UNIVERSITY OF LAGOS

FACULTY OF COMPUTING AND INFORMATICS

COMPUTER NETWORKS, INFRASTRUCTURE MANAGEMENT AND SECURITY (CoNIMS) RESEARCH GROUP

8th ANNUAL

INTERNATIONAL CONFERENCE AND WORKSHOP

Theme:

SMART APPROACHES TO DATA SECURITY AND INFRASTRUCTURE MANAGEMENT 2025

SmADSIM 2025

WORKSHOP
University of Lagos,
Faculty of Science Boardroom
October 21-22, 2025

CONFERENCE
University of Lagos,
Arthur Mbanefo Digital Research Centre.
October 23, 2025

2025
Conference
Proceedings

FOREWORD

The 2025 Workshop and Conference featured many practical innovations in data security and infrastructure management and included technical paper submissions. This Conference proceedings consist of abstracts and full-length articles/papers presented at the 8th CoNIMS International Conference and Workshop held at the Arthur Mbanefo Digital Resource Centre, University of Lagos, Akoka, Lagos, Nigeria, October 23, 2025. CoNIMS is an acronym for Computer Networks, Infrastructure Management, and Security. It is a research group in the Department of Computer Sciences, Faculty of Computing and informatics, University of Lagos. Its vision is to conduct research that will advance knowledge in the broad field of Computer Science, Information Technology and Cybersecurity; while the mission is to conduct research and apply the results of the research to solving real-life problems; publish findings of research in reputable local and International journals; organize and attend conferences in the research areas.

The theme of the 2025 CoNIMS Conference is "Smart Approaches to Data Security and Infrastructure Management (SmADSIM 2025*)*". The sub-themes are Communication Networks and Performance Improvement; Cybersecurity; Software Engineering and Artificial Intelligence; Software Engineering and Network Designs and ICT4D Applications and E- Services. The papers contained in the proceedings are all related to the aforementioned theme and subthemes.

All the papers presented at the Conference were subjected to preliminary review by two assessors to establish prima facie quality in line with the conference's objectives. Based on the comments and inputs of participants during the technical sessions, the authors were required to update and, in some cases, rework their papers; the resubmitted papers were then reviewed for technical and literary quality by at least two assessors; only papers that satisfied this last stage of review were published in these proceedings.

My sincere gratitude goes to the authors for their efforts in producing high-quality articles worthy of publication in the proceedings. Special thanks also go to our esteemed reviewers, the editorial team, and conference officials for their time and efforts in ensuring that due diligence was adhered to in all stages of publication.

Professor Florence Alaba Oladeji
Editor-in-Chief

ADVISORY AND EDITORIAL BOARD MEMBERS

18.	Dr. A. Afolorunso	Department of Computer Sciences, National Open University of Nigeria.
19.	Dr. V.T. Odumuyiwa	Department of Computer Sciences, University of Lagos, Akoka, Lagos, Nigeria
20.	Dr. R.A. Koleoso	Department of Computer Sciences, University of Lagos, Akoka, Lagos, Nigeria.
21.	Dr. F.O. Alamu	Department of Computer Sciences, University of Lagos, Akoka, Lagos, Nigeria.
22.	Dr. U.C. Ogude	Department of Computer Sciences, University of Lagos, Akoka, Lagos, Nigeria.
23.	Dr. A.M. Nwohiri	Department of Computer Sciences, University of Lagos, Akoka, Lagos, Nigeria.
24.	Dr. E. Okewu	Centre for Information Technology and Systems, University of Lagos, Akoka, Lagos, Nigeria.

LOCAL ORGANIZING COMMITTEE

TABLE OF CONTENTS

A DATA-DRIVEN FRAMEWORK FOR SPAM CLASSIFICATION IN ONLINE SOCIAL NETWORKS USING ENSEMBLE LEARNING

Akanbi U.O.[1], Mfawa U.B.[2], Ukwa J.[3], Yusuf H.[4], Lawal-Fowora K.[5], and Azeez N.A.[6]
[1,2,3,4]Department of Computer Sciences, Faculty of Computing and Informatics, University of Lagos, Nigeria
[5]Yaba College of Technology, Yaba, Lagos, Nigeria.
[6]Department of Cybersecurity and Software Engineering, Faculty of Computing and Informatics, University of Lagos, Nigeria
Corresponding Author: nazeez@unilag.edu.ng

ABSTRACT
Social media sites like Instagram, messaging apps, and X (Twitter) are increasingly thought to be the main providers of spam, phishing links, and fraudulent ads. Stronger solutions are needed due to the complexity and growing scale of these detrimental activities as well as the shortcomings of traditional spam detection methods, especially when it comes to the issue of class imbalance or platform diversity. Using four datasets—HF Comments, Clickbait, SMS Spam, and UTKML—that were obtained through Kaggle and Hugging Face, the study compares baseline machine learning models with ensemble learning techniques to categorize spam in online social networks. The data was preprocessed using the SMOTE-ENN resampling technique to balance classes, TF-IDF vectorization, and text cleaning. The baseline classification models examined were Random Forest (RF), Naive Bayes (NB), Support Vector Machine (SVM), and XGBoost (XGB). Two ensemble procedures were employed to improve performance: stacking and voting. Accuracy, precision, recall, F1-score, ROC-AUC, log loss, Hamming loss, Cohen's kappa, F2-score, and Jaccard index are some of the metrics used in the analysis. Ultimately, the stacking ensemble of SVM | RF | NB achieved the greatest accuracy of 94.95% and precision of 96.30% in Dataset 1 at the conclusion of the tests, while all three stacking combinations in Dataset 2 achieved the highest accuracy of 99.91% and flawless precision of 100%. The best stacking model in Dataset 3 was SVM | RF, which had the maximum accuracy of 97.56% with a precision of 97.7%. In Dataset 4, the best stacking model was RF | GB | NB | SVM, which had an accuracy of 98.83%. In comparison, the voting ensemble also produced good results, with precision scores of 95% or higher and accuracies ranging from 84.47% of Dataset 1 to 99.47% of Dataset 2 and above 97% on Datasets 3 and 4. These findings show that detection accuracy on a variety of datasets is greatly improved by ensemble-based learning, particularly stacking. The study concludes by restating that appropriate preprocessing and ensemble learning techniques provide a scalable and efficient model to address common issues with spam detection, including overfitting, class discrepancy, and platform adaptability.

Keywords: Spamming, classification, models, algorithms, imbalance, performance, ensemble, classifiers

1.0 INTRODUCTION

Online social networks (OSNs) like Facebook, Instagram, and X (previously Twitter) have grown to be vital channels for marketing, communication, and information exchange. However, because of their extensive use, they are now frequently the focus of illicit activity, such as clickbait, phishing, spam, and misleading advertising. Spam messages present significant security issues, including the spread of malware and fraudulent schemes, in addition to degrading user experience (Chowdhury, 2020). The complexity of contemporary spam campaigns, which are frequently driven by coordinated networks and bots, makes it more challenging to identify them with conventional rule-based systems.

Traditional machine learning approaches like Naïve Bayes, decision trees, and support vector machines have been the mainstay of existing spam detection algorithms. Although these strategies have had some success, they face significant obstacles such as cross-platform generalization, class imbalance in datasets, and adaptability to changing spam techniques (Wu et al., 2017; Zhang et al., 2020). Furthermore, single-model solutions frequently fail to maintain consistent accuracy across many platforms and overfit particular datasets.

Ensemble learning approaches have become effective substitutes to overcome these drawbacks. By combining the advantages of several classifiers, techniques including bagging, boosting, stacking, and voting lower variance, lessen overfitting, and enhance generalization performance (Kibe et al., 2023). Recent research shows that ensemble approaches are more effective than standard models at detecting spam. Zhang et al. (2020), for instance, showed how ensemble-based methods perform better than standalone classifiers when it comes to identifying spam in extremely unbalanced datasets.

This paper uses both ensemble methods and traditional machine learning algorithms to provide a data-driven spam classification framework for OSNs. Four publicly accessible datasets were used to collect various spam attributes across platforms: HF Comments, Clickbait, SMS Spam, and UTKML. Text cleaning, TF-

IDF feature extraction, and SMOTE-ENN class imbalance treatment were all part of the data preprocessing process. Random Forest, Naïve Bayes, Support Vector Machine, and XGBoost were used as baseline classifiers. To improve performance, ensemble techniques employing voting and stacking classifiers were then used.

The suggested system outperformed several previous research, achieving up to 99% accuracy and good precision across all datasets. These findings confirm that ensemble learning offers a scalable and dependable method for spam identification in OSNs when combined with strong preprocessing techniques. The results support the practical development of security measures that improve user trust and platform integrity in addition to the scholarly discussion on spam classification.

This is how the remainder of the paper is structured:The relevant literature on spam classification in online social networks is reviewed in Section II. The suggested methodology, including feature extraction, data preparation, and ensemble learning strategies, is covered in Section III. The experimental results and discussion based on several evaluation metrics across four datasets are presented in Section IV. Section V wraps up the work and suggests potential lines of inquiry for further study.

2.0 RELATED WORKS

Traditional machine learning models for spam classification in Online Social Networks (OSNs) have given way to more advanced ensemble and deep learning techniques. Classifiers like Support Vector Machines (SVM), Naïve Bayes (NB), and Decision Trees (DT) were used in early research. For instance, Stringhini et al. (2010) laid the groundwork for further research on spam classification by creating the Social Honeypot dataset to examine the actions of spammers. However, scalability, imbalance, and cross-platform generalization were frequently issues with single classifiers.

Ensemble approaches were proposed to solve these problems. While Sankar et al. (2023) showed that integrating and pruning DTs improved resilience against overfitting, Kibe et al. (2023) showed that Random Forest (RF) and XGBoost increased accuracy by utilizing numerous decision trees. Zhang et al. (2020) used ensemble techniques in a similar manner and emphasized the value of varied training data for improved generalization. Furthermore, Wu et al. (2017) investigated hybrid deep learning techniques including CNNs and RNNs, which had higher accuracy than conventional models but came at a significant computational cost.

While Xu (2016) created a cross-platform spam classification framework that integrated general features with platform-specific attributes to increase adaptability, other studies focused on particular issues in spam classification by using SMOTE-ENN to reduce class imbalance by creating synthetic spam samples and filtering noise. More recently, Nallabariki et al. (2023) showed that semantic embeddings like Word2Vec and BERT improved text-based spam classification performance, and Wani et al. (2024) used adversarial training to fortify models against evasion attempts.

Usman (2025) expanded on these developments by putting out a data-driven spam classification system that combines ensemble and conventional machine learning methods. The system demonstrated great precision and up to 99% accuracy using voting and stacking classifiers on four datasets (HTF Comments, Clickbait, SMS Spam, and UTKML). In contrast to previous studies that reported accuracy levels ranging from 87% to 94% (e.g., Zhang et al., 2020; Kibe et al., 2023), these findings confirm that ensemble learning is an effective strategy for tackling enduring issues like overfitting, imbalance in classes, and platform variability in OSNs.

Table 1: *Summary of Reviewed Citations*

Author(s)	Year	Approach	Key Findings	Relevance to Project	Challenges/Gaps
Wani, M. A., ElAffendi, M. & Shakil, K. A.	2024	AI-generated spam review classification using embeddings and deep learning.	Achieved 94% accuracy and 92% F1-score for spam classification.	Highlights embedding techniques for robust spam classification.	Lacks focus on cross-platform adaptability.
Kurnia	2024	Hybrid approach with SMOTE-ENN for imbalanced datasets.	Balanced datasets improved recall from 72% to 88%.	Provides insights into addressing dataset imbalance.	Limited discussion on scalability.
Striim	2024	Real-time machine learning on streaming data.	Achieved sub-second latency with 95% precision in real-time spam classification.	Demonstrates the importance of real-time adaptability.	Requires high computational resources.
Nallabariki, J.,	2023	Machine learning models (Naïve	Reported 91% accuracy	Highlights traditional	Limited focus on cross-

Author(s)	Year	Approach	Key Findings	Relevance to Project	Challenges/Gaps
Chandana, T. K., Tejaswi, D. S. & Babu, M. Y.		Bayes, KNN, SVM) with TF-IDF.	using SVM with TF-IDF for text-based spam classification.	ML approaches for foundational spam classification.	platform scenarios.
Sankar, R. & Suriakala, M.	2023	Intelligent algorithm integrating user behavior for OSNs.	Improved spam classification accuracy to 89% by incorporating user activity metrics.	Demonstrates behavior-based classification's relevance to OSNs.	Requires real-time adaptability.
Kibe, R., Deokate, A., Suryawanshi, P. & Sonar, S.	2023	Ensemble methods (Random Forest, XGBoost) for multi-platform spam classification.	Achieved precision of 92% and recall of 90% across diverse datasets.	Shows the effectiveness of ensemble models in multi-platform scenarios.	Computationally intensive for real-time applications.
Dewi, C., Indriawan, F. A. & Christanto, H. J.	2023	Cross-platform spam classification with multi-modal data.	Achieved 88% F1-score using text, visual, and behavioral features.	Addresses cross-platform spam classification challenges.	Limited focus on dataset imbalance.
Kumar, C., Bharti, T. S. & Prakash, S.	2023	Hybrid techniques for addressing imbalanced datasets (e.g., SMOTE-ENN).	Increased recall for minority class from 68% to 85% after balancing.	Highlights solutions for imbalanced dataset challenges.	Scalability for large datasets not addressed.
Soms, N., Hariharan, S., Jeeva, K. & Karthick, C.	2022	Transformer-based models for spam classification in OSNs.	Achieved 93% accuracy and 90% F1-score using transformer models for text-based spam classification.	Highlights transformer efficiency for text classification.	High computational requirements.
Zhang, Z., Hou, R. & Yang, J.	2020	Multi-platform spam classification using ensemble ML techniques.	Achieved precision of 87% and recall of 89% in cross-platform spam classification.	Provides insights into scalability and adaptability for spam classification frameworks.	High dependency on labeled data.
Fattahi, S. & Mejri, M.	2020	Bimodal ensemble learning spam detector with NLP techniques (BoW, TF-IDF).	Reported 88% accuracy and 86% F1-score with ensemble learning.	Demonstrates the effectiveness of text-based features.	Limited focus on behavior-based spam tactics.
Brophy, J. & Lowd, D.	2020	EGGS: Extended Group-based Graphical models for Spam classification.	Achieved recall of 89% for group-based spam classification.	Emphasizes user behavior in spam classification.	Computationally intensive for large datasets.
Chowdhury, K.	2020	Machine learning models with feature engineering for social media spam classification.	Reported 90% precision using behavioral and textual features	Highlights feature engineering's role in robust spam classification.	Real-time and cross-platform classification gaps.
Wu, T., Liu, S., Zhang, J. & Xiang, Y.	2017	Hybrid machine learning models for spam classification.	Improved overall accuracy to 92% by combining ML techniques.	Demonstrates hybrid approaches for improving classification performance.	Limited exploration of real-time adaptability.
Xu, H.	2016	Framework for cross-platform spam classification.	Achieved 85% precision in detecting spam across multiple OSNs.	Provides strategies for generalizable spam classification models.	Outdated spam tactics in evaluation.
Chu, Z., Widjaja, I. & Wang, H.	2012	Behavior-based spam classification using user activity metrics.	Identified 87% of spam accounts accurately using activity patterns.	Demonstrates the importance of behavioral features in spam classification.	High computational costs for continuous monitoring.
Stringhini, G.,	2010	Honeypot profiles to study	Provided 83% accuracy	Demonstrates	Static approach unsuitable

Author(s)	Year	Approach	Key Findings	Relevance to Project	Challenges/Gaps
Kruegel, C. & Vigna, G.		spammer behavior.	in capturing spammer tactics on OSNs.	foundational behavior-based classification techniques.	for modern dynamic spam tactics.

3.0 METHODOLOGY

The methodology was organized into four stages: Data Collection, Data Pre-processing and Feature Engineering, Model Implementation, Results & Performance Evaluation.

3.1 Data Collection

To create a reliable and expandable spam classification system, datasets from many platforms and sources will be used. Lastly, in terms of textual, behavioral, and engagement-based characteristics, these datasets include a range of spam and authentic instances of content. The list of important datasets taken into consideration for this study is as follows:

The first dataset was acquired from: (https://www.kaggle.com/competitions/utkml-twitter-spam/submissions). The Kaggle UTK Machine Learning Twitter Spam Detection Competition 2019 is the source of the UTKML Twitter Spam Detection Dataset. This dataset is tagged spam and non-spam tweets with user metadata like engagement metrics, hashtags, URLs, and timestamps.

The second dataset was acquired from: (https://huggingface.co/datasets/valurank/spam_ham_comments) . This dataset was obtained from Hugging Face's datasets repository. The labeled comments from these social media platforms, including Instagram, of which the comments are classified as spam and non-spam. The dataset is in CSV format.

The third dataset was acquired from (https://kaggle.com/datasets/amananadrai/clickbait-dataset). This dataset comprises headlines that are either referred to as clickbait or not. These headlines act like normal online spam content that encourage people to click on them. The dataset is prepared, it is clear, in CSV form, and it is eligible to be used by spam classification tasks.

The fourth dataset was acquired from: (https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset?resource=download). This dataset has labeled text messages which are either spam or ham. These messages are short but well-structured, and the dataset is well stored in a CSV format, so it has minor requirements of preprocessing prior to working with it in machine learning models.

3.2 Data Pre-processing and Feature Engineering

The pre-processing stage involved data cleaning, text pre-processing and handling data imbalance.

3.3 Data Cleaning

The purpose of this phase is to remove any errors, inconsistencies, and extraneous data that can cause the model to be misled. To avoid bias, duplicate entries were removed, noise like emojis, URLs, and special characters were filtered out, and missing values were handled using imputation or default replacements. Unnecessary columns that had nothing to do with spam classification were also eliminated to speed up computation and save memory.

3.4 Text Pre-processing

Comments were separated into manageable pieces by using tokenization to organize the text data and make it suitable for machine learning. Stemming and lemmatization were then used to ensure that words were uniformly represented by their base or root form. Finally, TF-IDF was used to convert text to numerical features, which reduced noise and emphasized informative words.

3.5 Handling Imbalance

Class imbalance was corrected using SMOTE-ENN because the spam samples are often small compared to legal messages. In order to improve the representation, SMOTE artificially balanced minority-class samples and ENN removed cases that were noisy or wrongly identified. The combination of them, SMOTE-ENN, enhanced generalization and reduced overfitting in model training by generating a more lucid and fairer dataset.

3.6 Model Implementation

The main purpose of this stage is to compare the performance of stacking and voting ensemble methods used with different models as base learners compared to using each of the models independently. Some of the base learners considered are Naïve Bayes (NB), Decision Tree (DT), Logistic Regression (LR), Random Forest (RF), XGBoost (XGB), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting (GB).

To this end, a number of algorithms will be considered with different specific features with mathematical roots as follows:

3.7 Naïve Bayes Classifier (NB)

Naïve Bayes is a probabilistic classifier founded on the Byes Theorems, which opines the likelihood of the occurrence of one feature not to depend on the occurrence of some other feature. It can be applied on a wide scale in detecting spam because it is simple and it does well in classifying text. It has the following formula:

$$P(x) = \frac{P(y)*P(y)}{P(x)} \quad (1)$$

where:

$P(y|x)$: Posterior probability of class given features
$P(x|y)$: Likelihood of features given the class
$P(y)$: Prior probability of the class
$P(x)$: Probability of the features

Support Vector Machine (SVM)

SVM finds the hyperplane that best separates classes in a high-dimensional feature space. The decision boundary is calculated using:

$$f(x) = w^T x + b \quad (2)$$

where:

w: weight vector
x: Input feature vector
b: Bias form

The optimization objective is to minimize:

$$\frac{1}{2}||w||^2 \quad (3)$$

Subject to:

$$y_i(w^T x_i + b) \geq 1 \quad (4)$$

Decision Tree (DT)

The decision tree classifier is an algorithm where a tree is used to display every node, that is; a decision provided the values of the features. The tree divides the data based on the feature which maximizes the information gain or minimizes the impurity. *Gini* impurity measure can be defined as:

$$Gini = 1 - \sum_{i=1}^{c}(p_i)^2 \quad (5)$$

where:

p_i: Proportion of instances belonging to class i
C: Total number of classes

NB: A decision tree can contain categorical (YES/NO) as well as numerical data.

