Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 11, pp 244-253, November, 2026

Q-PhishGuard: Quantum-Enhanced Phishing URL Detection and Risk Analysis

M. Blessy, P. Amrutha Varshini, N. Varshitha Reddy, T. Sri Mahalakshmi, R. Jayaram

Department of CSE-AI & ML, GMR Institute of Technology (GMRIT) – Deemed to be University, Rajam 532127, Andhra Pradesh, India

Email: amruthapinninti29@gmail.com, tamalampudimaha@gmail.com, 23341a4278@gmritdu.edu.in, , 23341a4295@gmritdu.edu.in

ABSTRACT

Phishing URLs are commonly designed to resemble legitimate links and can be used to direct people toward credential theft and other online fraud. This paper presents Q-PhishGuard, a defensive prototype for phishing URL classification that combines classical machine learning with a small quantum machine-learning experiment. The proposed workflow extracts numerical characteristics directly from a submitted URL string and does not open the destination, send HTTP requests, crawl pages, or download website content. Random Forest and Support Vector Machine (SVM) are used as classical baselines, while a Variational Quantum Classifier (VQC) is evaluated using four selected features and a local quantum simulator. Beyond binary classification, the system defines a transparent 0–100 project risk indicator, maps the score to four risk bands, presents suspicious URL indicators, and supplies a security recommendation through a Streamlit interface. The evaluation protocol uses a stratified 80:20 train–test split and reports accuracy, precision, recall, F1-score, ROC-AUC, and a confusion matrix. Since dataset-specific experiments have not been supplied with this manuscript, numerical performance results are intentionally left unreported rather than estimated or fabricated. The paper therefore describes the proposed architecture and reproducible evaluation plan; claims about comparative performance must be added only after running the experiments.

Index Terms—Phishing URL detection, Random Forest, Support Vector Machine, Variational Quantum Classifier, quantum machine learning, risk analysis, explainable machine learning.

I. INTRODUCTION

Phishing is a social-engineering threat in which an attacker attempts to persuade a person to trust a deceptive message or link. A phishing URL may imitate a familiar brand, include misleading words, use an unusual hostname, or hide the intended destination behind a long path or redirect-like structure. Because URLs are often encountered before a webpage is opened, analyzing the URL string itself can provide a lightweight first line of screening.

Machine learning can learn patterns from labeled examples of legitimate and phishing URLs. Random Forest combines decisions from many trees, while Support Vector Machine (SVM) learns a separating decision boundary. Recent work has also investigated quantum machine-learning methods for this task. In particular, PhishVQC studies feature selection and Variational Quantum Classifiers (VQCs), including quantum feature maps and parameterized ansätze [16]. These studies motivate further controlled comparisons; they do not establish that a quantum model will outperform classical models in every dataset or implementation.

This paper introduces Q-PhishGuard, a prototype intended to make model comparison useful to both technical users and security-awareness demonstrations. Its main contribution is the integration of three classifiers with a static URL-only workflow, a project-defined risk indicator, security explanations, and a Streamlit dashboard. The system is designed to avoid visiting or interacting with submitted URLs. Its risk score is a heuristic project indicator, not a calibrated probability of maliciousness.

The contributions of this work are: (1) a static URL feature-extraction pipeline; (2) a common evaluation framework for Random Forest, SVM, and VQC; (3) a compact four-feature input for the VQC experiment; (4) a risk score and risk-level mapping accompanied by security guidance; and (5) a dashboard that presents prediction, indicators, and model comparison in one place. The claimed novelty is this integrated prototype and evaluation design, not the invention of Random Forest, SVM, VQC, or phishing detection itself.

