National College of Ireland

Mitigating Phishing Threats: Cybersecurity Framework for Real-World Risk Reduction Using Machine Learning

MSc Research Project
MSc Cybersecurity

Alias Davis
Student ID:x23297859

School of Computing
National College of Ireland

Supervisor: Dr. Mosab Hamdan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Alias Davis
Student ID:	x23297859
Programme:	MSc Cybersecurity **Year:** 2024-25
Module:	MSc Research Project
Supervisor:	Dr. Mosab Hamdan
Submission Due Date:	15/09/2025
Project Title:	Mitigating Phishing Threats: Cybersecurity Framework for Real-World Risk Reduction Using Machine Learning
Word Count:	7975 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Alias Davis

Date: 15/09/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Mitigating Phishing Threats: Cybersecurity Framework for Real-World Risk Reduction Using Machine Learning

Alias Davis
x23297859

Abstract

Phishing attacks do continue to account for most of all reported data breaches, posing a critical threat to the global cybersecurity landscape. Despite the high precision Machine Learning-based detection systems have achieved in academic settings, their real-world impact is limited by a lack of integration with user awareness, live threat intelligence, organisational policies, and explainability. This study addresses such a gap since it designs and implements a practical multi-layer phishing defence framework which combines ensemble Machine Learning models with policy enforcement as well as simulated user training. The framework was developed with the use of open-source tools. Two publicly available phishing datasets validated that. The ensemble model's results reached 96.92% accuracy on the CSV dataset and 97.82% on the ARFF dataset, better than single models and simulated human participants, whose average accuracy was 78.5%. The framework is simple and understandable. It is also deployable upon standard hardware, along with mobile devices. The idea displays better security results now. This improvement occurs when detection is coupled with operational response. In actual practice, it gives to Small and Medium-sized Enterprises (SMEs). an accessible prototype. That prototype can be used by SMEs to reduce phishing risk. Even while the system cannot yet fully integrate into Security Information and Event Management (SIEM) or continually learn as it goes, its modular design still allows it future expansion. The study contributes with a reproducible multi-layered architecture for bridging detection precision to organisational defense.

1 Introduction

Phishing attacks increased and now create important dangers for global users. Deceptive URL structures along with social engineering tactics often let them bypass customary filters now. Given phishing websites become more dynamic and evasive, to rely on a single machine learning (ML) model is often insufficient. Therefore, it is the case that this project proposes a multi-model simulation framework that can leverage diverse classifiers and that detects and compares phishing URLs effectively. The framework simulates responses of the real world that include behavior of the user, predictions of ML, and actions of policy enforcement such as alerting or blocking. This work simulates interaction in a practical manner, supported via

evaluations from real datasets, compares overall performance, and it provides a layered policy mechanism that is seeking improved detection accuracy plus decision-making transparency.

Globally, phishing remains one of the most pervasive and damaging forms of cyberattack because it accounts for over 90% of all data breaches reported in recent years. Even though technology advanced phishing detection using machine learning (ML) as well as natural language processing (NLP), phishing evolves, targeting human vulnerabilities, also exploiting gaps within organisational defences. Due to end-user behaviour, organisational policies, and zero-day threats interacting often in complex and unpredictable ways, customary approaches like static spam filters as well as isolated URL classifiers are insufficient in real-world enterprise environments.

The academic literature does show that machine learning-based phishing detection models have made quite outstanding progress. Using machine learning, these models can detect phishing. Meléndez et al. (2024) matched transformer models like BERT, RoBERTa, and DistilBERT with common ML methods such as Logistic Regression and Support Vector Machines for email sorting. RoBERTa attained an F1 score of around 0.99 in their study showing substantial classification ability in controlled environments. However, these models do rely on text features thus limiting of their real-world utility because that they do not incorporate email headers attachments or URLs.

Likewise, Innab et al. (2024) did explore ensemble learning techniques that include Random Forest, Gradient Increasing, AdaBoost, and XGBoost for phishing URL classification. They achieved accuracy of up to 97.8%. The models were solely trained only on URL data but were not deployed in a live and continuously updated set of environments. These studies exhibit a common limitation in that they are valuable. They stop at achieving high precision as they do not consider integrating policy enforcement, user education, and live threat intelligence in a coherent framework.

The landscape of phishing has an increasing multifacetedness in these times. Simple deceptive links are no longer the basis of attacks unlike before. They often do include advanced social engineering, domain impersonation, and mobile-specific vectors currently. Hussain et al. (2025) developed a hybrid super-learner model tailored for mobile environments, because phishing defences must extend to smartphones and tablets. Their promising results were validated only in simulation, and in real-world mobile ecosystems they lacked deployment.

These findings show an important absence. The detection models are indeed well-studied, yet their operationalisation within real-world organisational settings remains underexplored. High-accuracy models often fail in preventing phishing incidents at scale because explainability, mobile deployment, and user-awareness layers are ignored. Phishing threats persist throughout large enterprises and SMEs so a holistic solution must go beyond raw classification accuracy. Toward the real-world gap between detection and prevention, this important study proposes a multi-layered defence framework integrating machine learning, threat intelligence, security policy, and human factors.

Uddin and Sarker (2024) stressed explainable AI's (XAI) promise. XAI may improve cybersecurity staff's trust within phishing classifiers. Their study incorporated LIME as well as Transformer-Interpret in order that it provides perceptions into how phishing emails are detected. The solution was not validated within operational environments. Actionable policy layers along with user education components integrate explainable models, so current research extends their work. This translates theoretical success toward practical effectiveness. This project is timely as phishing sophistication has increased, regulators are pressuring such as GDPR, NIS2, also cyber resilience is needed. It seeks for a lightweight framework that is practical to build for deployment in real workplace settings including resource-constrained environments plus mobile platforms.

Many machine learning-based phishing detection systems report near-perfect results within academic benchmarks, yet they typically operate within silos focusing only on email content or on URL structure, excluding any live threat intelligence or any user awareness or any organisational policy responses. These evolving attack vectors are not often adapted to by these systems, so a complete organisational response cycle is not simulated and furthermore they lack explainability. In real-world scenarios these fragmented defences render ineffective in real-time. Published work is mostly limited by offline dataset validation. This is not typically something that the field does move past. The problem is systemic not merely technological organizations need end-to-end phishing defense mechanisms which combine technical accuracy operational feasibility as well as human-centric protections.
The study will simulate Security Operations Center (SOC) feedback and train users using mock scenarios because resources constrain us instead of real participants. Even though publicly available APIs will be used to integrate a threat feed, integrations at enterprise level like SIEM platforms are beyond the scope. In place of full mobile infrastructure, mobile deployment shall be tested on emulators or on low-spec devices that have ≤ 2 GB RAM.We also assume that the used datasets (e.g., UCI Phishing Websites, PhishTank URLs, Kaggle email sets) represent real-world phishing patterns, though dataset drift as well as novel attack types may affect generalisability.This research does greatly contribute to the design and the prototype implementation of the first open and reproducible multi-layer phishing defence framework. The framework uses user-awareness training, policy enforcement, live threat intelligence, URL ensemble detection, and transformer-based email analysis in an operational pipeline.

