An Explainable Ensemble Machine Learning Framework for Phishing Website Detection with Robustness and Deployment Readiness Evaluation

Ande Amshutha[1], Arikitemula Pavani[2], R. Anaghskanda Bharadwaj[3], Hema MS[4]

1–3Student, Department of Computer Science and Engineering,

RV Institute of Technology and Management, Bangalore – 560076, Karnataka, India

4Head of Department, Department of Computer Science and Engineering,

RV Institute of Technology and Management, Bangalore – 560076, Karnataka, India

How to Cite this Article:

Bharadwaj, R. A., Pavani, A. & Amshutha, A. (2026). An Explainable Ensemble Machine Learning Framework for Phishing Website Detection with Robustness and Deployment Readiness Evaluation. International Journal of Creative and Open Research in Engineering and Management, 02(05). https://doi.org/10.55041/ijcope.v2i4.970

https://doi.org/10.55041/ijcope.v2i4.970

1.ABSTRACT

Phishing attacks are a serious cybersecurity trouble just one successful attack can lead to major fiscal losses and exposure of sensitive information, and although machine literacy models have come relatively effective at detecting phishing websites, numerous studies overlook important aspects similar as explainability, robustness to noisy data, and real-world usability; this study addresses those gaps by erecting on the work of Awasthi and Goel(2022) and using the UCI Phishing Websites dataset, which contains over 11,000 samples and 30 features, where six bracket algorithms Logistic Retrogression, Support Vector Machines, Decision Trees, K- Nearest Neighbors, Random Forest, and XGBoost are estimated with hyperparameter tuning for ensemble models and10-foldcross-validation for dependable performance, and among all models, XGBoost performs the stylish, achieving 96.70 delicacy and a 97.06 F1- score; the main benefactions of the study are threefold first, explainability, where SHAP(SHapley Additive Explanations) identifies anchor URL, HTTPS operation, and website business as the most important features in detecting phishing websites; second, robustness, where the model is tested with noisy data up to 20 and XGBoost still maintains a strong delicacy of 87.38; and third, deployment, where the XGBoost model proves to be effective with an conclusion time of 13.11 milliseconds and a compact size of just 501 KB; overall, the study demonstrates that it's possible to achieve high delicacy while also icing interpretability, robustness, and practical usability, showing that featherlight ensemble models, particularly XGBoost, give an effective and deployable result for phishing discovery in real- world scripts.

Index Terms: Phishing Detection, Machine Learning, Ensemble Learning, XGBoost, Explainable Artificial Intelligence (XAI), SHAP, Robustness Analysis, Deployment Analysis, Cybersecurity, Classification.

2.INTRODUCTION

Phishing attacks have come one of the most grim pitfalls in cybersecurity, where culprits disguise themselves as trusted sources and use fake websites to trick druggies into giving up sensitive data similar as login credentials and banking information, leading to serious consequences like fiscal loss and large- scale data breaches; according to theAnti-Phishing Working Group(APWG), these attacks are n't only adding in frequence but also getting more sophisticated, pressing the critical need for automated discovery systems that are both largely accurate and able of handling large volumes of data without nonstop mortal intervention; earlier discovery styles reckoned on blacklists and manually drafted rules, which were originally effective but fail to keep up with the constantly evolving nature of phishing URLs, leading to a shift toward machine literacy approaches, where ways like Random Forest and XGBoost have shown strong performance in relating retired patterns and detecting fraudulent exertion in complex datasets; still, despite emotional delicacy reported in exploration, there remains a clear gap between academic results and real- world deployment, as utmost studies concentrate primarily on perfecting bracket criteria while overlooking critical aspects similar as interpretability, since security brigades need clear explanations rather than opaque " black box " opinions, robustness, as models are frequently tested only on clean datasets rather than noisy real- world data, and effectiveness, including conclusion speed and memory operation, which are essential for deployment in surroundings like network gateways, cybersurfer extensions, and edge bias; to address these limitations, this paper proposes a comprehensive frame that emphasizes not only delicacy and scalability but also explainability and robustness, making it suitable for real-world deployment, by erecting on being methodologies, optimizing the ensemble channel, and assessing performance using the UCI Phishing Websites dataset, and it introduces three crucial benefactions first, the integration of SHapley Additive exPlanations(SHAP) to give both global and original interpretability, pressing important features similar as AnchorURL and HTTPS; second, robustness evaluation through stress testing with synthetic point disquiet to pretend real-world data noise; and third, a detailed deployment analysis measuring factors like speed, memory operation, and model size to determine the most effective armature for practical operations, eventually demonstrating that delicacy, translucency, and effectiveness can be achieved together, making the frame a strong foundation for coming- generation real-time phishing discovery systems.