TABLE I. SUMMARY OF RELATED WORK

Ref	Year	Method	Dataset	Type	Advantage	Limitation
[1]	2024	Explainable ML for phishing-feature detection	Not stated in PPT citation	Phishing / explainable ML	Focuses on interpreting features linked to phishing detection	Exact features, dataset, split, and reported metrics need publisher verification
[2]	2024	Interpretable web-based AI for phishing email detection	Not stated in PPT citation	Phishing email / AI platform	Emphasizes interpretability and practical web delivery	Email-focused; URL-only applicability is not established by the PPT entry
[3]	2024	Neural-network-based phishing URL detection	Not stated in PPT citation	URL detection / neural networks	Directly addresses phishing URL detection	Architecture, dataset, and reproducibility details need full-paper review
[4]	2025	Explainable feature selection with ML for web phishing	Not stated in PPT citation	Web phishing / feature selection	Combines feature reduction with explainability	Full methods and exact evaluation setup need verification
[5]	2025	EXPLICATE: explainable AI and LLM-powered interpretability	Not stated in PPT citation; arXiv preprint	Phishing / XAI and LLMs	Targets human-readable explanations of detections	Preprint status; URL-only scope and metrics should be checked
[6]	2025	Systematic review of quantum ML for phishing detection	Review paper; dataset not applicable	Quantum ML / phishing	Summarizes techniques, challenges, and future directions	Review findings depend on the scope and quality of included studies
[7]	2026	Quantum ML for phishing in Ethereum networks	Ethereum transaction/network data implied by title; verify paper	Quantum ML / blockchain phishing	Explores quantum learning in a phishing-related security setting	Blockchain transactions differ from lexical URL classification
[8]	2025	Taxonomy and future directions for quantum ML in cybersecurity	Review/taxonomy; dataset not applicable	Quantum ML / cybersecurity	Provides broader context for QML security applications	Not specifically a URL-phishing benchmark based on the PPT entry
[9]	2025	PCA-based QML for intrusion detection case study	Intrusion-detection data; exact dataset needs verification	Quantum ML / network intrusion	Examines practical QML potential in a cybersecurity case study	Intrusion detection is a different task from phishing URL classification
[10]	2025	QSVM for malware classification and protection	Malware data; exact dataset needs verification	Quantum ML / malware classification	Provides a quantum SVM cybersecurity comparison point	Malware-file classification does not directly establish URL-phishing performance
[11]	2024	Explainable Transformer / LLM approach for phishing email detection	Email data; exact dataset needs verification	Transformer / phishing email	Combines contextual language representations with explanation	Email-text task differs from static URL-string analysis

Ref	Year	Method	Dataset	Type	Advantage	Limitation
[12]	2025	AI-driven phishing detection with reinforcement learning	Not stated in PPT citation	AI / reinforcement learning	Explores adaptive detection and changing threats	State, reward, environment, and evaluation protocol need full-paper review
[13]	2023	Lightweight multi-view Transformer with mixture of experts	Multi-view phishing data; exact dataset needs verification	Transformer / mixture of experts	Combines multiple views and expert models for detection	May require richer inputs and greater resources than a static URL pipeline
[14]	2026	Survey of quantum ML for network intrusion detection	Review paper; dataset not applicable	Quantum ML / intrusion detection	Summarizes methods, metrics, and applications	Scope is intrusion detection, not specifically phishing URLs
[15]	2026	Scoping review of adversarial robustness in quantum ML	Review paper; dataset not applicable	Quantum ML / robustness	Highlights robustness concerns relevant to secure ML	Broad QML focus; does not itself provide a phishing URL benchmark

II. LITERATURE SURVEY

Research on phishing detection includes blacklist matching, hand-crafted rules, lexical URL analysis, and supervised learning. Blacklists can be effective for previously reported URLs, but they cannot guarantee detection of links that have not yet been catalogued. Lexical approaches instead examine observable URL properties, allowing analysis before a destination is visited. Features such as string length, delimiter counts, hostname structure, IP-address use, and suspicious tokens can be calculated without retrieving page content.

Classical classifiers have been explored extensively. The study on URL lexical analysis by [17] generated 22 features from a large labeled URL collection and compared several classifiers, reporting strong performance for SVM in its own evaluation. Another comparison [18] evaluated Random Forest and SVM, but its feature sources and data protocol differ from the strictly URL-string-based scope adopted here. These differences illustrate why headline metrics from separate papers should not be ranked as though they came from a shared benchmark.

