Comparison of Machine Learning Algorithms for Phishing Detection of Uniform Resource Locators

Fritz Noel C. Quilacio[1], Asher Paul M. Cuadra[2], Raj G. Redaja[3], Shane M. Gabaton[4], Merijoy P. Marinog[5]

[1,2,3,4,5]Guimaras State University
[1]fritznoel.quilacio@gsu.edu.ph, [2]asherpaul.cuadra@gsu.edu.ph, [3]raj.redaja@gsu.edu.ph,
[4]shane.gabaton@gsu.edu.ph [5]merijoy.marinog@gsu.edu.ph

***Abstract*— This study focuses on the development of a Machine Learning Algorithms and Hybrid Model that combines Naïve Bayes and all other algorithms which are the XGBoost, Support Vector Machine, Random Forest and Decision Tree to enhance the detection of phishing websites. The system was trained and tested using three datasets (Dataset1, Dataset2 and Dataset3) with various train-test split (50%-50%, 60%-40%, 70%-30%, 80%-20%, and 90%-10%) to evaluate its performance under different conditions. The preprocessing stage included normalization, feature selection, and data balancing using SMOTE to improve accuracy. To validate the model, 10-Fold Cross Validation was applied to ensure consistency and prevent overfitting. The system's performance was assessed using key evaluation metrics such as accuracy, precision, recall, and F1-score. The results revealed the XGBoost and hybrid model is Naïve Bayes and Decision Tree classifiers, demonstrating higher detection accuracy and reliability.**

***Keywords*—** *Phishing Detection, Hybrid Model, XGBoost, Naïve Bayes, Decision Tree, Random Forest, Support Vector Machine, Train-Test Split, Machine Learning*

I. INTRODUCTION

Phishing is a cybercrime whereby a target unsuspectingly discloses sensitive information, including usernames, passwords, and credit card data, through impersonation of a trusted entity in electronic communications. Normally carried out through emails, messages, or fake websites, phishing attacks prey on human psychology and social engineering tactics to trick a victim. According to the Anti-Phishing Working Group, phishing is still one of the most common online threats, with millions of phishing attempts being reported across the world each year [1].

This study supports the United Nations' Sustainable Development Goals (SDGs) [2], particularly SDG 9: Industry, Innovation, and Infrastructure and SDG 16: Peace, Justice, and Strong Institutions. SDG 9 focuses on building resilient infrastructure, promoting innovation, and advancing technology. As cyber threats lik phishing attacks continue to grow, strengthening cybersecurity measures is essential in protecting businesses, organizations, and individuals from online fraud. By improving phishing detection models, this research contributes to the development of more reliable and secure digital infrastructures, ensuring safer online transactions and communications.

Meanwhile, SDG 16 aims to promote peace, justice, and strong institutions, which includes ensuring security in both physical and digital spaces. Phishing is a form of cybercrime that targets individuals and organizations by tricking them into giving away sensitive information. This study helps in the fight against cybercrime by providing effective methods to detect and prevent phishing attacks. Strengthening cybersecurity systems supports the protection of personal data, financial assets, and institutional security, ultimately contributing to a safer and more trustworthy digital environment [2].

II. REVIEW OF RELATED LITERATURE

A. Comparative Study of Machine Learning Algorithms for Phishing Website Detection

The primary goal of this research project [12] was to determine the most Effective Machine-Learning Model for detecting phishing domains. To achieve this, they evaluated seven Machine-Learning Techniques: Logistic Regression (LR), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Naive Bayes (NB), Decision Trees (DT), Random Forests (RF), and Gradient Boosting.

The analysis concluded that Gradient Boosting and Random Forest emerged as the best-performing models across various evaluation metrics. Decision Trees, K-Nearest Neighbors, and Logistic Regression also demonstrated competitive performance. However, the Naive Bayes Classifier significantly underperformed compared to the other models, suggesting a need for further research and improvements. By understanding the strengths and limitations of each model, more informed decisions can be made when selecting the optimal model for phishing detection [12]. For detailed evaluation results, refer to Table 1.

B. Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection

The results [13] conclusively demonstrate that tuned classification algorithms outperform existing classification algorithms in terms of accuracy. The researchers' investigations reveal that data balancing leads to a minor improvement in performance. In hyper-parameter tuning, the experimental results indicate that Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting (GB) algorithms achieve higher accuracy when applied to Dataset-1. For Dataset-2, SVM, KNN, GB, and Decision Tree (DT) show significant improvements. Regarding feature selection, the results highlight the minimum number of features required to achieve a higher accuracy margin, which can guide future research in selecting the best scoring function with an optimal number of features. For Dataset-2, Gradient Boosting (GB) and Extreme Gradient Boosting (XGB) achieve accuracy values of 98.27% and 98.21%, respectively [13]. Detailed evaluation results are provided in Table 1.