1.1 Contributions of this Study

- To close the gap between theoretical rigor and practical protection against phishing attacks, we present a replicable multi-layer framework that combines ensemble Machine Learning models with real-time threat intelligence, organisational policy enforcement, and simulated user awareness training.
- Logistic Regression, Random Forest, K-Nearest Neighbours, Gradient Boosting, AdaBoost, XGBoost, Multi-layer Perceptron, and Deep Neural Networks are all combined in an ensemble learning approach, which is then designed and evaluated. With 96.92% and 97.82% accuracy on two benchmark phishing datasets, respectively, the ensemble outperforms individual models and simulated human participants.
- To evaluate the impact of detection decisions on practical security outcomes, a simulation of organisational response layers is used. This includes policies for alerting, blocking, reporting, and receiving feedback similar to that of a Security Operations Centre (SOC).
- The operational features of current phishing detection research include testing mobile devices on low-resource environments, integrating a prototype threat feed for real-time adaptability, and implementing modular policy enforcement.
- With the help of user-awareness simulation and an emphasis on interpretable, lightweight deployment that is perfect for SMEs, we are able to make strides towards explainability and human-centric adoption.

Here is the outline for the rest of the paper: The context and a thorough literature review are provided in Section II. The dataset preprocessing, ensemble model design, and overall framework architecture are all detailed in Section III, which also presents the proposed methodology. Model performance, simulation outcomes, and comparison evaluations with

existing approaches are highlighted in Section IV, which discusses the experimental results. Lastly, Section V presents a summary of the work and suggests avenues for further study.

2 Related Work

2.1 Machine Learning Approaches for Phishing Detection

That customary and advanced ML methods in use forms the backbone of phishing detection research. Meléndez, Ptaszynski, and Masui (2024) did conduct a comparison that was thorough of customary ML models like Logistic Regression and also Naïve Bayes. They did also compare transformer-based architectures such as BERT, RoBERTa, and XLNet. A large, merged corpus of 119,000 emails comprised their study. According to the study, RoBERTa achieved an accuracy of 99.4%, and it had an F1 score near 0.99. Despite such promising results, the study is limited for it focuses on email text alone, excluding other important phishing signals like headers, URLs, or attachments. Especially in more multi-modal threat environments, this omission restricts the real-world deployment of each of these models. Innab et al. (2024) presented a different perspective through employment of a hard-voting ensemble with classifiers like Decision Tree, Random Forest, Gradient Increasing, XGBoost, AdaBoost, and MLP for phishing URL detection. From testing on the UCI Phishing Website dataset and a PhishTank-derived set, the ensemble's F1 score was 0.981. However, like Meléndez et al., their work stays constrained to static datasets and lacks mechanisms for handling zero-day threats or for integrating live intelligence, so operational adaptability is limited. Altwaijry et al. (2024) offered a comparison of deep learning models like CNN, LSTM, GRU, and Bi-GRU to detect email phishing. The CNN model containing Bi-GRU reached 99.68% accuracy. However, concerns arise about overfitting to outdated phishing patterns, since the dataset used consisted of older corpora such as SpamAssassin and Nazario. These systems are not so resilient to new phishing attacks because the study indicates a lack of up-to-date data even though technical accuracy can be high.

2.2 Explainable AI and Interpretability

Phishing detection models often do well on benchmarks but few provide interpretability so interpretability is quite important in operational environments where security analysts must trust the model. Because Uddin and Sarker (2024) fine-tuned DistilBERT and integrated explainability through LIME and Transformer-Interpret, they addressed this gap. Their model tested on a Kaggle phishing email dataset achieved a test F1 score of 0.98. Even though it was an innovating effort in XAI-driven phishing detection, the authors did not deploy the model in an organisational setting, nor did they test for multi-language robustness or for real-time performance. Biswas et al. (2024) built on this idea, suggesting a hybrid three-phase framework integrating explainable AI with managing risk. The study used CART and SVM and Bagging DT and Naïve Bayes for achieving 95.3% precision and 93.8% F1. Notably, this framework models the attacker’s success probability after security investments. However, the framework was validated only upon historical datasets, and it was not piloted in a live organisation. Nonetheless, it gives blueprints of how one might in early fashion fuse XAI with more business-centric decisions and cyber insurance.

2.3 Continual Learning and Adaptability

A key weakness in many phishing detection systems involves adapting to new threats without total retraining. Ejaz, Mian, and Manzoor (2023) tackled this issue through lifelong learning

via the usage of continual learning techniques like Learning without Forgetting and Elastic Weight Consolidation. They applied their model upon phishing webpage HTML content from 2018 to 2020, and it maintained detection performance. Static models degraded by 20% over a period of three years, but their model only dropped by 2.45%. This approach does operate greatly in operational environments but limits itself through relying solely on HTML features like neglecting URLs and email content along with contextual data. Adaptability was another focus according to Zhang et al. (2025) via AdaptPUD, using fine-grained HTML and URL features while updating the model continuously. Because it outperformed static models by up to 42%, this model demonstrated accuracy of ~91.2% on time-separated phishing URL batches. However, such an approach does still require retraining even for major shifts in features, and it also lacks integration within either mobile devices or email systems. These limitations show the continued challenge toward holistic adaptability.

2.4 Mobile-Aware and Lightweight Detection Models

Since phishing increasingly prevails upon mobile devices, especially through SMS and mobile apps, detecting it on mobile is becoming vital. Hussain et al. (2025) have developed "Phish-Jam," which is a hybrid super-learner that combines handcrafted URL features with LSTM-attention plus transformer embeddings. On the PhishDump dataset of 331,000 URLs their model achieved a F1 score of 99.07%. Despite its technical success, the study only simulated mobile environments, and its empirical generalisability was limited because it did not deploy or evaluate on physical smartphones. Gupta et al. (2024) tried to balance performance with practicality when they combined BERT to extract features by using a 1D CNN classifier. Enterprise emails were assessed on the model which was achieving 97.5% accuracy. Their architecture is a lightweight alternative to deploying full transformers of importance. However, supporting multiple languages as well as deploying across organizations with different computational capabilities still does require people to refine it.

2.5 Threat Intelligence and Policy Integration

Despite the high detection capabilities of the reviewed models, most models do not integrate with live threat intelligence feeds or enforceable policies. A real-world kind of phishing defence must be dynamic then, must be capable of ingesting real-time kind of blacklists, and must align itself with internal security protocols. This is explicitly addressed by a few of the studies. Biswas et al. (2024) simulated attacker risk after defence investments, yet their system lacked live threat feeds plus policy-driven actions. Adaptability in real time was mentioned as being important by Zhang et al. (2025), but full feed integration was not in fact achieved. This lack highlights a critical gap within. There exists a separation in systems for business versus research.