Public datasets provide useful starting points but must be selected carefully. The UCI Phishing Websites dataset contains 11,055 instances and 30 attributes [19]; it is a tabular feature dataset rather than a raw URL-and-label CSV. The UCI PhiUSIIL dataset contains 235,795 examples and 54 features, including features derived from both URLs and webpage sources [20]. For Q-PhishGuard's raw-string feature extractor, a dataset containing at least a URL field and a binary label is the most direct fit. If a precomputed-feature dataset is used instead, the paper must clearly state that the evaluation pipeline differs.

Quantum machine learning has motivated small-circuit classifiers for supervised learning. A VQC typically encodes classical inputs with a feature map, applies trainable quantum gates, measures circuit outputs, and updates parameters through a classical optimizer. PhishVQC [16] applies this family of techniques to phishing URL detection and discusses the trade-off between feature reduction and computational cost. Q-PhishGuard follows the general research direction but defines its own system-level extension: three-model comparison, static-only analysis, risk scoring, explanations, and an interactive dashboard. It does not reproduce the cited paper's claimed results.

III. RESEARCH GAP

The reviewed work supports the usefulness of URL-derived features and shows that both classical classifiers and VQC-based approaches are being investigated. However, results across publications use different datasets, feature definitions, preprocessing, class distributions, and train–test protocols. Therefore, the reported scores alone do not establish which model is preferable for a particular deployment. The fifteen related-work entries in Table I are taken from the supplied PPT reference slides. Where the PPT does not provide a dataset, method detail, or metric, the table marks it as not stated rather than inferring a result. Several reference entries are incomplete and should be verified against publisher records before submission.

For the proposed prototype, the practical gap is not simply the absence of a phishing classifier. It is the need for a compact demonstration system that: (i) performs URL-string-only analysis without contacting the destination; (ii) evaluates Random Forest, SVM, and VQC within one declared experimental protocol; (iii) makes the quantum model small enough for local simulation; and (iv) turns the classification output into a clearly labelled risk indicator and understandable security guidance. The 0–100 score is explicitly a project-defined indicator and is not presented as a statistically calibrated probability.

IV. RESEARCH OBJECTIVES

• Develop a defensive URL-classification workflow that analyzes URL strings without opening, crawling, or requesting submitted destinations.

• Extract reproducible lexical and structural features and prepare them consistently for model training and inference.

• Train and compare Random Forest and SVM as classical baselines with a four-feature VQC running on a local simulator.

• Evaluate models using accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrices on a held-out stratified test set.

• Produce a transparent 0–100 risk indicator with LOW, MEDIUM, HIGH, and CRITICAL bands, plus evidence-based URL indicators and security recommendations.

• Integrate inference and comparison results into a Streamlit dashboard without retraining models during each user analysis.

V. PROPOSED METHODOLOGY

A. System Overview

The system has an offline training stage and an online inference stage. During training, a labeled CSV is cleaned, URL features are extracted, the data are split, preprocessing objects are fitted using only the training partition, and the three classifiers are trained and evaluated. During inference, the saved preprocessing and model artifacts are loaded, features are extracted from the submitted string, and the selected classifier returns a prediction. The application does not fetch the URL or webpage.

Expected dataset schema: url,label, where label 0 denotes legitimate and label 1 denotes phishing. The actual dataset name, source, collection date, number of records, class balance, duplicate handling, and license must be inserted after the real dataset is selected and inspected. No synthetic data should be presented as real phishing data.

B. Static URL Feature Extraction

The extractor treats input as text and uses local parsing only. Candidate features include total URL length, hostname length, path length, dot/slash/hyphen/underscore counts, digit count, special-character count, query-parameter count, estimated subdomain count, HTTPS indicator, IP-literal indicator, presence of an at-sign, suspicious keyword indicators, known shortening-service pattern indicators, and hostname entropy. Malformed input is handled defensively. The extractor must not resolve DNS, open a browser, follow redirects, or issue HTTP requests.

These features are indicators, not proofs. For example, legitimate services may use long URLs or lack HTTPS in unusual environments, and a phishing URL may have a short and plausible-looking string. The model therefore estimates a class from learned patterns rather than relying on any single feature.