C. Comparative Evaluation of Machine Learning Algorithms for Phishing Site Detection

The present research [14] evaluates the effectiveness of eight Machine Learning (ML) and Deep Learning (DL) algorithms—Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Decision Tree (DT), Extreme Gradient Boosting (XGBoost), Logistic Regression (LR), Convolutional Neural Network (CNN), and a DL model—in identifying phishing attacks. This study utilizes two real-world datasets, Mendeley and UCI, and employs performance metrics such as accuracy, precision, recall, false positive rate (FPR), and F1-score. The study demonstrates consistent model performance across both datasets, highlighting the stability and reliability of the proposed approach [14]. Detailed evaluation results are provided in Table 1.

Table 1 shows the results of a study of different Machine Learning Algorithms for Phishing Detection using the following legend:

A represents the Evaluation results in the study [12].
B represents the Dataset1 of Evaluation results in the study [13].
C represents the Dataset2 of Evaluation results in the study [13].
D represents the Evaluation results in the study [14].
X represents the algorithm that was not used for study of other researchers.

Table 1 - Summary Evaluation Results of Related Literature Studies

Classifier Algorithms	Accuracy (%)				F1 Score (%)				Recall (%)				Precision (%)			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
XG Boost	X	98	97	97	X	98	98	96	X	98	98	95	X	98	97	96
Random Forest	97	98	97	98	97	97	97	94	97	97	98	97	97	98	97	98
Decision Tree	96	96	97	97	97	96	97	97	97	96	97	97	97	96	97	97
Support Vector Machine	94	94	95	96	95	94	95	95	96	94	96	95	94	94	94	96
Logistic Regression	93	93	92	95	94	93	92	94	95	95	93	94	93	92	91	94
Gradient Boost	97	97	94	X	97	97	95	X	97	97	95	X	97	97	94	X
K-Nearest Neighbor	96	94	95	96	96	94	94	94	97	93	95	95	96	94	95	94
Naive Bayes	60	84	91	X	45	81	91	X	29	72	90	X	99	94	91	X
Convolutional Neural Network	X	X	X	99	X	X	X	99	X	X	X	98	X	X	X	98
Deep Learning	X	X	X	98	X	X	X	94	X	X	X	95	X	X	X	94

D. Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification

In this study [15], they conducted a thorough performance analysis of 11 Machine and Deep Learning algorithms using six different IoT-related datasets. One of the key findings of the study was that the performance of Machine Learning Algorithms varied significantly depending on the dataset used. For instance, Random Forests (RF) performed exceptionally well on most datasets, achieving high accuracy and precision.

The use of multiple datasets also allows for a more nuanced understanding of algorithm performance. In the study, the authors evaluated algorithms based on various metrics, including precision, recall, F1-score, accuracy, and ROC-AUC score. By using multiple datasets, they were able to provide a more detailed comparison of the algorithms, showing that no single algorithm outperformed others across all datasets. This finding supports the idea that the choice of algorithm should be tailored to the specific characteristics of the dataset and the problem at hand [15].

E. Phishing Detection Using Machine Learning Techniques

In this study [17], they highlighted that there is no guarantee that combining multiple classifiers in an ensemble always outperforms the best individual classifier. Their findings motivate future research to enhance model performance by incorporating additional

features into the dataset. Additionally, they propose exploring and developing new mechanisms to extract novel features from websites, ensuring the approach remains effective against evolving phishing techniques [17].

III. TECHNICAL BACKGROUND

A. Data Collection

1) Phishtank: This is a collaborative clearing house for data and information about phishing on the Internet [17]. This is where the Phishing URLs are collected.

2) Alexa Repository: Alexa, formally known as Alexa Internet, Inc., is a subsidiary company of Amazon.com. It provides commercial web traffic data, global rankings, browsing behavior, and other information on over 30 million websites [18]. This is where the Legitimate URLs are collected.

3) Github: GitHub is a web-based version control and collaboration platform for software developers [19]. This is where the dataset1and dataset2 gathers to be compared with the researchers' datasets.

B. Programming Language used and the Imported Libraries

Python. Is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation [20].

C. Process Flow of Classification Model

Fig. 1 shows the process starts with Dataset1 or Dataset2, which serve as the raw data sources. Since raw data is often unstructured or varies in scale, it first goes through preprocessing, where it is normalized to ensure consistency. After that, the data undergoes feature selection and balancing to improve model performance. The most important features are chosen using SelectKBest(), ensuring that only relevant information is used for training. Since datasets can sometimes be imbalanced—meaning there are more examples of one class than another—SMOTE() is applied to generate synthetic samples and balance the data.

Once the data is prepared, it is split into 70% for training and 30% for testing, so the model learns from most of the data while keeping some aside for evaluation. During training, 10-Fold Cross Validation is used to validate performance across different subsets of the training data, preventing the model from overfitting. Additionally, hyperparameters are fine-tuned to optimize the model's performance. After training, the model is tested using the 30% test data to evaluate its accuracy and reliability.