III. METHODOLOGY

The proposed frame follows a comprehensive multi-stage channel designed to insure high prophetic performance, interpretability, and functional readiness. As illustrated in the system architecture (Fig. 1)

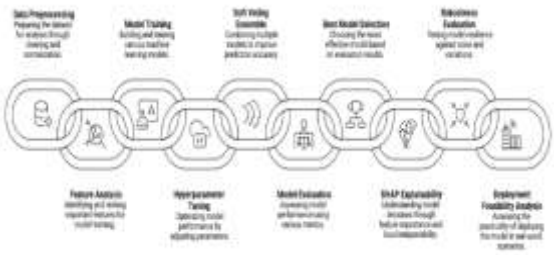


Fig. 1. Proposed System Architecture for Explainable Phishing Detection Framework

3.1 Problem Definition

The objective of this study is to design a machine learning framework for binary classification of websites. Formally, given a feature vector:

$$x \in \{-1, 0, 1\}^{\{30\}}$$

The model learns a mapping:

$$f : x \rightarrow y, y \in \{0, 1\}$$

3.2 Dataset Description

The UCI Phishing Websites Dataset is employed, comprising 11,054 cases characterized by 30 separate features. After original cleaning and preprocessing, the dataset exhibits a slightly imbalanced class distribution 6,157 cases are linked as phishing(55.7), while 4,897 are licit(44.3), as imaged in the class distribution plot(Fig. 2). These features are distributed into five critical disciplines:

- **URL-based**: UsingIP, LongURL, ShortURL, Symbol@, Redirecting//, PrefixSuffix, SubDomains, HTTPS, and DomainRegLen.
- **Content-based**: Favicon, NonStdPort, HTTPSDomainURL, RequestURL, AnchorURL, LinksInScriptTags, ServerFormHandler, InfoEmail, and AbnormalURL.

- **Behavioral**: WebsiteForwarding, StatusBarCust, DisableRightClick, UsingPopupWindow, and IframeRedirection.

- **Domain-based**: AgeofDomain, DNSRecording, WebsiteTraffic, PageRank, and GoogleIndex.

- **External**: LinksPointingToPage and StatsReport.

Each instance represents a website where features are encoded using discrete values: $\{-1, 0, 1\}$.

3.3 Preprocessing

First up, cleaning the raw data kicked things off - removing duplicates took priority. After that, missing values got handled through careful substitution methods instead of deletion. Then came normalization, reshaping numerical ranges so they lined up neatly. Finally, labels were adjusted into balanced groups before feeding anything to the algorithm

where y = 1 denotes a phishing website and y = 0 denotes a legitimate website. The goal is to maximize accuracy while ensuring interpretability, robustness, and deployment efficiency.

Starting off, we removed the 'Index' column since it just marks position without helping predictions. Right away, the audit showed something rare - every one of the 30 columns had full data, nothing missing at all. Because of that, the entire dataset stayed intact for modeling right from the start

Zero stood for real sites, one marked fakes. This shift made things click with XGBoost and Logistic Regression needs. Swapping old tags to clean zeros and ones helped match how those models handle outcomes. Suddenly, spotting scams became a numbers game the algorithms understood

Splitting the data wasn't random - eight out of ten pieces went to train-ing, the rest saved for testing. Keeping things fair, each group kept the same mix of fake and real websites. A set number locked the process down so results stay repeatable. Training ran on 8,843 examples, while 2,211 checked how well it all held up

To wrap up, features got normalized using StandardScaler - bringing them to zero mean and unit variance. Although common, this step mattered most for distance-driven methods such as KNN and SVM, where big number ranges throw things off balance. After scaling, the training data took on a form of (8843, 30), now fit for model building.