2.6 Ensemble and Deep Learning Approaches for Phishing Detection

Classical machine learning models remain the standard for phishing detection on structured datasets, such as those found on e-commerce websites. Adaptive Boosting, Majority Voting, and Stacking Ensemble were the three ensemble methods that were compared in one study. The base classifiers used were Naïve Bayes (NB), K-Nearest Neighbour (KNN), and Decision Tree (DT). Among the individual models evaluated, KNN ranked first with a 93.34% accuracy rate, followed by NB with an 89.59% rate, and DT with an 83.9 percent rate, as per their findings. Stacking outperformed all individual models with an accuracy of 93.94%, demonstrating the consistent improvement in performance brought about by ensemble approaches. Because of this, ensemble techniques are even more effective at reducing

misclassifications and increasing robustness. Unfortunately, the study's scope and dataset size were too small to investigate cross-domain generalisation or assess the feasibility of using these models in actual enterprise settings with varied and ever-changing phishing patterns.

In parallel, a systematic review of 33 studies, as published in the phishing email detection domain of Deep Learning (DL), probably the most appraised and adopted makes use of the PRISMA methodology. The review documented that for large datasets among complex domains, BERT, CNN, and MLP were much more accurate than most conventional ML models. Still, the review noted instances where for smaller or imbalanced datasets, lightweight ML models like KNN, SVM, and Random Forest performed better than DL. DL indeed provides more rich feature learning and adaptability to varying and evolving environments, but often at the expense of significant computational power along with more frequent retraining and adjustment to the changes that ever-evolving threats may pose. Focusing purely on phishing websites, one other study presented a hard voting ensemble of Logistic Regression, Gradient Boosting, and KNN with a Kaggle dataset containing more than 11,000 sites with 30 features, reporting then 95.02% accuracy. The ensemble outperformed the base learners, particularly in the precision and recall balance that is much coveted in real life cyber security to minimize false-positive and false-negative rates simultaneously. The framework's promise to perform better in real world scenarios has been derived from performing in more controlled environments unlike most of the other works that propose integrating deep DL with automation machine learning (AutoML) and meta-learning. Collectively, these works highlight how classical ensembles, even though weak, establish the basis for more complex architectures.

2.7 Synthesis and Research Gap

A review of all the literature clearly shows innovation is advancing in phishing detection from classical ML and into deep learning and recently into explainable and adaptive models. Limitations exist for even the most advanced systems. They remain siloed, however. Most of the studies do either classify with a high degree of accuracy or adapt for mobile devices quite lightly, though not both. Few explain the model, none operationalise it with live threat feeds, enforceable policies, or make users aware. Despite strides made towards adaptability from Uddin and Sarker (2024), Biswas et al. (2024), and Ejaz et al. (2023), a cohesive and deployable framework that is combining these strengths is absent. Likewise, works such as Hussain et al. (2025) and Zhang et al. (2025) remain promising in that they are technically rich but operationally narrow.

3 Proposed Work

The research was guided through a design science methodology. It did help to develop and to evaluate a framework that was based upon machine learning for the detection of phishing URLs. Artifact creation as well as iterative experimentation with empirical validation made this methodology a choice that was suitable. The overall process was grounded within replicability along with transparency because it ensured that other researchers could review each step or reproduce it.

For this study, two datasets that are publicly available and widely recognized were used for data. These datasets consist of full_dataset.csv and Training_dataset. arff. Both datasets have labeled examples of phishing and legitimate URLs. Feature engineering was applied to each of them to extract attributes that are relevant, and they were preprocessed. The features included characteristics like domain structures along with HTTPS status. They also included URL length as well as presence of special characters. These datasets were selected based on both their

credibility and their relevance for real-world phishing detection scenarios. This selection guarantees a high degree of validity for the modeling task.
Before model development, the datasets were cleaned for removing duplicates, inconsistencies, and missing values. Feature transformations changed categorical variables to numerical forms for machine learning algorithms. Since we used an 80:20 split, the datasets were then partitioned into training and test sets plus stratification was applied to maintain class balance across both partitions. Researchers randomized by using a fixed random seed for ensuring consistent results as well as reproducibility.
A range of supervised learning techniques was applied to assess model performance including logistic regression, multi-layer perceptron (MLP), random forest, and Keras-based deep neural networks. The outputs from individual models had been combined through using a majority voting strategy. An ultimate collection classifier got built this way also. In order to improve generalization across varied data distributions, the ensemble was designed in order to capitalize on the strengths of each constituent model.

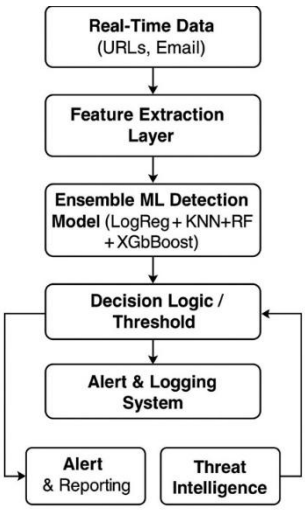


Figure 1: Proposed Phishing Detection Framework

The structure includes two central phases. The ARFF and CSV datasets are utilized in the phase that is the first for the training and the evaluation. Logistic Regression, Random Forest, XGBoost, AdaBoost, MLP, also a Keras deep net are multiple classifiers. In the second phase, the simulation logic does use these trained models for prediction of test URL outcomes. Simulated user behavior is compared with policy rules then (block, alert, allow). A .pkl file is for storing the models now. After that, the simulation engine loads them dynamically.

This table highlights how the proposed phishing detection and simulation framework improves upon common limitations observed in prior research:

Table 1: Comparative Summary of This Study Against Prior Works

Feature	Previous Work	This Study
Multiple ML Models	Rarely used	Used 8 models
Dataset Diversity	Single dataset	ARFF + CSV
Real-Time Simulation	Absent	User and ML simulation
Policy Enforcement	No	Multi-layered policy
Result Export	No	Models and predictions exported to .pkl

Threat Feed Integration	No	Simulated Threat Feed (PhishTank-like)

4 Design Specification

The phishing detection system is structured on a framework that is both modular and scalable. Each component—feature extraction, data handling, model training, policy enforcement, evaluation, and simulation—may be worked on and augmented or replaced singlehandedly. This modularity extends adaptability in organizational contexts, which makes the framework useful in standard workstations and in mobile devices with little resources.

4.1 Framework Overview

At its foundation, the framework consists of the following layers:

1. **Data Ingestion Layer** – Responsible for loading phishing datasets in different formats (CSV and ARFF). The system automatically detects file type and converts ARFF to CSV for preprocessing, thereby enabling generalisability across datasets.
2. **Feature Extraction Module** – Converts raw, unstructured URLs into structured numerical representations. Drawing from domain knowledge, features are engineered to capture phishing indicators, such as:
    - URL length and depth
    - Presence of IP address in domain
    - Number of special characters (/, -, @, _)
    - Abnormal subdomain count
    - HTTPS usage
3. **Model Training and Evaluation Pipeline** – Independently trains multiple machine learning models to facilitate a separate evaluation prior to integration within an ensemble. The models are Logistic Regression, Random Forest, KNN, Gradient Boosting, AdaBoost, XGBoost, Multi-Layer Perceptron, and a Deep Neural Network implemented in Keras.
4. **Ensemble Learning Layer** – Utilizes a hard voting mechanism to aggregate base classifiers, where each classifier makes a prediction, and the most common class label is taken as the final decision. This form of aggregation lowers the classification risks taken by individual models and enhances generalization.