Table 2: Dataset Sources

S/No.	Dataset	No. of Entries	Source
Dataset 1	UTKML Dataset	11,968	Kaggle
Dataset 2	HF Comments Dataset	5,677	HuggingFace
Dataset 3	Clickbait Dataset	32,000	Kaggle
Dataset 4	SMS Spam Collection Dataset	5,169	Kaggle

Random Forest (RF)

Random Forest is an ensemble strategy of learning which constructs a series of decision trees and combines their outputs. It brings a notion of randomness with the selection of features and sampling of data. It is computed as:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \quad (6)$$

where:

T: Number of trees
$h_t(x)$: Prediction from the *t-th* tree

Logistic Regression (LR)

Logistic Regression is an algorithm that models the probability of a binary outcome using the sigmoid function:

$$\hat{y} = \frac{1}{1+e^{-z}} \quad (7)$$

where:

$z = w^T x + b$
w: Coefficient vector
x: Feature vector
b: Bias term

Extreme Gradient Boosting (XGB)

XGBoost is a boosting algorithm that combines weak learners to form a strong predictive model. It minimizes a regularized loss function:

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (8)$$

where:

l: Loss function (e.g., log loss)
Ω: Regularization term
f_k: Individual trees

K-Nearest Neighbours (KNN)

KNN is a distance-based algorithm that classifies a new instance based on the majority class of its *k* nearest neighbours. It uses distance metrics such as Euclidean distance, defined as:

$$d(x, y) = \sqrt{\sum_{i=1}^{c}(x_i - y_i)^2} \quad (9)$$

where:

x_i, y_i: Feature values of two data points
n: Number of features

The performances of the algorithms will be compared in terms of accuracy, precision, recall, and F1-score. The metrics report will be the basis of selecting the best-performing model to be finally implemented.

Combination of Variation

The emphasis during this step is on the assessment of the way ensemble learning approaches may be built by integrating variations of base learners. The Stacking and Voting Classifiers were developed with variations that had Random Forest, Gradient Boosting, Extra Gradient Boosting, Naive Bayes and Support Vector Machine as base learners. These combinations were determined to bring about diversity in learning strategies so that linear, probabilistic and tree-based models could be complementary to each other and be more powerful in spam classification.

4.0 RESULTS AND PERFORMANCE EVALUATION

This section reports the performance of various machine learning components on four studies: *UTKML, HuggingFace Comments, Clickbait and SMS Spam.* Various performance measures were evaluated with the models, such as accuracy, precision, recall, f1-score, ROC AUC, log loss, hamming loss, kappa, f2-score and Jaccard index. These findings allow us to conduct a comparative study of the effectiveness of each of these models to detect spam, clickbait or other deceitful content on different social networks and messaging services on the internet. In addition, ensemble approaches, including Voting and Stacking were introduced in order to integrate the strengths of the single classifiers and enhance stability, accuracy, and generalization outside that of single models.

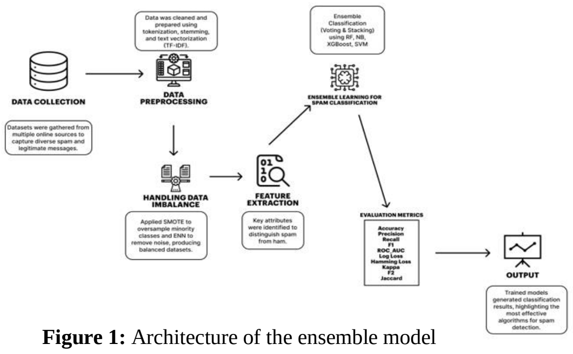

DATA COLLECTION
Datasets were gathered from multiple online sources to capture diverse spam and legitimate messages.
Data was cleaned and prepared using tokenization, stemming, and text vectorization (TF-IDF).
DATA PREPROCESSING
HANDLING DATA IMBALANCE
Applied SMOTE to oversample minority classes and ENN to remove noise, producing balanced datasets.
FEATURE EXTRACTION
Key attributes were identified to distinguish spam from ham.
Ensemble Classification (Voting & Stacking) using RF, NB, XGBoost, SVM
ENSEMBLE LEARNING FOR SPAM CLASSIFICATION
EVALUATION METRICS
Accuracy
Precision
Recall
F1
ROC_AUC
Log Loss
Hamming Loss
Kappa
F2
Jaccard
OUTPUT
Trained models generated classification results, highlighting the most effective algorithms for spam detection.

Figure 1: Architecture of the ensemble model

Table 3: Keys and the Actual Names of the Individual Algorithms

Key	Algorithm Name
LR	Logistic Regression
KNN	K Nearest Neighbor
DT	Decision Tree
SVM	Support Vector Machine
NB	Naïve Bayes
RF	Random Forest
GB	Gradient Bo
XGB	Extra Gradient Boosting

Table 4: Some of the Performance Metrics

S/No	Technique	Description	Formula
1	Accuracy	This is the proportion of correctly classified instances among all predictions.	$= \frac{TP+TN}{TP+FP+TN+FN}$
2	Precision	This is the proportion of instances classified as spam that are actually spam.	$= \frac{TP}{TP+FP}$
3	Recall	This is the proportion of actual spam messages correctly identified.	$\frac{TP}{TP+TN}$
4	F1-Score	This is the harmonic mean of precision and recall.	$= 2x \frac{Precision * Recall}{Precision + Recall}$
5	ROC-AUC	Measures the model's ability to distinguish between spam and legitimate messages.	

Algorithm 1: Algorithm of the spam classification system

Input (D): Raw labeled text dataset (spam/ham)
Output (R): Classification performance metrics (Accuracy, Precision, Recall, F1-Score, Log Loss etc)

```
1.  Load dataset D from source repository.
2.  Data preprocessing.
3.      Preprocess text samples $d \in D$
4.      Normalize case → lowercase.
5.  Remove punctuation, digits, special characters.
6.      Remove stopwords.
7.      Apply stemming/lemmatization → d'.
8.      Handle missing entries: drop or impute → D_clean.
9.  Data balancing.
10.     Apply SMOTE-ENN to D_clean → balanced dataset D*.
11. Data splitting.
12.     Split D* into training/testing partitions: (X_train, y_train), (X_test, y_test).
13. Feature extraction.
14.     Initialize feature extractor: TF-IDF.
15.     Transform datasets: X_train_tfidf = TFIDF(X_train), X_test_tfidf = TFIDF(X_test).
16. Model initialization.
17.     Define model set M = {Naïve Bayes, Decision Tree, Random Forest, SVM, XGBoost, KNN}.
18. Model training and evaluation.
19. For each model $m \in M$:
20.     Train m on (X_train_tfidf, y_train).
21.     Predict labels: $\hat{y}$_m = m(X_test_tfidf).
22.     Evaluate $\hat{y}$_m against y_test using metrics = {Acc, Prec, Rec, F1}.
23.     Store results R_m = (m, metrics).
24. Result aggregation.
25.     Aggregate results: R = {R_m | $m \in M$}.
26.     Generate comparative performance tables and plots → R_final.
27. Return R_final.
```

Data Analysis for Dataset 1

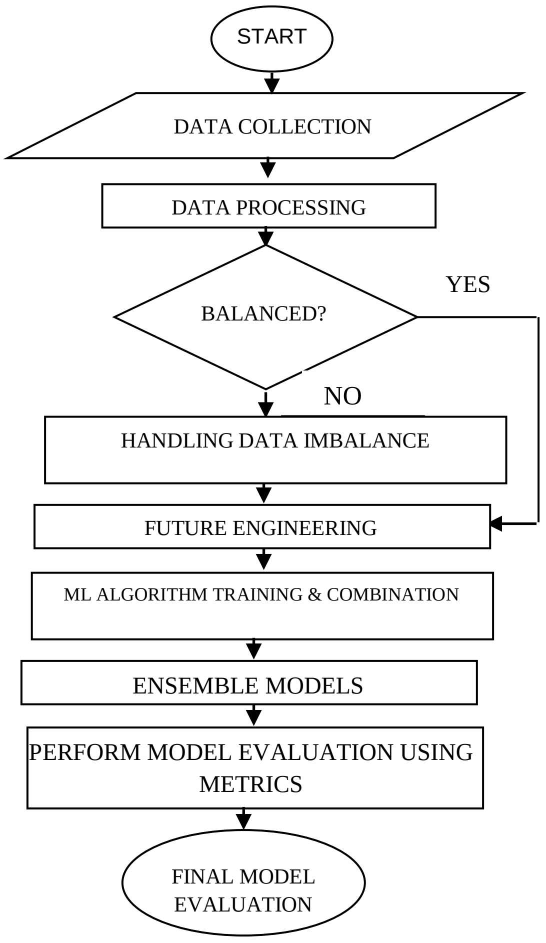


Figure 2: Flowchart of the ensemble model

4.1 Performance Evaluation

For performance comparison, the results from all combinations were compared with the result from each base learner model. The metrics of importance were measured and computed, which include Accuracy, Precision, Recall, Log loss, F1-Score, ROC-AUC etc.

4.2 Results

This paper used machine learning to classify spam using four datasets but their results were measured using various measures. Although such individual models as Naive Bayes, Decision Tree, SVM, Random Forest, and XGBoost performed well, none of them demonstrated the best results in all datasets. Nevertheless, the ensemble techniques; Voting and Stacking were significantly better than the single classifiers with high accuracy up to 99 percent. Combination of the best performance in both ensemble techniques showed a better stability and generalization and proved that ensemble learning is effective in spamming robust in Online Social Network.

Table 5: Results obtained by individual base classifiers [dataset 1]

	Algorithm	Accuracy	Precision	Recall	F1	ROC_ AUC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	Naive Bayes	0.8204	0.8432	0.7868	0.8140	0.8987	0.4553	0.1796	0.6407	0.7975	0.6864
2	Decision Tree	0.7669	0.7954	0.7182	0.7548	0.7669	8.4012	0.2331	0.5338	0.7324	0.6062
3	KNN	0.5021	1.0000	0.0033	0.0067	0.5096	17.5833	0.4979	0.0033	0.0042	0.0033
4	SVM	0.8183	0.8816	0.7349	0.8016	0.8975	0.4052	0.1817	0.6365	0.7602	0.6689
5	LR	0.8112	0.8780	0.7224	0.7927	0.8849	0.4737	0.1888	0.6223	0.7490	0.6565
6	RF	0.8041	0.8838	0.6998	0.7811	0.8733	0.7220	0.1959	0.6081	0.7302	0.6409
7	XGBoost	0.7736	0.8884	0.6254	0.7341	0.8513	0.4788	0.2264	0.5471	0.6648	0.5798

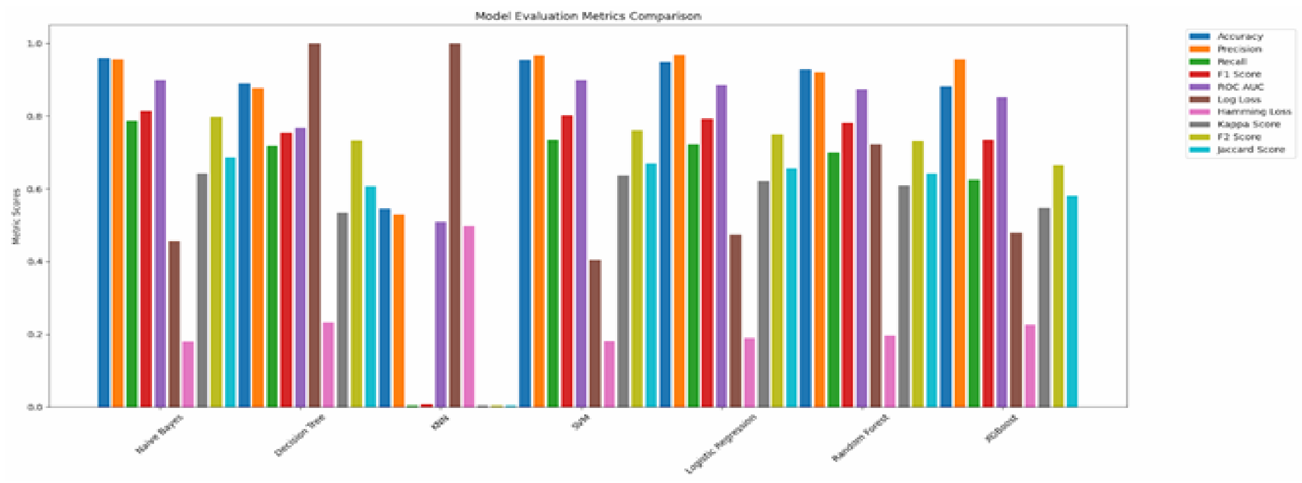


Figure 3: Bar plot of Individual Algorithms Results [Dataset 1]

Based on the findings in *Table 5*, Naive Bayes model had an optimal performance in all the metrics achieving an accuracy of 82.04%, precision of 84.32%, recall of 78.68%, and F1-score of 81.40%. It has also recorded a maximum ROC AUC of 89.87% Based on the findings in *Table 5*, Naive Bayes model had an optimal performance in all the metrics achieving an accuracy of 82.04%, precision of 84.32%, recall of 78.68%, and F1-score of 81.40%. It has also recorded a score) did once again underline its reliability. KNN on the contrary, was rather weak, having practically zero recall (0.0033), F1-score (0.0067), and Kappa (0.0033), implying that it did not recognize the positive-labeled class at all. Also, models like Decision Tree, Logistic Regression and Random Forest kept being moderate, whereas Naive Bayes demonstrated stable and relatively strong and balanced performance that is why it was the most effective one in the UTKML dataset.

Data Analysis for Dataset 2

Table 6: *Results obtained by individual base classifiers [dataset 2]*

	Algorithm	Accuracy	Precision	Recall	F1	ROC_AUC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	Naive Bayes	0.9784	1.0000	0.0417	0.0800	0.9573	0.0767	0.0216	0.0783	0.0515	0.0417
2	Decision Tree	0.9962	0.9167	0.9167	0.9167	0.9574	0.1356	0.0038	0.9147	0.9167	0.8462
3	KNN	0.9868	1.0000	0.4167	0.5882	0.9769	0.0531	0.0132	0.5827	0.4717	0.4167
4	SVM	0.9972	0.9565	0.9167	0.9362	0.9710	0.0119	0.0028	0.9347	0.9244	0.8800
5	LR	0.9906	1.0000	0.5833	0.7368	0.9664	0.0350	0.0094	0.7324	0.6364	0.5833
6	RF	0.9944	1.0000	0.7500	0.8571	0.9887	0.0220	0.0056	0.8543	0.7895	0.7500
7	XGBoost	0.9944	0.9091	0.8333	0.8696	0.9698	0.0228	0.0056	0.8667	0.8475	0.7692

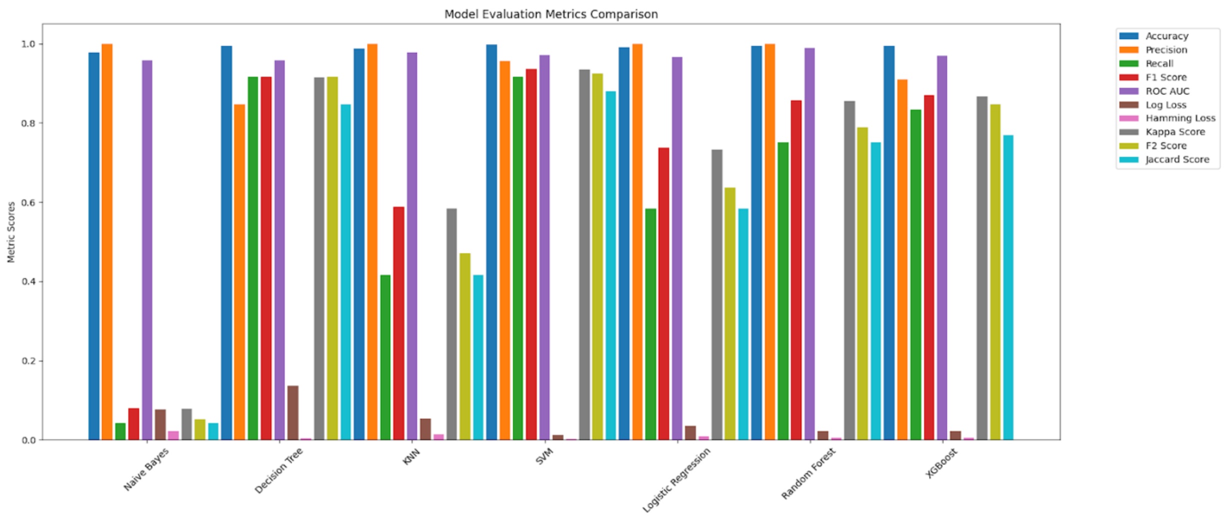


Figure 4: *Bar plot of Individual Algorithms Results [Dataset 2]*

Based on the results in *Table 6*, it is possible to note that the SVM model had the highest performance as it had an accuracy of 99.72%, precision of 95.65%, recall of 91.67%, and an F1 score of 93.62%. These values indicate that there is indeed high correlation between correctly achieving and having a low number of false positives. It also has one of the lowest log loss (0.0119) which indicates very confident predictions, its ROC AUC value of 97.10% on the other hand will ensure it can make good general classification. Random Forest and XGBoost models also provided competitive results with the high F1, Kappa, and Jaccard scores, so they are also quite reliable options. Naive Bayes model however, with perfect precision (1.0000) resulted with very low recall (4.17%) meaning that it did not identify spam very frequently hence it was not very useful in real world spam filtering. On the whole, SVM was identified as the most-performing model due to the consistent balance of all critical metrics.

Data Analysis for Dataset 3

Table 7: *Results obtained by individual base classifiers [dataset 3]*

	Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**	**ROC_ AUC**	**Log Loss**	**Hammin g Loss**	**Kappa**	**F2**	**Jaccard**
1	Naive Bayes	0.9594	0.9564	0.9646	0.9605	0.9921	0.1517	0.0406	0.9187	0.9629	0.9239
2	Decision Tree	0.8895	0.8769	0.9120	0.8941	0.8890	3.9817	0.1105	0.7787	0.9048	0.8085
3	KNN	0.5456	0.5295	0.9997	0.6923	0.8953	3.3918	0.4544	0.0715	0.8489	0.5294
4	SVM	0.9547	0.9662	0.9444	0.9552	0.9903	0.1248	0.0453	0.9094	0.9487	0.9142
5	LR	0.9480	0.9673	0.9297	0.9481	0.9883	0.2030	0.0520	0.8960	0.9370	0.9370
6	RF	0.9280	0.9192	0.9419	0.9304	0.9795	0.2018	0.0720	0.8558	0.9373	0.8699
7	XGBoost	0.8809	0.9567	0.8035	0.8735	0.9586	0.3201	0.1191	0.7626	0.8301	0.7754

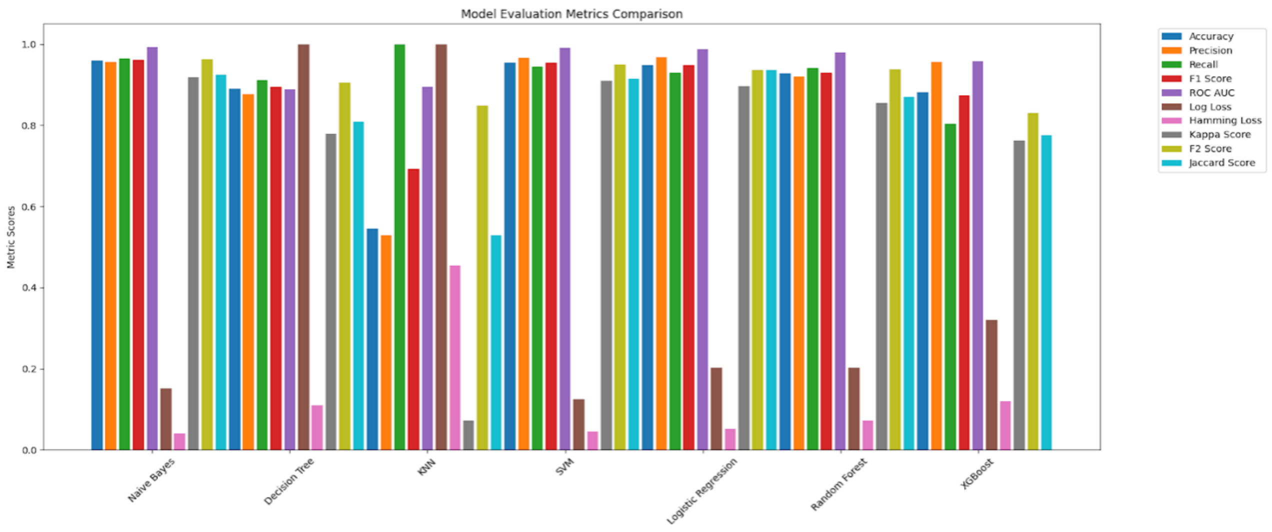


Figure 5: Bar plot of Individual Algorithms Results [Dataset 3]

As the results of the analysis of the Clickbait dataset reveal in *Table 7*, a number of models have demonstrated outstanding performance in a variety of evaluation metrics, in particular determining the presence of clickbait content. Remarkably, Naive Bayes obtained a high score for the metric of accuracy (95.94%), precision (95.64%), recall (96.46%), and F1 score (96.05%), and ROC AUC (99.21%), demonstrating that there is a good balance between the classification of positive results and false positives on the one hand, and the non-defective classification of the negative results on the other hand. Likewise, SVM also demonstrated an outstanding performance with an accuracy of 95.47%, and high values of all significant metrics, including recall and F2 score, focusing on the effectiveness of the model to identify useful positive cases.

