IJMIC Vol. 1, Issue 2, 26-37

Research Article

Enhancing Phishing Detection with Advanced Ensemble Learning Techniques

Nur Syakirin Mohd Shahiran, Mohammed Ahmed

DaTA Research Group, Faculty of Computing and Informatics, Universiti Malaysia Sabah, Malaysia
*Corresponding author: Mohammed Ahmed, mohammed.ahmed@ums.edu.my

Received: [April 16, 2025]
Accepted: [July 10, 2025]
Published: [July 31, 2025]

Abstract - Phishing attacks contribute to over 90% of data breaches, posing a severe cybersecurity threat by tricking users into divulging sensitive information. Traditional detection methods, such as blacklists and heuristic-based approaches, are often ineffective against new phishing websites due to their rapidly evolving nature. This study introduces an advanced phishing detection model that leverages ensemble learning techniques to improve accuracy, robustness, and adaptability. The model integrates Decision Tree, Support Vector Machine (SVM), and k-Nearest Neighbours (kNN) as base classifiers, combined through a stacking ensemble approach, with Logistic Regression serving as the meta-classifier. Feature selection is performed using Random Forest, selecting the most impactful attributes based on importance scores greater than 0.01. Principal Component Analysis (PCA) is applied to reduce dimensionality while retaining 95% of the variance, minimizing information loss. Hyperparameter optimization is achieved through Grid Search. The dataset was sourced from an open-access phishing detection repository and consists of 11,430 URLs, with 60% classified as phishing and 40% as legitimate. It includes 87 features that are categorized into URL structure, webpage content, and external service queries. The model's performance is evaluated using accuracy, precision, recall, and F1-score across various test sizes (10%, 20%, 30%, and 40%). Experimental results demonstrate that the stacking ensemble model achieves a peak accuracy of 97.64% with PCA (95%) and feature selection (importance score > 0.01) at a 10% test size, significantly outperforming traditional methods. Performance comparisons across different test sizes highlight the positive impact of feature selection and PCA on phishing detection. Statistical validation through t-tests ($p < 0.05$) further confirms the model's reliability, indicating substantial improvements over baseline methods. This study showcases the potential of ensemble learning and feature optimization in enhancing phishing detection, offering a robust solution for practical cybersecurity applications.

Keywords: Phishing detection, ensemble learning, feature selection, principal component analysis, stacking model.

1. INTRODUCTION

The Internet is a cornerstone of global connectivity and economic activity, yet it has also become a fertile ground for cybercriminals. Phishing attacks, in particular, have emerged as one of the most pervasive cybersecurity threats. Phishing is a deceptive cyberattack method that manipulates individuals into disclosing sensitive information, and it accounts for over 90% of data breaches, leading to losses exceeding $10 billion annually. Attackers employ various tactics, such as fraudulent emails and counterfeit websites, to exploit user trust and bypass traditional security measures.

Traditional phishing detection methods, such as blacklists and heuristic-based approaches, are becoming increasingly ineffective against modern threats. Blacklists depend on known malicious URLs, leaving systems vulnerable to zero-day attacks, while heuristic methods rely on static rules that struggle to adapt to the evolving nature of phishing techniques. Consequently, there is a growing demand for advanced detection systems capable of identifying and mitigating emerging threats in real-time.

Ensemble learning, a machine learning approach that integrates multiple models to enhance predictive accuracy, shows promise in phishing detection. Its ability to handle imbalanced datasets, reduce overfitting, and adapt to new attack patterns makes it particularly suitable for this domain. However, implementing ensemble learning for phishing detection comes with challenges, such as model selection complexity, the computational cost of hyperparameter tuning, and the risk of overfitting, which can undermine the system's generalizability to new threats.

This study explores the application of ensemble learning for phishing detection through a stacking approach combining Decision Tree, Support Vector Machine (SVM), and kNN classifiers. The model integrates advanced feature selection and dimensionality reduction techniques to improve accuracy, adaptability, and robustness. By addressing the limitations of traditional methods and leveraging the strengths of ensemble learning, this research aims to contribute to the development of more secure and resilient phishing detection systems, ultimately enhancing global cybersecurity.