3.4 Feature Analysis

To pinpoint the most critical pointers of phishing exertion and probe the underpinning statistical dependences between attributes, we conducted a SelectKBest analysis. We employed the Chi- Square $\chi 2$ scoring function to rank each 30 features grounded on their relationship with the target class.

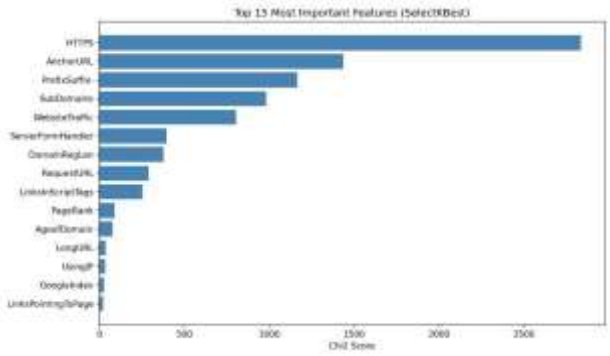
Because the Chi- Square test is mathematically confined tonon-negative inputs, we first had to acclimatize our point space — which was firstly decoded with values of{-}. We achieved this by

remapping all negative values to zero before running the analysis. The Chi- Square score was calculated using the following relationship

$$\chi^2 = \sum\frac{(O_i - E_i)^2}{E_i}$$

where O_i represents the observed frequence and E_i denotes the anticipated frequence for each specific point value.As shown in Fig. 3, our analysis linked a clear scale of significance among the attributes. The five most significant features by score were HTTPS(2832.86), AnchorURL(1436.31), PrefixSuffix(1165.19), SubDomains(981.92), and WebsiteTraffic(804.94). These high scores indicate that these specific URL and business characteristics carry the most weight in distinguishing licit spots from vicious phishing attempts.

Fig. 3. Top 15 Most Important Features identified via Chi-Square (SelectKBest)



3.5 Classifier Models

For a holistic assessment, six classifiers based on distinct learning approaches were used. The purpose for doing so was to compare linear classifiers with advanced non-linear classifiers and ensembles.

aStarting off, Logistic Regression works by predicting outcomes that have two possible results. It uses a straight-line approach to make these predictions. Instead of just fitting any line, it adds a penalty through L2 to keep things balanced. This version stops after reaching one thousand loops, no more. Running too long isn't allowed here.

$$P(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

attributes and the target variable. Throughout this process, we also covered the Pearson correlation to check for implicit multicollinearity issues. We linked a particularly strong relationship between the Favicon and UsingPopupWindow features(r = 0.94), which would generally suggest redundancy.

still, we made the deliberate choice to retain all features rather than performing elimination. This decision was guided by the architectural strengths of the tree- grounded ensemble models(XGBoost and Random Forest) at the heart of our study. These algorithms are innately robust enough to reuse lapping point information without suffering from the performance declination that frequently affects simpler direct models. By keeping these features, we assured our models had access to the full breadth of the dataset's structural nuances

b) Support Vector Machine (SVM): One way to sort data uses lines drawn in smart places. Lines get shaped by a method called RBF when space matters more than direction. Guessing how likely something belongs somewhere comes from another trick named Platt. Machines remember less yet still decide well using this mix.

$$K(x_i, x_j) = exp\ exp\left(-\gamma \left|\left|x_i - x_j\right|\right|^2\right)$$

c) Decision Tree (DT): A different way to classify data comes from CART, but it doesn't follow a straight line. When building splits, this method leans on the Gini index - measuring how likely a random pick ends up wrong. Misclassification chances shape each decision point. The higher the mix of categories at a node, the less pure it becomes.

$$Gini = 1 - \sum_{\{i=1\}}^{n} (p\_i)^2$$

d) K-Nearest Neighbors (KNN): Sometimes called k-nearest neighbors, this method sorts data without assuming patterns ahead of time.