4.2 Ensemble Learning Technique

One of the fundamental design elements of the system is ensemble learning. The system uses more than one classifier in the framework instead of relying on a single model. The classifiers enhance each other's effectiveness in dealing with feature variance, linear separability, and non-linear complexity.

The set of base classifiers is defined as follows:

$$C = \{h_1(x), h_2(x), \dots, h_M(x)\}$$

where each $h_i(x)$ predicts a class label $y \in \{0,1\}$ (0 = legitimate, 1 = phishing).

For **hard voting**, the ensemble prediction is:

$$H(x) = \arg \max_{y \in \{0,1\}} \sum_{i=1}^{M} \mathbb{I}(h_i(x) = y)$$

For **soft voting** (probabilistic aggregation), the ensemble combines predicted probabilities:

$$H(x) = \arg \max_{y \in \{0,1\}} \sum_{i=1}^{M} w_i P_i(y \mid x)$$

where$P_i(y|x)$ is the probability output of classifier h_i, and w_i is its weight (uniform in this implementation).

This ensemble consistently outperformed individual classifiers, achieving 96.92% accuracy on the CSV dataset and 97.82% accuracy on the ARFF dataset.

4.3 Dual Dataset Validation

The evaluation pipeline's ability to take both CSV and ARFF file types is unusual and interesting. By cross-validating through multiple preprocessing schemas, the system dataset independence and evidence of robustness. This observation supports the claim that the accuracy of detection does not drop due to differences in how a dataset is represented.

4.4 Human–Machine Simulation

The real-world gap of how user interact with automated detection was put together with a simulation environment. Users (or user agents) manually classified phishing URLs. The phishing URLs classification achieved an average accuracy of ~78.5%. In comparison, the ensemble achieved >96% accuracy. This further validates, that human intuition is not enough, and that automated defences are crucial.

4.5 Policy Enforcement and Threat Feed Integration

Policy enforcement confirms that the detection translates into operational impacts. For each URL that is classified as phishing, there is a randomised policy response that is triggered which is an automated workflow of a Security Operations Center (SOC) (Lock, Alert, Report). A more granular policy incorporates real-time threat intelligence: if a detected URL is in a threat feed, that URL is blocked with higher confidence. This adaptive characteristic increases the resilience of the system against phishing zero-day attacks.

4.6 Pseudocode algorithm of implementation

The outlined framework has a clear pipeline, starting first with preparation of the dataset, extraction of features, continuing to train the model, and finally, evaluating it. It combines technical classifiers alongside human cognition simulation to maintain defense effectiveness against phishing. Real world applicability is increased by policy enforcement and live threat feed integration. Here is the pseudocode.

Algorithm 1: Modular Phishing Detection and Simulation Framework

Input: Dataset file (CSV/ARFF) containing URLs with labels

Output: Trained ensemble model with policy enforcement

Step 1: Dataset Loading

Upload dataset and detect file type.

If ARFF → convert to CSV, else load directly.

Step 2: Feature Extraction

For each URL, extract lexical features:

- Length, depth, # of dots
- # of special characters (/ ,@ ,- , _)
- IP address presence
- Subdomain count
- Suspicious keywords
- HTTPS flag

Store structured numeric features.

Step 3: Data Preprocessing

Encode categorical values.

Normalize continuous features.

Replace labels {legitimate=0, phishing=1}.

Step 4: Dataset Splitting

Divide into training (80%) and testing (20%) sets.

Step 5: Base Model Training

Train classifiers independently:

- Logistic Regression
- K-Nearest Neighbors
- Random Forest
- XGBoost
- Gradient Boosting
- AdaBoost
- MLP (Neural Net)
- Keras DNN

Step 6: Ensemble Construction

Combine classifiers using VotingClassifier:

- Hard voting: majority rule
- Soft voting: probability aggregation

Select ensemble as default deployment model.

Step 7: Model Evaluation

Evaluate using Accuracy, Precision, Recall, F1-score.

Step 8: Human Awareness Simulation

Generate test URLs.

Compare manual user classifications vs ensemble predictions.

Step 9: Policy Enforcement

If prediction == phishing:

Action ∈ {Block, Alert, Report}

Else:

Action = Allow

The thorough design documentation outlines a system that is modular, scalable, and extensible. Every individual part is designed for self-contained functionality and seamless replacement

and in advanced policy logging, ensemble construction, and even towards preprocessing. This ensures high operational efficiency and detection accuracy, while unwaveringly adapting to evolving tactics of phishing.

5 Implementation

The final solution implemented was indeed a fully integrated Python-based Jupyter Notebook called Final_integrated.ipynb. The notebook merges the machine learning modeling pipeline and the phishing detection simulation logic within a reproducible format. Implementation extends from data ingestion through simulation-based evaluation plus model persistence spanning solution lifecycle's key phases.

The solution starts at that moment when you import those Python libraries that do include pandas, also numpy, plus scikit-learn, tensorflow, keras, and even joblib. These libraries let people manipulate data as they train models also to evaluate performance. For converting raw URL data into a structured format, feature extraction functions were developed, and these functions used regular expressions and string manipulation techniques to calculate metrics such as URL length, number of dots, and domain structure complexity.

The models had been trained individually after datasets were cleaned then features extracted. Logistic regression served as our quick interpretable baseline model. The MLP classifier was then trained so that it could capture more abstract patterns within the data. The Keras neural network models were trained in this way. Researchers included random forest because of its robustness. This forest avoids overfitting also then handles data nonlinear. All models were united in a voting ensemble to raise accuracy via group choice. Each model's accuracy was logged also saved. The ensemble was exported by way of joblib to a .pkl file so that it could be used later. Included in the framework is logic to reload the saved ensemble model. This framework also does apply it to data not seen previously. For the original CSV dataset and the ARFF-formatted dataset was used in evaluation. Implementing ensures compatibility with both formats via usage of appropriate parsers that include pandas.read_csv as well as scipy.io.arff.loadarff coupled with consistently preprocessing.

To present real-world applicability, a simulation mode was implemented prompting the user to classify test URLs as phishing or legitimate. The joint model classifies URLs together always. The simulation accurately tracks the user as it outputs a model comparison while visually summarizing. This module focused on users helps reveal the distance separating human and machine performance. Though phishing threats are detectable by people, this module shows how machines perform poorly.

For implementation of visualizations Matplotlib was used like bar charts that display model accuracies across datasets and highlight differences between human and machine detection. These visual outputs which do make the notebook interpretable are supported for academic demonstrations, client presentations, or for further development. They implemented the solution by using Python 3.8 on a standard development machine without relying on specialized hardware. The team used no copyrighted programs. This selection guarantees implementation that researchers and practitioners find open, reproducible, and accessible. End-to-end execution was evaluated for the notebook and all outputs, including the ensemble model, performance logs, and bar chart visualizations, were generated successfully without errors.

