



COMPARATIVE ANALYSIS AND OPTIMISATION OF SUPERVISED MACHINE LEARNING ALGORITHMS FOR REAL-TIME PHISHING URL DETECTION

¹Shipra Goyal, ² Dr. Rajeev Yadav (Associate Professor)

¹Research Scholar, ²Research Supervisor

¹⁻²Department of Computer Science, Sabarmati University, Ahmedabad, Gujarat

Abstract

Real-time phishing URL detection differs from offline classification because a useful detector must make accurate decisions within a strict computational budget. This analytical paper compares supervised machine learning algorithms from the joint perspectives of classification quality, feature-processing cost, inference latency, calibration, memory demand, interpretability, and operational false-positive risk. The analysis deliberately differs from feature-family-centred phishing studies by treating the model-selection and optimisation problem as the main research object. Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, k-Nearest Neighbour, Naive Bayes, AdaBoost and gradient-boosted trees are examined as candidate classifiers. Recent public URL datasets demonstrate that large-scale supervised benchmarking is feasible: LegitPhish contains more than 100,000 labelled URLs with engineered structural and lexical attributes, while another 2022 dataset contains 111,660 URLs with 22 numerical lexical and structural features. Recent optimisation research also shows that reducing feature counts can lower training complexity while preserving useful predictive performance. The paper proposes a latency-aware evaluation protocol in which feature extraction, preprocessing and model inference are timed separately; predictive metrics are assessed alongside false-positive rate and throughput; and model optimisation uses feature pruning, hyperparameter search, class weighting, calibration and threshold tuning.

Keywords: phishing URL, supervised machine learning, real-time detection, algorithm optimisation, inference latency, Random Forest, XGBoost, SVM, feature selection

1. Introduction

Phishing URLs are designed to lead users toward fraudulent pages that imitate trusted services, collect credentials, distribute malware, or support financial deception. URL screening is attractive because a decision can often be made before the user fully interacts with a destination. In real-time settings, however, the classifier operates under conditions that are different from a conventional offline machine learning experiment. A model may achieve excellent test-set accuracy yet be unsuitable for browser, email-gateway or endpoint deployment if its feature extraction is slow, memory demand is high, probability estimates are poorly calibrated, or false positives interrupt legitimate activity. This paper therefore frames phishing URL detection as a multi-objective optimisation problem. Predictive quality remains necessary, but it is considered together with computational efficiency. The central question is not simply which supervised algorithm produces the largest accuracy value, but which algorithm provides the most defensible trade-off between security effectiveness and real-time responsiveness.

Current public datasets enable systematic comparison. LegitPhish contains 101,219 labelled URLs and engineered URL characteristics including length, entropy, token count, subdomain use and TLD-related information (Potpelwar et al., 2022). Another feature-engineered dataset published in 2022 contains 111,660 URLs with 22 numerical lexical



INTERNATIONAL JOURNAL OF ARTS, COMMERCE & EDUCATION

A Multidisciplinary Peer-Reviewed, Refereed, Open Access Journal,
<https://avjournals.com/> e-ISSN: 2347-7687, Vol 12, Issue 2, July-December 2024

and structural features (Dam & Hung, 2022). These resources make controlled supervised comparisons possible, but they do not remove methodological risks such as class imbalance, duplicate campaigns, temporal drift and source-specific artefacts.

The present study is analytical rather than a report of newly executed experiments. It establishes a comparative framework, optimisation criteria, evaluation protocol and interpretation scheme that can be implemented empirically without inventing performance values.

2. Objectives of the Study

- To compare major supervised machine learning algorithms for real-time phishing URL classification.
- To examine the trade-off between predictive performance and computational efficiency.
- To identify optimisation strategies that reduce feature cost, training complexity and inference latency.
- To evaluate the role of hyperparameter tuning, feature selection, class weighting, probability calibration and decision-threshold optimisation.
- To develop a real-time evaluation protocol based on accuracy-related metrics, false-positive behaviour, latency, throughput and model complexity.
- To propose a deployment architecture that supports rapid decisions while preserving detection quality.

3. Research Questions and Analytical Propositions

The paper addresses four main questions. First, how do supervised classifiers differ in their suitability for real-time phishing URL detection? Second, which optimisation techniques can reduce computational cost without materially degrading security performance? Third, how should models be compared when precision, recall, latency and false-positive rate conflict? Fourth, how can a deployment pipeline separate fast screening from more computationally expensive analysis?

