
Alsariera et al. [45]	UCI	ForestPA-PWDM, Bagged-ForestPA-PWDM, sAdab-ForestPA-PWDM	96.26% 96.5% 97.4%
Pandey et al. [46]	Websites	SVM,RF	94.00%
Lakshmi et al. [47]	UCI	DNN +Adam	96.00%

IV. METHODOLOGY

In this research study, our main objective was to identify the most effective machine-learning model for detecting phishing domains. To achieve this, we conducted experiments with seven distinct machine-learning techniques: Logistic Regression (LR), k-Nearest Neighbors (KNN), Support Vector Machine (SVM), Naive Bayes (NB), Decision Tree (DT), Random Forest (RF), and Gradient Boosting.

Our dataset consisted of over 11,055 records, with 31 website parameters and a corresponding class label indicating whether it was a phishing website (1) or not (-1). To improve the models' accuracy, we applied the MinMax normalization feature as a preprocessing strategy.

To ensure robust evaluation, we employed a ten-fold cross-validation method during the classification process. This approach enabled us to obtain a more accurate performance evaluation of the models on the dataset, ensuring reliable results.

After the classification process, we thoroughly assessed the machine learning algorithms' performance using various evaluation metrics commonly used in the field, including accuracy, precision, recall, and F1-score. These metrics allowed us to make meaningful comparisons between the algorithms, ultimately identifying the most suitable approach for effectively detecting phishing websites.

To visually illustrate the concept, (see Fig. 8) presents a graphical representation of the process.

A. The Dataset

The research utilized a dataset obtained from the UCI machine-learning repository, which can be accessed at [48]. The dataset contains 11,055 records, and each sample within the dataset is composed of 31 website parameters. Among these parameters is a class label that indicates whether the website is classified as a phishing website or not, represented by values of 1 or -1 (Table II).

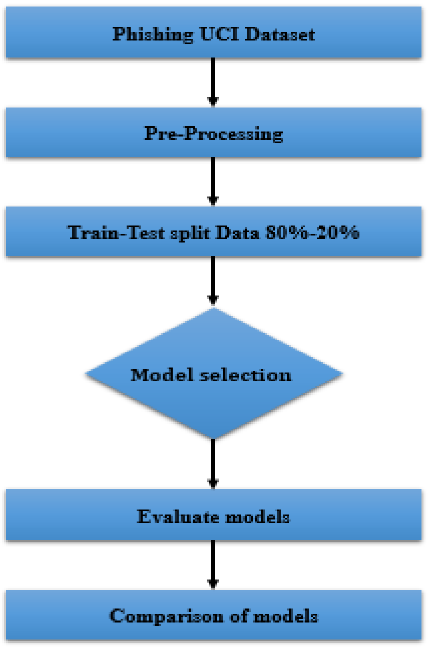

Fig. 8. Model's flowchart.

TABLE II. DESCRIPTION OF STUDIED PHISHING WEBSITE DATASET

Total number of attributes	31	
No. of independent variables	30	
No. of class variables	1	
Details of the class variable	Name: Result	
	Legitimate =-1	Phishing=1
	4898	6175
Total number of instances	11055	

B. Dataset Representation

The dataset utilized in this research incorporates novel features that have been experimentally introduced [48], including the assignment of new rules to certain well-known parameters. The dataset comprises 30 parameters, which are listed below:

'having_IP_Address', 'URL_Length', 'Shortening_Service', 'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix', 'having_Sub_Domain', 'SSLfinal_State', 'Domain_registration_length', 'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor', 'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL', 'Redirect', 'on_mouseover', 'RightClick', 'popUpWindow', 'Iframe', 'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank', 'Google_Index', 'Links_pointing_to_page', 'Statistical_report'.

C. Visualizing the Dataset: Heatmap of Feature Correlations

To gain further insights into the dataset and understand the relationships between its features, we generated a heatmap to visualize the pairwise correlations among the 30 parameters used in this research (see Fig. 9). The heatmap provides a clear and concise representation of the correlation matrix, allowing us to identify potential associations and patterns that might influence the classification of phishing websites [50].