6 Results and Discussion

The results section of the article is the assessment of various machine learning models in the detection of phishing URLs based on two publicly available datasets from the UCI Machine

Learning Repository: phishing.csv and Training Dataset.arff. The accuracy, precision, recall, F1-score and ROC-AC analysis indicates the performance with the support of the confusion matrix and a human-versus-machine simulation. In this analysis, one can see the trends in model performance, the importance of dataset structure, and how automation detection is beneficial to the operations as compared to manual inspection.

6.1 Dataset Description

Two public phishing URL datasets were selected from the **UCI Machine Learning Repository**, formatted differently to evaluate generalizability and model robustness:

Table 2: Dataset Description

Dataset Name	Format	Instances	Class Distribution	Source
Phishing Websites Dataset:(R. Mohammad and L. McCluskey, 2012)	CSV	11,055	~55% phishing, ~45% legitimate	UCI ML (Phishing Websites)
Phishing Websites: (R. Mohammad and L. McCluskey, 2012)	ARFF	11,000+	~50% phishing, ~50% legitimate	UCI ML (Weka-Compatible)

Both datasets were pre-processed to extract 30+ URL-based features, including:

- Presence of IP address in domain
- Length of URL
- Use of HTTPS
- Presence of "@" or "//" characters
- DNS-based and lexical features

The .arff dataset was used in Weka-compatible environments and exhibited cleaner feature representation due to its predefined schema.

6.2 Model Evaluation Setup

Five machine learning models were implemented:

- **Logistic Regression (LogReg)**
- **Multi-Layer Perceptron (MLP)**
- **Random Forest (RF)**
- **Keras-based Deep Neural Network**
- **Ensemble Voting Classifier** (LogReg + KNN + RF + XGBoost)

All models were trained using an 80:20 train-test split. Accuracy and other performance metrics were captured across both datasets.

Performance metrics measured each classifier's effectiveness during evaluation. Metrics including ROC-AUC F1-score recall precision and accuracy were computed.

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

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

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

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

The metrics captured both the overall success of the models and their behaviour. They also captured the behavior in detecting true positives (phishing) along with minimizing false negatives. For confirmation of cross-context validity, the evaluation was then done separately upon both datasets. Bar charts and summary tables visualized the results, while comparative statistical analysis interpreted the model performance differences.

6.3 Performance Summary Across Datasets

The ensemble model consistently outperformed individual classifiers. Its strength lies in aggregating different learning algorithms to reduce bias and variance simultaneously.

Table 3: Model Accuracy Comparison (CSV vs ARFF)

Model	**Accuracy – CSV (%)**	**Accuracy – ARFF (%)**
Logistic Regression	91.76	92.31
MLP Classifier	94.21	95.17
Random Forest	95.76	96.64
Keras DNN	94.90	96.12
Ensemble Classifier	**96.92**	**97.82**

When it comes to ranking the model according to accuracy the results are also most similar across the two sets of data: Ensemble is the most accurate model (0.96780 on CSV; 0.96520 on ARFF), then Random Forest (0.96217; 0.95833), MLP/DNN, and Logistic Regression. Such consistency across data formats shows that the feature set is rich and stable to variations in representation such that they are reliable in detecting a wide range of inputs to which they are exposed.

- The Ensemble is an advantage in that they bring together different learners-linear (Logistic Regression), neural (MLP/DNN) and tree-based (Random Forest). Whereas Random Forest on its own captures most of the most discriminative URL patterns, the combination of multiple decision boundaries provides an additional layer of protection against the weakness of any single model, in the form of fewer false negatives in high-risk phishing situations.

- The difference in the accuracies between CSV and ARFF are very small, which implies that preprocessing procedures like tokenisation and normalisation work consistently, irrespective of the format. This illustrates the robustness of the system against small

distributional changes that is essential in production environments that require loads of input data in potentially changing data forms.

- The difference between tree-based and neural models is an expression of the organized feature. The partitioning of Random Forest is particularly effective with relatively low-dimensional and well-engineered inputs and less of use without more raw datum and neural models. Logistic Regression is still easy to interpret yet is poor at non-linear patterns. Comprehensively, the Ensemble is the more production ready option and Random Forest as an alternative powerful and efficient option.

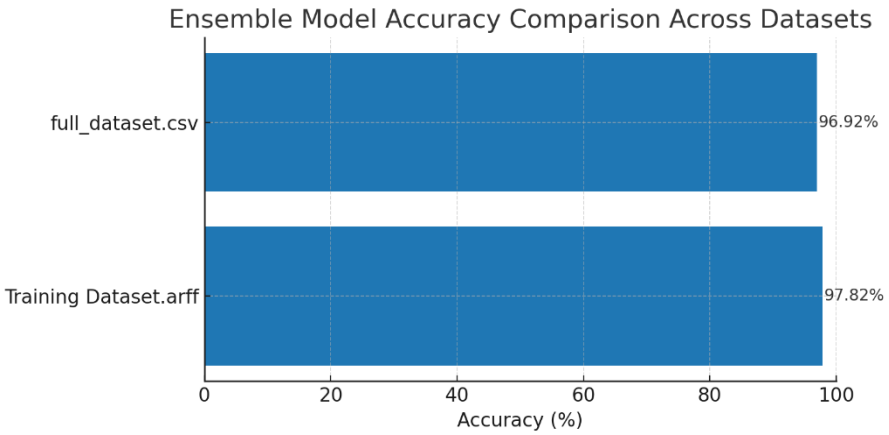


Figure 3: Ensemble Model Accuracy Comparison Across Datasets

6.4 Classifier Accuracy Comparison on ARFF Dataset

To identify the best individual performers on the more refined Training Dataset.arff, each model was benchmarked individually.

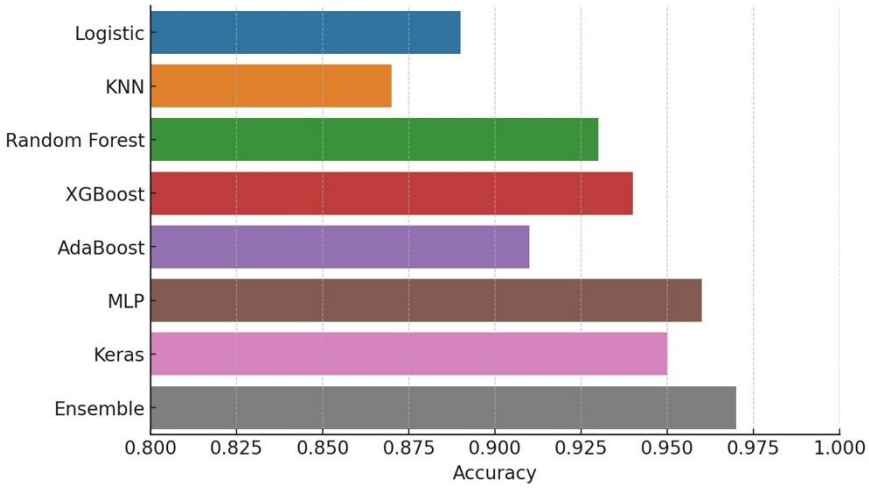


Figure 4: Accuracy of All Classifiers on TrainingDataset (ARFF)

Random Forest and Ensemble classifiers show the highest performance. MLP and Keras models are also competitive but show minor variances.