The primary analytical proposition is that model optimisation should be evaluated as a Pareto problem rather than as a single-metric ranking. A second proposition is that feature reduction can improve operational efficiency when redundant or expensive variables are removed carefully. A third proposition is that probability calibration and threshold tuning can be as important as the underlying classifier when the operational costs of false negatives and false positives differ.

4. Real-Time Detection as a Multi-Objective Problem

In offline benchmarking, researchers commonly optimize accuracy, F1-score or ROC-AUC. A real-time detector has additional constraints. The system must extract features, transform them, execute the classifier and return a decision quickly enough that the user or application does not experience unacceptable delay. The relevant latency is therefore end-to-end latency, not model inference alone.

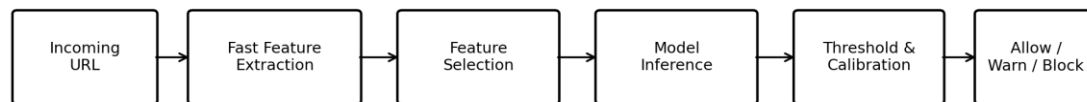
A model can also be computationally efficient during inference but expensive during training. This distinction matters because training may occur periodically on a server, whereas inference may occur continuously on large traffic volumes. Training time affects maintainability and retraining frequency; inference time affects user-facing responsiveness. Memory footprint and model size matter when detection is embedded in browsers, mobile applications or constrained endpoints.

The optimisation target can be represented conceptually as maximizing phishing recall and precision while minimizing false positives, feature-extraction time, inference latency and computational demand. Because these objectives can



conflict, the framework does not assume one universal winner.

Optimisation Pipeline for Real-Time Phishing URL Detection



Optimisation targets: predictive quality • feature cost • inference latency • false-positive control

Figure 1. Optimisation Pipeline for Real-Time Phishing URL Detection

5. Supervised Algorithms Selected for Comparative Analysis

5.1 Logistic Regression

Logistic Regression provides a transparent baseline and is attractive for real-time systems because prediction is computationally inexpensive once features are prepared. Coefficients can be inspected to understand directional associations, and probability outputs can be calibrated and thresholded. Its principal limitation is the linear form of the decision boundary unless interactions and transformations are engineered explicitly.

5.2 Decision Tree and Random Forest

A Decision Tree learns non-linear rules and is inexpensive to evaluate when tree depth is controlled. It is also interpretable at the rule level, but individual trees are unstable and can overfit. Random Forest reduces this instability by averaging many trees and is well suited to heterogeneous URL features. The trade-off is a larger model and greater inference cost than a single tree. Optimisation can focus on tree count, maximum depth, feature subsampling and pruning of weak inputs.

5.3 Support Vector Machine

Support Vector Machines can perform strongly in high-dimensional feature spaces, particularly when class boundaries are well separated. Linear SVM variants can be efficient, whereas non-linear kernels may become expensive as training data grows. Feature scaling is essential, and probability estimates generally require an additional calibration step. SVM is therefore a useful comparator but requires careful latency and scalability testing.

5.4 k-Nearest Neighbour

k-NN has almost no conventional model-training cost, but its inference cost can be high because each new URL must be compared with stored training observations or an indexed representation. This reverses the usual real-time preference: cheap training does not imply cheap prediction. Dimensionality reduction, prototype selection and approximate-neighbour methods can reduce the burden, but k-NN is often less attractive for very high-throughput URL screening.

5.5 Naive Bayes

Naive Bayes is computationally light and can provide a useful low-cost baseline. Its conditional-independence assumptions are restrictive because many URL features are correlated, such as URL length, token count, path length



INTERNATIONAL JOURNAL OF ARTS, COMMERCE & EDUCATION

A Multidisciplinary Peer-Reviewed, Refereed, Open Access Journal,
<https://avjournals.com/> e-ISSN: 2347-7687, Vol 12, Issue 2, July-December 2024

and special-character counts. Nevertheless, its speed makes it valuable as a reference point for determining whether more complex algorithms produce enough benefit to justify additional cost.