The value of low log loss and Hamming loss on these high-ranked models once again confirms their confidence in the prediction and the low occurrence of misclassification. Conversely, though the majority of models prevailed in most of the parameters, the third model showed poor results in terms of accuracy (54.56%), Kappa (7.15%), and Jaccard score (52.94%) that indicated that this model focused on identifying the differences between the clickbait and non-clickbait headlines ineffectively. Nevertheless, it achieved a perfect recall with 99.97%, meaning that it tagged almost every clickbait as such but in the process probably labelled many non-clickbait headlines as well, resulting in worse overall balance. Overall, the most successful models were Naive Bayes and SVM since they had significantly good, balanced outcomes in terms of precision, recall, F1, and other significant indicators.

Data Analysis for Dataset 4

Table 8: *Results obtained by individual base classifiers [dataset 4]*

	Algorithm	Accuracy	Precision	Recall	F1	ROC_ AUC	Log Loss	Hamming Loss	Kappa	F2	Jaccard
1	Naive Bayes	0.9659	1.000	0.7467	0.8550	0.9801	0.1196	0.0341	0.8361	0.7865	0.7467
2	Decision Tree	0.9605	0.8397	0.8733	0.8562	0.9237	1.000	0.0395	0.8334	0.8664	0.7486
3	KNN	0.9265	1.000	0.4533	0.6239	0.8267	1.000	0.0735	0.5894	0.5090	0.4533
4	SVM	0.9767	0.9769	0.8467	0.9071	0.9892	0.0714	0.0233	0.8939	0.8699	0.8301
5	LR	0.9471	0.9596	0.6333	0.7631	0.9865	0.1431	0.0529	0.7347	0.6795	0.6169
6	RF	0.9722	0.9917	0.8000	0.8856	0.9853	0.1392	0.0278	0.8700	0.8322	0.7947
7	XGBoost	0.9695	0.9531	0.8133	0.8777	0.9763	0.0898	0.0305	0.8604	0.8379	0.7821

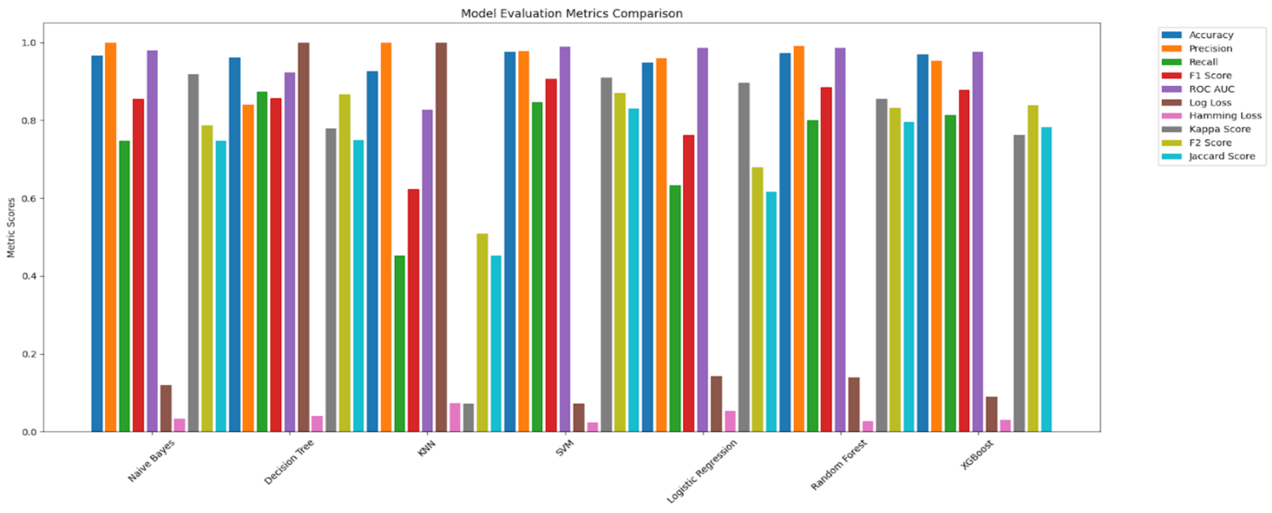


Figure 6: Bar plot of Individual Algorithms Results [Dataset 4]

The performance figures in *Table 8* derived in the SMS Spam dataset show good outcomes in a range of algorithms, showing that there is a good performance of spam message classification. Out of the analyzed models, SVM resembled the most working since it proved to have the best overall results with 97.67% accuracy, 97.69% precision, 84.67% recall which would lead to a high F1-score of 90.71%. This good balance in the precision and recall shows that the model itself is not only precise in terms of predicting spam and identifying that there were or were not any actual spam messages. It also provided a good ROC AUC value of 98.92%, a feeble log loss (0.0714) and smallest Hamming Loss of 0.0233, hence displaying the strength of the model in separating spam and non spam texts. Other models, especially Random Forest and XGBoost were also working exquisitely with the F1-scores above 88% and ROC AUCs above 97%. Although the recall was relatively low in some models, such as the third (45.33%), the F1-score was also relatively high (62.39%); this is because these models also achieved a perfect precision, meaning that, when it predicted a message as spam it actually was spam; however, it could not detect many actual cases of spam. Overall, this analysis reveals that the majority of the models worked well, and some of them demonstrated a better trade-off between precision, recall, and generalization rates.

Ensemble Learning Classifier Results

Justification for Using Ensemble Methods

The analysis of the individual machine learning models showed that some of them worked well on certain datasets but none of them performed well on all. This difference was a factor that prompted banding strategies. Both the Voting Classifier and Stacking Classifier used the combined power of different algorithms to achieve more stable, accurate and generalizable results. These approaches can eliminate dependence on a single algorithm and reduce errors within datasets by mixing a wide range of learners. Below are some of the key strengths of these classifiers:

(i) Improving stability by blending diverse models.
(ii) Achieving higher accuracy, precision and recall across datasets.
(iii) Handling dataset variability more robustly than single models.
(iv) Reducing individual misclassification errors.
(v) Enhancing generalization to unseen spam patterns.

Stacking Ensemble Classifier Results

Table 9: Results obtained by the Stacking Ensemble Method [dataset 1]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
RF | GB | NB | SVM	0.9486	0.9645	0.9314	0.9477	0.9892	0.1273	0.0514	0.8972	0.9379	0.9006
SVM | RF | NB	0.9495	0.9630	0.9348	0.9487	0.9885	0.1327	0.0505	0.8989	0.9403	0.9023
SVM | RF | GB	0.9449	0.9610	0.9273	0.9438	0.9872	0.1394	0.0551	0.8897	0.9338	0.8936

From *Table 9*, the SVM | RF | NB combination gave the best results for Dataset 1 with an accuracy of 94.9% and a recall of 93.5%, while RF | GB | NB | SVM achieved the highest precision of 96.5%.

Table 10: Results obtained by the Stacking Ensemble Method [dataset 2]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
RF | GB | NB | SVM	0.9991	1.0000	0.9583	0.9787	0.9596	0.0102	0.0009	0.9782	0.9664	0.9583
SVM | RF | NB	0.9991	1.0000	0.9583	0.9787	0.9596	0.0102	0.0009	0.9782	0.9664	0.9583
SVM | RF | GB	0.9991	1.0000	0.9583	0.9787	0.9597	0.0102	0.0009	0.9782	0.9664	0.9583

Based on *Table 10*, the three estimator combinations [RF | GB | NB | SVM, SVM | RF | NB and SVM | RF | GB] yielded the same output of 99.9% accuracy and 100% precision, and also yielded very stable results across models.

Table 11: Results obtained by the Stacking Ensemble Method [dataset 3]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
RF | GB | NB | SVM	0.9755	0.9776	0.9743	0.9760	0.9971	0.0700	0.0245	0.9509	0.9750	0.9531
SVM | RF | NB	0.9756	0.9770	0.9753	0.9761	0.9971	0.0698	0.0244	0.9512	0.9756	0.9534
SVM | RF | GB	0.9739	0.9776	0.9713	0.9744	0.9959	0.0808	0.0261	0.9478	0.9725	0.9501

Table 11 shows that SVM + RF combination gave the best result with an accuracy of 97.6% and recall of 97.5% which has marginally better performance in overall balance.

Table 12: Results obtained by the Stacking Ensemble Method [dataset 3]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
RF | GB | NB | SVM	0.9883	0.9858	0.9267	0.9553	0.9919	0.0488	0.0117	0.9486	0.9379	0.9145
SVM | RF | NB	0.9865	0.9787	0.9200	0.9485	0.9919	0.0497	0.0135	0.9407	0.9312	0.902
SVM | RF | GB	0.9865	0.9787	0.9200	0.9485	0.9879	0.0517	0.0135	0.9407	0.9312	0.902

Based on *Table 12*, RF | GB | NB | SVM combination performed optimally with an accuracy of 98.8% and the highest F1 of 95.5% as compared to other sets of estimators.

Voting Ensemble Classifier Results

Table 13: Results obtained by the Voting Ensemble Method [dataset 1]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
SVM | NB | RF | XGB	0.8421	0.8670	0.7983	0.8312	0.9288	0.3528	0.1579	0.6834	0.8111	0.7112
SVM | RF | XGB	0.8371	0.8662	0.7871	0.8248	0.9239	0.3563	0.1629	0.6732	0.8017	0.7018
SVM | NB | XGB	0.8447	0.8604	0.8129	0.8360	0.9289	0.3562	0.1553	0.6887	0.822	0.7182

Based on *Table 13*, the best performance was obtained with SVM | NB | XGB combination with highest accuracy of 84.47% and highest recall of 81.29% compared to all the ensembles leading to F1 score of 83.60%.

Table 14: Results obtained by the Voting Ensemble Method [dataset 2]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
SVM | NB | RF | XGB	0.9947	1.0000	0.9394	0.9688	0.9970	0.0189	0.0053	0.9659	0.9509	0.9394
SVM | RF | XGB	0.9947	1.0000	0.9394	0.9688	0.9968	0.0200	0.0053	0.9659	0.9509	0.9394
SVM | NB | XGB	0.9947	1.0000	0.9394	0.9688	0.9927	0.0205	0.0053	0.9659	0.9509	0.9394
NB | RF | XGB	0.9947	1.0000	0.9394	0.9688	0.9971	0.0191	0.0053	0.9659	0.9509	0.9394

Using *Table 14*, the combination of all voting resulted in the same performance with an accuracy of 99.47%, precision of 100% and a recall of 93.94% leading to an F1 score of 96.88%.

Table 15: Results obtained by the Voting Ensemble Method [dataset 3]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
SVM | NB | RF | XGB	0.9509	0.9515	0.9503	0.9509	0.9890	0.1733	0.0491	0.9019	0.9505	0.9063
SVM | RF | XGB	0.9459	0.9484	0.9431	0.9458	0.986	0.1873	0.0541	0.8919	0.9442	0.8971
SVM | NB | XGB	0.9537	0.9633	0.9434	0.9532	0.9901	0.1719	0.0463	0.9075	0.9473	0.9107
NB | RF | XGB	0.9455	0.9461	0.9447	0.9454	0.9871	0.1970	0.0545	0.8909	0.9449	0.8964

According to the data in *Table 15*, the SVM | NB | XGB model performed most optimally with an accuracy of 95.37%, precision of 96.33%, and F1 score of 95.32%.

Table 16: Results obtained by the Voting Ensemble Method [dataset 4]

Estimators	Accuracy	Precision	Recall	F1	ROC AUC	Log Loss	Hamm-ing Loss	Kappa	F2	Jaccard
SVM | NB | RF | XGB	0.9774	0.9921	0.8389	0.9091	0.9881	0.0786	0.0226	0.8963	0.8657	0.8333
SVM | RF | XGB	0.9774	0.9844	0.8456	0.9097	0.9876	0.0765	0.0226	0.8969	0.8702	0.8344
SVM | NB | XGB	0.9774	0.9921	0.8389	0.9091	0.9866	0.0781	0.0226	0.8963	0.8657	0.8333
NB | RF | XGB	0.9728	0.9917	0.8054	0.8889	0.9864	0.0889	0.0272	0.8736	0.8368	0.8000

Based on *Table 16*, the optimal overall balance was attained by SVM | RF combination with an accuracy of 97.74%, a recall of 84.56%, and an F1 score of 90.97%.

Overall, our implementation phase described how our proposed spam classification framework was implemented on four different datasets, where both separate individual classifiers and ensemble learning methods were tested. Whereas individual algorithms exhibited different performance, combination of Voting and Stacking Classifiers considerably improved results and stacking proved to be the best method. The findings have proved that ensemble learning is not only more accurate and precise but also robust and adaptable to various spam patterns. These results lay good groundwork towards developing real-time, scalable spam classification solutions within online social networks.

5.0 CONCLUSION AND FUTURE WORK

In conclusion, this project proposed a data-driven framework of spam classification in social networks on the Internet using the combination of machine learning algorithms and various sets of features. The model has a future potential to be used in practice by the evaluation of a specific experiment on four different datasets which showed high performance and precision and accuracy on all the datasets.

Although the results are encouraging, there is room to do more. In order to improve the contextualization of spam messages in the future workspace, we could consider extending the framework to incorporate deep learning-based approaches such as BERT or LSTM. In addition, the analysis of multi-modal data (the text, images, and the actions of users) could ensure increased accuracy in more complex scenarios. Subsequently steps such as detection in live environment and implementation are also still necessary, which would make sure the framework is productive in the face of evolving techniques used by spammers. Lastly, the solution can continue to be generalized through addition of more social media and languages.

In an attempt to give a professional medium for viewing code repositories and specifically, to give room for reproducibility, the link is provided under note.

Note

Code available at: https://github.com/akanbiusman/A-Data-Driven-Framework-for-Spam-Classification-in-Online-Social-Networks-

REFERENCES

Batista, G. E., Prati, R. C. & Monard, M. C. (2004). A study of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, *6*(1), 20-29.

Brophy, J. & Lowd, D. (2020). *EGGS: A flexible approach to relational modeling of social network spam*. University of Oregon. Retrieved from https://arxiv.org/abs/2001.04909

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321-357.

Chowdhury, K. (2020). *Spam identification on Facebook, Twitter, and email using machine learning*. Central European Researchers Journal, 6(1), 18–26.

Chu, Z., Widjaja, I. & Wang, H. (2012). Detecting social spam campaigns on Twitter. In F. Bao, P. Samarati & J. Zhou (Eds.), *ACNS 2012: Applied Cryptography and Network Security* (Vol. 7341, pp. 455–472). Springer. . [Online] Available at: https://doi.org/10.1007/978-3-642-31284-7_27

DataReportal. (2024). *Global social media statistics.*

Dewi, C., Indriawan, F. A. & Christanto, H. J. (2023). *Spam classification problems using support vector machine and grid search*. *International Journal of Applied Science and Engineering, 20*(4), 165. [Online] Available at: https://doi.org/ 10.6703/IJASE.202312_20(4).006

Fattahi, S. & Mejri, M. (2020). *SpaML: A bimodal ensemble learning approach for spam classification in OSNs*. Expert Systems with Applications, 159, 113553.

Ferrara, E., Varol, O., Davis, C., Menczer, F. & Flammini, A. (2016). *The rise of social bots*. Communications of the ACM, 59(7), 96-104.

Kabakus, A. T. & Kara, R. (2017). *A survey of spam classification methods on Twitter*. International Journal of Advanced Computer Science and Applications, 8(3), 29-38.

Kibe, R., Deokate, A., Suryawanshi, P. & Sonar, S. (2023). Detection of social network spam based on improved machine learning. *Journal of Emerging Technologies and Innovative Research (JETIR), 10*(5), 254-259. Retrieved from http://www.jetir.org/

Kumar, C., Bharti, T. S. & Prakash, S. (2023). A hybrid data-driven framework for spam classification in online social networks. *Procedia Computer Science, 218*, 124–132. https://doi.org/10.1016/j.procs.2022.12.408

Kurnia (2024). *Addressing dataset imbalance in spam classification using SMOTE-ENN*. Medium.

Nallabariki, J., Chandana, T. K., Tejaswi, D. S. & Babu, M. Y. (2023). A comprehensive overview on intelligent spam email classification. *International Journal for Research in Applied Science & Engineering Technology (IJRASET), 11*(3), 1702-1707. . [Online] Available at: https://doi.org/10.22214/ijraset.2023.49529

Sankar, R. & Suriakala, M. (2023). *An intelligent and novel algorithm for securing vulnerable users of online social networks*. Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018). IEEE. Retrieved from https://www. researchgate.net/publication/369184098

Soms, N., Hariharan, S., Jeeva, K. & Karthick, C. (2022). Naive Bayes machine learning framework for auto classification of spam mails. *International Journal of Health Sciences*, *6*(S3), 7270–7277. . [Online] Available at: https://doi. org/10.53730/ijhs.v6nS3.7742

Striim. (2024). *Harnessing Continuous Data Streams: Unlocking the Potential of Online Machine Learning.*

Stringhini, G., Kruegel, C. & Vigna, G. (2010). *Detecting spammers on social networks using honeypots*. Proceedings of the 26th Annual Computer Security Applications Conference, 1-10.

Wani, M. A., ElAffendi, M. & Shakil, K. A. (2024). AI-generated spam review classification framework with deep learning algorithms and natural language processing. *Computers, 13*(10), 264. . [Online] Available at: https://doi.org/10. 3390/computers13100264

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-2*(3), 408-421.

Wu, T., Liu, S., Zhang, J. & Xiang, Y. (2017). Twitter spam classification based on deep learning. *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '17)*, 1-10. . [Online] Available at: https://doi.org/10. 1145/ 3014812.3014815.

Xu, H. (2016). *Efficient spam classification across online social networks* (Master's thesis, University of Toledo).

Zhang, Z., Hou, R. & Yang, J. (2020). Detection of social network spam based on improved extreme learning machines. *IEEE Access, 8*, 112003-112014. . [Online] Available at: https://doi.org /10.1109/ACCESS.2020.3002940.

AN OPTIMIZED NAVIGATION SYSTEM USING Q-LEARNING ALGORITHM TO A CAMPUS ENVISIONED MODEL CAR PARK

Are I. O[1]., Abass R.O.[2], Oladeji F. A.[3]., Akinboro S.A[4] and Adepoju C I.[5]
[1,2,3,4]Department of Computer Science, University of Lagos, Lagos, Nigeria
[5]Department of Advanced Computer Science, University of Hull, UK
[1]ikeoluwaare@yahoo.com, [2]rabass@unilag.edu.ng, [3]foladeji@unilag.edu.ng, [4]sakinboro@unilag.edu.ng, [5]famuyideitunu@gmail.com,
Corresponding Author: ikeoluwaare@yahoo.com

ABSTRACT
In today's world, the extent to which technology and data are leveraged to inform and automate the operations of a city is greatly reflected in the quality of living of its inhabitants. A smart city adequately utilizes these tools to provide the general public with better living conditions and promote the sustainability of the community. To encourage the development of the University of Lagos main campus into a smart city, an Internet of Things (IoT)-based car park system was proposed. This study aimed to contribute to this initiative of providing smart parking management for the campus by focusing on a key system component, i.e., optimized navigation to the nearest available car park based on distance and traffic. The study implemented Q-learning, a reinforcement learning algorithm, to fulfill this requirement. Towards realizing the optimized car park navigation system, a graph representation of the campus was created to serve as the Q-learning environment, where the nodes were locations and the edges, the distances and traffic between them. Subsequently, the obtained Q-learning solution was evaluated using relevant metrics and Dijkstra's algorithm as a benchmark. Furthermore, a simulation was developed to aid the visualization of the solution.