2. LITERATURE REVIEW

a) Ensemble Learning: A Technical Overview

Ensemble learning is a machine learning technique designed to improve model performance by combining predictions from multiple models. Instead of relying on a single model, ensemble methods aggregate outputs from different models to reduce errors and improve accuracy. The primary types of ensemble learning include bagging, boosting, and stacking.

(i) Bagging (Bootstrap Aggregation)

Bagging enhances model stability and accuracy by generating multiple training datasets through bootstrap sampling—randomly selecting data points with replacement. Individual models are trained on these datasets, and their predictions are aggregated. Random Forest, a common implementation, constructs multiple decision trees and combines their predictions. Bagging reduces variance and helps mitigate overfitting, making it suitable for complex datasets.

(ii) Boosting

Boosting is an ensemble technique that combines multiple weak learners to build a strong predictive model. Algorithms like AdaBoost and Gradient Boosting train weak learners sequentially, with each iteration correcting errors made by the previous model. Boosting excels at addressing imbalanced datasets and enhancing classification accuracy.

(iii) Stacking

Stacking, or stacked generalization, combines classifiers built with different algorithms on the same dataset. The process involves two stages: first, base classifiers are trained on the original dataset, and second, a meta-classifier learns to combine their outputs for final predictions. Stacking leverages the strengths of diverse models, improving predictive performance and robustness.

b) Advantages of Ensemble Learning

Ensemble methods improve the accuracy and stability of machine learning models by integrating multiple perspectives. Techniques such as voting or weighted averaging mitigate the weaknesses of

individual models, resulting in more reliable predictions. Ensemble learning also enhances robustness by aggregating predictions from diverse models, increasing adaptability and reliability. This approach reduces the impact of errors or outliers in individual models, ultimately improving overall performance. Additionally, ensemble learning helps reduce overfitting by training each model on random subsets of data. Bagging introduces variability, while boosting assigns higher weights to difficult cases, enabling the construction of robust models capable of capturing intricate data patterns.

c) Current Trends in Phishing Detection

Recent advancements in phishing detection methods incorporate machine learning, list-based detection, and heuristic approaches to improve detection accuracy and response times. Research has explored a variety of methodologies, each offering its own set of strengths and challenges.

(i) Machine Learning Techniques

Machine learning plays a crucial role in classifying and detecting phishing-related features, enabling automated fraud detection. For example, [8] applied a Support Vector Machine (SVM) to differentiate between phishing and legitimate websites, achieving an accuracy of 95.66%. However, this study faced limitations due to a small dataset and reliance on a single classifier. Similarly, [9] evaluated multiple classifiers, including Decision Trees, Logistic Regression, Naïve Bayes, and Random Forest, with Random Forest achieving an accuracy of 83.0%. This lower accuracy highlights the need for improved feature selection and model tuning. Additionally, [10] utilized Random Forest alongside heuristic approaches, reaching 95% accuracy. Despite its success, the study was constrained by a small feature set, underlining the need for larger datasets to improve model adaptability.

(ii) List-Based Detection

List-based detection compares web pages based on content similarity, such as text, CSS, images, and other visual elements. For example, [11] combined blacklist-based, visual similarity, heuristic, and machine learning techniques, achieving 99.33% accuracy with the PART algorithm. [12] employed list-based, visual similarity, and machine learning techniques, achieving 97.00% accuracy but facing challenges due to reliance on third-party features. [13] utilized white-list-based and visual similarity techniques, attaining 96.17% accuracy with a small dataset of 200 websites. Finally, [14] integrated list-based, visual similarity, heuristic, and machine learning techniques, achieving 98.72% accuracy but encountering challenges related to dataset size and computational complexity.

3. METHODOLOGY

The dataset used in this study is extracted from [15], containing 11,430 URLs with 87 extracted features categorized into three classes: 56 features derived from the structure and syntax of URLs, 24 features extracted from the content of corresponding web pages, and 7 features obtained through external service queries. The dataset is balanced, comprising 50% phishing and 50% legitimate URLs, ensuring unbiased model evaluation. The dataset is provided in a comma-separated values (CSV) format, where the extracted feature values are directly utilized as input for classification models. Table 1 presents several attributes from the dataset along with their descriptions.