6. Comparative Algorithm Matrix

Algorithm	Real-time advantage	Main operational weakness	Primary optimisation focus
Logistic Regression	Very fast inference; compact; interpretable	Limited non-linearity without engineered interactions	Regularization, feature scaling, threshold calibration
Decision Tree	Fast rule-based prediction; explainable	Overfitting and instability	Depth, minimum leaf size, pruning
Random Forest	Robust non-linear ensemble; handles mixed features	Larger model and higher inference cost	Tree count, depth, max-features, feature pruning
SVM	Strong margins; effective in high-dimensional spaces	Kernel methods scale poorly; calibration may be needed	Kernel choice, C, gamma, scaling, feature reduction
k-NN	Minimal training; simple decision logic	Prediction cost grows with reference set	k, distance metric, prototype reduction, dimensionality
Naive Bayes	Extremely light and fast	Strong independence assumptions	Smoothing and feature representation
AdaBoost	Focuses on difficult observations	Sensitive to noise/outliers	Estimator count, learning rate, weak learner
XGBoost / GBDT	Strong tabular non-linear modelling	Tuning and model complexity	Depth, estimators, learning rate, subsampling, regularization

7. Dataset Design for a Fair Real-Time Comparison

A fair algorithm comparison requires a common dataset, identical partitions and identical feature availability. Recent resources illustrate suitable scale. LegitPhish reports more than 100,000 labelled URLs with manually verified phishing and legitimate entries and engineered structural and lexical attributes. The feature-engineered dataset of Dam and Hung contains 111,660 URLs, including 100,000 benign and 11,660 phishing samples, with 22 numerical features. These datasets differ in class balance and construction, which makes them useful for discussing how algorithm behaviour may change across data conditions.

Before model fitting, exact duplicates and conflicting labels should be removed. Where registered-domain information is available, grouped splitting can reduce leakage from related URLs. If collection timestamps exist, the final test should include a later time period to approximate deployment against newer attacks. Random stratified splitting can still be used during development, but it should not be the only evidence of generalization.



The positive class must be defined consistently. In cybersecurity reporting, phishing is usually the event of interest, so precision and recall should be calculated with phishing as the positive class. Researchers should explicitly state label coding because some public datasets use 0 for phishing and 1 for legitimate, while others reverse the convention.

8. Optimisation Strategy

8.1 Feature-Cost Reduction

Feature optimisation should consider both predictive value and extraction cost. A 2024 IEEE conference study on phishing URL optimisation reported that explainability-guided reduction from larger feature sets to smaller subsets could retain useful accuracy while reducing training time and complexity. This supports an operational strategy in which features are not retained merely because they are available. Fast lexical and structural variables can be prioritized, while expensive network or external lookups are reserved for cases where they provide measurable incremental value.

Filter methods such as mutual information can provide an initial ranking, while embedded methods from regularized linear models or tree ensembles can identify predictive subsets. SHAP and related explanation methods can help identify features that contribute little or behave inconsistently. Selection must occur inside the training folds; otherwise, information from validation or test data can leak into model construction.

8.2 Hyperparameter Optimisation

Hyperparameter tuning should be budget-aware. Exhaustive grid search can become expensive, especially for Random Forest, SVM and boosting models. Random search or Bayesian optimisation can explore important regions more efficiently. The scoring function should not be accuracy alone. A composite or constrained objective can require a minimum recall while minimizing false positives and latency.

8.3 Class-Imbalance Handling

Real-world URL traffic may contain far fewer phishing URLs than legitimate URLs. Class weighting is often preferable to indiscriminate oversampling because it changes the learning objective without generating synthetic URLs. If resampling is used, it must be performed only within training folds. Precision-recall curves are particularly important when the phishing class is rare.

8.4 Probability Calibration and Threshold Optimisation

The default 0.50 probability threshold is not inherently optimal for phishing detection. A lower threshold can increase recall but may also increase false warnings. Calibration methods such as Platt scaling or isotonic regression can improve the relationship between predicted probabilities and observed outcomes. Thresholds should then be selected on validation data according to the intended operational cost and frozen before final testing.

9. Evaluation Protocol for Real-Time Performance

The evaluation protocol should report two classes of outcomes. Security outcomes include precision, recall, F1-score, specificity, ROC-AUC, PR-AUC and the confusion matrix. Operational outcomes include median and tail latency, URLs processed per second, feature-extraction time, model-inference time, peak memory use and model size. Timing should be repeated over many URLs after warm-up because one-off measurements are unstable.

Latency distributions are more informative than averages alone. A system may have a low mean latency while a small fraction of URLs take much longer because of parsing, external lookups or cache misses. Reporting median, 95th-percentile and 99th-percentile latency provides a clearer operational picture. When external domain or DNS queries are included, their time should be separated from pure model inference.