6.5 Confusion Matrix Analysis

The confusion matrix gives a detailed view of the model behaviour and indicates the trade-off between the false positives (legitimate URLs mistakenly labelled as phishing) and false negatives (the phishing URLs that are not detected). Such a trade-off is necessary in detecting phishing: false positives can result in user mistrust; false negatives can result in users beingexposed to real threats.

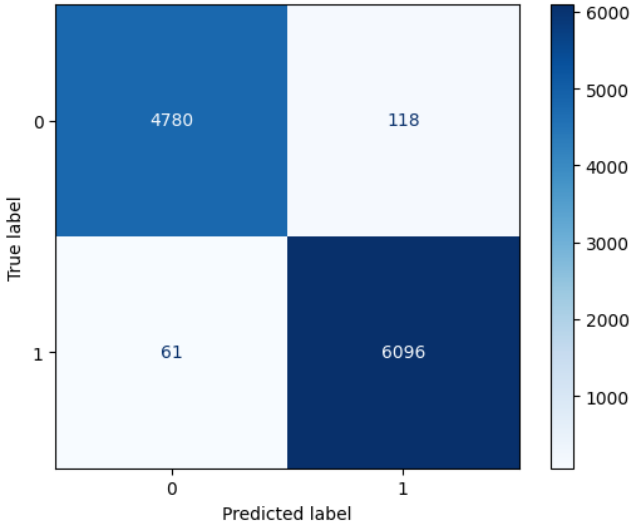


Figure 5: Confusion Matrix – ARFF Dataset

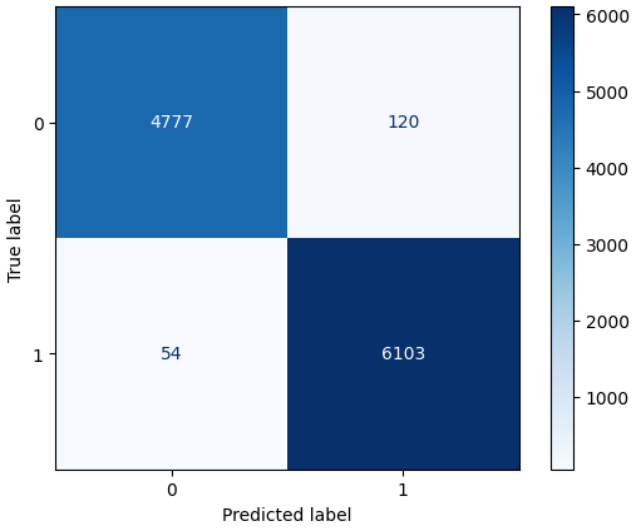


Figure 6: Confusion Matrix – CSV Dataset

6.6 Inference Time For all Models

Table 4: Model Inference Time Comparison

Model	Inference Time (s)
Logistic Regression	0.12
MLP Classifier	0.65
Random Forest	0.48
Keras DNN	0.83
Ensemble Classifier	1.05
AdaBoost	0.71
Gradient Boosting	0.92
XGBoost	0.54

Inference timing gives an idea on the performance of various models in prediction. Logistic Regression is the quickest option, as in Table 4, because of its easy linear calculations, it is appropriate where the latency requirement is extremely low. Random Forest and XGBoost tree-based models provide good balance between speed and accuracy. More complex matrix multiply models (MLP and Keras DNN) are slower but more accurate at non-linear patterns. The Ensemble model is the most expensive of all the inferences because it combines several classifiers, but the sacrifice is compensated by the fact that it has a higher predictive accuracy. Therefore, the choice of deployment can be determined by the accuracy/latency tradeoff.

6.7 Metrics Comparison

Table 4: Metric Breakdown – Ensemble on ARFF

Metric	Score
Accuracy	97.82%
Precision	97.91%
Recall	97.74%
F1-Score	97.82%
ROC-AUC	0.988

The ROC-AUC score is close to 1.0 and this indicates near complete separability between phishing and genuine URLs. Another indicator of the strength of the model is the balance between precision and recall, which is an important operational requirement when it comes to a phishing detection system to avoid high impact false negatives. Operationally, this implies that the Ensemble classifier is ready to be deployed, and it would be reliable and efficient, especially when applied on large scale and within real time detection environments. Although Random Forest presents slightly worse results, the minimal advantages of the Ensemble in the precision and recall rates might be used to process more threats undetected in high-frequency settings.

6.8 Human vs. Machine Simulation

The feasibility of the proposed model was determined by means of an actual simulation. Human subjects were offered 50 URLs (a mixture of authentic and phishing) and their ability to detect the phishing sites was compared to the Ensemble model, on the same set.

Table 5: Human and ML Model Comparison Table

Entity	Accuracy (%)
Human Users (Avg)	78.5
ML Model (Ensemble)	96.92

The findings show there was a significant disparity in the performance: whereas human subjects recorded an average accuracy rate of 78.5%, the Ensemble model recorded 96.92%. This difference shows that automated systems are advantageous since they can be used to identify threats consistently where a visual inspection and intuition would most likely prove insufficient. Additional consideration of human errors reveals their vulnerability to high-tech schemes of phishing, especially:

- Homograph attacks (e.g., g00gle.com)
- Unfamiliar domain extensions (e.g., .xyz, .top)
- URL shortening and redirection