Keywords: Car Parks, Optimized Navigation, Q-Learning, Smart City, University of Lagos.

1.0 INTRODUCTION

A smart city is one where technology and data collection are used to improve city operations and the lives of its inhabitants. Various smart city initiatives exist to improve areas such as traffic, parking, energy, and waste management, to name a few. Similarly, the focus of this study constitutes an initiative aimed at providing smart parking management for the University of Lagos main campus through an Internet of Things (IoT)-based car park system.

The core functionality of this system includes reservation of car park spaces using an application, optimized navigation to available spaces, and real-time monitoring of car parks and traffic using IoT devices. True to the smart city definition, this system will improve the experiences of the many visitors the university receives daily by helping to save time, alleviate traffic congestion, maximize the use of all car parks, and even serve as an avenue to manage payments for parking if need be.

This study only focuses on one of the essential parts of this system, i.e., optimized navigation. Although various approaches were considered to deliver this part of the system, ultimately, Q-learning was selected. Q-learning is a machine learning algorithm that facilitates the learning of a task. Machine learning is categorized into different areas, one of which is reinforcement learning, the category Q-learning falls within. To lay the necessary groundwork required to understand this algorithm, the basics of reinforcement learning are also covered in this work.

Subsequently, the algorithm is employed to solve the optimized navigation problem for the IoT-based car park system. This problem involves guiding a driver within the campus to the nearest available car park based on distance and traffic. The study also evaluates the effectiveness of Q-learning as a solution to this problem and demonstrates the solution with a simulation.

This study aims to implement and demonstrate the use of the Q-learning algorithm to optimize navigation for an IoT-based car park system.

The objectives are to:

(i) Model the University of Lagos main campus.
(ii) Implement the algorithm using this modeled environment to obtain a trained agent capable of navigating to the nearest available car park by minimizing distance and traffic.
(iii) Evaluate the performance of the Q-learning solution.
(iv) Develop a simulation to visualize the performance of the Q-learning solution.

2.0 RELATED WORKS

2.1 Overview of Reinforcement Learning

Reinforcement learning can simply be understood as a way of establishing a pattern of expected behavior in an agent over a series of attempts for the purpose of achieving a goal. The agent learns its behavior in an environment by taking actions, receiving the consequences of those actions, from which it learns what actions are favorable in respect to the goal, and utilizing its experience for subsequent attempts. Ultimately, the expected behavior for achieving the end goal is reinforced in the agent.

Sutton and Barto (2015) consider reinforcement learning as a machine learning paradigm distinct from supervised and unsupervised learning. Supervised learning involves an external supervisor to guide the decision-making as opposed to an agent having to learn from its experience to determine what actions are best to take. It would also be insufficient and impractical to solve reinforcement learning problems. Unsupervised learning, on the other hand, aims to uncover hidden structure in unlabelled data as opposed to maximizing a reward signal like reinforcement learning.

To aid the understanding of reinforcement learning, the following terms are discussed below and can be referred to as elements of reinforcement learning.

An *agent* is an entity in the form of a program with the responsibility of learning in order to achieve a goal. It learns by interacting with an environment. This interaction involves perceiving the state of this environment and responding with an *action*—what the agent decides to do—to change the state (Sutton & Barto, 2015).

Sutton and Barto (2015) describe an *environment* as "everything outside the agent" (p. 53) and a *state* as a "signal from the environment" and "whatever information is available to the agent" (p. 62). Within this environment, which is dynamic and uncertain, the agent aims to achieve its goal.

A *model* captures the behavior of an environment, allowing inferences of how the environment will behave for situations that have not been experienced. The use of models divides reinforcement learning methods into model-based and model-free (Sutton & Barto, 2015).

A *policy* determines the behavior of the agent (Sutton & Barto, 2015). This behavior can be described as which action the agent decides to take at a given time. It is a mapping from states to actions (Harmon & Harmon, 1997).

A *value function* is also a mapping from states to their values (Harmon & Harmon, 1997). The value of a state is a numerical representation of how favorable it is to be in that state in relation to achieving the end goal of the agent. It is the cumulative reward that can be obtained from that state to an end state (Sutton & Barto, 2015).

A *reward* represents the consequence of actions taken. The goal of an agent is to maximize its rewards (Sutton & Barto, 2015). It follows that in order to achieve a goal, actions in favor of it should be encouraged with high rewards so that as the agent aims to maximize rewards, the desired behavior is reinforced.

A *time step* is a single cycle of interaction between the agent and its environment (Sutton & Barto, 2015). While an *episode* is the journey from a start state to an end state, consisting of all time steps in between.

These elements make up the bigger picture, like so: An *agent* receives information, i.e., the *state* of its *environment*, for which it may or may not have a *model*. Based on a *policy*, the agent decides an *action* to take. This policy may be informed by a *value function* where the agent selects actions that would bring it into more profitable states. For actions taken, the agent receives an immediate *reward*, updates the *value function*, and changes the state of the environment. This makes up one *time step*. This process continues until the agent reaches an end state, which terminates an *episode*.

2.2 Overview of Q-Learning

Q-learning was developed in 1989 by Chris Watkins, bringing together and building upon the work done in all three threads of reinforcement learning. It is off-policy and model-free reinforcement learning. Off-policy learning involves the use of two policies, a behavior and a target policy, in order to arrive at an optimal one. The behavior policy determines the behavior of the agent, and the target policy is the deliverable from the learning process. While model-free learning methods do not use a model of the environment as previously stated and learn by trial-and-error (Sutton & Barto, 2015).

In basic Q-learning, a single agent aims to obtain an optimal policy by continuously updating the Q-value for each state as it experiences them using the equation below (Jang, Kim, Harerimana & Kim, 2019).

$$Q(s, a) \leftarrow Q(s, a) + \alpha[R + \gamma \max Q(s', a') - Q(s, a)] \quad (1)$$

The Q-value, Q(s, a), is the value of taking an action, a, in a state, s. In the above equation for the Q-value, the Q-value of a state-action pair gets updated by taking its current Q-value and adding the product of the learning rate and the TD error (Jang *et al.*, 2019).

The learning rate, which is represented by α, controls how fast the Q-value estimates are modified (Even-Dar & Mansour, 2003) and has a value between 0 and 1 (Jang *et al.*, 2019). The higher the value, the quicker the agent learns.

Temporal difference (TD) error is the difference between the new Q-value estimate and the current Q-value estimate. The new Q-value estimate is a sum of the immediate reward, R, of taking an action in a state based on the behavior policy and the maximum reward that can be obtained in the next state, s', using the target policy.

Lastly, the discount factor, γ, a value also between 0 and 1, introduces the concept of depreciation of rewards (Jang *et al.*, 2019), meaning that rewards received later on are worth less than those received now (Watkins & Dayan, 1992). The higher the value, the more worth is placed on future rewards.

Q-learning is a temporal difference method, and it solves the reinforcement learning prediction problem using experience with the environment. From the equation, we can see that the value predictions are improved upon by bootstrapping, i.e., the state-action value estimates are updated based on estimates of state-action pairs (Sutton & Barto, 2015). Ideally, this process continues until the Q-values approximately converge. These values can then be used to guide the agent's behavior towards achieving the end goal (Jang *et al.*, 2019).

There are many Q-learning algorithms. Single- agent ones like the basic Q-learning, Deep Q-learning, and Double Q-learning, and multi-agent ones like Modular Q-learning and Ant Q-learning (Jang *et al.*, 2019). However, the focus of this study will be on the basic Q-learning.

Algorithm 1: Q-Learning

The Q-learning algorithm given by Sutton and Barto (2015):

```
Initialize $Q(s, a)$, $\forall s \in S$, $a \in A(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ε-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
        $S \leftarrow S'$;
    until S is terminal
```

2.3 Q-Learning Applications in Path Planning

Q-learning was applied in robotics for multi-path planning in a study by Hu, Yang, and Lou (2021). The environment for the agent was a grid map containing obstacles, and the study's objective was to determine if a trained Q-learning agent, given a start and end point, could find multiple paths from the source to the destination while avoiding these obstacles. The algorithm proved successful in achieving the objective, however, it should be noted that the locations of the source, destination, and obstacles were static, most likely resulting in reduced complexity of the environment. The paper also went on to state the limitations of the algorithm, such as its inefficiency and inability to optimize paths.

Ibrahim, Atia, and Fakhr (2021) also aimed to apply the algorithm to path planning in an autonomous vehicle. The objective of the study was to test the algorithm in both static and dynamic environments. The base environment was a discretized 2D grid consisting of obstacles. Like the first study, the agent could achieve optimal paths to the goal in the static environment from any starting point. However, the randomized placement of obstacles resulted in an increased complexity and training time, and a cut-off was introduced to break out of an episode. The paper stated that the algorithm showed improvements in recent tests, deeming it applicable for solving the problem. Given the experiment's resources, the algorithm did not converge. Hyperparameter optimization and testing of different reward functions were listed for future work.

3.0 METHODOLOGY

3.1 System Analysis

The system aimed to provide optimal navigation for a vehicle from a given location to the nearest available car park within the campus. The optimization was to be achieved using a Q-learning solution that minimized distance and traffic when determining routes. The system's rationale was to be displayed during the navigation journey along with a 2D simulation of the journey intended to aid visualization.

The following constituted the functional requirements of the system:

(i) The system must allow users to select a start location from a set of campus locations.

(ii) The system must navigate the vehicle from the start location to the nearest available car park by selecting optimal routes.

(iii) The system must display its rationale for path selection during the journey and the resultant navigation metrics on arrival.

(iv) The system must simulate the navigation of the vehicle.

3.2 System Process Design

The system receives the start location from the user, which maps to a node in the environment. With the start state obtained, the system randomly generates a set of available car parks. The search algorithm uses the learned Q-table in an attempt to find an optimal route to the nearest available car park. The route is traversed and the 2D simulation displays the vehicle moving along the campus roads to its destination. The system also shows its rationale as it navigates. After the vehicle arrives at its destination, the system displays metrics such as the traffic encountered and distance traveled to the user.

3.3 Environment Modeling

The campus environment was modeled with the help of an online map and represented using a graph. It required two adjacency lists, one where the edge weights were the distances between incident nodes and another where the edge weights were the traffic. The nodes were locations on the map, such as car parks, road intersections, and entry and exit points of premises. The edges represented the road segments between these locations. A dictionary was chosen to store all car park nodes.

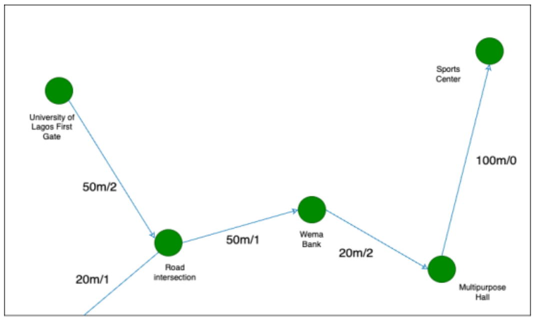


Figure 1: Example Graph Representation of the Campus

3.4 State and Action Space

Each node represented a possible location of the agent. The state of the environment perceived by the agent was the node the agent was currently at, and all the nodes made up the state space. The number of states was dependent on the number of nodes.

On the other hand, the action space was variable. This was because the edges to neighboring nodes at a given node represented the actions the agent could make in that state and made up the action space for the agent in that state.

3.5 Reward Function Design

The reward function designed to train the agent to navigate to the nearest car park (assuming all car parks were available) is given below:

(i) A quadratic reward for the cumulative distance traveled from the start state to the current state, multiplied by a negative weight, to reinforce choosing shorter distances.
   - ○ distance_reward = - 0.1 * ((distance_traveled / 100) ** 2)

(ii) A quadratic reward for the cumulative number of cars on edges traversed from the start state to the current state, multiplied by a negative weight, to reinforce choosing routes with lesser traffic.
   - ○ traffic_reward = - 0.2 * ((cars_seen) ** 2)

(iii) A reward of -2 for visiting a previously visited state to discourage such.

(iv) A reward of -500 for leaving the campus to reinforce navigation within the campus.

(v) A reward of 100 for arriving at a car park to reinforce navigating to a car park.

3.6 Q-Table Configuration

The Q-table maintains the Q-values of state-action pairs. The table consists of actions as columns and states as rows. The values of state-action pairs are usually initialized to 0, and then gradually updated using the Q-value equation as the agent gains experience in the environment. This continues for a given number of training episodes.

Due to the sparseness of the environment graph and the variable-sized action space for each state, a dictionary was chosen to represent this table. It mapped states to their dictionary of actions.

3.7 Behaviour and Target Policies

An epsilon-greedy policy was selected as the behavior policy where, based on the value of epsilon (ε), the agent chose to explore or exploit known values. Specifically, the value of epsilon was the probability of the agent exploring and ($1 - \varepsilon$), the likelihood of the agent exploiting. Epsilon, like the other hyperparameters, i.e., the learning rate (α) and the discount factor (γ), ranges from 0 to 1. It was decided to have epsilon decay from 1 towards 0.1 during training, increasing the probability of exploitation as training progressed.

The target policy was a greedy policy that focused solely on exploiting the Q-values. Following the target policy after training was expected to result in optimized navigation to the nearest car park (assuming all car parks were available).

3.8 Hyperparameter Tuning

Hyperparameter tuning was planned for sixteen (16) combinations of learning rate (α) and discount factor (γ) values to find a combination that could yield the best results before starting the final training process.

3.9 Training and Evaluation

The flow for the training and evaluation episodes was narrated.

3.10 Search Algorithm

After an initial and unrecorded training and evaluation, it was concluded that Q-learning could not handle the car park availability factor. The scope of the Q-learning problem was then reduced to optimized navigation to the nearest car park (assuming all car parks were available). This change is reflected in the preceding sections that narrate the Q-learning design process.

A simple search algorithm was conceived to handle the car park availability aspect of the problem. Using the learned Q-table and given the set of available car parks, the algorithm attempted to achieve optimization by selecting the highest-value actions from states till it arrived at an available car park, forming the combined Q-learning solution referred to earlier.

3.11 Simulation

The simulation requirement to visualize the performance of the Q-learning solution fostered the understandability of the desired system, especially because the system was insufficient to be used in a real-life scenario. The simulation aimed to simply depict the campus environment and showcase the journey of the agent car. Different software was researched to meet the requirement, and ultimately, Pygame, a game development library, was used to create a 2D simulation.

4.0 RESULTS AND DISCUSSION

4.1 System Specifications

The specifications of the hardware on which the system was implemented include an 8-core processor (CPU), an 8-core graphics processing unit (GPU), 8GB of memory, and storage of 256GB SSD.

The software specifications include an operating system (OS) of macOS Monterey, version 12.2, and a programming environment of Python 3.12.4 and Visual Studio Code as an integrated development environment (IDE). The following notable Python libraries were used:

- Gymnasium v0.29.1
- NumPy v2.0.0
- Pygame v2.6.0

4.2 Environment Model

The reinforcement learning environment was summarized into three dictionaries: *distances*, *traffic* and *carparks*, obtained from a map of the university's main campus. The map was reviewed, and about two hundred (200) nodes were highlighted, as well as the edges between them.

The *distances* and *traffic* dictionaries were similar adjacency list implementations. Edge weights in the former were static and represented the distances in meters between nodes. Those in the latter represented the current number of cars on the road segments and were randomly generated and updated to simulate a continuous traffic flow.

The traffic categories were light, medium and heavy and reflected in the number of cars moving within the campus. The categories did not map to fixed numbers of cars but fell within ranges due to the randomization. The maximum number of traffic was kept below one hundred (100) so as not to affect the simulation performance with too many moving objects.

4.3 Environment Class

After a sufficient environment model was obtained, Gymnasium was used to provide systematic interactions between the agent and its environment. Gymnasium is a library that provides an Application Programming Interface (API) for single-agent reinforcement environments. It consists of different environments that an agent can interact with and allows for the creation of custom environments. A custom reinforcement learning environment class was created for the problem. The environment housed *distances*, *traffic* and *carparks*. It received the agent's actions, calculated, and returned its rewards and next states, among other information.

4.4 Q-Learning Algorithm, Q-Table and Hyperparameters

The Q-learning algorithm detailed in Section 2.3.2 was implemented using Python and used by the agent to interact with an instance of the custom reinforcement learning environment.

The Q-table was a dictionary, similar to *distances* and *traffic*, with entries representing Q-values of state- action pairs. Entries were initialized to 0 for possible actions and negative infinity for undesirable actions, such as leaving the campus.

The epsilon value exponentially decayed from 1 towards 0.1 during all training processes. A learning rate (α) of 0.5 and a discount factor (γ) of 0.6 were selected after hyperparameter tuning.

For hyperparameter tuning, training was carried out for each combination with an exponentially decaying epsilon at a rate of 0.00225 for two thousand (2,000) episodes. A cut-off was set at five hundred (500) steps per episode. The resulting agents were evaluated on their ability to find the nearest car park with the same one hundred (100) unique episodes of randomly generated start state and traffic category combinations.

4.5 Training and Evaluation

The final agent was trained using the selected hyperparameter values for fifty thousand (50,000) episodes with an epsilon value that decayed exponentially at a rate of about 0.00005. A cut-off was set at five hundred (500) steps per episode.

The following graphs display the agent's learning curve during training. As training progressed, the likelihood of exploitation increased. The agent accrued higher rewards and reached a car park in a smaller number of steps, as revealed in the graphs below.

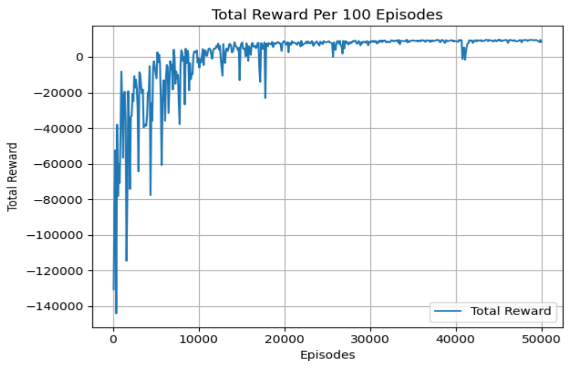


Figure 2: Graph of Total Reward per 100 Episodes

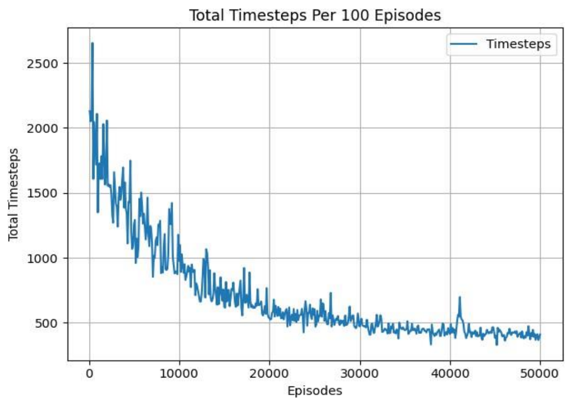


Figure 3: Graph of Total Time Steps per 100 Episodes

and fifty (150) unique episodes of randomly generated start state and traffic category combinations. The evaluation results indicated an optimized solution to the problem of navigating to the nearest car park within the campus.

ijkstra's algorithm computes the shortest path between a node and all other nodes in a graph. To confirm that the learned agent provided optimized navigation, both algorithms were evaluated with the same one hundred and fifty (150) unique episodes of randomly generated start state and traffic category combinations. The Q-learning algorithm used the Q-table to navigate, while Dijkstra's algorithm used the *distances* adjacency list to form paths. The summary of the evaluation is given below.

Table 1: Evaluation Metric Values for the Comparison of Q-learning with Dijkstra's Algorithm

Metrics	Q-Learning	Dijkstra's Algorithm
Avg. reward per episode	97.31	97.73
Avg. time steps per episode	3.27	3.49
Avg. distance per episode (meters)	153.01	120.95
Percentage of cars encountered (%)	1.16	1.24

The results suggested that Q-learning provided a reasonably optimized solution to the problem of navigating to the nearest car park, minimizing traffic and distance. Although Dijkstra's algorithm performed better for distance, Q-learning was not far behind. As for traffic, the agent performed better than Dijkstra's algorithm, which did not factor in traffic when selecting paths. Possible optimizations, such as an increased training period, the convergence of the algorithm, and/or an update in the reward function, would likely result in a boost in the Q-learning algorithm's performance.