Table 1: A Subset of Attributes Extracted from The Dataset with Their Descriptions

Attribute	Description
url	The full URL string representing an individual web page.
length_url	The total number of characters in the URL.
length_hostname	The total number of characters in the hostname.
ip	Binary indicator (0 or 1) for whether the URL contains an IP address.
nb_dots, nb_hyphens, nb_at, nb_qm, etc.	Count of specific characters or symbols within the URL.
http_in_path	Binary indicator showing the presence of "http" in the path.
https_token	Binary indicator of "https" appearing in the tokenized URL.
ratio_digits_url, ratio_digits_host	The proportion of numerical digits within the URL and hostname, respectively.
port	Indicates the presence of a port number in the URL.

a) Machine Learning Algorithms

(i) Decision Tree

A Decision Tree classifier was employed due to its interpretability and capability to handle both numerical and categorical data. The tree recursively partitions the dataset based on information gain or Gini impurity, aiming to create homogeneous subsets. However, Decision Trees are prone to overfitting, which was mitigated through pruning and depth restriction.

(ii) SVM (Support Vector Machine)

SVM was utilized to find an optimal hyperplane that maximally separates phishing and legitimate URLs. Kernel functions, including linear and radial basis function (RBF) kernels, were explored to capture complex decision boundaries. The support vectors play a crucial role in maintaining robustness against outliers.

(iii) k-Nearest Neighbours (KNN)

KNN, a non-parametric algorithm, classifies URLs based on their proximity to K nearest training instances. The Euclidean distance metric was used to determine similarity between instances. KNN was chosen for its simplicity and effectiveness in handling complex decision boundaries.

b) Feature Importance Score Using Random Forest

Feature importance analysis was conducted using the Random Forest algorithm to determine the most influential features for phishing website detection. The importance scores were sorted and only features with a significance level above 0.00 and 0.01 were retained. This selection process resulted in a reduced dataset with the most relevant predictors. A bar plot was used to visualize these top features, providing insights into their significance in the model's decision-making process. Figure 1 displays features with importance scores greater than 0.00, while Figure 2 shows feature with importance scores greater than 0.01.