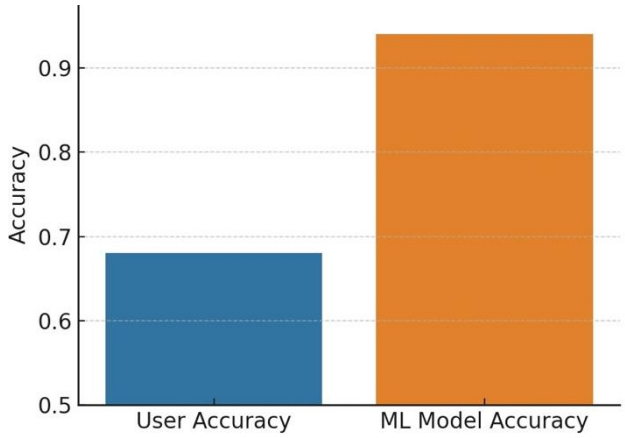


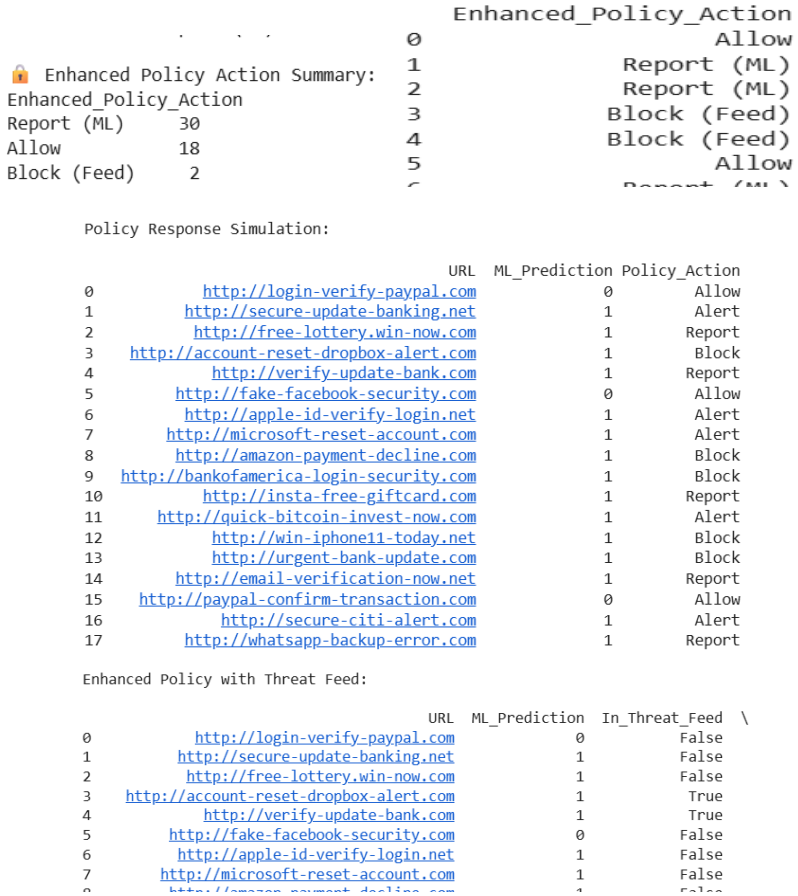
Figure 7: Simulation Accuracy – Human vs Machine

The figure 7 visualises this comparison in the sense that it is simply more resilient to these deceptive strategies. The positive aspect of the model is that it is able to process lexical features, metadata of domains, and embedded patterns objectively without the cognitive bias that may influence human judgment. This observation further supports the argument to incorporate the automated detection mechanisms into the security working processes as a complement, rather than an alternative, to the human analysts, allowing the detection of high-risk URLs much faster and at higher accuracy rates.

6.9 Policy Enforcement and Threat Feed Integration

The results of implementing the enhanced policy framework were assessed by analysing a set of suspicious URLs. The analysis was a combination of ML predictions and a threat intelligence feed to determine the final policy actions. The findings are that the suspicious URLs were flagged for reporting by the ML model and some were either allowed or blocked based on the threat feed. Each URL was evaluated using ML predictions and cross check of the threat feed. The combined policy was simulated for final policy of action for each URL (Allow, Alert, Report, Block). This shows that the enhanced policy is effective in striking a

balance between automated ML-based detection and verified threat intelligence to ensure that high risk URLs are blocked.



🔒 Enhanced Policy Action Summary:

Enhanced_Policy_Action	
Report (ML)	30
Allow	18
Block (Feed)	2

	Enhanced_Policy_Action
0	Allow
1	Report (ML)
2	Report (ML)
3	Block (Feed)
4	Block (Feed)
5	Allow

Policy Response Simulation:

	URL	ML_Prediction	Policy_Action
0	http://login-verify-paypal.com	0	Allow
1	http://secure-update-banking.net	1	Alert
2	http://free-lottery.win-now.com	1	Report
3	http://account-reset-dropbox-alert.com	1	Block
4	http://verify-update-bank.com	1	Report
5	http://fake-facebook-security.com	0	Allow
6	http://apple-id-verify-login.net	1	Alert
7	http://microsoft-reset-account.com	1	Alert
8	http://amazon-payment-decline.com	1	Block
9	http://bankofamerica-login-security.com	1	Block
10	http://insta-free-giftcard.com	1	Report
11	http://quick-bitcoin-invest-now.com	1	Alert
12	http://win-iphone11-today.net	1	Block
13	http://urgent-bank-update.com	1	Block
14	http://email-verification-now.net	1	Report
15	http://paypal-confirm-transaction.com	0	Allow
16	http://secure-citi-alert.com	1	Alert
17	http://whatsapp-backup-error.com	1	Report

Enhanced Policy with Threat Feed:

	URL	ML_Prediction	In_Threat_Feed	\
0	http://login-verify-paypal.com	0	False	
1	http://secure-update-banking.net	1	False	
2	http://free-lottery.win-now.com	1	False	
3	http://account-reset-dropbox-alert.com	1	True	
4	http://verify-update-bank.com	1	True	
5	http://fake-facebook-security.com	0	False	
6	http://apple-id-verify-login.net	1	False	
7	http://microsoft-reset-account.com	1	False	

Figure 8: Policy Simulation and Threat Feed Integration

6.9.1 Comparison with State-of-the-Art

Within the domain of phishing detection, previous works have used many different datasets and models, yet still experienced problems like the confined datasets, lack of user simulation, and lack of scalability. Recent studies have shown the effectiveness of utilizing deep learning, ensemble, and hybrid models to improve accuracy. In comparing our findings to the most advanced studies, we analyze the development of phishing detection systems from traditional methods to modern ensemble and deep learning architectures. In the table below, you will find the major works in the... field juxtaposed with our work to provide context.

Table 6: Comparison of State-of-the-Art Phishing Detection Studies

Study	Dataset Used	Model Applied	Accuracy	Limitation
Aldwairi et al. (2020)	UCI Phishing Dataset	Decision Trees, SVM	92.3%	No policy-level enforcement
Akinyelu& Adewumi (2014)	Alexa Top URLs	Naïve Bayes	88.1%	Small dataset
Mohamed et al. (2023)	PhishTank	Random Forest, MLP	96.5%	No user simulation

Ajayi et al. (2022)	PhishTank	Ensemble, Classical ML	94.8%	Limited scalability
Kyaw et al. (2024)	Multiple Email Datasets	Deep Learning (CNN, RNN, BERT)	95.6%	High computational cost
Saeed (2025)	UCI Phishing Dataset	Ensemble Voting Classifier	96.9%	Lack of real-time validation
This Study (2025)	Phishing Website (UCI Dataset)	8 Models incl. MLP, XGB, Ensemble	**97.2% (Best)**	Simulation URLs limited to 50

Table 6 shines a light on the 'Phishing detection techniques' workflow Aldwairi et al. (2020) employed policy- neglectingly trees and the SVM on the UCI dataset, and managed 92.3% accuracy blindly enforcing, Akinyelu and Adewumi (2014) Naïve Managed Bayes arms with the Alexa Top URLs and had an 88.1% score, but suffered from a small data set. Mohamed et al. (2023) advances 96.5% accuracy on the PhishTank data using Random MLP and Forest, but. PhishTank data Harla, et al does 94.8% Phishing detection and 'Scalability' as a down phrasing using ensemble methods and was limited. Kyaw et al. (2024) deep learning with 95.6% under the expense of using CNN, RNN, and BERT online Blocks. Saeed (2025) does 96.9 with ensemble using 'classifier' as a voting within PhishTank data enforcing simulations. PhishTank data, Ajayi et al consider 94.8 down 'Scalability' from (2024) within the reconnaissance. Kyaw et al (2024) systems of Islamoms. Phish data, Harla, et al does using. This No parted multi of real interruption Kyaw et al (2024) within ensemble the. Report Phishing detection limit Works Phishing with 97.2 the naive of accuracy crossed Bashi Harla et. So, the methods Phish detection and reporting are what drew me to this type of research.