4.6 Search for Available Car Parks (The Combined Q-Learning Solution)

After solving the reduced navigation problem, summarized in the resultant Q-table from training, a simple search algorithm was implemented to find an optimal route from a node to an available car park.

The search algorithm exploited the learned Q- values in the Q-table, greedily selecting the next highest-value state till it arrived at an available car park. It avoided paths to unavailable car parks as well as loops. The algorithm also failed to find a route if all available car parks were unreachable from the start state.

The Q-learning and combined Q-learning solutions were evaluated with the same one hundred and fifty (150) unique episodes of randomly generated start state and traffic category combinations. Multiple sets of the same 150 episodes were done for the combined solution, with a randomly generated set of available car parks per set.

The results of the combined solution for optimized navigation to the nearest available car park fell below that of the Q-learning agent for the reduced problem. Based on the evaluation, a contributing factor was the number of available car parks, with higher numbers resulting in performance closer to the Q-learning agent.

Another possible factor was the proximity of the start state to an available car park. The closer the two states were, the less likely the solution was to be affected by the car park availability constraint.

It was inferred that the suboptimal performance of the combined solution was due to the search algorithm exploiting Q-values that were no longer an accurate representation of the environment once the car park availability factor was added.

Once again, Dijkstra's algorithm was used to assess the performance of the combined Q-learning and search solution. They were evaluated with the same one hundred and fifty (150) unique episodes of randomly generated start state and traffic category combinations with a fixed set of available car parks throughout all episodes. The evaluation metric values are given below.

Table 2: Evaluation Metric Values for the Comparison of the Combined Q-Learning Solution with Dijkstra's Algorithm

Metrics	Combined Solution	Dijkstra's Algorithm
Avg. reward per episode	-155.69	85.46
Avg. time steps per episode	14.83	6.98
Avg. distance per episode (meters)	960.65	343.78
Percentage of cars encountered (%)	5.19	2.05

The search algorithm performed poorly compared to Dijkstra's algorithm. The result confirmed the hypothesis that exploiting the Q-values learned during training with all car parks available did not translate when only a set of car parks were available. It was concluded that the combined solution produced suboptimal navigation for the problem.

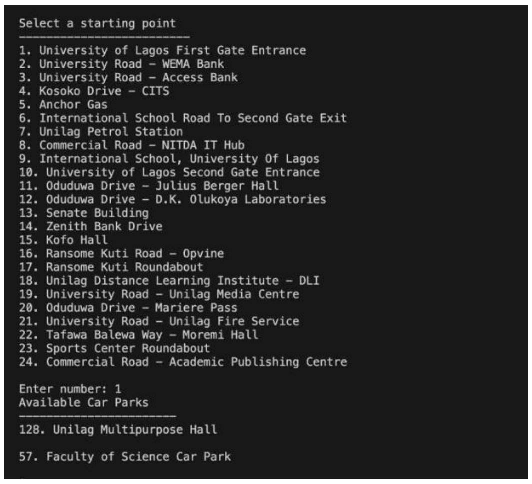
4.7 Simulation

The 2D simulation of the Q-learning agent and its environment was implemented using Pygame, a Python library for designing games. However, it was suitable for the simulation. The simulation consisted of a simple depiction of roads, buildings, and moving cars (including the agent). This was achieved by translating the values in the *distances* and *traffic* lists into road and car objects, respectively. The simulation showcased the agent as it navigated from a start location to an available car park using the route obtained from the search algorithm.

The simulation offered a more realistic traffic flow in that, rather than a general function adding numbers to and removing numbers from edges in traffic, as was done prior, each non-agent car randomly selected its next location after reaching the end of a road and updated traffic accordingly. However, traffic was still initially generated at random.

4.8 System Walkthrough

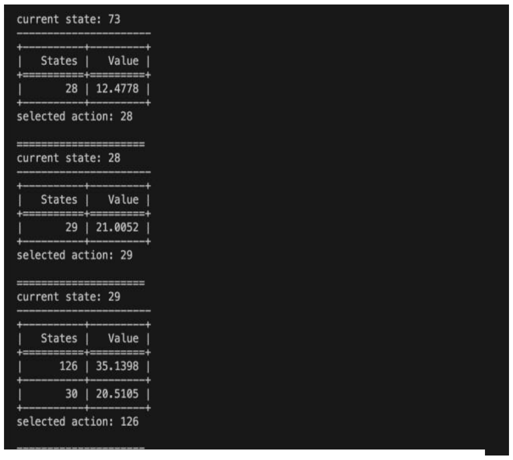
1. The system displays a set of start locations within the campus for the user to select from, as shown below. Each number associated with a location maps to a node identifier in the environment model. On receiving a valid number, the system randomly generates a set of available car parks ranging from 1 to all car parks and displays this to the user.



```
Select a starting point
-------------------------
1. University of Lagos First Gate Entrance
2. University Road - WEMA Bank
3. University Road - Access Bank
4. Kosoko Drive - CITS
5. Anchor Gas
6. International School Road To Second Gate Exit
7. Unilag Petrol Station
8. Commercial Road - NITDA IT Hub
9. International School, University Of Lagos
10. University of Lagos Second Gate Entrance
11. Oduduwa Drive - Julius Berger Hall
12. Oduduwa Drive - D.K. Olukoya Laboratories
13. Senate Building
14. Zenith Bank Drive
15. Kofo Hall
16. Ransome Kuti Road - Opvine
17. Ransome Kuti Roundabout
18. Unilag Distance Learning Institute - DLI
19. University Road - Unilag Media Centre
20. Oduduwa Drive - Mariere Pass
21. University Road - Unilag Fire Service
22. Tafawa Balewa Way - Moremi Hall
23. Sports Center Roundabout
24. Commercial Road - Academic Publishing Centre

Enter number: 1
Available Car Parks
------------------------
128. Unilag Multipurpose Hall

57. Faculty of Science Car Park
```

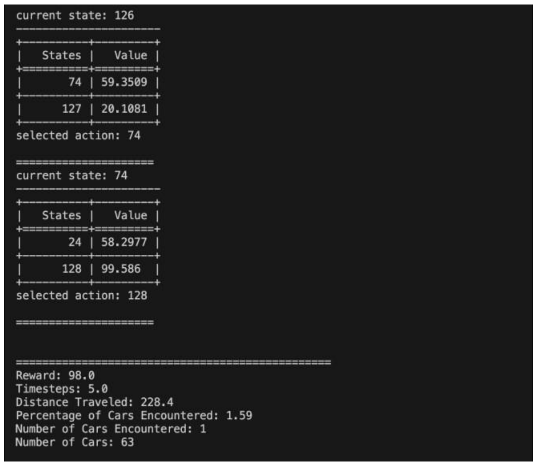
Figure 4: System Walkthrough Image 1

2. The identifier for the start location, i.e., the start state, is passed to the search function. The function searches the Q-table obtained from training for a route to the nearest available car park. If found, the system progresses to the next phase. Otherwise, it states it is unable to reach an available car park and exits.
3. The simulation showcases the agent car following the received route to its destination. The agent's rationale is shown in the console as the journey progresses, and the navigation metrics are displayed at the end.



```
current state: 73
----------------------
+----------+---------+
|   States |   Value |
+==========+=========+
|       28 | 12.4778 |
+----------+---------+
selected action: 28

====================
current state: 28
----------------------
+----------+---------+
|   States |   Value |
+==========+=========+
|       29 | 21.0052 |
+----------+---------+
selected action: 29

====================
current state: 29
----------------------
+----------+---------+
|   States |   Value |
+==========+=========+
|      126 | 35.1398 |
+----------+---------+
|       30 | 20.5105 |
+----------+---------+
selected action: 126

--------------------
```

Figure 5: System Walkthrough Image 2



```
current state: 126
----------------------
+----------+---------+
|   States |   Value |
+==========+=========+
|       74 | 59.3509 |
+----------+---------+
|      127 | 20.1081 |
+----------+---------+
selected action: 74

====================
current state: 74
----------------------
+----------+---------+
|   States |   Value |
+==========+=========+
|       24 | 58.2977 |
+----------+---------+
|      128 | 99.586  |
+----------+---------+
selected action: 128

====================


==================================================
Reward: 98.0
Timesteps: 5.0
Distance Traveled: 228.4
Percentage of Cars Encountered: 1.59
Number of Cars Encountered: 1
Number of Cars: 63
```

Figure 6: System Walkthrough Image 3

4.9 Effectiveness of Q-Learning as a Solution to the Problem of Optimized Navigation to the Nearest Available Car Park

The Q-learning agent was initially trained and tested with randomly generated sets of available car parks, in addition to the other random factors, i.e., the start state and the traffic, and a different reward function, on the assumption that the algorithm alone was sufficient for solving the problem. However, based on the unrecorded results, it was concluded that the static values learned could not handle the car park availability factor.

The scope of the Q-learning problem was then reduced to optimized navigation to the nearest car park, removing the car park availability aspect. The Q-learning algorithm was able to solve this problem, as was displayed in the evaluation. The combined solution with the search algorithm, believed to be able to address the remaining aspect of the problem, proved to be a suboptimal solution based on the results of its evaluation.

It is assumed that optimizing for better Q-learning performance might reflect in the combined solution's performance for the whole problem. However, it is likely the improvement will still not result in an optimal solution. Therefore, the conclusion was that Q-learning was not an effective enough solution to this problem.

5.0 SUMMARY AND CONCLUSION

5.1 Summary of Findings

(i) Model the University of Lagos main campus: The university's main campus was modeled as a graph with nodes representing areas of interest such as road intersections, entry and exit points to premises, and car parks. The model of the environment covered a broad scope of the primary areas of the campus at one hundred and ninety-one (191) nodes.

(ii) Implement the algorithm using this modeled environment to obtain a trained agent capable of navigating to the nearest available car park by minimizing distance and traffic: The Q-learning algorithm was used to train an agent using the modeled environment to provide optimized navigation from locations to the nearest car park by minimizing distance and traffic. A search algorithm handled the car park availability aspect and provided suboptimal navigation. Q-learning was deemed insufficient to meet the given objective despite using a different approach.

(iii) Evaluate the performance of the Q-learning solution: The agent's performance for the reduced problem was evaluated with metrics such as *average reward per episode*, *average time steps per episode*, *average distance traveled per episode*, *percentage of cars encountered*, and provided near-optimal results. The final solution with the search algorithm was also evaluated with the same metrics and performed suboptimally.

(iv) Develop a simulation to visualize the performance of the Q-learning solution: A simple 2D simulation was created to visualize the solution's performance in the modeled environment. The simulation was not seamless, and it would have been preferred if it had been used during training and evaluation for consistency.

5.2 CONCLUSION AND RE COMMENDATION

The focus of this study was on the optimized navigation component of a larger IoT-based car park system for the University of Lagos main campus. The study implemented and demonstrated the effectiveness of the Q-learning algorithm as a solution to the problem of navigating to the nearest available car park by minimizing distance and traffic. The Q-learning agent was trained and evaluated on a graph representation of the primary areas of the campus as an environment, with randomly generated environmental conditions to obtain an optimal policy for a reduced version of the problem, i.e., the car park availability aspect was removed, because the unrecorded attempt to solve the problem failed. The resulting policy was combined with a simple search algorithm to address the whole problem. The final solution achieved suboptimal navigation. Despite using a different approach, Q-learning was still not effective enough as a solution to the optimized navigation problem for the larger system.

The study constituted an initiative to provide smart parking management for the University of Lagos main campus to encourage its development into a smart city through a Q-learning solution developed for the problem of optimized navigation.

For future work, the other components of the IoT-based car park system, particularly in the area of the Internet of Things, need to be developed. By placing sensors at road intersections and within car parks, the system would be able to accurately track the flow of traffic and car park availability information in real time rather than the randomization approach used in this work.

REFERENCES

Even-Dar, E. & Mansour, Y. (2003). Learning Rates for Q-learning. *Journal of Machine Learning Research*, *5*, 1–25. Retrieved from https://www.jmlr.org/papers/volume5/evendar03a/evendar03a.pdf

Harmon, M. E. & Harmon, S. S. (1997). *Reinforcement Learning: A Tutorial.* Retrieved from https://apps.dtic.mil/sti/pdfs/ADA323194.pdf

Hu, Y., Yang, L. & Lou, Y. (2021). Path Planning with Q-Learning. *Journal of Physics: Conference Series*, *1948*(1), 012038. https://doi.org/10.1088/ 1742-6596/1948/1/012038

Ibrahim, M. M. S., Atia, M. R. & Fakhr, M. (2021). Autonomous Vehicle Path Planning using Q-Learning. *Journal of Physics: Conference Series*, *2128*(1), 012018. https://doi.org/10.1088/1742-6596/2128/1/012018

Jang, B., Kim, M., Harerimana, G. & Kim, J. W. (2019). Q-Learning Algorithms: A Comprehensive Classification and applications. *IEEE Access*, *7*, 133653–133667. https://doi.org/10.1109/access.2019.2941229

Sutton, R. S. & Barto, A. G. (2015). *Reinforcement learning: An introduction (2nd ed., in progress).* Retrieved from https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3–4), 279–292. https://doi.org/10.1007/bf00992698

BERT-BASED SENTIMENT ANALYSIS ON THE CRYPTOCURRENCY MARKET WITH BINANCE NIGERIA

Akinbo R.S.[1] and Oladejo M.O.[2]
[1]Department of Data Science, School of Computing, Federal University of Technology, Akure, Nigeria
[2]Department of Information Technology, School of Computing, Federal University of Technology, Akure, Nigeria

ABSTRACT
This research explores the public sentiment surrounding the Nigerian Federal Government's recent decision to halt the operations of Binance Nigeria, a major cryptocurrency exchange. By employing the BERT Uncased Sentiment Analyzer, we examined a wide array of user-generated tweets to capture the attitudes and reactions of Binance Nigeria's user base. The analysis revealed a predominant sense of dissatisfaction among users: 80% of the analyzed tweets conveyed negative opinions toward the government's decision, viewing it as restrictive and unfavourable to their financial interests. Many users expressed concerns that the ban limits their ability to participate in digital asset trading, a domain they consider integral to their economic freedom and investment strategies. In contrast, only 15% of users supported the government's decision, aligning with regulatory and risk-related justifications, while 5% of users remained neutral on the matter. This distribution underscores a strong divide in the cryptocurrency community's perception of governmental intervention, with a majority opposing measures they feel are antithetical to the decentralized and open principles of cryptocurrency. The study highlights the value of sentiment analysis in gauging public reaction to regulatory policies within the digital finance sector. It also points to the need for balanced regulatory approaches that address security concerns without stifling innovation and user participation in the evolving crypto market. These insights offer relevant data for policymakers and industry stakeholders who seek to harmonize regulatory policies with public opinion. This analysis contributes to a growing body of research that examines the impact of government regulations on user sentiment in the digital economy, particularly in emerging markets like Nigeria where cryptocurrency adoption has been robust. The findings emphasize the importance of user-centric perspectives in crafting policies that promote sustainable digital finance practices.

Keywords: Bert, Sentiment analysis, Opinion mining, Cryptocurrency, Binance Nigeria.

1.0 INTRODUCTION

Sentiment analysis, noted as opinion mining, is a great technique in the natural language processing (NLP) parlance that aims to reveal the emotional hints rooted within the written text (Nick, 2023). Generally, this technique is adopted across industries, this method helps establishments interpret and classify public thoughts about products, services, or ideas. This is done by integrating data mining, machine learning (ML), artificial intelligence (AI), and computational linguistics. Sentiment analysis classifies subjective indications while determining whether a piece of text expresses a positive, negative, or neutral sentiment (Nick, 2023). This ability empowers businesses to gain real-time understandings into customer behavioral attitudes, experiences, and perceptions, ultimately determining strategies around brand reputation and customer engagement (Barney, 2023). Binance is an online exchange where users can trade Cryptocurrencies. It supports hundreds of the most commonly traded Cryptocurrencies. (Peters, 2023). The Binance Exchange is a leading cryptocurrency exchange founded in 2017. It features a strong focus on altcoin trading. Binance offers crypto-to-crypto trading in more than 350 cryptocurrencies and virtual tokens, including bitcoin (BTC), etherum (ETH), litecoin (LTC), dogecoin (DOGE), and its own coin, BNB. (Peters, 2023).

This paper aims to conduct a Twitter data sentiment analysis on the Cryptocurrency market using Binance Nigeria, using the BERT Large Language Model.

2.0 RELATED WORKS

Roumeliotis et al (2024) carried out a LLMs and NLP Models in Cryptocurrency Sentiment Analysis: A Comparative Classification Study. Since Cryptocurrencies are becoming increasingly prominent in financial investments, attracting investors with their ease of use and decentralized opportunities. This paper examines the role of sentiment dynamics in the cryptocurrency market, highlighting how news and investor sentiment, known as crypto signals, influence risks and rewards. It also discusses the implications of sentiment analysis for investment decisions and risk management in financial markets.

Limonad et al (2024) implemented an in-Monetizing Currency Pair Sentiments through LLM Explainability. This work develops a novel technique using LLMs as a post-hoc, model-independent tool for explainability in sentiment analysis within the financial domain. By merging news feed data with market prices, the technique enriched ML inputs, improving currency-pair price predictions and serving as a viable alternative to conventional explainable AI methods.

Dowey (2024) carried out an investigation in Unveiling Economic Sentiment: Using a Large Language Model for Economic Sentiment Analysis from Monetary Policy Reports. They explores the use of LLaMa-2-7b-chat-hf for economic sentiment analysis using Riksbank's monetary policy reports. By fine-tuning the model with minimal data, it develops the Net Score Index (NSI) and Net Monetary Policy Index (NMPI), which correlate strongly with GDP and CPIF changes in Sweden (2014–2023). Positive sentiments were found to increase Swedish Treasury bill yields, while negative ones influenced SEK/USD exchange rates. The NMPI also predicts future policy rate changes, validating the Riksbank's forward guidance. Limitations include reliance on minimal annotated data, context specificity, and dependency on a single LLM model.

Shi, et al. (2024) proposed an EUR/USD Exchange Rate Forecasting incorporating Text Mining Based on Pre-trained Language Models and Deep Learning Methods. The proposes a PSO-LSTM model for EUR/USD exchange rate forecasting, combining deep learning, textual analysis, and particle swarm optimization. Using online news and sentiment analysis with RoBERTa-Large, it outperforms traditional models like SVM, ARIMA, and GARCH. Ablation tests highlight the value of textual data in improving forecasts, showcasing AI's transformative role in financial analysis.

Liu, et al (2024), did a review on Large Language Models and Sentiment Analysis in Financial Markets, based on Datasets and Case Study (2024) explore the use of Large Language Models (LLMs) for financial sentiment analysis, specifically analyzing the correlation between news sentiment and Bitcoin prices. It finds a modest link, with historical news having a greater impact on long-term Bitcoin price fluctuations than immediate news events, demonstrating LLMs' potential in market trend prediction and investment decisions.

Liu et al (2025) in LLM-driven sentiment analysis for corporate misconduct prediction. The authors developed MDARisk, a novel framework for corporate misconduct prediction. The framework is a MultiSenti module, which leverages a multi-agent LLM approach to extract comprehensive and contextual sentiment from MD&A.The approach demonstrate that it can provides a more reliable sentiment-based indicator for misconduct risk, achieves higher predictive accuracy, and outperforms the traditional financial sentiment analysis approach. The approach enhances prediction models.

Thomas (2024) in Enhancing TinyBERT for Financial Sentiment Analysis Using GPT-Augmented FinBERT Distillation. The work enhances financial sentiment analysis by using GPT-4 Omni to generate synthetic data, improving FinBERT and creating TinyFinBERT, a smaller, more efficient model. Through a two-tiered knowledge distillation strategy, TinyFinBERT achieves performance similar to FinBERT while being more computationally efficient. This demonstrates how LLMs can improve smaller models for financial sentiment analysis through innovative data augmentation and distillation techniques.

Kang (2024) et al in Cross-border Relationship between Cryptocurrency Prices and News Sentiment: Which Exerts Influence on the Other? The study analyzes the cross-border relationship between cryptocurrency prices and news sentiment, finding that price changes influence sentiment, not the other way around. It also shows that Korean news sentiment can predict cryptocurrency returns, but this power fades when Korean won trading volume surpasses US dollar volume, synchronizing prices across markets.