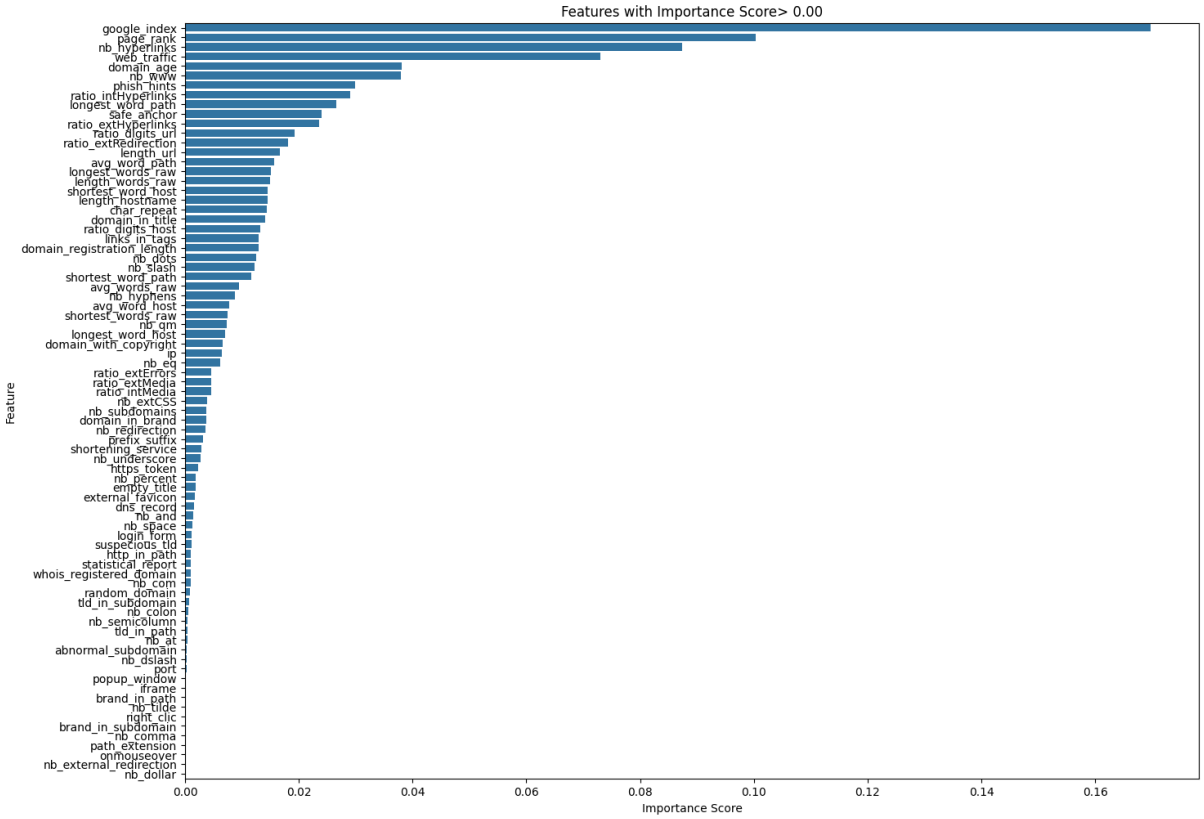

Figure 1: Feature with Importance Score Greater Than 0.00

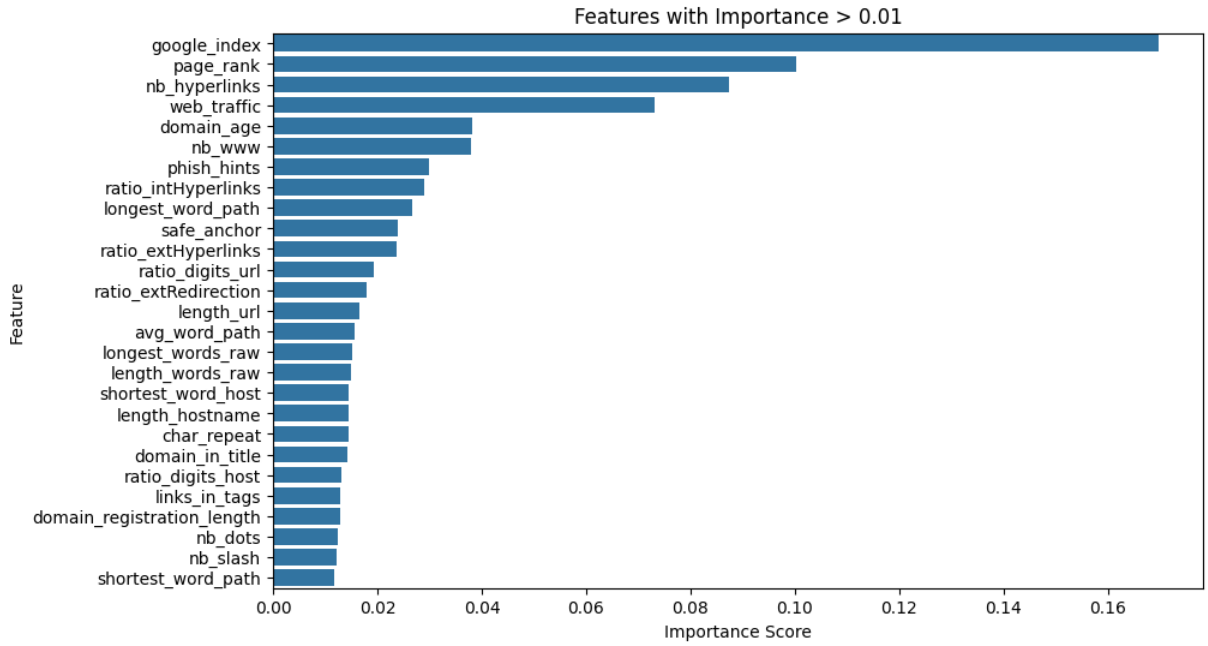

Figure 2: Feature selection with Importance Score Greater Than 0.01

c) Feature Standardization and Dimensionality Reduction

To enhance the quality of input data, the selected features were standardized using the StandardScaler. Standardization ensured that all features had a mean of 0 and a standard deviation of 1, which is crucial for maintaining consistency in feature scaling.

Subsequently, Principal Component Analysis (PCA) was applied for dimensionality reduction. PCA helps retain the most significant variance in the dataset while eliminating less important features. In this study, PCA was performed with a variance retention threshold of 95%, ensuring that the transformed dataset retained most of the original data's variance while reducing the number of dimensions. This step aimed to improve computational efficiency and enhance the performance of the machine learning models.

d) Hyperparameter Tuning Using Grid Search

Grid Search was implemented to optimize the hyperparameters of Decision Tree, SVM, and KNN classifiers. A predefined set of hyperparameter values was systematically evaluated to identify the optimal configuration that maximized classification performance. This exhaustive search method ensured optimal selection of model parameters to enhance generalization. Table 2, Table 3, and Table 4 display the hyperparameters and their respective values used in hyperparameter tuning.