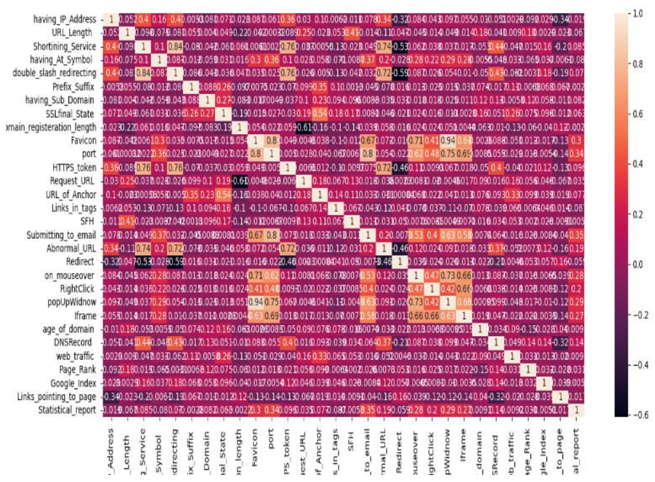
Fig. 9. The heatmap of dataset features.

Each cell in the heatmap is color-coded based on its correlation value, ranging from highly positive (dark shades) to highly negative (light shades) correlations. A correlation value close to 1 indicates a strong positive relationship, while a value close to -1 denotes a strong negative relationship. Conversely, a correlation value near 0 suggests little to no linear correlation between the respective features.

Upon analyzing the heatmap, several interesting observations emerge. For instance, we observe a strong positive correlation between the 'having_Sub_Domain' and 'Links_pointing_to_page' features, indicating that websites with more subdomains tend to have more links pointing to their pages. Conversely, there appears to be a negative correlation between 'URL_Length' and 'Page_Rank', suggesting that longer URLs may be associated with lower page ranks.

Additionally, the presence of certain novel features, as experimentally introduced in the dataset, reveals potential correlations with other established parameters. For example, the 'Abnormal_URL' feature exhibits a moderate negative correlation with 'SSLfinal_State,' suggesting that websites with abnormal URLs might be less likely to have a valid SSL certificate.

The heatmap not only facilitates the identification of such associations but also aids in assessing potential multicollinearity among the features. Identifying multicollinearity is crucial, as it can impact the performance and interpretability of predictive models.

D. The MinMax Normalization

In our study, our main focus was to boost the precision of our proposed models by introducing MinMax normalization as a critical preprocessing measure. This technique, widely acknowledged in the realm of machine learning, significantly enhances model accuracy, particularly for specific models that rely on it [49]. By employing MinMax normalization in our suggested model, we effectively rescaled the data to a domain of [0, 1], leading to notable improvements in the input quality during model training (see Eq. (1)).

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Where:

$X_{normalized}$ is the normalized value of the data point X.

X is the original value of the data point.

X_{min} is the minimum value in the dataset.

X_{max} is the maximum value in the dataset.

E. The Ten-fold Cross-validation Method

The ten-fold cross-validation method is a widely used technique in machine learning and statistical analysis to evaluate a model's performance on a dataset [51]. It involves ten iterations with different data splits for training and testing, yielding a robust estimate of the model's abilities. This approach mitigates bias and variance issues, providing a comprehensive evaluation of generalization to unseen data. The final performance metric is obtained by averaging the results from all ten iterations, ensuring an accurate assessment of the model's capabilities.

F. The Evaluation Metrics

The evaluation metrics are essential tools for assessing the performance of machine learning models. They provide quantitative measures of the model's accuracy, precision, recall, and F1-score. By using these metrics, researchers can make meaningful comparisons between different models and identify the most effective approach for their specific task.

1) Accuracy: Accuracy is a fundamental performance metric used to assess the overall correctness of a machine learning model. It represents the ratio of correctly predicted instances to the total number of instances in the dataset. In other words, it measures how often the model makes correct predictions. It is a simple and intuitive metric, but it might not be the best choice when dealing with imbalanced datasets.

$$\text{Accuracy} = \frac{\text{Number of correctly predicted instances}}{\text{Total number of instances}} \tag{2}$$

2) F1 Score: The F1 score is a balanced metric that takes into account both precision and recall. It is particularly useful when dealing with imbalanced datasets, where one class might dominate the others. The F1 score computes the harmonic mean of precision and recall, providing a single value that balances the trade-off between false positives (FP) and false negatives (FN).

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

3) Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive instances that are correctly identified by the model. It is essential when the cost of false negatives is high, as it focuses on minimizing the number of missed positive instances.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4}$$

4) Precision: Precision represents the proportion of true positive predictions among all positive predictions made by the model. It is crucial when the cost of false positives is high, as it aims to reduce the number of incorrectly classified positive instances.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (5)$$

In summary, accuracy measures overall correctness, F1 score balances precision and recall, recall focuses on minimizing false negatives, and precision aims to minimize false positives.