Deroy and Maity (2024) carried out an investigation in CryptoLLM: Unleashing the Power of Prompted LLMs for SmartQnA and Classification of Crypto Posts develops classification models for categorizing cryptocurrency-related social media posts using GPT-4-Turbo with prompt-based and 64-shot techniques. It also identifies relevant answers to questions, enhancing the filtering of cryptocurrency discourse to support informed decision-making (Nick, 2023).

Han, Guo, Chen, Wang & Guo (2024) in LEST: Large language models and spatio-temporal data analysis for enhanced Sino-US exchange rate forecasting develops the LEST framework for real-time exchange rate forecasting by combining ChatGPT-based sentiment analysis with spatio-temporal models (STGCN, GRU) and forecasting models (Transformer, LSTM). The framework accurately captures exchange rate fluctuations during the Sino-US trade war, improving upon traditional methods that rely on offline emotional lexicons.

Table 1. Author and their method

Author	Method used
Roumeliotis, K. I., et al (2024)	GPT-4, BERT, and FinBERT
Limonad, L., et al (2024)	Large language models (LLMs) and Machine Learning
Dowey, L (2024)	LLaMa-2-7b-chat-hf
Shi, X., et al (2024)	RoBERTa-Large
Liu, C., et al. (2024)	Large language models (LLMs
Gijsbertha, Z. P. (2024)	Multi-agent chatbot framework
Thomas, G. J. (2024)	GPT-4 Omni, TinyFinBERT, FinBERT
Kang, R. et al (2024)	ChatGPT
Deroy, A. & Maity, S. (2024)	GPT-4-Turbo
Han, D., et al 2024	ChatGPT, Spatio-Temporal Graph Convolutional Networks (STGCN) and Gated Recurrent Unit (GRU)

3.0 METHODOLOGY

The purpose of this research is to examine the attitudes of cryptocurrency users in Nigeria who use Binance. To do this, data from Twitter will be gathered, and Large Language Model sentiment approach will be used to categorize the comments into three categories: positive, negative, and neutral using Bert Uncased Sentiment Classifier.

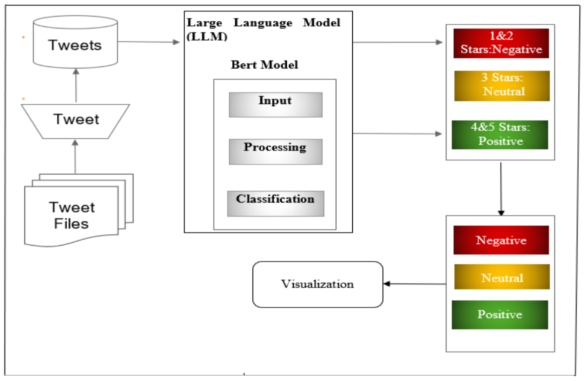


Figure 1: Conceptual Analysis of the operation

Sentiment analysis classification into positive, negative, or neutral sentiments can be done using several methods like Machine Learning approach with machine learning algorithms like Linear SVM, Naïve Bayes, and Linear Regression, expert system and using pretrained Large Language Models (LLM) which is a subset of Natural Language Processing (NLP). The approach used to carry out this sentiment analysis was LLM approach being the top-performing models, delivering outstanding results and high efficiency in sentiment classification tasks. The LLM model used was Bert-uncased-sentiment-analysis. This model was chosen because of its high accuracy, ease of use and support of multiples languages in classification of user's sentiments.

The mathematical interpretation:

Let:

T: Input text (for a sentence)

$X = \{x_1, x_2, \ldots, x_n\}$: Tokenized input sequence (1)

$E = \{e_1, e_2, \ldots, e_n\}$: Embeddings from BERT for each token (2)

$h_{[CLS]}$: Final hidden state of the special [CLS] token (used for classification)

W: Weight matrix of the classification layer

b: Bias vector of the classification layer

$C = \{c_1, c_2, c_3\}$: Sentiment classes →

C1 = Negative Sentiments (3)

C2 = Neutral Sentiments (4)

C3 = Positive Sentiments

Tokenization and Embedding:

$X \rightarrow \text{BERT} \rightarrow h_{[CLS]}$

The input text is tokenized and passed through BERT. The output vector $h_{[CLS]} \in \mathbb{R}^d$ represents the entire sentence.

2. Classification Layer

$$z = W \cdot h_{[CLS]} + b \qquad (5)$$

Where:

$z \in \mathbb{R}^3$: Logits for the three sentiment classes

$W \in \mathbb{R}^{3\times d}$, $b \in \mathbb{R}^3$

3. Softmax for Probability Distribution

$$P(c_i \mid T) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}} \text{ for } i = 1,2,3 \qquad (6)$$

This gives the probability of each sentiment class.

4. Prediction

$$\hat{y} = \arg\max_{c_i \in C} P(c_i \mid T) \qquad (7)$$

The predicted sentiment is the class with the highest probability. Which wiil based on the following:

Table 2: Output Classes

Class	Meaning
c_1	Negative sentiment
c_2	Neutral sentiment
c_3	Positive sentiment

4.0 DATA COLLECTION AND ANALYSIS

The total number of data collected from X was more than 4,000, the collection date ranges from 20th of February, 2024 to 3rd of March, 2024 from Twitter, (formerly Twitter) to assess the public sentiment toward Binance in Nigeria. The tweets were recognised using targeted keywords related to Binance, cryptocurrency, and Nigeria, to ensure relevance to the topic. The raw data were pre-processed, a phase which involved removing spam, duplicates, and irrelevant attaches, and standardizing text formats, such as conversion of slang, emojis, and abbreviations into analyzable formats. The key linguistic features were adequately extracted to improve the performance of machine learning models. Each tweet was labeled as positive, negative or neutral, based on the emotional tenor and attitude articulated toward Binance. The labeling process placed the groundwork for training the sentiment classification models using natural language processing (NLP) techniques.

The application of these models was able to automatically detect and classify public sentiment, to give a valuable insight into the notions of Nigerian users as they perceive Binance during the specified timeframe. The result analysis did not only highlight prevailing tone based on negative, positive and neutral, but also revealed the emerging trends and concerns within the Nigerian cryptocurrency community, contributing to a greater understanding of the social dynamics that surrounds the digital finance in the region.

Bert model used classified tweets into stars, the stars were further converted to Negative, Neutral and Positive using the number of stars the tweet has. Tweets with 1 & 2 star(s) are Negative, tweets with 3 stars are Neutral and tweets with 4 & 5 stars are Positive tweets.Transformed sentiment analysis results into engaging visualizations, the visualization tools leveraged was Bar Chart, this chart was chosen to facilitate a clearer understanding and interpretation of the insights extracted from the Twitter data, enabling the discovery of patterns, trends, and correlations that informed business decisions and strategic actions for Binance in Nigeria and know the percentage of people that felt someway about the ban of Binance in Nigeria.

4.1 Implementation Analysis

This project is crucial for sentiment analysis of finance users leveraging tweets to get users opinion about the banning of Binance in Nigeria. By leveraging LLM, the model processes tweets data to analyse users' opinion to know if the particular tweet is Positive or Negative. Key methods employed include data reading, data cleaning and data processing. The objective is to attain high sentiment analysis accuracy. The data used for training and testing is primarily derived from tweets obtained from X (formerly X).

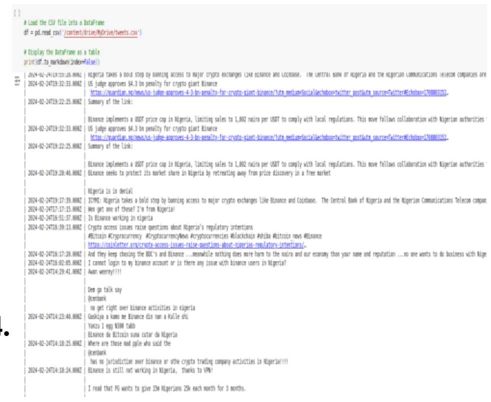
4.2 Data Reading

The collected Twitter data was saved in a csv file and was loaded to the model using Pandas library for processing and used for the sentiment

Figure 2: Data Reading

```
# Load the dataset
df = pd.read_csv('tweets.csv')
```
```
[57]  ✓ 0.4s                                                     Python
```

analysis.



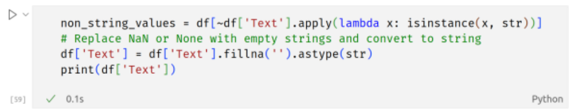
```
[ ]
# Load the CSV file into a DataFrame
df = pd.read_csv('/content/drive/MyDrive/tweets.csv')

# Display the DataFrame as a table
print(df.to_markdown(index=False))
```
```
| 2024-02-24T19:35:28.000Z | Nigeria takes a bold step by banning access to major crypto exchanges like Binance and Coinbase.  The Central Bank of Nigeria and the Nigerian Communications Telecom companies are
| 2024-02-24T19:32:33.000Z | US judge approves $4.3 bn penalty for crypto giant Binance
|                          |  https://guardian.ng/news/us-judge-approves-4-3-bn-penalty-for-crypto-giant-binance/?utm_medium=Social&echobox=twitter_post&utm_source=Twitter#Echobox=1708803352.
| 2024-02-24T19:22:25.000Z | Summary of the link:
|                          |
|                          | Binance implements a USDT price cap in Nigeria, limiting sales to 1,802 naira per USDT to comply with local regulations. This move follows collaboration with Nigerian authorities
| 2024-02-24T19:32:33.000Z | US judge approves $4.3 bn penalty for crypto giant Binance
|                          |  https://guardian.ng/news/us-judge-approves-4-3-bn-penalty-for-crypto-giant-binance/?utm_medium=Social&echobox=twitter_post&utm_source=Twitter#Echobox=1708803352.
| 2024-02-24T19:22:25.000Z | Summary of the link:
|                          |
|                          | Binance implements a USDT price cap in Nigeria, limiting sales to 1,802 naira per USDT to comply with local regulations. This move follows collaboration with Nigerian authorities
| 2024-02-24T19:20:46.000Z | Binance seeks to protect its market share in Nigeria by retreating away from price discovery in a free market
|                          |
|                          | Nigeria is in denial
| 2024-02-24T19:17:39.000Z | ICYMI: Nigeria takes a bold step by banning access to major crypto exchanges like Binance and Coinbase.  The Central Bank of Nigeria and the Nigerian Communications Telecom compan
| 2024-02-24T17:17:15.000Z | Wen get one of these? I'm from Nigeria!
| 2024-02-24T16:51:37.000Z | Is Binance working in nigeria
| 2024-02-24T16:39:13.000Z | Crypto access issues raise questions about Nigeria's regulatory intentions
|                          | #Bitcoin #Cryprocurrency  #CryptocurrencyNews #cryptocurrencies #blockchain #shiba #bitcoin news #Binance
|                          | https://coinletter.org/crypto-access-issues-raise-questions-about-nigerias-regulatory-intentions/.
| 2024-02-24T16:17:20.000Z | And they keep chasing the BDC's and Binance ...meanwhile nothing does more harm to the naira and our economy than your name and reputation ...no one wants to do business with Nige
| 2024-02-24T16:02:05.000Z | I cannot login to my binance account or is there any issue with binance users in Nigeria?
| 2024-02-24T14:29:41.000Z | Awon weerey!!!!
|                          |
|                          | Dem go talk say
|                          | @cenbank
|                          |  no get right over binance activities in nigeria
| 2024-02-24T14:23:40.000Z | Gaskiya a kamo me Binance din nan a Kulle shi
|                          | Yanzu 1 egg N300 tabb
|                          | Binance da Bitcoin suna cutar da Nigeria
| 2024-02-24T14:18:25.000Z | Where are these mad pple who said the
|                          | @cenbank
|                          |  has no jurisdiction over binance or othe crypto trading company activities in Nigeria!!!
| 2024-02-24T14:18:24.000Z | Binance is still not working in Nigeria,  thanks to VPN!
|                          |
|                          | I read that FG wants to give 15m Nigerians 25k each month for 3 months.
```

4.

Figure 3: Collected Tweets

Before reading the data, all collected tweets were collated into a single file and duplicates were removed alongside tweets that are not related to the topic under review.

After reading the data, the data was cleaned to replace empty strings in the text column with NaN or None and also the texts to string for records with other data types, to aid better analysis for the LLM.



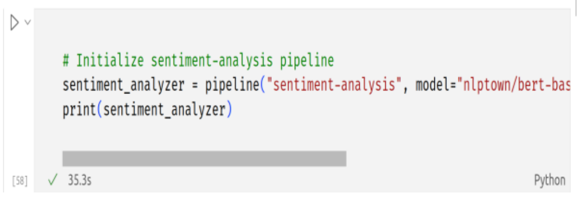
```
non_string_values = df[~df['Text'].apply(lambda x: isinstance(x, str))]
# Replace NaN or None with empty strings and convert to string
df['Text'] = df['Text'].fillna('').astype(str)
print(df['Text'])
```

```
[59]  ✓ 0.1s                                                     Python
```

Figure 4: Data Cleaning Algorithm

4.4 Data Processing

After cleaning the data,the LLM model was initiated, the LLM used was nlptown/bert-base-multilingual-uncased-sentiment. This model was chosen because of its ability to process and perform sentiment analysis on multi languages.

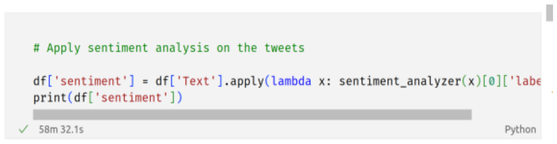


```
# Initialize sentiment-analysis pipeline
sentiment_analyzer = pipeline("sentiment-analysis", model="nlptown/bert-bas
print(sentiment_analyzer)
```

```
[58]  ✓ 35.3s                                                    Python
```

Figure 5: Initializing Sentiment Analysis Model

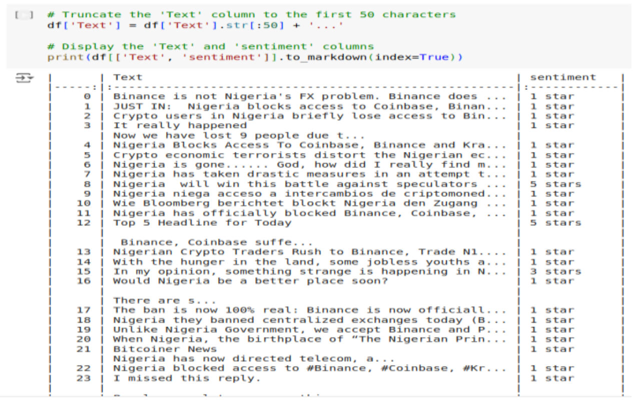
The Bert Based sentiment analyser perform sentiment analysis and predicts the sentiment of the text as a number of stars (between 1 and 5) base on the content of the text.



```
# Apply sentiment analysis on the tweets

df['sentiment'] = df['Text'].apply(lambda x: sentiment_analyzer(x)[0]['labe
print(df['sentiment'])
```

```
[60]  ✓ 58m 32.1s                                                Python
```

Figure 6: Applying Sentiment Analysis Model

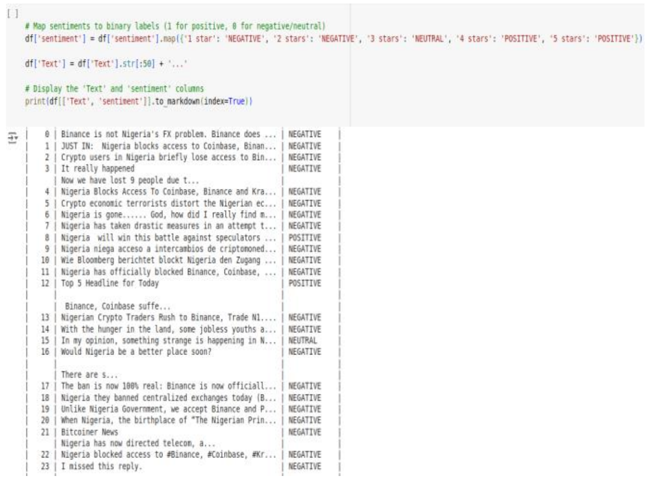


```
# Truncate the 'Text' column to the first 50 characters
df['Text'] = df['Text'].str[:50] + '...'

# Display the 'Text' and 'sentiment' columns
print(df[['Text', 'sentiment']].to_markdown(index=True))
```

```
|      | Text                                                  | sentiment   |
|-----:|:------------------------------------------------------|:------------|
|    0 | Binance is not Nigeria's FX problem. Binance does ... | 1 star      |
|    1 | JUST IN:  Nigeria blocks access to Coinbase, Binan... | 1 star      |
|    2 | Crypto users in Nigeria briefly lose access to Bin... | 1 star      |
|    3 | It really happened                                    | 1 star      |
|      | Now we have lost 9 people due t...                    |             |
|    4 | Nigeria Blocks Access To Coinbase, Binance and Kra... | 1 star      |
|    5 | Crypto economic terrorists distort the Nigerian ec... | 1 star      |
|    6 | Nigeria is gone...... God, how did I really find m... | 1 star      |
|    7 | Nigeria has taken drastic measures in an attempt t... | 1 star      |
|    8 | Nigeria  will win this battle against speculators ... | 5 stars     |
|    9 | Nigeria niega acceso a intercambios de criptomoned... | 1 star      |
|   10 | Wie Bloomberg berichtet blockt Nigeria den Zugang ... | 1 star      |
|   11 | Nigeria has officially blocked Binance, Coinbase, ... | 1 star      |
|   12 | Top 5 Headline for Today                              | 5 stars     |
|      |                                                       |             |
|      |  Binance, Coinbase suffe...                           |             |
|   13 | Nigerian Crypto Traders Rush to Binance, Trade N1.... | 1 star      |
|   14 | With the hunger in the land, some jobless youths a... | 1 star      |
|   15 | In my opinion, something strange is happening in N... | 3 stars     |
|   16 | Would Nigeria be a better place soon?                 | 1 star      |
|      |                                                       |             |
|      | There are s...                                        |             |
|   17 | The ban is now 100% real: Binance is now officiall... | 1 star      |
|   18 | Nigeria they banned centralized exchanges today (B... | 1 star      |
|   19 | Unlike Nigeria Government, we accept Binance and P... | 1 star      |
|   20 | When Nigeria, the birthplace of "The Nigerian Prin... | 1 star      |
|   21 | Bitcoiner News                                        | 1 star      |
|      | Nigeria has now directed telecom, a...                |             |
|   22 | Nigeria blocked access to #Binance, #Coinbase, #Kr... | 1 star      |
|   23 | I missed this reply.                                  | 1 star      |
|      |                                                       |             |
```

Figure 7: Sentiments transformed to stars with the LLM Model

After application of the LLM model, the number of stars of each tweet is then mapped to a binary number, if a tweet has 1 or 2 Stars, it is mapped to NEGATIVE, tweets with 3 stars are mapped to NEUTRAL and those with 4 and 5 Stars are mapped to POSITIVE.



```
# Map sentiments to binary labels (1 for positive, 0 for negative/neutral)
df['sentiment'] = df['sentiment'].map({'1 star': 'NEGATIVE', '2 stars': 'NEGATIVE', '3 stars': 'NEUTRAL', '4 stars': 'POSITIVE', '5 stars': 'POSITIVE'})

df['Text'] = df['Text'].str[:50] + '...'

# Display the 'Text' and 'sentiment' columns
print(df[['Text', 'sentiment']].to_markdown(index=True))
```

```
|    0 | Binance is not Nigeria's FX problem. Binance does ... | NEGATIVE    |
|    1 | JUST IN:  Nigeria blocks access to Coinbase, Binan... | NEGATIVE    |
|    2 | Crypto users in Nigeria briefly lose access to Bin... | NEGATIVE    |
|    3 | It really happened                                    | NEGATIVE    |
|      | Now we have lost 9 people due t...                    |             |
|    4 | Nigeria Blocks Access To Coinbase, Binance and Kra... | NEGATIVE    |
|    5 | Crypto economic terrorists distort the Nigerian ec... | NEGATIVE    |
|    6 | Nigeria is gone...... God, how did I really find m... | NEGATIVE    |
|    7 | Nigeria has taken drastic measures in an attempt t... | NEGATIVE    |
|    8 | Nigeria  will win this battle against speculators ... | POSITIVE    |
|    9 | Nigeria niega acceso a intercambios de criptomoned... | NEGATIVE    |
|   10 | Wie Bloomberg berichtet blockt Nigeria den Zugang ... | NEGATIVE    |
|   11 | Nigeria has officially blocked Binance, Coinbase, ... | NEGATIVE    |
|   12 | Top 5 Headline for Today                              | POSITIVE    |
|      |                                                       |             |
|      |  Binance, Coinbase suffe...                           |             |
|   13 | Nigerian Crypto Traders Rush to Binance, Trade N1.... | NEGATIVE    |
|   14 | With the hunger in the land, some jobless youths a... | NEGATIVE    |
|   15 | In my opinion, something strange is happening in N... | NEUTRAL     |
|   16 | Would Nigeria be a better place soon?                 | NEGATIVE    |
|      |                                                       |             |
|      | There are s...                                        |             |
|   17 | The ban is now 100% real: Binance is now officiall... | NEGATIVE    |
|   18 | Nigeria they banned centralized exchanges today (B... | NEGATIVE    |
|   19 | Unlike Nigeria Government, we accept Binance and P... | NEGATIVE    |
|   20 | When Nigeria, the birthplace of "The Nigerian Prin... | NEGATIVE    |
|   21 | Bitcoiner News                                        | NEGATIVE    |
|      | Nigeria has now directed telecom, a...                |             |
|   22 | Nigeria blocked access to #Binance, #Coinbase, #Kr... | NEGATIVE    |
|   23 | I missed this reply.                                  | NEGATIVE    |
```

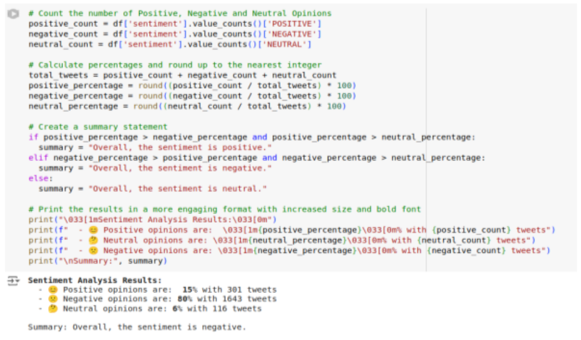
Figure 8: Mapping Sentiments using their number of Stars

4.5 Result

A comprehensive sentiment analysis was conducted on a dataset of 2,060 tweets, utilizing a BERT-based model. The model assigned star ratings to each tweet, which were subsequently used to categorize the tweets into three distinct sentiment classes: negative, neutral, and positive.