Table 2: The Decision Tree Hyperparameter Tuning

Hyperparameter	Values
Max depth	5,10,15,20, None
Min samples split	2,5,10
Criterion	Gini, entropy

Table 3: The SVM Hyperparameter Tuning

Hyperparameter	Values
C	0.1,1,10
Kernel	Linear, Radial Basis Function (RBF), Poly
Gamma	Scale, auto

Table 4: The kNN Hyperparameter Tuning

Hyperparameter	Values
N neighbours	3, 5, 7, 9
Weights	Uniform, Distance
Metric	Euclidean, Manhattan

e) Evaluation Metrics

The selected model was trained on the training subset and evaluated on the testing subset. Predictions were made for the testing data, and several performance metrics were calculated for each test split. These metrics included accuracy, which measures the proportion of correct predictions made by the model; precision, which represents the ratio of true positive predictions to all positive predictions; recall,

which is the ratio of true positives to the total actual positives; and the F1 score, which provides a balanced evaluation by calculating the harmonic mean of precision and recall.

In addition to these metrics, a t-test was conducted to assess the statistical significance of differences in these metrics across different test splits, ensuring a comprehensive analysis of the model's performance.

4. RESULTS & DISCUSSIONS

a) Experiment 1: Using All Features

This experiment evaluates the performance of Decision Tree, Support Vector Machine (SVM), kNN, and Stacking Classifier across various test sizes (10%, 20%, 30%, and 40%) with an importance score greater than 0.00. Additionally, Principal Component Analysis (PCA) with 95% variance retention was applied to reduce dimensionality while preserving data variability. The Decision Tree performed well at smaller test sizes, achieving 90.81% accuracy at 10%, though accuracy declined slightly as the test size increased, reaching 89.26% at 40%. Precision and recall followed a similar pattern, indicating potential overfitting to smaller training sets. SVM demonstrated high and stable accuracy across all test sizes, reaching 97.11% at 10% and decreasing slightly to 95.80% at 40%, with consistently high precision, recall, and F1 scores, reflecting its strong generalization ability. kNN showed stable accuracy above 94% across all test sizes, peaking at 94.75% at 20%, with recall improving as the test size grew, suggesting the model effectively adapts to increasing data. The Stacking Classifier achieved the highest accuracy among all models, with 97.38% at 10% and a minor drop to 95.84% at 40%, demonstrating strong generalization and superior performance, reinforcing its effectiveness over individual models.

b) Experiment 2: Feature Importance Score > 0.00

In this experiment, machine learning models were evaluated using only features with importance scores greater than 0.00. PCA with 95% variance retention was applied to reduce dimensionality while preserving variability. This experiment aimed to assess how filtering features based on importance scores influences model performance across different test sizes. The Decision Tree performed well with smaller test sizes, achieving 91.34% accuracy at 10%. However, its accuracy declined slightly as the test size increased, reaching 89.34% at 40%, with precision and recall following a similar trend, and recall peaking at a 30% test size. SVM maintained high and stable accuracy across all test sizes, achieving 97.38% accuracy at 10% and only a minor drop to 95.93% at 40%, demonstrating strong generalization ability and a balanced F1 score. kNN showed consistent performance, with slight recall improvements as test size increased. Accuracy remained above 94% across all splits, peaking at 94.51% at 30%, while recall improved with larger test sizes before slightly decreasing at 40%. The Stacking Classifier achieved the highest accuracy among all models, with 97.64% at 10% and a minor decline to 96.22% at 40%. The model demonstrated robust generalization and well-balanced performance, reaffirming its superiority over individual models.