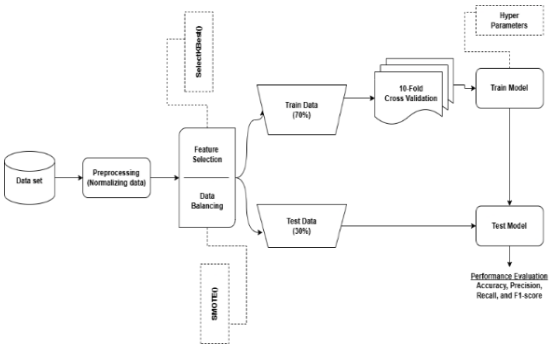

Figure 1. Process Flow of the Classification Model for Dataset1 and Dataset2

Fig. 2 shows, outlines the structured approach used to develop and evaluate a classification model. The process begins with a dataset, which undergoes preprocessing where the data is normalized to ensure consistency across all values.

Following preprocessing, the feature selection step identifies the most relevant attributes that contribute to the model's performance. Additionally, data balancing techniques, such as SMOTE (Synthetic Minority Over-sampling Technique), are applied to handle any class imbalances and improve model fairness.

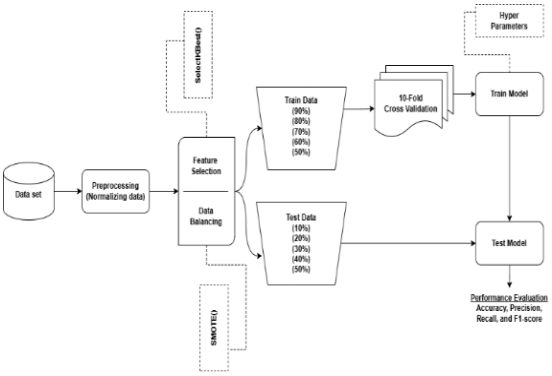

Figure 2. Process Flow of the Classification Model for Dataset3

D. How the Project Worked

This comparison of machine learning algorithms for Phishing Detection of Uniform Resource Locators gives the ranking of the most accurate and precise algorithm in phishing detection.

1) Data Collection. First is the Data Collection. To train the phishing detection model, the researchers gathered a dataset of both phishing and legitimate URLs. These URLs were collected from sources like PhishTank, the Alexa Repository, and GitHub. The researcher gathered 20,000 URLs, to be exact 10,000 URLs for phishing and 10,000 URLs for legitimate.

2) Feature Extraction. Next is the Feature

Extraction. Once the dataset was collected, the next step was to analyze the URLs by extracting features that help differentiate phishing sites from legitimate ones. This was done using Python, which processed each URL to identify key characteristics. Here is the process on how the Researchers gets the Feature Extracted datasets:

Data Preprocessing and Model Training

And then next is Data Preprocessing and Model Training. After the feature extraction, the datasets were preprocessed to prepare it for training. This involved removing non-relevant columns like raw URLs, normalizing numerical values using StandardScaler to improve model performance, and also handling imbalance data using SMOTE (Synthetic Minority Over-sampling Technique) to ensure fair model training.

Train the five (models) the Decision Tree, XGBoost, Random Forest, Support Vector Machine, Naïve Bayes and the Hybrid Model of Naïve Bayes and Decision Tree. For the Hybrid Model here is the researchers do it:

Trains Naïve Bayes and Decision Tree models separately. Uses majority voting for prediction: If both models agree, their prediction is used. If they disagree, Decision Tree's prediction is used.

Model Evaluation

And then lastly is the Model Evaluation. Once the models were trained, they were evaluated based on their ability to accurately classify phishing and legitimate URLs.

Metrics to Compare Model Performance

The researchers used the following metrics to In Fig. 3 shows the Accuracy Score it is the percentage of correct predictions. In Fig. 4 shows, Precision Score measures how many URLs classified as phishing are actually phishing. In Fig. 5 shows, Recall Score measures how many actual phishing URLs were correctly detected. And in Fig. 6 shows, F1-Score is a balance between precision and recall.

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

Figure 3. Accuracy

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

Figure 4. Precision

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

Figure 5. Recall

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Figure 6. F1-Score

IV. RESULTS AND DISCUSSION

Finally, the results were compared and visualized using heatmaps and performance charts to determine which algorithm performed best in detecting phishing URLs. Plots confusion matrices for all models. Plots model performance comparison. Plots model training and prediction times. Then convert the results into Data Frame and sort it based on accuracy.