V. Findings and Analysis

In this section, we present the experimental results of our comparative study (Table III). We analyze the performance of each machine-learning algorithm using the evaluation metrics defined in the previous section. We provide a comprehensive analysis of the results, highlighting the strengths and weaknesses of each algorithm.

TABLE III. Evaluation Results in (%).

Classifier	Accuracy	F1 score	Recall	Precision
Gradient Boost	97.2%	96.9%	97%	96.8%
Random Forest	97.1%	97.3%	97.4%	97.2%
Decision Tree	96.3%	96.7%	96.7%	96.6%
K-Nearest Neighbors	95.6%	96.2%	96.8%	95.7%
Support Vector Machine	93.9%	95%	96.4%	93.7%
Logistic Regression	92.7%	93.8%	95%	92.7%
Naive Bayes Classifier	60.1%	45.3%	29.3%	99.2%

In this analysis, we evaluate the performance of various machine-learning models based on key metrics, including Accuracy, F1-score, Recall, and Precision. The models under consideration are Gradient Boost, Random Forest, Decision Tree, K-Nearest Neighbors, Support Vector Machine, Logistic Regression, and Naive Bayes Classifier.

Starting with Gradient Boost and Random Forest, both models showcase impressive results. Gradient Boost achieves a remarkable Accuracy of 97.2%, indicating its ability to make correct predictions for a majority of instances. The F1-score of 96.9% suggests a well-balanced trade-off between precision and recall. Additionally, with Recall and Precision scores of 97% and 96.8% respectively, it effectively identifies most positive instances while maintaining a high level of accuracy in positive predictions.

Random Forest, another strong performer, demonstrates an Accuracy of 97.1%, marginally trailing behind Gradient Boost. Nevertheless, its F1-score of 97.3% indicates an excellent balance between precision and recall. The model boasts a high Recall score of 97.4%, suggesting its proficiency in correctly identifying positive instances. Furthermore, its Precision score of 97.2% underscores its accuracy in positive predictions.

The Decision Tree model also shows promise with a respectable Accuracy of 96.3%. Its F1-score of 96.7% reflects a good balance between precision and recall. The Recall and Precision scores of 96.7% and 96.6% respectively affirm the model's effectiveness in correctly identifying positive instances and making accurate positive predictions.

K-Nearest Neighbors performs well, attaining an Accuracy of 95.6%. Its F1-score of 96.2% demonstrates a commendable balance between precision and recall. With Recall and Precision scores of 96.8% and 95.7% respectively, the model effectively identifies positive instances and makes accurate positive predictions.

The Support Vector Machine achieves an Accuracy of 93.9%, somewhat lower than the previously mentioned models. Nevertheless, its F1-score of 95% suggests a satisfactory balance between precision and recall. A Recall score of 96.4% indicates its effectiveness in identifying positive instances, and its Precision score of 93.7% underscores its accuracy in positive predictions.

Logistic Regression, with an Accuracy of 92.7%, provides a reasonable performance. Its F1-score of 93.8% signifies a good balance between precision and recall. A Recall score of 95% indicates its ability to effectively identify positive instances, while its Precision score of 92.7% reflects accurate positive predictions.

On the other hand, the Naive Bayes Classifier lags significantly behind the other models with an Accuracy of 60.1% and an F1-score of 45.3%. The low Recall score of 29.3% suggests its struggle to effectively identify positive instances. However, it exhibits an unexpectedly high Precision score of 99.2%, indicating that when it predicts a positive instance, it is usually correct. This discrepancy might imply a bias towards negative instances.

In conclusion, the analysis showcases Gradient Boost and Random Forest as top-performing models, excelling in various metrics. The Decision Tree, K-Nearest Neighbors, and Logistic Regression also demonstrate competitive performances. However, the Naive Bayes Classifier significantly underperforms in comparison to the other models, necessitating further investigation and improvements. By understanding the strengths and weaknesses of each model, we can make informed decisions when selecting the most suitable model.

After a thorough examination of our research findings, we will proceed to conduct a comparative analysis with other relevant studies in the field (Table IV).