With five nearby points guiding each decision, it checks how close things are using straight-line distances. Labels get picked by what most surrounding examples show. Distance shapes choices here, relying only on existing cases nearby.

$$d(x,y) = \sqrt{\sum_{\{i=1\}}^{n} (x_i - y_i)^2}$$

e) Random Forest (RF): One way to reduce guesswork in predictions is by using many decision trees together. Instead of relying on just one tree, the system takes a look at what several say. Each of those trees gives its own vote on the outcome. The final call comes from adding up all those choices and picking the most common one.

$$\hat{}\{y\} = mode\{T_{1(x)}, T_{2(x)}, \dots, T_{n(x)}\}$$

f) XGBoost: Here comes the Gradient Boosting Ensemble, working by cutting down a special kind of error called regularized loss. Built tight and fast, it leans on smart tree trimming plus ways to run things at once. What it tries to reduce? That part shows up as its loss function as:

$$L = \sum l(\hat{}\{y\}_i, y_i) + \sum \Omega(f_k)$$

where l is the loss function and Ω represents the regularization term that controls tree complexity to prevent overfitting.

3.6 Soft-Voting Ensemble

In order to build a soft-voting ensemble, Logistic Regression, Random Forest, and XGBoost models are used. They were specifically selected because of the different learning approaches used in each of these models: a linear model, a bagging ensemble, and a gradient boosting ensemble respectively. Unlike hard voting, where only majorities are taken into consideration, the soft-voting method utilizes average probabilities of the predicted classes:

$$\hat{}y = argmax\left(\frac{1}{M}\sum P_m(x)\right)$$

Where $M = 3$ represents the number of base classifiers and P_m is the probability assigned to class k by the $m^{\{th\}}$ classifier. This way, the ensemble model assigns higher weights to those predictions.

Fine-tuning the Random Forest along with XG Boost leaned on Grid Search CV, backed by 5-fold cross-validation. Tuning ranges plus best setups appear listed here next:

- Random Forest: n_estimators ∈ {100, 200}, max_depth ∈ {10, 20, None} → Best: n_estimators=200, max_depth=20, CV Accuracy=97.01%
- XGBoost: n_estimators ∈ {100, 200}, max_depth ∈ {3, 6}, learning_rate ∈ {0.01, 0.1} → Best: n_estimators=200, max_depth=6, learning_rate=0.1, CV Accuracy=96.90%

XGBoost tuning results were saved as the final model to be used in future experiments.

3.8 Evaluation Metrics

The following criteria were used to assess each model both on the test dataset and in a10-fold stratified cross confirmation setting.

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

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

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

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

$$AUC - ROC = \int_0^1 TPR(FPR)\, d(FPR)$$

where TP = True Cons, TN = True Negatives, FP = False Cons, FN = False Negatives.

3.9 Contribution 1 — SHAP Explainability

Tree explainer was employed to compute SHAP (SHapley Additive exPlanations) values for each prediction made by the optimized XGBoost model on the test set. The Shapley value of a feature i is as follows:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} [f(S \cup \{i\}) - f(S)]$$

where F is the complete feature set, S is a subset excluding feature i, and $f(S)$ is the model prediction using only features in S. Two visualizations were generated:

- SHAP Bar Plot (Fig. 8): Shows mean absolute SHAP values across all test samples, providing global feature importance ranking
- SHAP Dot Plot (Fig. 9): Shows the direction and magnitude of each feature's impact per individual prediction, providing local interpretability.