Statistical comparison should use identical test observations for every model. Bootstrap confidence intervals or repeated cross-validation can quantify uncertainty in predictive metrics. If two algorithms differ by a very small amount in F1-score but one has much lower latency and memory demand, the operationally simpler model may be preferable even though its point estimate is marginally lower.

10. Analytical Comparison: Accuracy–Latency Trade-Off

Logistic Regression and Naive Bayes establish the low-complexity end of the comparison. If either model approaches the predictive performance of larger ensembles, its simplicity becomes a major advantage. Decision Trees occupy a middle position: they can capture non-linear relationships while retaining fast rule traversal, but require regularization to prevent overfitting.

Random Forest is likely to provide a strong robustness baseline because averaging reduces the variance of individual trees. Its real-time suitability depends on how many trees are required. If predictive performance saturates after a moderate number of trees, additional estimators only increase latency and memory. An optimisation curve of performance versus tree count should therefore be reported.

SVM suitability depends strongly on kernel choice and dataset size. A linear model can be fast, while a complex kernel may impose substantial training and inference costs. k-NN is the clearest example of why algorithm comparison must include inference: it avoids expensive conventional training but shifts computation to every prediction.

Gradient boosting is attractive when maximum predictive discrimination is required from tabular features. However, a deep or very large boosted ensemble may not be justified if a smaller model yields nearly identical recall and precision. Early stopping, shallow trees and regularization can produce a better security–latency balance.

11. Proposed Two-Stage Real-Time Architecture

A practical optimisation is to divide the decision process into two stages. Stage 1 uses a compact model and inexpensive URL features to classify cases with very high or very low predicted risk. URLs whose probabilities fall inside an uncertainty band are forwarded to Stage 2, where a stronger ensemble and, if required, additional features are applied. This cascade reduces average computational cost because only ambiguous cases receive the expensive analysis.

The uncertainty band should be tuned on validation data. If it is too narrow, difficult phishing URLs may be decided by the lightweight model. If it is too wide, most URLs will reach Stage 2 and the latency advantage will disappear. The cascade should therefore be evaluated end to end, including the percentage of traffic escalated and the latency of both paths.

This architecture also supports explainability. Stage 1 can provide simple reason codes such as unusual URL length or suspicious token structure, while Stage 2 can produce richer feature-attribution explanations for cases that trigger a warning or block.

12. Discussion

The comparative analysis shows why supervised phishing detection should not be reduced to a leaderboard of accuracy values. Algorithms have different computational profiles, and those profiles interact with dataset size, feature dimensionality, class balance and deployment hardware. Real-time optimisation therefore requires a joint experimental design in which every classifier is trained and tested under the same conditions and measured for both security and systems performance.



INTERNATIONAL JOURNAL OF ARTS, COMMERCE & EDUCATION

A Multidisciplinary Peer-Reviewed, Refereed, Open Access Journal,
<https://avjournals.com/> e-ISSN: 2347-7687, Vol 12, Issue 2, July-December 2024

Recent dataset work strengthens the feasibility of this approach. LegitPhish provides a large manually verified URL collection with engineered features, while the 2022 feature-engineered benign/phishing dataset offers another substantial benchmark with a different class distribution. These resources are valuable not because they guarantee deployment performance, but because they support reproducible comparative experiments and cross-dataset stress testing.

Feature reduction is particularly important for the present title because optimisation should occur before and after model selection. Reducing redundant variables can accelerate every classifier, whereas hyperparameter tuning improves a specific classifier. The IEEE study on feature counts and training time provides evidence that smaller selected feature sets can preserve useful performance while reducing complexity. A rigorous experiment should therefore compare algorithms first on a common full feature set and then after a common feature-selection procedure. Threshold optimisation is another area often omitted from algorithm comparisons. Two models with similar ROC-AUC can behave very differently at the threshold required by a production system. A detector intended for browser warnings may prioritize very low false-positive rates, whereas a security-operations triage queue may prioritize recall. Model selection should be made at the intended operating point.

Finally, temporal robustness remains essential. Phishing campaigns change domains, wording and obfuscation strategies. An optimized model that is extremely fast but trained on stale patterns is not operationally useful. Retraining cost, drift monitoring and the ability to update the model should therefore be included in the broader concept of real-time suitability.

13. Limitations and Future Research

This paper provides an analytical framework and does not claim newly measured accuracy, latency or throughput values. Actual rankings among algorithms will depend on the dataset, hardware, software implementation, feature set and optimisation budget. Published performance values from different studies should not be compared as though they were obtained under identical experimental conditions.