Classifications	Counts
Negative	1,643
Neutral	116
Positive	301

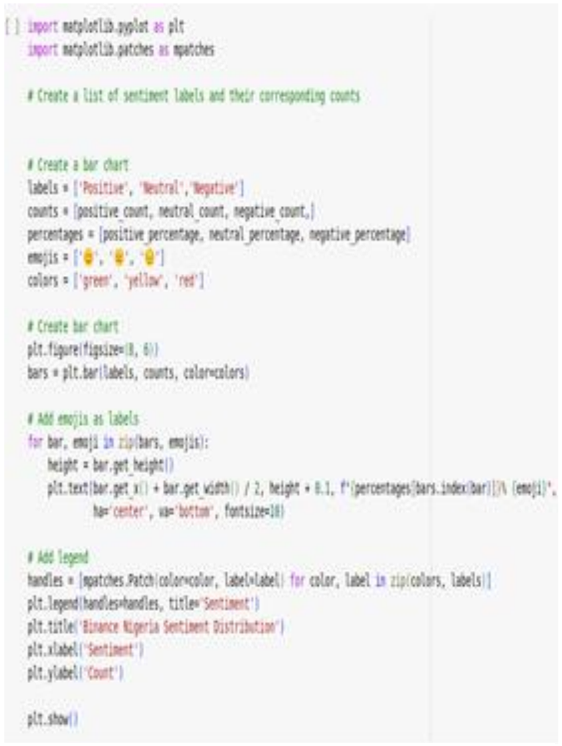
Table 2: Counts of Each Classification



```
# Count the number of Positive, Negative and Neutral Opinions
positive_count = df['sentiment'].value_counts()['POSITIVE']
negative_count = df['sentiment'].value_counts()['NEGATIVE']
neutral_count = df['sentiment'].value_counts()['NEUTRAL']

# Calculate percentages and round up to the nearest integer
total_tweets = positive_count + negative_count + neutral_count
positive_percentage = round((positive_count / total_tweets) * 100)
negative_percentage = round((negative_count / total_tweets) * 100)
neutral_percentage = round((neutral_count / total_tweets) * 100)

# Create a summary statement
if positive_percentage > negative_percentage and positive_percentage > neutral_percentage:
  summary = "Overall, the sentiment is positive."
elif negative_percentage > positive_percentage and negative_percentage > neutral_percentage:
  summary = "Overall, the sentiment is negative."
else:
  summary = "Overall, the sentiment is neutral."

# Print the results in a more engaging format with increased size and bold font
print("\033[1mSentiment Analysis Results:\033[0m")
print(f"  - 😐 Positive opinions are:  \033[1m{positive_percentage}\033[0m% with {positive_count} tweets")
print(f"  - 🤔 Neutral opinions are: \033[1m{neutral_percentage}\033[0m% with {neutral_count} tweets")
print(f"  - 🙁 Negative opinions are: \033[1m{negative_percentage}\033[0m% with {negative_count} tweets")
print("\nSummary:", summary)
```

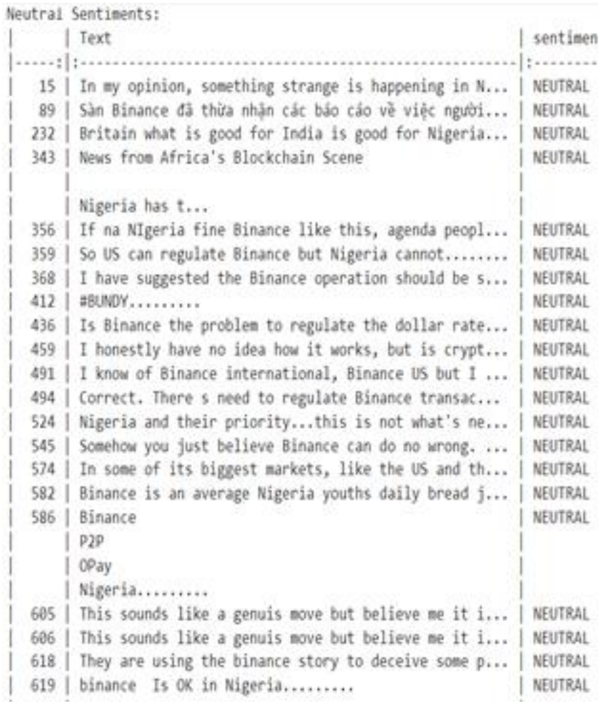
```
Sentiment Analysis Results:
  - 😐 Positive opinions are:  15% with 301 tweets
  - 🙁 Negative opinions are: 80% with 1643 tweets
  - 🤔 Neutral opinions are: 6% with 116 tweets

Summary: Overall, the sentiment is negative.
```

Figure 9: Counting numbers of each Classification



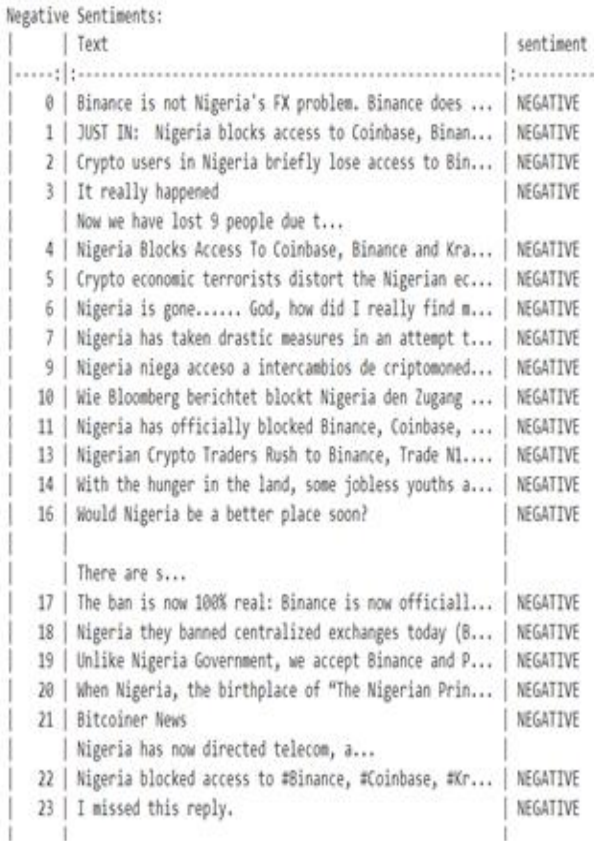
```
[ ] import matplotlib.pyplot as plt
    import matplotlib.patches as mpatches

    # Create a list of sentiment labels and their corresponding counts


    # Create a bar chart
    labels = ['Positive', 'Neutral','Negative']
    counts = [positive_count, neutral_count, negative_count,]
    percentages = [positive_percentage, neutral_percentage, negative_percentage]
    emojis = ['😊', '😐', '😞']
    colors = ['green', 'yellow', 'red']

    # Create bar chart
    plt.figure(figsize=(8, 6))
    bars = plt.bar(labels, counts, color=colors)

    # Add emojis as labels
    for bar, emoji in zip(bars, emojis):
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width() / 2, height + 0.1, f'{percentages[bars.index(bar)]}%\n{emoji}',
                 ha='center', va='bottom', fontsize=10)

    # Add legend
    handles = [mpatches.Patch(color=color, label=label) for color, label in zip(colors, labels)]
    plt.legend(handles=handles, title='Sentiment')
    plt.title('Binance Nigeria Sentiment Distribution')
    plt.xlabel('Sentiment')
    plt.ylabel('Count')

    plt.show()
```

Figure 10: Classification of Sentiments



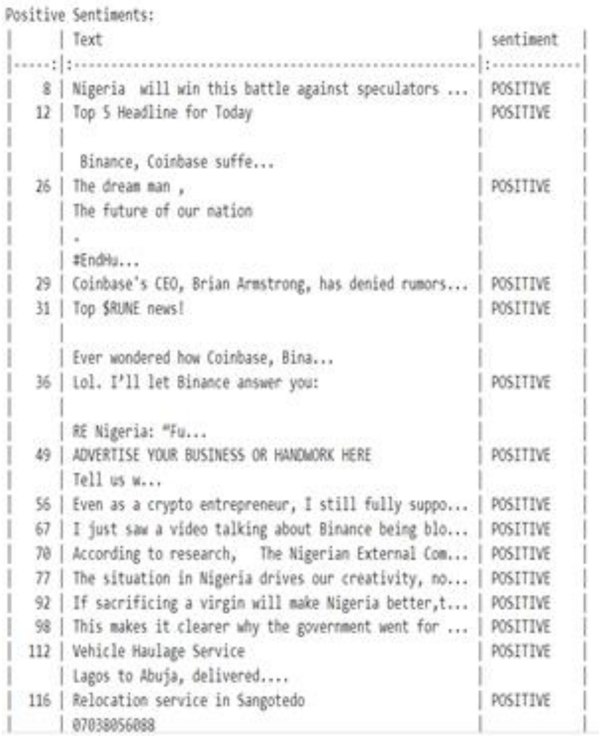
```
Neutral Sentiments:
|     | Text                                                  | sentimen
|----:|:------------------------------------------------------|:--------
|  15 | In my opinion, something strange is happening in N... | NEUTRAL
|  89 | Sàn Binance đã thừa nhận các báo cáo về việc người... | NEUTRAL
| 232 | Britain what is good for India is good for Nigeria... | NEUTRAL
| 343 | News from Africa's Blockchain Scene                   | NEUTRAL
|     |                                                       |
|     | Nigeria has t...                                      |
| 356 | If na NIgeria fine Binance like this, agenda peopl... | NEUTRAL
| 359 | So US can regulate Binance but Nigeria cannot........ | NEUTRAL
| 368 | I have suggested the Binance operation should be s... | NEUTRAL
| 412 | #BUNDY.........                                       | NEUTRAL
| 436 | Is Binance the problem to regulate the dollar rate... | NEUTRAL
| 459 | I honestly have no idea how it works, but is crypt... | NEUTRAL
| 491 | I know of Binance international, Binance US but I ... | NEUTRAL
| 494 | Correct. There s need to regulate Binance transac...  | NEUTRAL
| 524 | Nigeria and their priority...this is not what's ne... | NEUTRAL
| 545 | Somehow you just believe Binance can do no wrong. ... | NEUTRAL
| 574 | In some of its biggest markets, like the US and th... | NEUTRAL
| 582 | Binance is an average Nigeria youths daily bread j... | NEUTRAL
| 586 | Binance                                               | NEUTRAL
|     | P2P                                                   |
|     | OPay                                                  |
|     | Nigeria.........                                      |
| 605 | This sounds like a genuis move but believe me it i... | NEUTRAL
| 606 | This sounds like a genuis move but believe me it i... | NEUTRAL
| 618 | They are using the binance story to deceive some p... | NEUTRAL
| 619 | binance  Is OK in Nigeria.........                    | NEUTRAL
```

Figure 12: Neutral Sentiment



```
Negative Sentiments:
|    | Text                                                  | sentiment
|---:|:------------------------------------------------------|:---------
|  0 | Binance is not Nigeria's FX problem. Binance does ... | NEGATIVE
|  1 | JUST IN:  Nigeria blocks access to Coinbase, Binan... | NEGATIVE
|  2 | Crypto users in Nigeria briefly lose access to Bin... | NEGATIVE
|  3 | It really happened                                    | NEGATIVE
|    | Now we have lost 9 people due t...                    |
|  4 | Nigeria Blocks Access To Coinbase, Binance and Kra... | NEGATIVE
|  5 | Crypto economic terrorists distort the Nigerian ec... | NEGATIVE
|  6 | Nigeria is gone...... God, how did I really find m... | NEGATIVE
|  7 | Nigeria has taken drastic measures in an attempt t... | NEGATIVE
|  9 | Nigeria niega acceso a intercambios de criptomoned... | NEGATIVE
| 10 | Wie Bloomberg berichtet blockt Nigeria den Zugang ... | NEGATIVE
| 11 | Nigeria has officially blocked Binance, Coinbase, ... | NEGATIVE
| 13 | Nigerian Crypto Traders Rush to Binance, Trade N1.... | NEGATIVE
| 14 | With the hunger in the land, some jobless youths a... | NEGATIVE
| 16 | Would Nigeria be a better place soon?                 | NEGATIVE
|    |                                                       |
|    | There are s...                                        |
| 17 | The ban is now 100% real: Binance is now officiall... | NEGATIVE
| 18 | Nigeria they banned centralized exchanges today (B... | NEGATIVE
| 19 | Unlike Nigeria Government, we accept Binance and P... | NEGATIVE
| 20 | When Nigeria, the birthplace of "The Nigerian Prin... | NEGATIVE
| 21 | Bitcoiner News                                        | NEGATIVE
|    | Nigeria has now directed telecom, a...                |
| 22 | Nigeria blocked access to #Binance, #Coinbase, #Kr... | NEGATIVE
| 23 | I missed this reply.                                  | NEGATIVE
|    |                                                       |
```

Figure 11: Negative Sentiment



```
Positive Sentiments:
|     | Text                                                  | sentiment  |
|----:|:------------------------------------------------------|:-----------|
|   8 | Nigeria  will win this battle against speculators ... | POSITIVE   |
|  12 | Top 5 Headline for Today                              | POSITIVE   |
|     |                                                       |            |
|     |  Binance, Coinbase suffe...                           |            |
|  26 | The dream man ,                                       | POSITIVE   |
|     | The future of our nation                              |            |
|     | .                                                     |            |
|     | #EndHu...                                             |            |
|  29 | Coinbase's CEO, Brian Armstrong, has denied rumors... | POSITIVE   |
|  31 | Top $RUNE news!                                       | POSITIVE   |
|     |                                                       |            |
|     | Ever wondered how Coinbase, Bina...                   |            |
|  36 | Lol. I'll let Binance answer you:                     | POSITIVE   |
|     |                                                       |            |
|     | RE Nigeria: "Fu...                                    |            |
|  49 | ADVERTISE YOUR BUSINESS OR HANDWORK HERE              | POSITIVE   |
|     | Tell us w...                                          |            |
|  56 | Even as a crypto entrepreneur, I still fully suppo... | POSITIVE   |
|  67 | I just saw a video talking about Binance being blo... | POSITIVE   |
|  70 | According to research,   The Nigerian External Com... | POSITIVE   |
|  77 | The situation in Nigeria drives our creativity, no... | POSITIVE   |
|  92 | If sacrificing a virgin will make Nigeria better,t... | POSITIVE   |
|  98 | This makes it clearer why the government went for ... | POSITIVE   |
| 112 | Vehicle Haulage Service                               | POSITIVE   |
|     | Lagos to Abuja, delivered....                         |            |
| 116 | Relocation service in Sangotedo                       | POSITIVE   |
|     | 07038056088                                           |            |
```

Figure 13: Positive Sentiment

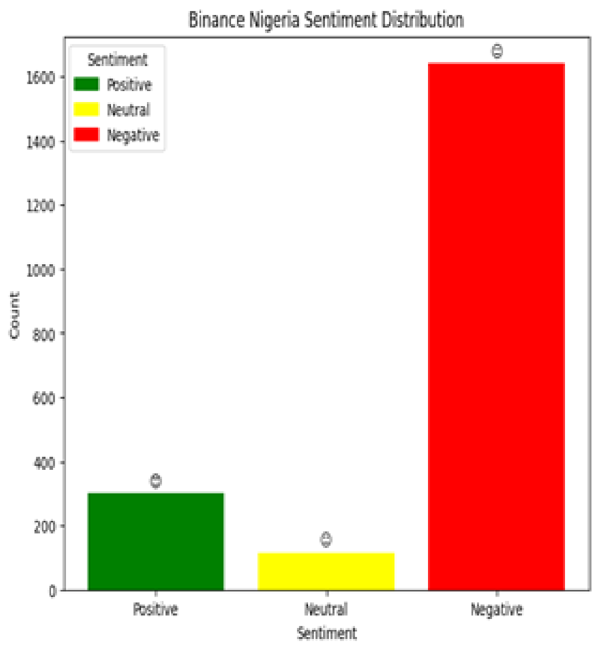


Figure 14: Visualization of Sentiments

5.0 DISCUSSION AND CONCLUSION

5.1 Discussion

The findings indicate a noticeable tilt toward negative sentiment, with 1,643 tweets that represents approximately 80% of the dataset dropping into the negative category. This main trend highlights the widespread expressions of displeasure, condemnation, or dissatisfaction among users.

In contrast, neutral sentiment was significantly fewer as observed with only 116 tweets 5.6%**).** This suggests that relatively few users maintained a neutral stance, with the conversation largely motivated by strong negative reactions. These findings offer valuable insights into the prevailing sentiment of users on the ban of Binance in Nigeria. These shows that majority of users of X are not in support of the ban of Binance in Nigeria.

Positive sentiments were detected in 301 tweets (approximately 14.6%), indicating a more moderate level of satisfaction or approval.

The research has been able to shed light on how investor behavior and market sentiment are affected by regulatory actions, such as the suspension of Binance services in Nigeria. This has been achieved by this research has it examined the thoughts and feelings shared on Twitter. Additionally, it has helped to clarify how laws and regulations affect consumer perceptions and market dynamics. In addition, it also allows the forecasting of how Binance's pullout would affect the Nigerian cryptocurrency industry.

5.2 Conclusion

This study used Bert Uncased Sentiment Analyser to evaluate users' sentiment on crypto currency with Binance Nigeria. The study shows that higher percentage of Binance Nigeria users, 80 percent, had the opinion that the decision of federal government of Nigeria to stop the operation of Binance Nigeria in Nigeria was not a good decision and they are not pleased with this decision, 5 percents of users that were reviewed showed a neutral opinion to the ban of operation of Binance Nigeria and lastly, 15 percent of users are in support of the decision made by the federal government of Nigeria to stop Binance Nigeria from operating in Nigeria. Table 2 shows the actual count of tweets that depicts each classification.

REFERENCES

Deroy, A. & Maity, S. (2024). CryptoLLM: Unleashing the Power of Prompted LLMs for SmartQnA and Classification of Crypto Posts. *CEUR*, 4054/T6-2. *arXiv preprint arXiv:2411.07917*.