c) Experiment 3: Feature Importance Score > 0.01

This study evaluates the performance of Decision Tree, SVM, kNN, and Stacking Classifier models using only features with importance scores greater than 0.01 across different test sizes. The Decision Tree showed stable performance, with accuracy ranging from 91.12% to 92.04% across test sizes. Slight fluctuations in precision and recall indicated consistent generalization. SVM achieved the highest accuracy, reaching 97.20% at 10%, but it slightly declined to 95.56% at 40%, demonstrating strong generalization across different test sizes. The kNN model exhibited stable performance with minor variations, maintaining an accuracy of 95.63% at 10% and 94.62% at 40%. Recall fluctuated slightly but remained strong throughout. The Stacking Classifier outperformed all models with high and balanced metrics, starting at 96.94% accuracy at 10% and decreasing to 95.54% at 40%, maintaining robustness and generalization across test sizes.

d) Comparison of Model Performance Using the Same Dataset

Table 5 shows that the proposed study outperforms previous research by integrating stacking classifiers, PCA-based feature selection, and feature importance scoring, leading to superior accuracy and better generalization in phishing detection. This demonstrates the advantage of advanced ensemble learning and dimensionality reduction techniques in improving detection performance.

Table 5: Comparison of Model Performance Using the Same Dataset

Study/Model	Accuracy (%)	Key Features & Approach	Notes
This study	97.64	Stacking Classifier with PCA 95% and Features Importance Score >0.01, Test Size 10%	Results from ensemble learning with feature selection using PCA
Hannanousse, Abdelhakim & Yahiouche, Salima [16]	95.63	Hybrid feature set with stacking ensemble	Focused on benchmark datasets for ML-based phishing detection
Abdulhanan Rafique [17]	89.66	Combine Classifier (RF +XGB)	Uses Random Forest (RF) and XGBoost (XGB) models for web page phishing detection
Ahmed Islam [18]	95.50	Random Forest with feature scaling and train-test split. Test Size :25%	Focus on URL-based features for phishing detection

e) T-Test Results

The results of the t-tests comparing the Stacking Classifier with Decision Tree, SVM, and kNN using different feature selection criteria provide valuable insights into model performance.

(i) T-Test between Stacking Classifier and Other Individual Models Using All Features

Table 6 shows the comparison between the Stacking Classifier and other models. There is a highly significant difference between Stacking and Decision Tree ($t = 45.37$, $p = 0.000024$), confirming that the Stacking Classifier outperforms the Decision Tree. However, when comparing Stacking with SVM, no statistically significant difference was found ($t = 2.39$, $p = 0.096563$), suggesting that both models perform similarly. In contrast, the comparison between Stacking and kNN shows a significant difference ($t = 6.71$, $p = 0.006757$), indicating that Stacking performs better than kNN.

Table 6: Result of T-Test Between Stacking Classifier and Other Individual Models Using All Features

Model Comparison	T- statistic	P-value
Stacking Classifier vs. Decision Tree	45.366705	0.000024
Stacking Classifier vs. SVM	2.392051	0.096563
Stacking Classifier vs. kNN	6.709469	0.006757

(ii) T-Test between stacking classifier and other individual model using features with importance score > 0.00

Table 7 shows the comparison between the Stacking Classifier and other models, highlighting Stacking's superior performance in several cases. There is a highly significant difference between Stacking and Decision Tree ($t = 41.40$, $p = 0.000031$), reaffirming Stacking's advantage. When compared to SVM, a statistically significant difference was observed ($t = 5.57$, $p = 0.011433$), indicating that Stacking performs better, particularly when focusing on important features. Lastly, the comparison between Stacking and kNN shows a significant difference ($t = 8.31$, $p = 0.003649$), confirming Stacking's advantage over kNN.

Table 7: Result of T-Test Between Stacking Classifier and Other Individual Model Using Features with Importance Score > 0.00

Model Comparison	T- statistic	P-value
Stacking Classifier vs. Decision Tree	41.396553	0.000031
Stacking Classifier vs. SVM	5.567764	0.011433
Stacking Classifier vs. kNN	8.311789	0.003649

(iii) T-Test between stacking classifier and other individual model using features with importance score > 0.01

Table 8 shows the comparison between the Stacking Classifier and other models, revealing key insights. There is a statistically significant difference between Stacking and Decision Tree ($t = 20.72$, $p = 0.000246$), although the performance gap has narrowed compared to previous comparisons. When comparing Stacking to SVM, no significant difference was found ($t = -0.95$, $p = 0.413174$), suggesting that both models perform similarly when focusing on the most important features. On the other hand, the comparison between Stacking and kNN shows a significant difference ($t = 10.44$, $p = 0.001876$), confirming Stacking's superiority over kNN, even when using high-importance features.