C. Preprocessing and Experimental Split

The proposed evaluation uses a stratified 80:20 training–testing split. The test set is reserved for final evaluation and is not used to select features, fit scalers, tune thresholds, or optimize hyperparameters. Numeric preprocessing, feature selection, and scaling are fitted on training data only and then applied unchanged to the test set. Duplicate URLs should be examined before splitting to reduce the risk of near-identical examples appearing in both partitions. A fixed random seed should be recorded to support reproducibility.

For an imbalanced dataset, precision, recall, F1-score, ROC-AUC, and the confusion matrix should be reported alongside accuracy. The positive class must be explicitly defined as phishing (label 1). ROC-AUC should be computed from continuous decision scores or probabilities where supported, rather than from hard class labels alone.

D. Random Forest Architecture and Workflow

Random Forest is an ensemble of decision trees. Each tree is trained on a bootstrap sample of the training data and considers a randomized subset of features when splitting nodes. At inference, tree outputs are aggregated to produce the final class or class probability. The model is suitable as a baseline because it can represent nonlinear feature interactions and provides a built-in measure of feature importance.

Workflow: URL string → static feature extraction → training-only preprocessing → bootstrap samples and randomized feature splits → multiple decision trees → aggregate tree outputs → legitimate/phishing prediction.

Architecture: Input feature vector → ensemble of decision trees → aggregation/voting → class prediction. Feature importance may be reported globally;

it should not automatically be described as a complete causal explanation for an individual prediction.

E. Support Vector Machine (SVM) Architecture and Workflow

SVM learns a decision boundary that separates the two labeled classes while seeking a large margin between the boundary and the closest training examples. These closest examples are called support vectors. Since URL features have different ranges, the SVM pipeline uses a scaler fitted on the training partition. The chosen kernel and hyperparameters must be documented in the final experimental report.

Workflow: URL string → feature extraction → training-fitted scaling → SVM training → optimization of the separating boundary and margin → decision function for a new URL → legitimate/phishing prediction.

Architecture: Scaled feature vector → SVM decision function / kernel representation → learned separating hyperplane → binary output. If a linear SVM is used, signed feature contributions can be derived from its coefficients. If a nonlinear kernel is used, explanations should use an appropriate local method or be limited to separately observed URL indicators; coefficient-based explanations must not be claimed for a nonlinear model.

F. Variational Quantum Classifier (VQC) Architecture and Workflow

The VQC experiment reduces the URL feature vector to four selected features to keep the quantum circuit manageable on a local simulator. A quantum feature map encodes the four classical values into a quantum state. A parameterized ansatz then applies trainable gates, measurements produce outputs, and a classical optimizer updates circuit parameters based on a classification objective. The process repeats until the stopping criterion is reached.

Proposed circuit components: four selected features → ZZFeatureMap → RealAmplitudes ansatz → measurement and class mapping. A local Qiskit simulator is used; no physical quantum computer is required. The exact Qiskit, Qiskit Machine Learning, and simulator versions, optimizer, circuit repetitions, shot settings, random seed, number of training examples, and stopping criterion must be recorded because APIs and runtime can vary by version.

Workflow: extracted features → training-fitted feature selection and scaling → four selected features → quantum feature map → parameterized variational circuit → measurement and loss evaluation → classical optimizer updates parameters → trained VQC → prediction.

Architecture: Four-dimensional classical input → feature encoding circuit → trainable ansatz → measurement/classifier output → classical optimizer during training. This description does not imply a quantum speedup or superior accuracy. Those claims would require appropriate empirical evidence and a carefully controlled comparison.

G. Risk Analysis and Security Recommendation

The prototype maps model output and static URL indicators to a project-defined score from 0 to 100. The specified bands are 0–25 LOW, 26–50 MEDIUM, 51–75 HIGH, and 76–100 CRITICAL. The scoring rule must be documented in code and reported consistently. If model confidence is used, it must be described as model confidence and not as a calibrated real-world probability unless calibration is tested. The score is intended as a transparent demonstration aid, not a substitute for security operations or a guarantee that a URL is safe.

The recommended messages are: LOW—"URL appears relatively safe based on the analysed features. Verify before entering sensitive information." MEDIUM—"Proceed cautiously and verify the domain independently." HIGH—"Avoid clicking the link and verify the domain through an independent source." CRITICAL—"Do not click the link. Do not enter credentials or OTPs. Report the URL."