Compilation of Results

In Table 3 shows how well different machine learning models performed in detecting phishing websites using three datasets: Dataset1 from 2019, Dataset2 from 2024, and Dataset3 collected by researchers. The results clearly show that the quality and freshness of the dataset have a big impact on accuracy. Since Dataset1 is from 2019, it has older phishing patterns that are not as strong, making it harder for the models to detect phishing sites accurately. This is why the accuracy and recall scores for Dataset1 are lower compared to the newer datasets. In contrast, Dataset2, which is from 2024, has more updated features, allowing all models to achieve 100% accuracy. The same goes for Dataset3, which was carefully collected by researchers, ensuring it includes strong phishing indicators that help the models perform well.

Among the models tested, XGBoost and SVM stand out as the top performers, achieving high accuracy even when dealing with the older Dataset1. However, SVM struggles a bit with recall, meaning it might miss some phishing sites. Decision Tree and Random Forest also performed well, especially on the newer datasets, but they had some minor drops in accuracy for Dataset1. Naïve Bayes, on the other hand, had the lowest recall and F1-score on Dataset1, which means it had trouble correctly identifying phishing websites from older data.

Looking at efficiency, Naïve Bayes and Decision Tree trained and made predictions the fastest, making them good choices for situations where speed is important. Meanwhile, XGBoost and Random Forest took longer to train but delivered excellent accuracy. SVM had the slowest prediction time on Dataset1,

possibly because it struggled with older phishing patterns.

Table 3 - Dataset1, Dataset2 And Dataset3 In 70-30 Train-Test Splits

Algo	Accuracy (%)			Precision (%)			Recall (%)			F1-score (%)			Train Time (s)			Prediction Time (s)		
	D1	D2	D3	D1	D2	D3	D1	D2	D3	D1	D2	D3	D1	D2	D3	D1	D2	D3
XGB	87	100	100	92	100	100	81	100	100	86	100	100	0.21	0.20	0.16	0.01	0.00	0.00
SVM	80	100	100	97	100	100	63	100	100	76	100	100	0.68	0.08	0.02	0.11	0.01	0.00
DT	81	100	100	98	100	100	64	100	99	78	100	100	0.01	0.02	0.01	0.00	0.01	0.00
RF	82	100	100	98	100	100	65	100	99	78	100	100	0.40	0.87	0.45	0.02	0.03	0.03
NB	79	100	99	98	100	99	60	100	98	74	100	99	0.00	0.01	0.01	0.00	0.01	0.00

Table 3 shows the result of Dataset1, Dataset2 and Dataset3 in 70-30 Train-Test Splits using the following legend:

D1 represents Dataset1
D2 represents Dataset2
D3 represents Dataset3
XGB stands for XGBoost
SVM stands for Support Vector Machine
DT stands for Decision Tree
RF stands for Random Forest
NB stands for Naïve Bayes

In Table 4 it shows the performance of different machine learning models across various train-test splits, ranging from an even 50-50 split to a more training-heavy 90-10 split. The results indicate that the amount of training data has an impact on model performance, but some algorithms are more affected than others.

XGBoost and Support Vector Machine (SVM) achieved perfect accuracy, precision, recall, and F1-score across all train-test splits, showing that these models are highly reliable regardless of the dataset size used for training. Decision Tree and Random Forest also performed exceptionally well, with only minor differences in accuracy. Their scores remained consistent across all splits, proving their robustness in detecting phishing URLs. However, there were slight fluctuations in recall values, particularly in lower training ratios.

Naïve Bayes had the lowest overall scores among the models, though its accuracy remained high, ranging from 98.5% to 98.8%. Unlike other models, Naïve Bayes showed small dips in recall, meaning it might struggle slightly with correctly identifying some phishing URLs when less training data is available. However, its precision remained strong, suggesting it is still reliable when it does classify a site as phishing.

Table 4 - All Train-Test Split Of Algorithm Models Performance

Algo	Accuracy (%)					Precision (%)					Recall (%)					F1-score (%)				
	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
XGB	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
SVM	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
DT	99.7	99.7	99.7	99.6	99.7	100	100	100	100	100	99.4	99.4	99.4	99.3	99.4	99.7	99.7	99.7	99.6	99.7
RF	99.6	99.7	99.7	99.6	99.7	100	100	100	100	100	99.3	99.4	99.4	99.3	99.4	99.6	99.7	99.7	99.6	99.7
NB	98.8	98.8	98.7	98.8	98.8	99.5	99.5	99.4	99.4	99.3	98.2	98.1	98.0	98.1	98.2	98.8	98.8	98.7	98.7	98.8

Table 4 shows all the results of Train-Test Split of Algorithm Models Performance Matrix using the following legend:

A represents 50%-50% Train-Test Splits
B represents 60%-40% Train-Test Splits
C represents 70%-30% Train-Test Splits
D represents 80%-20% Train-Test Splits
E represents 90%-10% Train-Test Splits
XGB stands for XGBoost
SVM stands for Support Vector Machine
DT stands for Decision Tree
RF stands for Random Forest
NB stands for Naïve Bayes

In Table 5 it provides insights into the training and prediction times of different machine learning models across various train-test splits. Training time refers to how long a model takes to learn from the data, while prediction time indicates how fast it classifies a new URL.