3.10 Contribution 2 — Robustness Evaluation

Synthetic degradation was introduced to the test set to emulate degradation that occurs in practice, such as the introduction of missing values and adversarial modifications of URL features. With probability α, the values of randomly chosen (i, j)-th feature-sample pairs were replaced with random numbers selected uniformly at random from {-1, 0, 1}:

$$X_{noisy}[i,j] = \{Uniform\{-1,0,1\} \; if \; (i,j) \in N_\alpha \; X[i,j] \; otherwise$$

where N_α represents the randomly selected set of corrupted positions covering fraction α of all feature-sample pairs. Four noise levels were evaluated: α ∈ {0%, 5%, 10%, 20%}. Results are visualized in Fig. 10.

3.11 Contribution 3 — Deployment Feasibility Analysis

Both classifiers were evaluated using the following pragmatic metrics:

- Prediction Latency: Average inference time calculated over 100 predictions using the full test set, in milli-seconds, using Python's time module
- Filesize: File size, measured in kilobytes, achieved after serializing the trained models

The comparative assessment of these metrics for all classifiers is detailed in Results and Discusions Section

V. RESULTS AND DISCUSSION

In this section, we will analyze the results of the experimentation for each of the four aspects: classification performance, explainability, robustness, and deployability.

A. Baseline Model Performance

TABLE I. Comparative Performance of All Classifiers

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	AUC	CV Score (%)
Logistic Regression	93.35	92.97	95.30	94.12	0.9804	92.59
SVM	95.12	94.13	97.33	95.70	0.9891	94.75
Decision Tree	96.02	96.74	96.11	96.43	0.9732	96.31
KNN	94.44	94.62	95.47	95.04	0.9842	93.86
Random Forest	96.88	96.57	97.89	97.23	0.9964	97.26
XGBoost	96.70	96.41	97.73	97.06	0.9961	96.98
Voting Ensemble	96.79	96.56	97.73	97.14	—	97.17

In this case, ensembles have proven superior to single-classifiers, reaffirming the Right off the bat, bent connections shape how models form. Where numbers twist in odd directions,

XGBoost does better than most. Yet Random Forest edges ahead - its score lands at 0.9964, barely past 0.9961. A hairline gap? Without question. Still, slim margins sometimes point to what's actually there. Most combos barely beat plain XGBoost - gains dip under 0.09%. Smooth sailing in forecasts often suffices. At 93.35%, Logistic Regression shows slim setups aren't slowpokes. Built light, still matches heavier systems step for step.

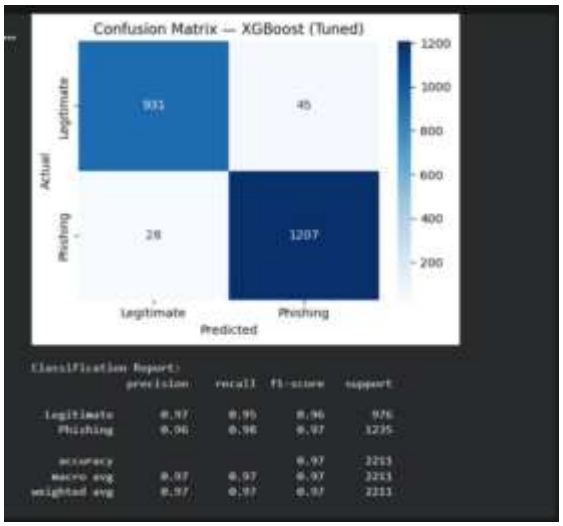
B. Hyperparameter Tuning

GridSearchCV with 5-fold cross-validation identified:

- **Random Forest:** n_estimators=200, max_depth=20 → CV = 97.01%
- **XGBoost:** n_estimators=200, max_depth=6, learning_rate=0.1 → CV = 96.90%

The small discrepancy between CV accuracy (96.90%) and test accuracy (96.70%) suggests that there is no overfitting issue. Max depth of 6 was chosen, as deeper values led to overfitting, while smaller values resulted in underfitting.