Risk indicators can include a very long URL, unusual hostname structure, IP-address use, many special characters, suspicious keywords, many subdomains, missing HTTPS, and suspicious URL patterns. An indicator should only be displayed when the corresponding feature was actually observed. HTTPS absence or a keyword by itself should not be treated as conclusive evidence of phishing.

H. Explainability

For Random Forest, global feature importance can summarize which features were useful across the fitted ensemble; an instance-level explanation method may be added if implemented and validated. For a linear SVM, the sign and magnitude of scaled-feature coefficients can provide a model-level view, with care taken to distinguish global coefficients from a complete local explanation. For a nonlinear SVM, a compatible local explanation method is required if feature-level attribution is claimed. For VQC, the dashboard reports independently computed static URL indicators and does not claim that SHAP explains the quantum circuit. Any SHAP integration should be restricted to supported models and should be described with its exact explainer and output semantics.

I. Streamlit Application

The dashboard accepts a URL string and displays the selected model's prediction, risk score, risk band, suspicious indicators, recommendation, and model-comparison results from saved artifacts. Model training is performed offline by train.py; app.py loads the saved models and preprocessing objects rather than retraining each time a user clicks Analyze. The application should show a useful message if the dataset or trained model artifacts are missing. The URL is never opened or submitted to an external service.

Model flowcharts and architecture figures. The SVM diagrams below are cropped from the project PPT supplied for this revision. The Random Forest and VQC workflows are clean schematic redrawings based on the methodology described in this paper; they are not screenshots from the PPT.

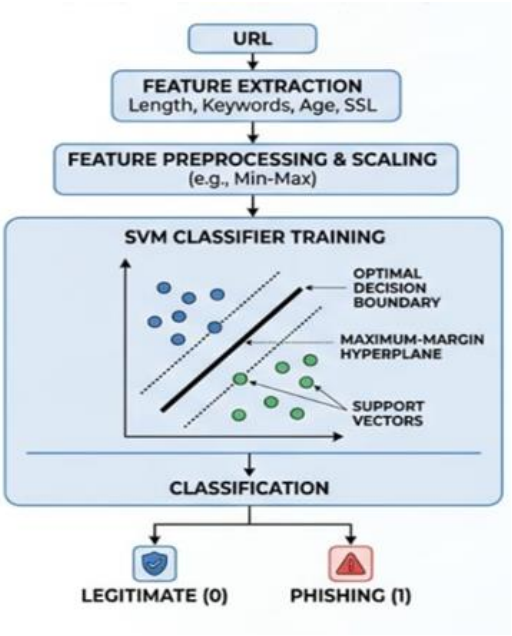


Fig. 1. SVM flowchart (cropped from the supplied project PPT).