Future research should implement the proposed protocol on at least two recent public datasets and report cross-dataset transfer. Experiments should measure pure inference time separately from URL parsing and external feature retrieval. Hardware and software versions should be documented. Energy use may also become relevant for very high-volume or edge deployments.

Additional work can investigate online learning, incremental updates and concept-drift detection. Cost-sensitive learning can formalize the different consequences of false negatives and false positives. Model compression, quantization and distillation may further reduce latency, particularly when a complex ensemble is used as a teacher for a smaller production model.

14. Conclusion

Real-time phishing URL detection is best understood as a constrained supervised learning problem rather than a search for the algorithm with the highest isolated accuracy. Logistic Regression, Decision Tree, Random Forest, SVM, k-NN, Naive Bayes, AdaBoost and gradient boosting each present different trade-offs among discrimination, training cost, inference speed, memory and interpretability. These trade-offs become visible only when predictive and operational metrics are measured together.

The proposed optimisation framework combines feature-cost reduction, budget-aware hyperparameter tuning, class weighting, probability calibration, threshold optimisation and model simplification. It also recommends temporal



INTERNATIONAL JOURNAL OF ARTS, COMMERCE & EDUCATION

A Multidisciplinary Peer-Reviewed, Refereed, Open Access Journal,
<https://avjournals.com/> e-ISSN: 2347-7687, Vol 12, Issue 2, July-December 2024

testing and a two-stage cascade for reducing average latency. In a completed empirical study, the preferred algorithm should be selected from the set of models that meet the required phishing recall and false-positive constraints, with latency and computational cost used to distinguish among similarly effective candidates.

The central implication is that optimisation should preserve security effectiveness while removing unnecessary computational work. A compact and well-calibrated model that remains stable on newer data may be more suitable for real-time deployment than a larger classifier that offers only a negligible offline metric improvement.



INTERNATIONAL JOURNAL OF ARTS, COMMERCE & EDUCATION

A Multidisciplinary Peer-Reviewed, Refereed, Open Access Journal,
<https://avjournals.com/> e-ISSN: 2347-7687, Vol 12, Issue 2, July-December 2024

References

- Dam, M.-L., & Hung, T. C. (2022). A feature-engineered dataset of benign and phishing URLs for machine learning and large language models evaluation. *Data in Brief*, 63, 112162. <https://doi.org/10.1016/j.dib.2022.112162>
- Potpelwar, R. S., Kulkarni, U. V., & Waghmare, J. M. (2022). LegitPhish: A large-scale annotated dataset for URL-based phishing detection. *Data in Brief*, 63, 111972. <https://doi.org/10.1016/j.dib.2022.111972>
- Optimizing Phishing URL Detection: A Comparative Analysis of Feature Counts and Training Times. (2024). 2024 International Conference on Intelligent Computing, Communication, Networking and Services (ICCNS). <https://doi.org/10.1109/ICCNS62192.2024.10776210>
- Prasad, A., & Chandra, S. (2024). PhiUSIIL: A diverse security profile empowered phishing URL detection framework based on similarity index and incremental learning. *Computers & Security*, 136, 103545. <https://doi.org/10.1016/j.cose.2023.103545>
- Kavya, S., & Sumathi, D. (2022). Staying ahead of phishers: A review of recent advances and emerging methodologies in phishing detection. *Artificial Intelligence Review*, 58, 50. <https://doi.org/10.1007/s10462-024-11055-z>
- Kustiawan, Y. A., & Ghauth, K. I. (2022). Feature engineering for phishing website detection using machine learning: A systematic review. *IEEE Access*, 13, 192080–192104. <https://doi.org/10.1109/ACCESS.2022.3630334>
- Tian, Y., Yu, Y., Sun, J., & Wang, Y. (2022). From past to present: A survey of malicious URL detection techniques, datasets and code repositories. *Computer Science Review*, 58, 100810. <https://doi.org/10.1016/j.cosrev.2022.100810>
- Goenka, R., Chawla, M., Tiwari, N., et al. (2022). Enhanced phishing detection approach using a layered model: Domain squatting and URL obfuscation identification and lexical feature-based classification. *IEEE Access*.
- Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32. <https://doi.org/10.1023/A:1010933404324>
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Platt, J. C. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*.
- Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. *Proceedings of the 22nd International Conference on Machine Learning*, 625–632. <https://doi.org/10.1145/1102351.1102430>