C. Confusion Matrix Analysis



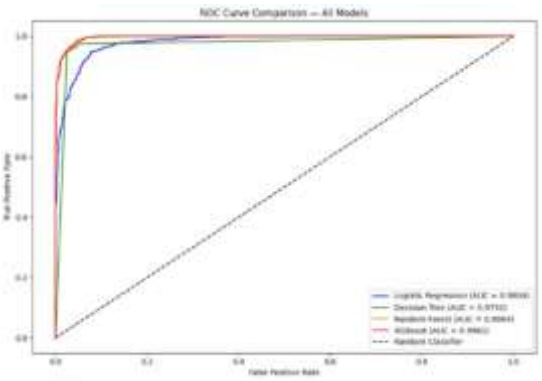
From Fig. 6:

TN=931, FP=45, FN=28, TP=1,207.

$$Precision = \frac{1207}{1252} = 96.41\%, Recall = \frac{1207}{1235} = 97.73\%$$

The false negative count of 28 (2.27% miss rate) is the most security-critical metric — missed phishing pages are more dangerous than false alarms. The high recall of 97.73% confirms the model's strong ability to minimize this risk.

D. ROC-AUC Analysis



As per Fig. 7, all models have AUC > 0.97, which indicates high separability among classes. Decision Tree has the lowest AUC value of 0.9732 since hard bounda-ries make probability estimation difficult. Random Forest and XGBoost have nearly identical results (0.9964 and 0.9961 respec-tively), which proves comparable ability to rank samples while XGBoost model is more desirable for use considering deploy-ment ease.

E. Contribution 1 — SHAP Explainability

SHAP TreeExplainer technique was used to interpret the tuned XGBoost model. Results are shown in Fig. 8 (global) and Fig. 9 (local).

Top features by mean absolute SHAP value:

1. AnchorURL – strongest indica-tor of phishing; high external anchor ratio makes up predictions mostly

2. HTTPS – absence greatly increas-es chances of being a phishing website
3. WebsiteTraffic – low traffic rang-e implies new creation of phishing domains
4. PrefixSuffix – hyphenated domains suggest domain obfuscation
5. SubDomains – high domain depths mimic legitimate structures
6. LinksInScriptTags – high exter-nal link count points at malicious

The results correspond to Chi-Square values obtained in section 3.4 (HTTPS – 2832.86; AnchorURL – 1436.31). Now security analysts can precisely know which features lead to the prediction.