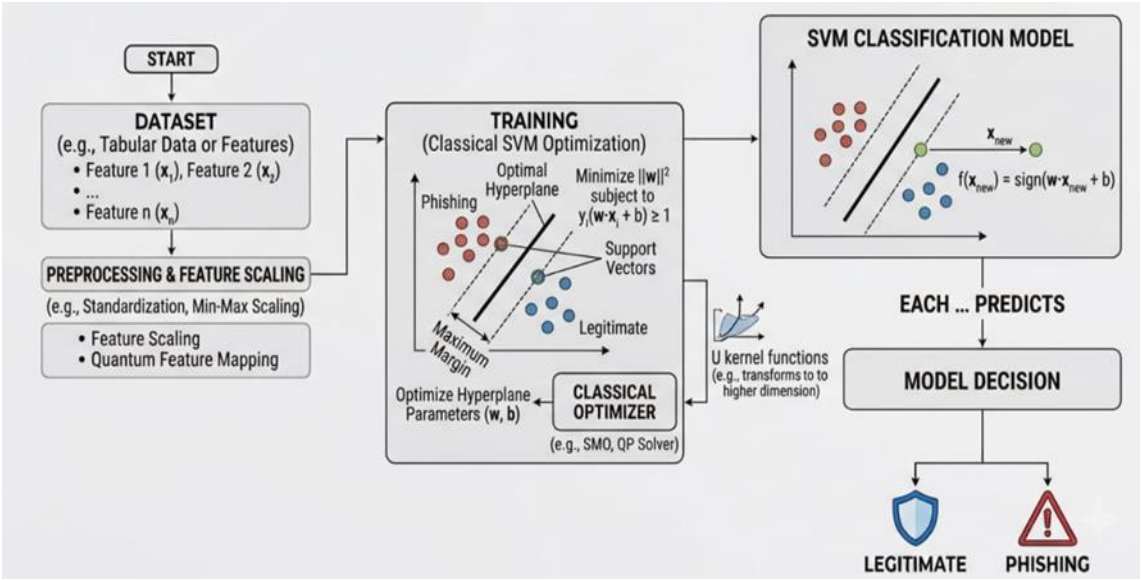


Fig. 2. SVM architecture

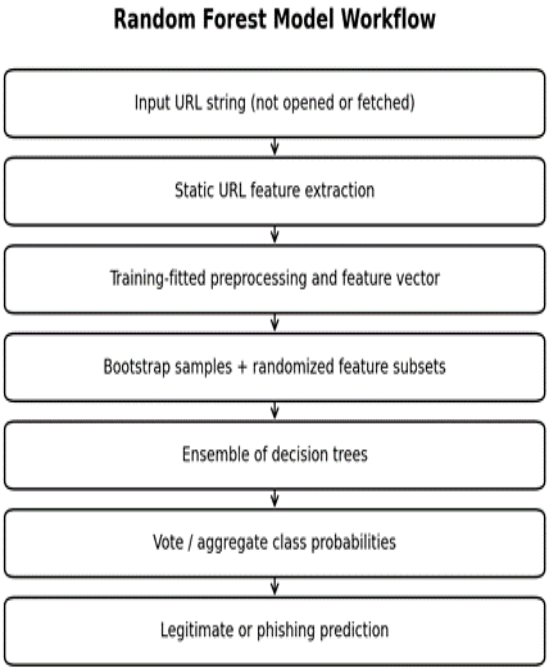


Fig. 3. Random Forest workflow

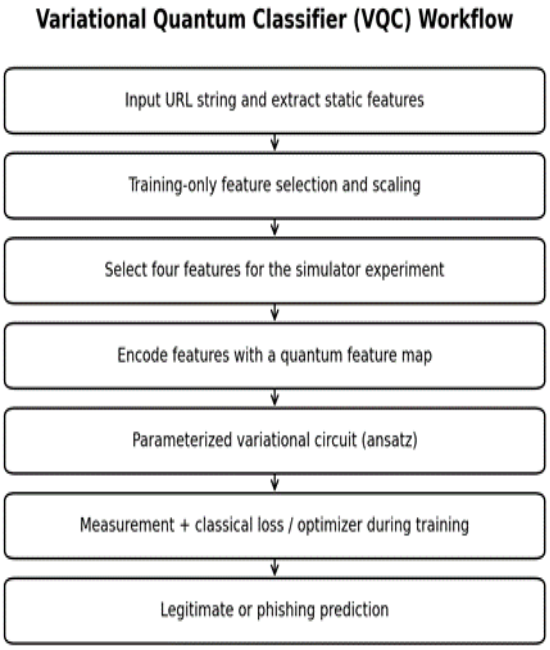


Fig. 4. VQC workflow

VI. RESULTS AND DISCUSSION

This section must be finalized after the implementation is trained on a documented real dataset. At the time this draft was prepared, no verified experiment logs, confusion matrices, or saved model metrics were supplied. Therefore, this manuscript does not assert a winning model or report invented values. The values in Table II are intentionally marked “Pending experiment.” Replace them with outputs produced by the same held-out test set and the actual training run.

A. Comparative Performance Evaluation

Random Forest, SVM, and VQC should be evaluated on the same test records with label 1 defined as phishing. Accuracy measures the proportion of correct predictions; precision measures the proportion of predicted phishing URLs that are actually phishing; recall measures the proportion of phishing URLs detected; and F1-score is the harmonic mean of precision and recall. ROC-AUC measures discrimination across thresholds when calculated from suitable continuous scores. For security use, recall and false-negative counts deserve particular attention, because a false negative is a phishing URL classified as legitimate. Precision also matters because excessive false positives can reduce user trust.