TABLE IV. Evaluation of Current Phishing Domain Detection Models

Authors	Dataset	Algorithm	Accuracy
Ubing et al. [44]	UCI	Ensemble bagging, boosting, stacking	95.4%
Alsariera et al. [45]	UCI	ForestPA-PWDM, Bagged-ForestPA-PWDM, and Adab-ForestPA-PWDM	96.26% 96.5% 97.4%
Lakshmi et al. [47]	UCI	DNN +Adam	96.00%
Alnemari et al. [49]	UCI	Random Forest	97.3%

Ubing et al. [44], in their work, harnessed the power of ensemble learning techniques such as bagging, boosting, and stacking, securing an impressive accuracy of 95.4% on the UCI dataset. This commendable outcome highlights the prowess of ensemble methods in accurately identifying phishing attacks, emphasizing the model's capacity to make precise predictions.

Equally noteworthy, Alsariera et al. [45] proposed multiple meta-learner models rooted in ForestPA-PWDM, Bagged-ForestPA-PWDM, and Adab-ForestPA-PWDM. Their experimental results yielded outstanding accuracies of 96.26%, 96.5%, and a remarkable 97.4%, respectively, on the UCI dataset. This exemplifies the efficacy and versatility of their ensemble-based approaches, which outperformed a majority of existing techniques, underscoring their potential as powerful solutions for phishing detection.

Venturing into the realm of deep learning, Lakshmi et al. [47] introduced the DNN +Adam model, accomplishing an impressive accuracy of 96.00% on the UCI dataset. This result vividly illustrates the effectiveness of deep learning methodologies in tackling phishing attacks, showcasing the model's ability to discern malicious websites with a remarkable degree of accuracy.

Furthermore, Alnemari et al. [49] adopted the Random Forest model, achieving an exceptional accuracy of 97.3% on the UCI dataset. The success of this approach showcases the formidable strength of Random Forest in detecting phishing attacks, operating at a high level of accuracy and outperforming several alternative methods.

Now, shifting our focus to our own research, our results reinforce the notion of competitive performance in the realm of phishing attack detection. With accuracy values ranging from 93.9% to 97.2%, the machine learning models utilized in our study prove their efficacy. Particularly, the Gradient Boost and Random Forest models demonstrated remarkable accuracies of 97.2% and 97.1%, respectively, rivaling or even surpassing the accuracy rates reported in the aforementioned studies.

The alignment of our results with the findings of previous studies accentuates the effectiveness of diverse machine learning techniques in detecting phishing attacks.

Overall, our research findings substantiate commendable performance, with the competitive accuracies achieved by our models showcasing their potential practicality in real-world phishing detection scenarios. The insights gleaned from our study empower us to make informed decisions when selecting the most appropriate machine learning technique to combat phishing threats in diverse applications.

VI. Conclusion

In this research, we have delved into the critical domain of phishing website detection, exploring diverse machine learning techniques and their effectiveness in countering the ever-evolving cybersecurity threat posed by phishing attacks. The rampant increase in such fraudulent activities has presented significant challenges to individuals and organizations worldwide, necessitating the development of robust and efficient methods to detect and thwart these deceitful websites.

Through an extensive analysis of our research results and a thorough comparison with other relevant studies, we have uncovered promising insights into the effectiveness of various machine learning models for detecting phishing attacks. Notably, the Gradient Boost and Random Forest models have demonstrated exceptional performance, showcasing accuracy rates that either align with those reported in the existing literature. This remarkable potential positions these models as viable candidates for real-world applications in phishing detection scenarios, playing a pivotal role in fortifying cybersecurity measures and shielding users from the dangers posed by phishing attacks.

References

[1] Z. Alkhalil, C. Hewage, L. Nawaf, and I. Khan, "Phishing Attacks: A Recent Comprehensive Study and a New Anatomy," Front. Comput. Sci., vol. 3, p. 563060, 2021, doi: 10.3389/fcomp.2021.563060.

[2] H. Aldawood and G. Skinner, "An Advanced Taxonomy for Social Engineering Attacks," International Journal of Computer Applications, vol. 177, pp. 975-8887, 2020. doi: 10.5120/ijca2020919744.

[3] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2006, pp. 581-590.

[4] C. Herley, "So long, and no thanks for the externalities: The rational rejection of security advice by users," in Proceedings of the 2009 workshop on New security paradigms, 2011, pp. 133-144.

[5] K. L. Chiew, K. Yong, and C. C. L. Tan, "A Survey of Phishing Attacks: Their Types, Vectors and Technical Approaches," Expert Systems with Applications, vol. 106, 2018, pp. 10.1016/j.eswa.2018.03.050.

[6] D. Gavrilut and I. Zaporojan, "The use of machine learning algorithms for phishing detection," in 2019 International Conference on Innovations in Intelligent Systems and Applications (INISTA), 2019, pp. 1-5.