Naïve Bayes had the shortest training and prediction times, making it the most efficient model in terms of speed. Its training time was nearly negligible, especially in higher train-test splits, and its prediction time was also the lowest. This makes it a great option for scenarios where quick classification is needed.

Support Vector Machine (SVM) had moderate training and prediction times. Its training time varied slightly across splits but remained reasonable. However, its prediction time was slightly higher than Naïve Bayes, meaning it takes a bit longer to classify URLs, but the difference is not significant.

Random Forest showed fluctuations in training time, with higher values in some train-test splits. It had one of the longest training times in the 80-20 split, indicating that it requires more resources to train when more data is available. Its prediction time, however, remained relatively low, making it efficient once trained.

XGBoost required more training time than some other models but was still reasonable. Despite taking longer to train, its prediction time remained low, making it an excellent trade-off between performance and efficiency.

Decision Tree had varying training times, with some splits taking longer to train than others. However, its prediction time was quite low, similar to Naïve Bayes.

This means that while it may take some time to prepare, it classifies URLs quickly once trained.

Table 5 - All Train-Test Splits of Algorithms Models Training And Predicition Time

Algorithms	Training Time (s)					Prediction Time (s)				
	A	B	C	D	E	A	B	C	D	E
Naïve Bayes	0.0045	0.0052	0.0045	0.0003	0.000	0.0047	0.0045	0.002	0.000	0.000
Support Vector Machine	0.0381	0.0444	0.0229	0.0364	0.0353	0.0061	0.0069	0.0032	0.009	0.0043
Random Forest	0.015	0.4189	0.0372	0.5488	0.5049	0.003	0.0372	0.0278	0.0227	0.0104
XGBoost	0.1464	0.224	0.1612	0.1513	0.153	0.0025	0.004	0.002	0.0031	0.002
Decision Tree	0.3885	0.0099	0.0092	0.0157	0.0115	0.0418	0.000	0.0005	0.0031	0.000

Table 5 shows all the results of Train-Test Splits of Algorithm Models Training and Prediction Time using the following legend:

A represents 50%-50% Train-Test Splits
B represents 60%-40% Train-Test Splits
C represents 70%-30% Train-Test Splits
D represents 80%-20% Train-Test Splits
E represents 90%-10% Train-Test Splits

In Table 6 it shows how different hybrid models performed across various train-test splits. The hybrid models combine Naïve Bayes with other algorithms like Decision Tree, XGBoost, Random Forest, and Support Vector Machine (SVM). The results reveal that all hybrid models performed consistently well across all train-test splits, with only slight variations in accuracy, precision, recall, and F1-score.

The Naïve Bayes + Decision Tree (NB+DT) model had the highest accuracy, maintaining a steady 99.4% across all splits. It also showed strong precision, recall, and F1-score values, making it a solid choice. The Naïve Bayes + XGBoost (NB+XGB), Naïve Bayes + Random Forest (NB+RF), and Naïve Bayes + SVM (NB+SVM) models all had identical accuracy and precision values, staying at 98.8% in every case. Their recall and F1-score were also very similar, with slight variations but no significant drops.

These hybrid models demonstrated excellent performance, with NB+DT slightly outperforming the rest. This suggests that combining Naïve Bayes with Decision Tree may offer a slight advantage. However, the other combinations, such as NB+XGB, NB+RF, and NB+SVM, still provide strong and reliable results. The consistency across different train-test splits indicates that these models are stable and effective for phishing detection, regardless of the data distribution.

Table 6 - All Train-Test Split of Hybrid Models Performance Matrix

Hybrid Models	Accuracy (%)					Precision (%)					Recall (%)					F1-score (%)				
	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E	A	B	C	D	E
NB+ DT	99.4	99.4	99.5	99.4	99.4	99.5	99.5	99.4	99.4	99.3	99.3	99.4	99.4	99.3	99.4	99.4	99.4	99.4	99.4	99.4
NB+ XGB	98.8	98.8	98.8	98.8	98.8	99.5	99.5	99.4	99.4	99.3	98.2	98.1	98.1	98.1	98.2	98.8	98.8	98.8	98.7	98.8
NB+ RF	98.8	98.8	98.8	98.8	98.8	99.5	99.5	99.4	99.4	99.3	98.2	98.1	98.08	98.1	98.2	98.8	98.8	98.7	98.7	98.8
NB+ SVM	98.8	98.8	98.8	98.8	98.8	99.5	99.5	99.4	99.4	99.3	98.2	98.1	98.08	98.1	98.2	98.8	98.8	98.7	98.7	98.8