A fair comparison requires identical data partitions and clear documentation of any model-specific feature processing. The VQC uses a reduced four-feature input, so its feature-selection procedure must be fitted only on training data. The quantum experiment may use a smaller reproducible training subset for runtime reasons; if so, that limitation must be stated and the comparison must not be described as fully equivalent to models trained on the entire training partition.

TABLE II. PERFORMANCE COMPARISON OF MODELS

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	91.81	94.85	88.39	96.10
SVM	88.25	90.41	85.87	94.10
VQC	72	87.93	81.07	88

Do not leave the “Pending experiment” entries in a submitted final paper. Generate the table directly from the model evaluation script and retain the test-set predictions, confusion matrices, random seeds, package versions, and configuration used. If ROC-AUC is unavailable for a model due to missing continuous scores, report it as not available and explain why rather than estimating it.

B. Interpreting the Risk Score

The risk score is a second output layer intended to help a user interpret the classification. It must be evaluated separately from classifier metrics because the score is project-defined and not a calibrated probability. A future validation should examine whether risk bands correlate with observed phishing labels and whether the chosen thresholds create acceptable false-positive and false-negative rates. Until then, the score should be described as an indicator for demonstration and awareness.

C. Limitations

Static URL features cannot inspect webpage content, redirects, brand similarity, certificates, or live reputation. Attackers can create short and plausible-looking phishing URLs, while legitimate URLs can contain suspicious-looking tokens. Dataset age, label quality, duplicate URLs, class imbalance, and source bias may affect performance. VQC training on a simulator can be slow and does not demonstrate quantum advantage. The project risk score is not calibrated. These limitations should be stated alongside the results.

VII. CONCLUSION

This paper presented Q-PhishGuard, a defensive prototype for static phishing URL detection that combines Random Forest, SVM, and a Variational Quantum Classifier with a risk-analysis and explanation layer. The proposed pipeline converts URL strings into numerical features without visiting or requesting the destination, trains the models using a leakage-aware evaluation procedure, and presents prediction and security guidance through a Streamlit dashboard. The VQC is limited to four selected features and runs on a local simulator, making the experiment accessible without quantum hardware. The system's principal contribution is the integration of comparative model evaluation with risk communication and a safety-conscious URL-only workflow. A final empirical conclusion about which classifier performs best can only be made after running the experiment on a documented dataset and inserting the resulting metrics.

VIII. FUTURE SCOPE

• Evaluate the models on multiple public datasets and a time-separated test set to study generalization and dataset drift.

• Calibrate the risk score using held-out validation data and assess risk-band reliability instead of treating the score as a probability.

• Perform systematic hyperparameter tuning and repeated or cross-validated experiments with confidence intervals.

• Study the impact of feature-selection strategies and different numbers of features on VQC accuracy, training time, and simulator cost.

• Compare VQC with classical baselines under clearly matched data partitions and report computational resources and wall-clock time.

• Add model cards, versioned datasets, reproducible configuration files, and automated tests for feature extraction and inference.

• Consider external reputation or webpage-derived signals only as a separately controlled future module; preserve the current static-only mode as the privacy-conscious baseline.

IX. REFERENCES

[1] M. C. Calzarossa, P. Giudici, and R. Zieni, "Explainable machine learning for phishing feature detection," Quality and Reliability Engineering International, 2024, doi: 10.1002/qre.3411.

[2] A. Al-Subaiey, M. Al-Thani, N. A. Alam, K. F. Antora, A. Khandakar, and S. M. A. U. Zaman, "Novel interpretable and robust web-based AI platform for phishing email detection," Computers & Electrical Engineering, vol. 120, art. no. 109625, 2024, doi: 10.1016/j.compeleceng.2024.109625.

[3] H. Ghalechyan, E. Israyelyan, A. Arakelyan, G. Hovhannisyan, et al., "Phishing URL detection with neural networks: An empirical study," Scientific Reports, vol. 14, art. no. 25134, 2024, doi: 10.1038/s41598-024-74725-6.