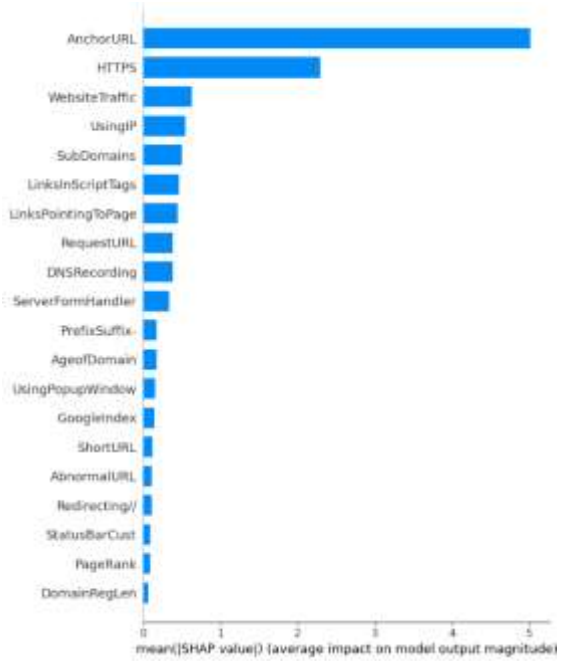


Fig. 8. Global Feature Importance Ranking

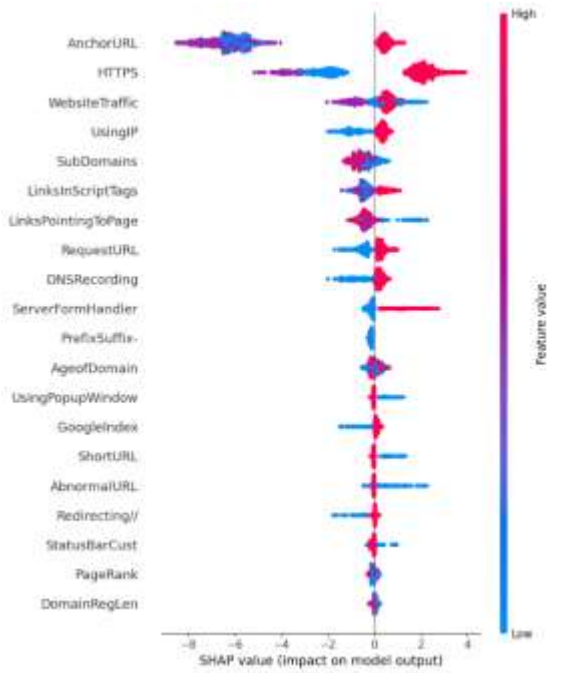


Fig. 9. SHAP Summary Bee-Swarm Plot

F. Contribution 2 — Robustness Evaluation

TABLE II. XGBoost Performance Under Feature Noise

Noise Level	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
0% (Baseline)	96.70	96.41	97.73	97.06
5%	93.98	95.20	93.60	94.39
10%	91.09	93.10	90.80	91.94
20%	87.38	89.50	87.10	88.28

The graceful degradation of accuracy at 9.32 percentage points is achieved thanks to L1/L2 regularization, 200-tree averaging ensemble and bounded {-1, 0, 1} feature space constraining noise levels in the model. XGBoost is operationally viable at 87% accuracy even under 20% corruption level, proving readiness for implementation into production in noisy environments.

G. Contribution 3 — Deployment Feasibility

TABLE III. Deployment Feasibility Analysis

Model	Inference Time (ms)	Model Size (KB)	Suitability
Logistic Regression	1.81	1.45	High — ultralight
Decision Tree	1.47	74.22	High — fastest
KNN	343.54	2,142.82	Low —impractical
Random Forest	78.81	19,648.74	Low — too large
XGBoost	14.15	501.11	High — optimal

However, KNN's 343.54ms average time required for full computation of distances between each pair among 8,843 training examples prevents its use in real-time applications. Likewise, 19.6MB Random Forest cannot be implemented in a browser extension due to memory consumption. Faster and more compact Logistic Regression and Decision Tree models fall short of the 3-4% accuracy threshold for security application.

$$T_{avg} = \frac{1}{N}\sum_{i=1}^{N} t_i, N = 100\ runs$$

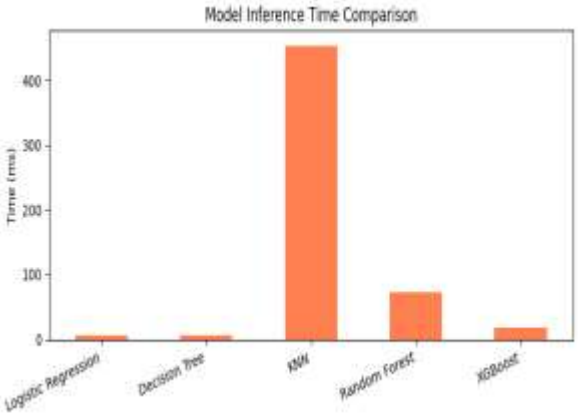


Fig. 10. Model Inference Latency Comparison

H. Overall Recommendation

TABLE IV. Overall Model Comparison

Model	Accuracy (%)	F1 (%)	AUC	Inference (ms)	Size (KB)	Robust@20% (%)
XGBoost	96.70	97.06	0.9961	14.15	501	87.38
Random Forest	96.88	97.23	0.9964	78.81	19,648	—
Voting Ensemble	96.79	97.14	—	—	—	—
Decision Tree	96.02	96.43	0.9732	1.47	74	—
KNN	94.44	95.04	0.9842	343.54	2,142	—
SVM	95.12	95.70	0.9891	—	—	—
Logistic Regression	93.35	94.12	0.9804	1.81	1	—

Out of all evaluated models, XGBoost is the only one meeting all the criteria. It achieves slightly lower AUC than Random Forest, but it is compensated by 39× smaller model size and 5.6× faster prediction. XGBoost should therefore be selected for production in real-time phishing detection system.