Table 6 shows all the results of the Train-Test Split of Hybrid Models Performance Matrix using the following legend:

A represents 50%-50% Train-Test Splits
B represents 60%-40% Train-Test Splits
C represents 70%-30% Train-Test Splits
D represents 80%-20% Train-Test Splits
E represents 90%-10% Train-Test Splits
NB+DT stands for Hybrid Model Naïve Bayes + Decision Tree
NB+XGB stands for Hybrid Model Naïve Bayes + XGBoost
NB+RF stands for Hybrid Model Naïve Bayes + Random Forest
NB+SVM stands for Hybrid Model Naïve Bayes + Support Vector Machine

In Table 7 it shows how long it takes for different hybrid models to train and make predictions across various train-test splits. The models combine Naïve Bayes with Decision Tree, XGBoost, Random Forest, and Support Vector Machine (SVM). Training time refers to how long the model takes to learn from the data, while prediction time is the time it takes to classify new data.

Among the models, Naïve Bayes + Decision Tree (NB+DT) had the fastest training time, with values ranging from 0.0117 to 0.0214 seconds. It also had one of the quickest prediction times, making it a good option for real-time applications. On the other hand, Naïve Bayes + XGBoost (NB+XGB) took slightly longer to train, ranging from 0.188 to 0.2825 seconds, but it's prediction time remained low, ensuring fast classification. Naïve Bayes + Random Forest (NB+RF) had the highest training time, reaching up to 0.4336 seconds, which could be due to the complexity of the Random Forest model. However, its prediction time was still competitive. Lastly, Naïve Bayes + SVM (NB+SVM) showed a balanced training and prediction time, though it was slightly higher than NB+DT.

Table VI - All Train-Test Splits Of Hybrid Models Training And Prediction Time

Hybrid Models	Training Time (s)					Prediction Time (s)				
	A	B	C	D	E	A	B	C	D	E
NB+ DT	0.0148	0.0117	0.017	0.0206	0.0214	0.0074	0.0077	0.0036	0.0019	0.0022
NB+ XGB	0.2325	0.2307	0.2682	0.188	0.2825	0.0101	0.0075	0.0062	0.003	0.0036
NB+ RF	0.3688	0.4084	0.39	0.4383	0.4336	0.0363	0.0075	0.03	0.0265	0.026
NB+ SVM	0.1169	0.1564	0.133	0.1278	0.1606	0.0187	0.0054	0.0102	0.0154	0.000

Table 7 shows all the results of Train-Test Splits of Hybrid Models Training and Prediction Time using the following legend:

A represents 50%-50% Train-Test Splits
B represents 60%-40% Train-Test Splits
C represents 70%-30% Train-Test Splits
D represents 80%-20% Train-Test Splits
E represents 90%-10% Train-Test Splits
NB+DT stands for Hybrid Model Naïve Bayes + Decision Tree
NB+XGB stands for Hybrid Model Naïve Bayes + XGBoost
NB+RF stands for Hybrid Model Naïve Bayes + Random Forest
NB+SVM stands for Hybrid Model Naïve Bayes + Support Vector Machine

V. SUMMARY, CONCLUSIONS, AND RECOMMENDATION

CONCLUSION

The phishing URL detection system developed in this study was designed to identify and classify phishing and legitimate websites with high accuracy and efficiency. The system was evaluated by comparing its predictions with actual phishing and legitimate URLs and analyzing its performance using standard Machine Learning Metrics.

To achieve its objectives, the system first collected datasets from publicly available sources such as the Phishtank archive, Alexa Repository, and GitHub Repository. These datasets were cleaned and preprocessed before being used to train and test different machine learning models. Five machine learning algorithms—XGBoost, Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF), and Decision Tree (DT)—were compared based on accuracy, precision, recall, and F1-score. The results showed that XGBoost and SVM achieved perfect classification accuracy, while Random Forest, Decision Tree, and the Hybrid NB+DT model also performed well with only slight variations in efficiency.

A key part of this study was combining Naïve Bayes with other Machine Learning Algorithms to form a hybrid model. The results indicated that the Hybrid Model of Naïve Bayes + Decision Tree approach maintained high accuracy while balancing computational efficiency and prediction time. Additionally, different train-test splits (50-50, 60-40, 70-30, 80-20, and 90-10) were examined to understand their effect on model performance. It was observed that increasing the training data improved accuracy, with the best results occurring at 80-20 and 90-10 splits.

The overall performance evaluation demonstrated that the system is highly functional, reliable, efficient, and user-friendly. These findings confirm that the system meets essential cybersecurity and machine learning standards for phishing detection. The Hybrid NB+DT model has proven to be a strong and viable approach for accurately classifying URLs while maintaining computational efficiency, making it a valuable tool for improving online security.