[7] D. Hosmer and S. Lemeshow, "Applied Logistic Regression," 2004. doi: 10.1002/9781118548387.

[8] A. Agresti, "Foundations of Linear and Generalized Linear Models," John Wiley & Sons, 2015.

[9] "Logistic Regression in Machine Learning—Javatpoint." Available online: https://www.javatpoint.com/logistic-regression-in-machine-learning (accessed on 19 July 2023).

[10] S. Menard, "Applied Logistic Regression Analysis," Sage Publications, 2002.

[11] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," in 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[12] E. Rodrigues, "Combining Minkowski and Cheyshev: New Distance Proposal and Survey of Distance Metrics Using k-Nearest Neighbours Classifier," Pattern Recognition Letters, vol. 110, 2018, pp. 10.1016/j.patrec.2018.03.021.

[13] P. Cunningham, M. Cord, and S. Delany, "Supervised Learning," in Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, IGI Global, 2010, pp. 20-36. doi: 10.1007/978-3-540-75171-7_2.

[14] B. V. Dasarathy, "Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques," IEEE Computer Society Press, 1991.

[15] "K-Nearest Neighbor (KNN) Algorithm for Machine Learning—Javatpoint." Available online: https://www.javatpoint.com/logistic-regression-in-machine-learning (accessed on 19 July 2023).

[16] V. Kecman, "Support Vector Machines – An Introduction," in Support Vector Machines: Theory and Applications, 2nd ed., Springer, 2015, pp. 1-25. doi: 10.1007/10984697_1.

[17] "Support Vector Machine Algorithm—Javatpoint." Available online: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm (accessed on 19 July 2023).

[18] B. Schölkopf and A. J. Smola, "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond," MIT Press, 2002.

[19] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," Journal of Machine Learning Research, vol. 2, Dec. 2002, pp. 265-292.

[20] P. Flach and N. Lachiche, "Naive Bayesian Classification of Structured Data," Machine Learning, vol. 57, 2004, pp. 233-269. doi: 10.1023/B:MACH.0000039778.69032.ab.

[21] "Naïve Bayes Classifier from Scratch with Hands-on Examples in R." Available online: https://insightimi.wordpress.com/2020/04/04/naive-bayes-classifier-from-scratch-with-hands-on-examples-in-r (accessed on 19 July 2023).

[22] T. Almeida, J. Almeida, and A. Yamakami, "Spam filtering: How the dimensionality reduction affects the accuracy of Naive Bayes classifiers," J. Internet Services and Applications, vol. 1, 2011, pp. 183-200. doi: 10.1007/s13174-010-0014-7.

[23] A. Priyam, R. Gupta, A. Rathee, and S. Srivastava, "Comparative Analysis of Decision Tree Classification Algorithms," International Journal of Current Engineering and Technology, vol. 3, pp. 334-337, June 2013. doi: ISSN 2277-4106.

[24] "Machine Learning Decision Tree Classification Algorithm—Javatpoint." Available online: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm (accessed on 19 July 2023).

[25] P. Sen, M. Hajra, and M. Ghosh, "Supervised Classification Algorithms in Machine Learning: A Survey and Review," in Proceedings of the International Conference on Advanced Computing Technologies and Applications (ICACTA), 2020, pp. 142-155. doi: 10.1007/978-981-13-7403-6_11.

[26] K. P. Murphy, "Machine Learning: A Probabilistic Perspective," MIT Press, 2012.

[27] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[28] "What is a Random Forest?" Available online: https://www.tibco.com/reference-center/what-is-a-random-forest (accessed on 19 July 2023).

[29] D. R. Cutler, T. C. Edwards Jr, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson, and J. J. Lawler, "Random forests for classification in ecology," Ecology, vol. 88, no. 11, pp. 2783-2792, 2007.

[30] Kavzoglu, T., Teke, A. Predictive Performances of Ensemble Machine Learning Algorithms in Landslide Susceptibility Mapping Using Random Forest, Extreme Gradient Boosting (XGBoost) and Natural Gradient Boosting (NGBoost). *Arab J Sci Eng* 47, 7367–7385 (2022). https://doi.org/10.1007/s13369-022-06560-8.

[31] "Gradient Boosting Algorithm in Machine Learning." Available online: https://pythongeeks.org/gradient-boosting-algorithm-in-machine-learning/ (accessed on 19 July 2023).

[32] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of Statistics, pp. 1189–1232, 2001.