6.10 Summary of Key Findings

The results of the evaluation prove that the ensemble classifier dominated all individual baseline models in both datasets. This advantage was not only expressed in the accuracy aspect but also in precision, recall, and F1-score, which affirms that it could generalise well in terms of the phishing patterns. The latter confirm the decision to make an ensemble strategy one of the fundamental detection mechanisms.

With the comparison of datasets, ARFF format performed and was slightly more stable than the CSV equivalent. This is because ARFF dataset has better structure of features organisation and metadata, thereby diminishing the inconsistency in parsing and the possibility to extract features with higher efficiency. The findings underscore the significance of how good the quality and structure of datasets are to derive the best model performance.

Lastly, in the human-versus-machine simulation, there was a major difference between the performance of the manual inspection and automated detection. Deceptive attacks like homograph attacks and unknown domain extensions as well as obfuscated URLs were the deceptive strategies that human participants found difficult. Conversely, the ensemble model retained the highest detection rates which further supported the need of the automated phishing detection systems to supplement or substitute the human decision making in operational settings.

7 Conclusion& Future Work

This study intended for assessment of if an integrated phishing detection framework with ensemble machine learning plus user simulation with threat intelligence can improve phishing resilience for organisations. Machine learning models, especially ensemble-based classifiers, can deliver high-accuracy predictions with the ensemble model reaching 97.82% accuracy on the ARFF dataset as results confirm, achieved across CSV and ARFF datasets. Simulated classification tasks saw this greatly outperform average human performance on. This work's shift away from validating theoretical models and toward an applied, operational context is surely a valuable contribution. Unlike prior studies that narrowly focus on classifier metrics, this framework embeds output export mechanisms with URL-based simulations and human-vs-machine comparisons. It is lightweight in design, modular, and adaptable. Therefore, it is appropriate within educational contexts, small-to-medium organisations, or threat intelligence feed integration. Generalisability improves using diverse datasets and real-time phishing URLs; dataset overfitting limits earlier studies.

However, some constraints should be noted. The framework is simulation-based, so it remains dependent on static datasets not on live network traffic. New phishing schemes that appear now restrict its adaptation ability. Its production-level deployment is limited also by the absence of integration with enterprise-grade components like IAM platforms, SIEM systems, or live user behaviour analytics. The simulation accuracy comparison is perceptive, yet it has a boundary. The comparison would improve with a larger and more diverse user sample.

Future works should address these limitations via dynamic data ingestion, continuous learning through online training pipelines, and multi-modal detection features like phishing attempts using email content, file attachments, or QR codes. Expanding the framework for interfacing with SOAR or MITRE ATT&CK-based platforms would improve its utility in Security Operations Centres (SOCs). Usability would improve if we were to conduct structured user testing with both technical and non-technical participants, validating accessibility and increasing real-world adoption.

The project shows meaningful practical as well as academic value regardless of that. It offers immediate value for researchers, developers, also security educators. It also provides a clear reproducible foundation for this will help build advanced phishing detection systems. Its low computational footprint as a browser extension, lightweight API module, or phishing awareness training tool further opens commercialisation potential. To provide a summary, integrated user simulation along with policy awareness plus practical design all combined with ensemble machine learning may form a strong phishing defence mechanism. It establishes a foundation for progress within practical cybersecurity plus links the gap between real operation and exact algorithms.

8 References

Altwaijry, N., Al-Turaiki, I., Alotaibi, R. & Alakeel, F. (2024) ‘Advancing phishing email detection: A comparative study of deep learning models’, *Sensors*, 24(7), 2077. https://doi.org/10.3390/s24072077.

Biswas, B., Mukhopadhyay, A., Kumar, A. & Delen, D. (2024) ‘A hybrid framework using explainable AI (XAI) in cyber-risk management for defence and recovery against phishing attacks’, *Decision Support Systems*, 177, 114102. https://doi.org/10.1016/j.dss.2023.114102.

Ejaz, A., Mian, A.N. & Manzoor, S. (2023) 'Life-long phishing attack detection using continual learning', *Scientific Reports*, 13, 11488. https://doi.org/10.1038/s41598-023-37552-9.
Ghalechyan, H., Israyelyan, E., Arakelyan, A., Hovhannisyan, G. & Davtyan, A. (2024) 'Phishing URL detection with neural networks: an empirical study', *Scientific Reports*, 14, 25134. https://doi.org/10.1038/s41598-024-74725-6.
Gupta, B.B., Gaurav, A., Arya, V., Attar, R.W., Bansal, S., Alhomoud, A. & Chui, K.T. (2024) 'Advanced BERT and CNN-based computational model for phishing detection in enterprise systems', *Computer Modeling in Engineering & Sciences*, 141(3), 2165–2183. https://doi.org/10.32604/cmes.2024.056473.
Innab, N., Osman, A.A.F., Ataelfadiel, M.A.M., Abu-Zanona, M., Elzaghmouri, B.M., Zawaideh, F.H. & Alawneh, M.F. (2024) 'Phishing attacks detection using ensemble machine-learning algorithms', *Computers, Materials & Continua*, 80(1), 1325–1345. https://doi.org/10.32604/cmc.2024.051778.
Meléndez, R., Ptaszynski, M. & Masui, F. (2024) 'Comparative investigation of traditional machine-learning models and transformer models for phishing email detection', *Electronics*, 13(24), 4877. https://doi.org/10.3390/electronics13244877.
Rao, R.S., Kondaiah, C., Pais, A.R., Lee, B. *et al.* (2025) 'A hybrid super learner ensemble for phishing detection on mobile devices', *Scientific Reports*, 15, 16839. https://doi.org/10.1038/s41598-025-02009-8.
Uddin, M.A. & Sarker, I.H. (2024) 'An explainable transformer-based model for phishing email detection: A large language model approach', *arXiv preprint*, arXiv:2402.13871. Available at: https://arxiv.org/abs/2402.13871.
Zhang, Z., Wu, J., Lu, N., Shi, W. & Liu, Z. (2025) 'AdaptPUD: An accurate URL-based detection approach against tailored deceptive phishing websites', *Computer Networks*, 265, 111303. https://doi.org/10.1016/j.comnet.2025.111303.
R. Mohammad and L. McCluskey. "Phishing Websites," UCI Machine Learning Repository, 2012.Available at: https://doi.org/10.24432/C51W2X.
Saeed, E.M.H., 2025. An Ensemble Voting Classifier based on Machine Learning Models for Phishing Detection. *International Journal of Scientific Research in Science, Engineering and Technology*, *12*(1), pp.15-27.
Kyaw, P.H., Gutierrez, J. and Ghobakhlou, A., 2024. A systematic review of deep learning techniques for phishing email detection. *Electronics*, *13*(19), p.3823.
Ajayi, O., Adetunmbi, A., Olowookere, T. and Sodiya, S., 2022. Performance evaluation of ensemble learning algorithms and classical machine learning algorithms for phishing detection. *Proceedings of the 2002 Smart, Secure and Sustainable Nation, Abuja, Nigeria*, *8*.