RECOMMENDATIONS

Based on the findings of this study, the following recommendations are proposed:

1. The choice of a model should be guided by the specific needs of the application. If accuracy is the top priority, XGBoost and SVM are the best choices. If efficiency is preferred, Decision Tree and Hybrid NB+DT offer a good balance. Meanwhile, Naïve Bayes is the best option when speed is the main concern.
2. Combining multiple algorithms is not always necessary for achieving high accuracy in phishing detection. Some models are already robust enough on their own. However, experimenting with model combinations remains a worthwhile exploration for further improvement.
3. For train-test splits, the 70%-30% or 80%-20% is recommended, as it ensures a well-balanced dataset for training and evaluation. Additionally, when acquiring datasets, it is best to prioritize newer ones, as they contain clearer and more relevant phishing features compared to older datasets. This will enhance the model's ability to detect modern phishing threats more effectively.
4. Future studies should explore the effect of feature selection techniques to optimize model efficiency further.

ACKNOWLEDGMENT

A lot of people have helped and willingly contributed their efforts to make this study a successful one, therefore, the researchers would like to convey their heartfelt gratitude and appreciation to the following:

To our panelist Instr. Loveson C. Gallo, Instr.

Alfred John G. Borreros, Prof. Divon G. Tamdang, Instr. James Ryan B. Ga, and Prof. Adrian J. Forca, College Dean, shared their expertise for the success of this study.

Dr. Lea P. Ymalay, Research Adviser, for her unconditional support, guidance, encouragement, and patience in forgoing the entire study.

To our parents, Mr. Noel Y. Quilacio and Mrs. Eleanor C. Quilacio, Mr. George Cuadra and Mrs. Maria Lea Cuadra, Mr. Ricardo G. Redaja and Mrs. Jepsy G. Redaja, Mr. Jose M. Marinog and Mrs. Ma.Cecilia P. Marinog, Mr.Rodolfo M. Gabaton and Mrs. Anita M. Gabaton for their moral and financial support throughout the study.

To our brothers, specially, Mr. Sean Noel C. Quilacio for providing a laptop to complete our research, Mr. Rev G. Redaja and Mr. John A. Alibugha for their encouragement and financial support also throughout the study.

Classmates and friends who encouraged, and endless support, extended to the researchers during the study.

Specifically, the project manager would like to thank his girlfriend. Jhaya May D. Balderama, for her constant support, love, and understanding throughout this journey.

Above all, to our Almighty God for his continued blessings and support, extending their help and motivation in pursuing the researcher's work.

All those who were not mentioned but helped complete this study, the researchers would like to say THANK YOU TO ALL OF YOU!

REFERENCES

[1] APWG | Phishing Activity Trends Reports. Retrieved March 28, 2025 from https://apwg.org/trendsreports/

[2] THE 17 GOALS | Sustainable Development. Retrieved March 30, 2025 from https://sdgs.un.org/goals

[3] Republic Act No. 10175 | Official Gazette of the Republic of the Philippines. Retrieved March 30, 2025 from https://www.officialgazette.gov.ph/2012/09/12/republic-act-no-10175/

[4] Phishing, ransomware attacks still top cyber threats in PH. Retrieved March 30, 2025 from https://business.inquirer.net/441921/phishing-ransomware-attacks-remain-top-cyber-threats-in-ph

[5] Cybersecurity Awareness: Definition, Importance & More | Spanning. Retrieved March 30, 2025 from https://www.spanning.com/blog/cybersecurity-awareness/

[6] What is a Data Breach and How to Prevent It? | Fortinet. Retrieved March 30, 2025 from https://www.fortinet.com/resources/cyberglossary/data-breach

[7] Feature Extraction Explained - MATLAB & Simulink. Retrieved March 30, 2025 from https://www.mathworks.com/discovery/feature-extraction.html

[8] What is Machine Learning? Guide, Definition and Examples. Retrieved March 30, 2025 from https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

[9] PHISHING Definition & Meaning - Merriam-Webster. Retrieved March 30, 2025 from https://www.merriam-webster.com/dictionary/phishing

[10] Phishing Detection - an overview | ScienceDirect Topics. Retrieved March 30, 2025 from https://www.sciencedirect.com/topics/computer-science/phishing-detection

[11] Train Test Split: What it Means and How to Use It | Built In. Retrieved March 30, 2025 from https://builtin.com/data-science/train-test-split

[12] Kamal Omari. 2023. Comparative Study of Machine Learning Algorithms for Phishing Website Detection. *International Journal of Advanced Computer Science and Applications* 14, 9 (2023), 417–425. https://doi.org/10.14569/IJACSA.2023.0140945

[13] Saleem Raja Abdul Samad, Sundarvadivazhagan Balasubaramanian, Amna Salim Al-Kaabi, Bhisham Sharma, Subrata Chowdhury, Abolfazl Mehbodniya, Julian L. Webber, and Ali Bostani. 2023. Analysis of the Performance Impact of Fine-Tuned Machine Learning Model for Phishing URL Detection. *Electronics 2023, Vol. 12, Page 1642* 12, 7 (March 2023), 1642. https://doi.org/10.3390/ELECTRONICS12071642