[33] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in Proceedings of the SIGCHI conference on Human Factors in Computing Systems, 2006, pp. 581-590.

[34] A. A. Yavuz and H. Polat, "Phishing websites detection using machine learning techniques," Expert Systems with Applications, vol. 55, pp. 225-233, 2016.

[35] M. Alazab, R. Broadhurst, and H. Chen, "Predictive data mining for combating phishing attacks," IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 41, no. 3, pp. 468-479, 2011.

[36] S. Wang, X. Jiang, W. Cui, T. Huang, B. Li, and Y. Qian, "Robust detection of web phishing using RBF based on an improved extreme learning machine," Expert Systems with Applications, vol. 42, no. 21, pp. 7374-7382, 2015.

[37] M. Alazab, R. Broadhurst, and J. Slay, "PhishNet: Predictive phishing detection system," Information Sciences, vol. 305, pp. 65-80, 2015.

[38] S. K. Mahanta, N. Sarma, D. Das, and A. Das, "A novel hybrid approach for phishing detection using random forest," Procedia Computer Science, vol. 89, pp. 120-127, 2016.

[39] P. K. Biswas, M. I. Hossain, D. K. Bhattacharyya, M. Nasipuri, D. K. Basu, and M. Kundu, "A feature selection mechanism for phishing detection," Applied Soft Computing, vol. 13, no. 8, pp. 3464-3475, 2013.

[40] A. Subasi, E. Molah, F. Almkallawi, and T. Chaudhery, "Intelligent phishing website detection using random forest classifier," in Proceedings of the 2017 International Conference on Engineering and Computer Technology (ICECTA), 2017, pp. 1-5. doi: 10.1109/ICECTA.2017.8252051.

[41] D. Salem, "On determining the most effective subset of features for detecting phishing websites," International Journal of Computer Applications, vol. 122, pp. 1-7, 2015. doi: 10.5120/21813-5191.

[42] V. Patil, P. Thakkar, C. Shah, T. Bhat, and S. P. Godse, "Detection and Prevention of Phishing Websites Using Machine Learning Approach," in 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5. doi: 10.1109/ICCUBEA.2018.8697412.

[43] A. Joshi and Prof. T. R. Pattanshetti, "Phishing Attack Detection using Feature Selection Techniques," in Proceedings of International Conference on Communication and Information Processing (ICCIP) 2019.

[44] A. Ubing, S. Kamilia, A. Abdullah, N. Zaman, and M. Supramaniam, "Phishing Website Detection: An Improved Accuracy through Feature Selection and Ensemble Learning," Int. J. Adv. Comput. Sci. Appl., vol. 10, pp. 252-257, 2019.

[45] Y. A. Alsariera, A. V. Elijah, and A. O. Balogun, "Phishing Website Detection: Forest by Penalizing Attributes Algorithm and Its Enhanced Variations," Arab. J. Sci. Eng., vol. 45, pp. 10459-10470, 2020.

[46] A. Agresti, "Foundations of Linear and Generalized Linear Models," John Wiley & Sons, 2015.

[47] L. Lakshmi, M. P. Reddy, C. Santhaiah, and U. J. Reddy, "Smart Phishing Detection in Web Pages Using Supervised Deep Learning Classification and Optimization Technique ADAM," Wirel. Pers. Commun., vol. 118, pp. 3549-3564, 2021.

[48] UCI Machine Learning Repository, "Phishing Websites Data Set," Available online: https://archive.ics.uci.edu/ml/datasets/phishing+websites.

[49] S. Alnemari and M. Alshammari, "Detecting Phishing Domains Using Machine Learning," Applied Sciences, vol. 13, article no. 4649, 2023. doi: 10.3390/app13084649.

[50] K. Fu, D. Cheng, Y. Tu, and L. Zhang, "Credit Card Fraud Detection Using Convolutional Neural Networks," in Proceedings of the 2016 International Conference on Neural Information Processing, pp. 483-490, 2016. doi: 10.1007/978-3-319-46675-0_53.

[51] M. Gaag, T. Hoffman, M. Remijsen, R. Hijman, L. de Haan, B. Meijel, P. van Harten, L. Valmaggia, M. Hert, A. Cuijpers, and D. Wiersma, "The five-factor model of the Positive and Negative Syndrome Scale - II: A ten-fold cross-validation of a revised model," Schizophrenia research, vol. 85, pp. 280-7, 2006. doi: 10.1016/j.schres.2006.03.021.