Table 8: Result of T-Test Between Stacking Classifier and Other Individual Model Using Features with Importance Score > 0.01

Model Comparison	T- statistic	P-value
Stacking Classifier vs. Decision Tree	20.722000	0.000246
Stacking Classifier vs. SVM	-0.947756	0.413174
Stacking Classifier vs. kNN	10.439567	0.001876

This study made comparisons between models, Decision Trees, Support Vector Machine (SVM), and kNN. These models were chosen because of their fundamental roles in classification tasks. The Decision Tree classifier is widely recognised for its simplicity and interpretability, allowing for clear decision rules. The SVM model was selected for its robustness in high-dimensional spaces, particularly when dealing with non-linear decision boundaries. The kNN classifier, known for its simplicity and effectiveness, was included to contrast the more complex ensemble methods. These baseline models served as a benchmark to evaluate the performance improvements achieved by the stacking ensemble approach used in this study.

5. CONCLUSIONS

This study underscores the significant potential of ensemble learning methods in enhancing phishing website detection. By integrating multiple classifiers, the ensemble approach effectively combines the strengths of individual models while minimizing their weaknesses. This results in improved accuracy, robustness, and adaptability, which are essential when dealing with the constantly evolving tactics used by cybercriminals in phishing attacks. In this study, the stacked ensemble model demonstrated the highest performance among the various ensemble techniques explored, surpassing traditional machine learning models such as decision trees, support vector machines (SVM), and kNN.

The stacked ensemble model's ability to capture complex, non-linear relationships in the data and to identify subtle differences between legitimate and phishing websites contributed to its superior performance. Unlike single classifiers, which may struggle to generalize effectively, especially when dealing with imbalanced or noisy data, the stacking model leveraged the complementary strengths of its base classifiers. This enabled it to make more informed decisions, reducing the likelihood of misclassifications, particularly for hard-to-detect phishing websites.

Additionally, the study highlighted the importance of feature selection and dimensionality reduction techniques, such as Principal Component Analysis (PCA), in improving the model's efficiency and accuracy. By reducing the dimensionality of the feature space while retaining critical information, these techniques helped mitigate the curse of dimensionality, enhancing learning efficiency without compromising performance.

One of the key findings of this research is the robustness of ensemble learning methods in real-world phishing detection. The stacked ensemble model demonstrated consistent performance across different test sizes, emphasizing its reliability in various scenarios. The model's high generalization ability makes it suitable for real-time phishing detection systems, where new, unseen phishing websites are regularly encountered. This makes the stacked ensemble model a practical and scalable solution for enhancing cybersecurity measures.

These findings have important implications for improving cybersecurity defences against phishing attacks. As phishing remains one of the most prevalent and damaging online threats, adopting ensemble

learning approaches can help organizations better safeguard sensitive data and protect users from fraudulent websites. Moreover, the use of advanced feature engineering and model optimization techniques, such as hyperparameter tuning and feature importance analysis, further enhances the effectiveness of these detection systems.

Despite the impressive performance of the stacking ensemble model, real-world deployment faces several challenges. One such challenge is the adaptability of the model to zero-day phishing attacks, which involve previously unseen attack patterns. While the model demonstrates strong generalization capabilities, continuous updates and retraining with fresh data are essential to ensure its effectiveness against new, evolving phishing techniques. Moreover, the model's interpretability remains a critical concern for practical cybersecurity systems. The Decision Tree classifier in the ensemble contributes to model transparency, allowing practitioners to trace decision-making paths. However, more complex models like SVM and kNN may require additional interpretability tools such as LIME (Local Interpretable Model-agnostic Explanations) to ensure the trustworthiness of the model's decisions. Furthermore, real-time deployment may require adjustments in computational efficiency to handle large-scale, dynamic data environments without compromising performance. Addressing these challenges is essential for integrating this model into production environments, where detection speed and adaptability to new threats are paramount.

In conclusion, this study demonstrates that ensemble learning, particularly stacking techniques, offers a promising avenue for advancing phishing website detection. The ability of these models to handle complex data patterns and adapt to emerging threats positions them as a critical tool in the ongoing battle against online fraud. Future research could further refine these models by incorporating additional feature sets, exploring other ensemble techniques, and testing the models on larger, more diverse datasets to ensure continued improvements in detection accuracy and real-world applicability.