[14] Noura Fahad Almujahid, Mohd Anul Haq, and Mohammed Alshehri. 2024. Comparative evaluation of machine learning algorithms for phishing site detection. *PeerJ Comput Sci* 10, (June 2024), e2131. https://doi.org/10.7717/PEERJ-CS.2131/SUPP-22

[15] Meysam Vakili, Mohammad Ghamsari, and Masoumeh Rezaei. 2020. Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification. (January 2020). Retrieved March 30, 2025 from https://arxiv.org/abs/2001.09636v1

[16] Vahid Shahrivari, Mohammad Mahdi Darabi, and Mohammad Izadi. 2020. Phishing Detection Using Machine Learning Techniques. *arXiv preprint arXiv:2009.11116* (2020). Retrieved March 30, 2025 from https://arxiv.org/abs/2009.11116.

[17] PhishTank > See all suspected phish submissions. Retrieved March 30, 2025 from https://phishtank.org/phish_archive.php

[18] Alexa Defined | iContact. Retrieved March 30, 2025 from https://www.icontact.com/define/alexa/

[19] What Is GitHub? | Definition from TechTarget. Retrieved March 30, 2025 from https://www.techtarget.com/searchitoperations/definition/GitHub

[20] What is Python? Executive Summary | Python.org. Retrieved March 30, 2025 from https://www.python.org/doc/essays/blurb/

[21] IEEE Xplore Full-Text PDF: Retrieved March 30, 2025 from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8970564

[22] Schematic diagram of the XGBoost algorithm | Download Scientific Diagram. Retrieved March 30, 2025 from https://www.researchgate.net/figure/Schematic-diagram-of-the-XGBoost-algorithm_fig2_369416646

[23] V Kecman. 2005. Support Vector Machines – An Introduction BT - Support Vector Machines: Theory and Applications. *Support Vector Machines: Theory and Applications* 47, (2005), 1–47. Retrieved March 30, 2025 from http://dx.doi.org/10.1007/10984697_1

[24] Javatpoint. n.d. Support Vector Machine Algorithm. Retrieved from https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm (accessed July 19, 2023)

[25] Bernhard Schölkopf, Alex Smola Rsise, Alexander J Smola, Kernel Algorithms, B Schölkopf, and A J Smola. 2002. Support Vector Machines and Kernel Algorithms. (2002).

[26] Leo Breiman. 2001. Random forests. *Mach Learn* 45, 1 (October 2001), 5–32. https://doi.org/10.1023/A:1010933404324/METRICS

[27] Anuja Priyam, Rahul K. Gupta, A. Rathee, and S. Srivastava. 2013. Comparative Analysis of Decision Tree Classification Algorithms. (2013).

[28] Javatpoint. n.d. Machine Learning Decision Tree Classification Algorithm. Retrieved March 30, 2025 from https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[29] Pratap Chandra Sen, Mahimarnab Hajra, and Mitadru Ghosh. 2020. Supervised Classification Algorithms in Machine Learning: A Survey and Review. *Advances in Intelligent Systems and Computing* 937, (2020), 99–111. https://doi.org/10.1007/978-981-13-7403-6_11

[30] Kevin P.. Murphy. 2012. Machine learning : a probabilistic perspective. (2012), 1067.

[31] Peter A. Flach and Nicolas Lachiche. 2004. Naive Bayesian classification of structured data. *Mach Learn* 57, 3 (December 2004), 233–269. https://doi.org/10.1023/B:MACH.0000039778.69032.AB/METRICS

[32] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. https://doi.org/10.1145/2939672.2939785

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

[34] Tiago A. Almeida, Jurandy Almeida, and Akebo Yamakami. 2011. Spam filtering: How the dimensionality reduction affects the accuracy of Naive Bayes classifiers. *Journal of Internet Services and Applications* 1, 3 (February 2011), 183–200. https://doi.org/10.1007/S13174-010-0014-7/METRICS

[35] Phishing-Website-Detection-by-Machine-Learning-Techniques/Phishing Website Detection by Machine Learning Techniques Presentation.pdf at master · shreyagopal/Phishing-Website-Detection-by-Machine-Learning-Techniques · GitHub. Retrieved March 30, 2025 from https://github.com/shreyagopal/Phishing-Website-Detection-by-Machine-Learning-Techniques/blob/master/Phishing%20Website%20Detection%20by%20Machine%20Learning%20Techniques%20Presentation.pd

[36] PhiUSIIL Phishing URL (Website) - UCI Machine Learning Repository. Retrieved March 30, 2025 from https://archive.ics.uci.edu/dataset/967/phiusiil+ph

ishing+url+dataset