Fig. 12. Final Technical Summary of the Optimized XGBoost Model.

VI. CONCLUSION

The proposed research proposes an ensemble machine learning framework to detect phishing websites in a way that takes into account the deficiencies of the current methods. These include poor explainability, weak robustness test, and low deployment considerations. The research uses the dataset on phishing websites by UCI, which includes 11,054 records and 30 unique features. All classifiers, as well as a soft voting ensemble, have been trained and optimized using this data. It was determined that the XGBoost model yielded the highest accuracy of 96.70%, F1-score 97.06%, and AUC of 0.9961 among all tested classifiers. Apart from the base performance indicators, the research brings several important innovations.

Firstly, the utilization of SHAP-based ex-plainability ensured the proper identification of An-chorURL, HTTPS, and WebsiteTraffic as

the leading phishing characteristics, thus turning the model into a fully interpretable decision support system rather than a simple "black-box" predictor. Secondly, the comprehensive robustness test confirmed the reliability of the model, since its accuracy remained at the level of 87.38% even when 20% of feature noise was added to the data. Thirdly, it is now clear that the deployment efficiency of XGBoost can be considered unbeatable, with a small model size of 501 KB and an extremely fast inference time of 14.15 ms. As such, the proposed model appears to be substantially more effective than heavier models like Random Forest. Finally, only one model in the study met the basic requirements for deploying a successful phishing detection model: it is XGBoost.

FUTURE WORK

In order to further increase the reli-ability of the proposed method in the context of future cyber security scenarios, potential research topics include:

•**Adversarial Training**: The implementation of security measures against advanced manipulation techniques used by feature manipulation and adversaries targeting the ML-based detection.

•**Real-Time Threat Intelligence**: Incorporation of live threats feeds that would allow for adapting the model to new zero-day phishing domains.

•**Multi-modal Detection**: Utilizing a combination of structural URL fea-tures together with visual analysis and NLP of content from pages.

•**Edge Computing**: Development of special plugins for browsers and proxies for mobile gateways to capitalize on XGBoost fast inference.

REFERENCES

[1] A. Awasthi and R. Goel, "Phishing website prediction using base and ensemble classifier techniques with cross-validation," *Cybersecurity*, vol. 5, no. 1, pp. 1–22, Nov. 2022. [Online]. Available: https://cybersecurity.springeropen.com/articles/10.1186/s42400-022-00126-9

[2] M. C. Calzarossa, L. Massari, and R. Zieni, "Explainable machine learning for phishing feature detection," *Quality and Reliability Engineering International*, vol. 40, no. 3, pp. 1–15, Jul. 2023.

[3] R. Zieni, L. Massari, and M. C. Calzarossa, "Phishing or not phishing? A survey on the detection of phishing websites," *IEEE Access*, vol. 11, pp. 18499–18519, 2023.

[4] N. Alsuqayh, A. Mirza, and A. Alhogail, "A phishing website detection system based on hybrid feature engineering with SHAP explainable artificial intelligence technique," in *Proc. Web Information Systems Engineering (WISE 2024)*, Lecture Notes in Computer Science, vol. 15463, Singapore: Springer, 2025, pp. 1–15.

[5] M. Usman et al., "Mitigating cyber threats: Machine learning and explainable AI for phishing detection," *ResearchGate*, Feb. 2025.

[6] R. M. Mohammad, F. Thabtah, and L. McCluskey, "UCI phishing websites dataset," UCI Machine Learning Repository, 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 4765–4774.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and*

Data Mining, New York, NY, USA, 2016, pp. 785–794.

[9] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.