REFERENCES

[1] M. A. Ganaie, M. Hu, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105151, 2021.

[2] H. Nguyen, D. T. Bui, A. Khosravi, K. Chapi, and B. Pradhan, "Enhancing prediction performance of landslide susceptibility models using hybrid machine learning approaches," *Applied Sciences*, vol. 8, no. 7, p. 1046, 2018.

[3] R. Dey and R. Mathur, "Ensemble learning method using stacking with base learner: A comparison," in *Proceedings of the International Conference on Recent Innovations in Computing,* 2023, pp. 167-176. doi: 10.1007/978-981-99-3878-0_14.

[4] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the International Workshop on Multiple Classifier Systems,* 2000, pp. 1–15. doi: 10.1007/3-540-45014-9_1.

[5] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1, pp. 1–39, 2010. doi: 10.1007/s10462-009-9124-7.

[6] R. Polikar, "Ensemble learning," *Scholarpedia*, vol. 4, no. 1, p. 2776, 2009. doi: 10.4249/scholarpedia.2776.

[7] A. Zhou, K. Xie, S. Wang, and X. Wang, "A survey of ensemble learning: Concepts, applications, and future directions," *ACM Computing Survey*, vol. 54, no. 2, pp. 1–36, 2021. doi: 10.1145/3447556.

[8] J. Rashid, T. Mahmood, M. W. Nisar, and T. Nazir, "Phishing detection using machine learning technique," in *Proceedings of the 1st International Conference on Smart Systems and Emerging Technologies,* 2020.

[9] B. Geyik, K. Erensoy, and E. Kocyigit, "Detection of phishing websites from URLs by using classification techniques on WEKA," in *Proceedings of the 6th International Conference on Inventive Computation Technologies (ICICT)*, 2021, pp. 120–125. doi: 10.1109/ICICT50816.2021.9358642.

[10] S. Parekh, D. Parikh, S. Kotak, and S. Sankhe, "A new method for detection of phishing websites: URL detection," in *Proceedings of the International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 949–952. doi: 10.1109/ICICCT.2018.8473085.

[11] P. A. Barraclough, G. Fehringer, and J. Woodward, "Intelligent cyber-phishing detection for online," *Computers & Security*, vol. 104, p. 102123, 2021. doi: 10.1016/j.cose.2020.102123.

[12] S. Maroofi, M. Korczynski, C. Hesselman, B. Ampeau, and A. Duda, "COMAR: Classification of compromised versus maliciously registered domains," in *Proceedings of the 5th IEEE European Symposium on Security and Privacy (Euro S&P)*, 2020, pp. 607–623. doi: 10.1109/EuroSP48549.2020.00045.

[13] N. A. Azeez, S. Misra, I. A. Margaret, L. Fernandez-Sanz, and S. M. Abdulhamid, "Adopting automated whitelist approach for detecting phishing attacks," *Computers & Security*, vol. 108, p. 102328, 2021. doi: 10.1016/j.cose.2021.102328.

[14] R. S. Rao and A. R. Pais, "Two-level filtering mechanism to detect phishing sites using lightweight visual similarity approach," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 9, pp. 3853–3872, 2020. doi: 10.1007/s12652-019-01637-z.

[15] "Phishing Website Dataset," Mendeley Data, 2021. [Online]. Available: https://data.mendeley.com/datasets/c2gw7fy2j4/3

[16] A. Hannousse and S. Yahiouche, "Towards benchmark datasets for machine learning based website phishing detection: An experimental study," *Engineering Applications of Artificial Intelligence*, vol. 104, Sep. 2021, doi: 10.1016/j.engappai.2021.104347.

[17] "Web page Phishing Detection Dataset." Accessed: Aug. 01, 2025. [Online]. Available: https://www.kaggle.com/datasets/shashwatwork/web-page-phishing-detection-dataset/code

[18] "Phishing No More: Creating a URL Classifier with Machine Learning | by Ahmed Islam | Python's Gurus | Medium." Accessed: Aug. 01, 2025. [Online]. Available: https://medium.com/pythons-gurus/phishing-no-more-creating-a-url-classifier-with-machine-learning-a8a1b3e90a0f