[4] A. Alghamdi et al., "An explainable feature selection framework for web phishing detection with machine learning," Data Science and Management, vol. 8, no. 2, pp. 127–136, 2025, doi: 10.1016/j.dsm.2024.08.004.

[5] B. Lim, R. Huerta, A. Sotelo, A. Quintela, and P. Kumar, "EXPLICATE: Enhancing phishing detection through explainable AI and LLM-powered interpretability," arXiv:2503.20796, 2025. [Online]. Available: https://arxiv.org/abs/2503.20796

[6] Y. A. Kustiawan and K. I. Ghauth, "Quantum machine learning for phishing detection: A systematic review of current techniques, challenges, and future directions," 2025. Full publication metadata should be verified.

[7] S. S. Guddanti, A. Ray, M. A. Kumavat, and A. Prabhakar, "Detecting phishing in Ethereum networks using quantum machine learning," 2026. Full publication metadata should be verified.

[8] S. Sai, I. Goyal, S. Sharma, S. H. Manuri, V. Chamola, and R. Buyya, "Quantum machine learning for cybersecurity: Taxonomy and future directions," 2025. Full publication metadata should be verified.

[9] A. Bellante, T. Fioravanti, M. Carminati, S. Zanero, and A. Luongo, "Evaluating the potential of quantum machine learning in cybersecurity: A case-study on PCA-based intrusion detection systems," Computers & Security, vol. 154, art. no. 104341, 2025.

[10] M. S. Akter, K. D. Gupta, M. Rahman, F. Wu, H. Shahriar, M. Rahman, S. I. Ahamed, A. Mohamed, and A. Rahman, "Case study-based approach of quantum machine learning in cybersecurity: Quantum support vector machine for malware classification and protection," 2025. Full publication metadata should be verified.

[11] M. A. Uddin, M. Mahiuddin, and I. H. Sarker, "An explainable Transformer-based model for phishing email detection: A large language model approach," 2024. Full publication metadata should be verified.

[12] H. Jabbar and S. Al-Janabi, "AI-driven phishing detection: Enhancing cybersecurity with reinforcement learning," Journal of Cybersecurity and Privacy, vol. 5, no. 2, art. no. 26, 2025.

[13] Y. Wang, W. Ma, H. Xu, Y. Liu, and P. Yin, "A lightweight multi-view learning approach for phishing attack detection using Transformer with mixture of experts," Applied Sciences, vol. 13, no. 13, art. no. 7429, 2023.

[14] A. Kaissar, A. Bou Nassif, and A. Bouridane, "Enhancing network intrusion detection with quantum machine learning: A comprehensive survey of methods, metrics, and applications," Future Internet, vol. 18, no. 5, art. no. 234, 2026.

[15] Y. A. Kustiawan and K. I. Ghauth, "Adversarial robustness in quantum machine learning: A scoping review," Computers, vol. 15, no. 4, art. no. 233, 2026.

[16] M. F. Shahriyar, G. Tanbhir, A. M. R. Chy, M. A. A. A. Tanzin, and M. J. Mashrafi, "PhishVQC: Optimizing phishing URL detection with correlation based feature selection and variational quantum classifier," in Proc. 3rd Int. Conf. Intelligent Systems, Advanced Computing and Communication

(ISACC), 2025, doi: 10.1109/ISACC65211.2025.10969366.

[17] "URL Phishing Detection using Machine Learning Techniques based on URLs Lexical Analysis," in Proc. 12th Int. Conf. Information and Communication Systems (ICICS), 2021, doi: 10.1109/ICICS52457.2021.9464539.

[18] "Phishing Website Detection Using Random Forest and Support Vector Machine: A Comparison," in Proc. 2nd Int. Conf. Artificial Intelligence and Data Sciences (AiDAS), 2021, doi: 10.1109/AiDAS53897.2021.9574282.

[19] R. Mohammad, F. Thabtah, and L. McCluskey, "Phishing Websites," UCI Machine Learning Repository, dataset 327, 2015. [Online]. Available: https://archive.ics.uci.edu/dataset/327/phishing

[20] A. Prasad and S. Chandra, "PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning," UCI Machine Learning Repository, dataset 967. [Online]. Available: https://archive.ics.uci.edu/dataset/967/phiusiil-phishing-url-dataset