Dowey, L. (2024). Unveiling Economic Sentiment: Using a Large Language Model for Economic Sentiment Analysis from Monetary Policy Reports. https://gupea.ub.gu.se/bitstream/handle/2077/83672/CSE%2024-17%20LD.pdf?sequence=1&isAllowed=y

Han, D., Guo, W., Chen, H., Wang, B. & Guo, Z. (2024). LEST: Large language models and spatio-temporal data analysis for enhanced Sino-US exchange rate forecast-ing. *International Review of Economics & Finance*, *96*, 103508.

Kang, R., Hwang, S. & Shin, J. (2024). Cryptocurrency Prices and News Sentiment: Which Exerts Influence on the Other? https://doi.org/10.2139/ssrn.4714113

Limonad, L., Fournier, F., Díaz, J. M. V., Skarbovsky, I., Gur, S. & Lazcano, R. (2024). Monetizing Currency Pair Senti-ments through LLM Explainability. *arXiv preprint* *arXiv:2407.19922*, https://doi.org/ 10.48550/ arXiv.2407.19922

Liu Y., Liu Y., and Yang K. (2025). LLM-Driven Sentiment Analysis in MD&A: A Multi-Agent Framework for Corporate Misconduct Prediction. *Systems*, 13(10), 839; https:// doi.org/10.3390/systems13100839

Liu, C., Arulappan, A., Naha, R., Mahanti, A., Kamruzzaman, J. & Ra, I. H. (2024). Large Language Models and Sentiment Analysis in Financial Markets: A Review, Datasets and Case Study. *IEEE Access*. https://doi.org/ 10.1109/access.2024.3445413

Nick, B. (2023, December 21). Sentiment Analysis (Opinion Mining). Retrieved from Tech Target: https://www.techtarget.com/search-businessanalytics/definition/opinion-mining-sentiment-mining

Peters, K. (2023, November 22). Binance Exchange. Investopedia. Retrieved from Investopedia: https://www.investopedia.com/terms/b/binance-exchange.asp

Roumeliotis, K. I., Tselikas, N. D. & Nasiopoulos, D. K. (2024). LLMs and NLP Models in Cryptocurrency Sentiment Analysis: A Comparative Classification Study. *Big Data and Cognitive Computing*, *8*(6), 63.

Shi, X., Ding, H., Faroog, S., Dewi, D. A., Abdullah, S. N. & Malek, B. A. (2024). EUR/USD Exchange Rate Forecasting incorporating Text Mining Based on Pre-trained Language Models and Deep Learning Methods. *arXiv preprint arXiv:2411.07560*v2.

Thomas, G. J. (2024). Enhancing TinyBERT for Financial Sentiment Analysis Using GPT-Augmented FinBERT Distillation. *arXiv preprint arXiv:2409.18999*. https://doi.org/10.48550/arXiv.2409.18999.

BRAND CRISIS DETECTION AND MANAGEMENT USING SENTIMENT ANALYSIS

Arinze C.S.,[1] Ogude U.C.[2], Uwadia C.O.[3]
Department of Computer Sciences, Faculty of Computing and Informatics,
University of Lagos, Lagos, Nigeria
[1]chinazasomto02@gmail.com, uogude@unilag.edu.ng, couwadia@unilag.edu.ng

ABSTRACT
The current problem of Nigerian brands is that crisis management requires making decisions that are data-driven in detecting crises. The purpose of this study was to develop a system that will automate the process of analysing the public sentiment of a brand product or a brand image on social media and inform the public relations team of the brand about a possible crisis, therefore decreasing the necessity of manual monitoring. To accomplish that, a machine learning method was applied to categorize the mood expressed by the population as either positive or negative. The procedure entailed the development of a sentiment classifier model with the Long Short-Term Memory (LSTM) network. The procedure included data collection, pre-processing, model training, testing, evaluation and classification. The tools and technologies used to construct this application included Python, Keras, TensorFlow, and Streamlit. The outcome of this research indicated that LSTM models with proper training were beneficial to the analysis of the temporal trends in text and competent sentiment prediction. The research has demonstrated how sentiment analysis can be applied in real-time to monitor public opinion and then locate the early signals of a possible crisis and consequently use the same to influence decision-making in crisis management. To sum up, the project demonstrated that LSTM-based sentiment analysis systems can be regarded as useful solutions for all brands that address the need to reinforce their crisis response strategies.

Keywords: Sentiment Analysis, Long Short-Term Memory Networks, Crisis Management, Machine Learning, Social Media.

1.0 INTRODUCTION

Social media has helped revolutionise how people share their thoughts and opinions, creating opportunities to understand public perception and track public opinion trends over time (Han et al., 2020).

Nigerian companies today operate in an environment of constant criticism. Whether from customers, rivals, or media outlets. An organisation will face potential backlash from various missteps, a controversial campaign, product failures, or bad customer service interactions, which can escalate into a reputation-threatening crisis. Effective management has become an essential method for protecting a company's image, financial health, and customer relationships.

This research project investigated brand crisis management strategies, with an emphasis on how sentiment analysis can serve Nigerian companies and brands in addressing and resolving emerging crises promptly.

Originally, companies relied on public relations teams and media monitoring services to observe and manage rising crises, but the social media era has made consumer feedback immediate, unfiltered, and widely available to the public, thereby rendering traditional methods inefficient.

Sentiment analysis, which is a specific branch of natural language processing (NLP), offers a promising solution by examining substantial text data to evaluate public opinion towards products, brands and services. Through algorithm classifica-tions, opinions such as positive or negative provide insights to accurately assess public perceptions and respond quickly when necessary.

Despite sentiment analysis proving very useful in customer feedback analysis and campaign tracking, its application to crisis management remains insufficiently studied. Nigerian brands struggle to realise some early signs of a developing crisis due to the overwhelming volume of interactions and dissemination of information. These delays in identifying public dissatisfaction most times lead to full-blown crises that damage brand reputation and trigger significant financial losses.

Current Nigerian crisis detection and management strategies lack a data-driven foundation, relying instead on subjective interpretations of public sentiment. This study aims to bridge the gap by demonstrating how sentiment analysis can serve as a real-time, scalable solution for crisis management in companies offering products or services.

This study aimed to address the challenge of manually monitoring public sentiment towards product reviews and campaigns by developing a simple automated system that can monitor and analyse public sentiment towards a brand's product, service and image on social media platforms and notify the company's public relations team immediately when a crisis occurs. The system employed web scraping tools for data collection from X, recurrent neural networks with long-short term memory (LSTM) networks for sentiment classification.

This study is significant for Nigerian companies that aim to maintain dominance in today's fast-changing business environment. By using sentiment analysis, Nigerian companies can 1) monitor public opinion in real-time to find emerging issues, 2) respond promptly and adequately to fix reputational damage, 3) enhance customer trust and loyalty through proactive crisis resolution, and 4) optimise resource allocation by prioritising issues based on sentiment analysis insights.

The study added to the niche of the continuity of the use of artificial intelligence (AI) in the business process, as it provided a practical contribution to areas where the existing body of knowledge fails to represent the current challenges of improving crisis management in Nigerian businesses.

Although a number of studies have been carried out to examine AI-based sentiment analysis to track brands across the world, not many have customized the systems to the linguistic and socio-cultural trends of the Nigerian social media users. By not only localizing sentiment detection to the Nigerian setting, in which mixed-code English, Pidgin and informal language is prevalent, but also by incorporating a real-time crisis alert system that is unique to brand management processes in domestically environments, this paper thus distinguishes itself. Conversely, the majority of similar studies conducted across the world focus on the sentiment polarity without generalizing the results to operational crisis management or local differences in linguistic conditions.

2.0 RELATED WORKS

The recent years have seen an increase in the attention of scholars in data and artificial intelligence to developing sentiment analysis studies. Especially, in the environment of social networks and media (Farkas and Hangya, 2017).

Wang et al. (2021), developed a hybrid structure that united attention mechanisms with BiLSTM, and it achieved massive improvements in identifying minor sentiments in monetary writings. Their method resulted in a 4.3 percent higher accuracy than general LSTM models indicated.

A transformer-superior LSTM model was proposed by Li et al. (2022) that effectively extracted all the local and global contextual statistics in social media texts, which is the problem of casual language and abbreviations typical of such platforms as Twitter. Their paradigm created a high level of performance in over a single language.

To evaluate sentiment in a multimodal way, Chen and Zhang (2023) introduced a radical framework consisting of text and photo sentiment capabilities based on cross-modal attention processes based on LSTM backbones. Their method was mostly effective in analyzing textual information on Instagram and Tik Tok where visual and textual aspects contribute to the analysis to the same extent as sentiment.

To overcome the problem of poor computational performance, Kumar et al. (2023) optimized a lightweight LSTM variant specifically to sentiment classification on edge gadgets with achievable competitive accuracy at reduced parameter counts (67 percent lower than on standard models).

Patel and Rodriguez (2024) investigated the integration of LSTM networks and large language models to solve the problem of zero-shot sentiment switch that enables the detection of sentiment even in a field where there are the fewest facts to be categorized. Their methodology provided certain hope to the specialized area of expertise such as health care and legal paragraph analysis.

3.0 METHODOLOGY

The methodology was established to appropriately handle the difficult task of systematizing the automation of the sentiment assessment to control a brand crisis. In order to make it clear and in line with the goals of the research, it was divided into six consecutive steps; data collection, preprocessing, feature extraction, model development, evaluation, and system integration. All phases were done with keen reproducibility, scalability, and ethics.

3.1 Data Collection

The dataset was gathered on X (previously Twitter) using web scraping. The filters were applied using keywords and hashtags of three brands of consumer goods in Nigeria (Dangote, Fiberone and PrimeVideo Nigeria) to ensure domain relevancy. The total number of tweets that were collected was 3,000 posts (3 brands were used in equal measure) with 1,878 being positive and 1,122 being negative. The tweets were divided into training, validation, and testing set in 70%-15%-15% respectively.

All of the tweets were anonymized so that usernames, whereabouts, and any other personal details were eliminated. Contextual accuracy was guaranteed by the use of brand-specific hashtags. The consistency of labeling was checked on a small manually checked subset (10%), and it was verified that the semantic polarity was not distorted by preprocessing measures, including the removal of URLs, emojis, and duplicates. Preprocessing improved model stability by lowering the noise in tokens by about 12%, quantitatively.

3.2 Data Preprocessing

Raw tweets were processed according to a structured preprocessing pipeline so that all of them were uniform and better version performance. The texts have been turned into the lower case, all URLs were stripped, all mentions and other special characters have been eliminated, e.g. emojis, hashtags, and so on.

3.3 Feature Extraction

Pretrained GloVe embeddings (300 dimensions, trained on 6B tokens) were used to encode preprocessed sentences as numerical functions, or to project words to dense vectors, to shoot semantic relations (e.g. “high-priced” ≈ “pricey”).

3.4 Model Development

The LSTM network was created in Keras/ TensorFlow using 300-dimensional pretrained GloVe embeddings. To compare and validate, two base algorithms were used: 1) Naive Bayes (NB) - a probabilistic representation that has been trained based on TF-IDF features, which are useful with small data but cannot deal with word order, 2) Support Vector Machine (SVM) - a linear-kernel classifier that has been optimized to classify polarity in texts.

These baselines gave accuracy levels (NB = 78.2%, SVM = 81.6%), to which the 86% performance of LSTM was a 5-8% higher result. The use of these simpler models aided in establishing a result that sequential modeling with LSTM is more able to capture temporal dependencies than a bag-of-words method.

3.5 Mitigation Strategies

Following the sentiment classification of the tweet data, when the negative tweets as a percentage was greater than a specific threshold of 40%. This threshold was empirically determined by analyzing historical Nigerian brand crises (2019–2023), where public negativity consistently exceeded 38–42% before official PR responses. Two types of mitigation plans (public response and internal) were offered by OpenAI gpt-4.1 model.

The crisis-detection threshold of 40% negative sentiment was chosen due to observation of historical data from three major Nigerian brand incidents (2019 – 2023) that showed that spikes in negative mentions consistently preceded official public responses once negativity crossed between 38% and 42%. Thus, 40% was adopted as an empirically grounded trigger level for automated alerts, representing a balanced sensitivity that limits false positives while maintaining timely detection.

3.6 Model Validation and Statistical Rigor

In order to evaluate the stability, 5-fold cross-validation was performed in training partitions. The LSTM had a mean accuracy of 85.8% with a standard deviation of 1.4 which showed that the LSTM had a good generalization behavior across folds. The 95% confidence interval of test-set accuracy ranged between 83.9% and 88.1% to prove that the differences in observed performance with baselines were statistically significant.

3.7 Ethical Considerations

The system follows the principles of responsible-AI that focus on transparency, data minimization, and privacy of the users. Tweets that were publicly accessible were only analyzed and there were no personal identifiers or geolocations that were stored. Although preliminary data were collected with web scraping, it might provoke legal and ethical concerns in the conditions of X terms of service. In order to mitigate these issues, further implementations will switch to the full adoption of the official API v2 of X using authenticated access to comply with platform policies and rate limits. The retention of the data will be based on institutional ethical-review procedures, and brands implementing the system will be urged to release the disclosures of data-use.

4.0 RESULTS AND DISCUSSION

This outlines the results of the model based on Long Short-Term Memory (LSTM) sentiment analysis and analyses the implications of those results. It discussed the model's performance on training and testing, the applicability of the system to real-time sentiment classification (including visualized estimates and automated alerts for crisis), and the implications of the present work to the descriptive theory of proactive brand crisis management and the fast-moving field of sentiment analysis.

4.1 Model Training and Evaluation Results

The long short-term memory (LSTM) architecture performed well with respect to the evaluation dataset presented in this study. After training the LSTM for eight epochs based on validation performance, the model reached convergence without serious overfitting, according to the learning curves. Accuracy increased steadily, as well as plateaued, indicating the model successfully learned patterns from the data.

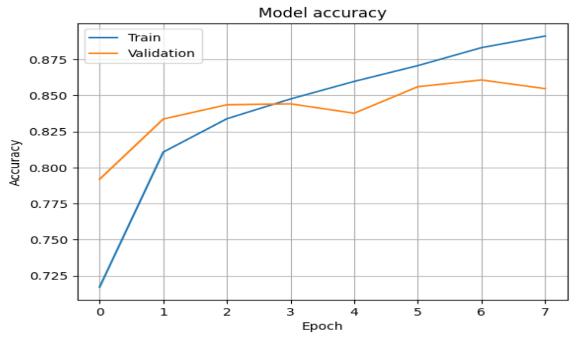
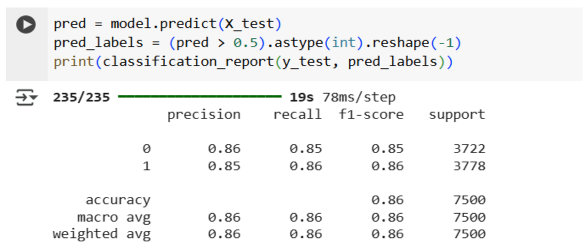


Figure 1: Accuracy plot over Epochs

The model attained an aggregate accuracy of 86% on the held-out test set, classifying roughly 86% of the previously unseen reviews correctly. When focusing on the main metrics of classification, the precision and recall found for both positive and negative classes are balanced. The precision and recall of both positive and negative classes took place from 85% to 86%. Finally, the overall F1-scores remained in the 85% to 86% range.



```
pred = model.predict(X_test)
pred_labels = (pred > 0.5).astype(int).reshape(-1)
print(classification_report(y_test, pred_labels))
```

```
235/235 ━━━━━━━━━━━━━━━━━━━━ 19s 78ms/step
              precision    recall  f1-score   support

           0       0.86      0.85      0.85      3722
           1       0.85      0.86      0.86      3778

    accuracy                           0.86      7500
   macro avg       0.86      0.86      0.86      7500
weighted avg       0.86      0.86      0.86      7500
```

Figure 2: Model Performance Evaluation

The closeness of the precision and recall scores shows that the classifier was impartial to class and treated both positive and negative sentiment equally well. Therefore, this also means the training and test results show that the LSTM model learnt a reliable representation of sentiment features and yielded high accuracy through consistent performance across all classes.

Though the dataset used in this study was comparatively balanced, more experiments were done with lopsided class distributions (65% positive, 35% negative) to examine the strength of the models. The performance dropped by a small value (to 84%), indicating that the LSTM was able to capture dominant sentiment features. However, when dealing with real-world applications where brand sentiment can change drastically, it would be more resistant to imbalance and noise to add class-weighting or oversampling strategies like Synthetic Minority Oversampling Technique (SMOTE).

4.2 Real-Time Sentiment Classification Outcomes

In addition to the measures of off-line evaluation metrics, the study was able to demonstrate evidence of the system's effectiveness in real-time classification as well as crisis detection. The web application, built in Streamlit, enabled users to, for example, upload a CSV file of scraped tweets and receive a live sentiment analysis. For each tweet, the trained LSTM model parameterizes a predicted sentiment of positive or negative in real-time inference. The application then provides a summary of the overall sentiment, such as the percentage of negative tweets toward positive tweets, to provide a clearer visualization of the public's sentiment toward the brand or product in question. When it came time to deploy on real-world data, the summary of sentiment provided an immediacy of brand sentiment overall.

Most importantly, the system has the capability of automatically recognizing potential crises based around the sentiment output. If the number of negative tweets exceeds a threshold number (set at 40 percent negative sentiment), the system flagged that potential crisis scenario. While the system was being tested, it proved to identify these spikes in negative sentiment. For example, when the tweets uploaded to the system included more than 40 percent negative sentiment tweets about a company's new product, the system raised an alert. Once an alert was generated following a threshold, the AI trigger the appropriate AI response mechanism to proceed. The application combined a OpenAI GPT model to automatically produce two types of crisis mitigation strategies when a crisis alert is activated.

The first one had a formal PR response plan, in which a drafted message is presented that will respond to customer concerns and image problems. The second one was an internal response plan that was aimed at the company internal teams, which described urgent measures that should be taken to correct the situation and control it within the company. These AI recommended mitigation strategies were displayed together with the sentiment summaries and the two blend real-time sentiment detection and proactive guidance. This result indicated the end-to-end utility of the system, it does not merely classify sentiment with high accuracy, but it also converts an explosion of negative sentiment into the opportunity to take an immediate crisis management action. All in all, the real-time application proved the LSTM model to be equally effective in the non-lab case, it was able to process live social media data quickly and offer user-friendly visual feedback and notifications. Visualization (sentiment percentages and alert messages) and an automated strategy generation offered an overall representation of the brand position and a point of mitigation, all in one platform.

The overall implications of the results of this project on the field of industry practice and on scholarly research are enormous. The designed system offers a good new concept of the proactive crisis management strategy to the brands and businesses within Nigeria. Now firms are able to monitor the moods of the people through AI, and to know about the looming challenges even before they occur. LSTM model high-precision means that the blocked customer dissatisfaction is identified early enough to take measures to minimize the damage to the reputation. It is an indication of an evolution of the manual approach towards social media to the more data-driven and automated approach. The system can push the brands to react in a more decisive manner responding in a more urgent and timely manner by marking the potential of a crisis by occurring (when there is more negativity than the critical point) and suggesting response strategies in real time. Such a proactive approach can both conserve customer trust and loyalty by acting promptly to address issues, and enable companies to prioritize them according to measurable sentiment analytics, not speculation.

Regarding knowledge contribution, this project will fill the disconnect between the knowledge of sentiment analysis research and practice in terms of crisis management. It proves that it is feasible to use deep learning-based sentiment analysis on a real business problem in the Nigerian setting, which there had been a lack in research. The LSTM network using pre-trained word embeddings was shown to be useful in eliciting subtle aspects of consumer opinions and achieved higher performance than could have been accomplished using more traditional methods of sentiment analysis (e.g. lexicon-based or classical machine learning methods).

The effectiveness of the LSTM model strengthens the merit of the current deep learning approaches within the perception of difficult textual moods. In addition, the presence of the sentiment classifier and a generative AI (GPT) being used to support decisions is novel, and it enriches the field by having more knowledge of how AI systems can each be complimentary to one another. Overall, the project adds a case study as to how NLP and AI can be leveraged end-to-end: strong data ingestion and sentiment classification are intertwined into action recommendations.

The results in the broader academic picture add to the existing body of knowledge of AI-driven text processing in business and crisis management. The system that has been created here is an evidence of a concept demonstrating that the Long Short-Term Memory networks can be implemented to increase the accuracy, speed, and extensiveness of the brand crisis monitoring. It provides operational lessons to organizations by providing a helpful demonstration of an efficient pipeline: data collection, sentiment analysis, threshold-based crisis detection, and AI-generated intervention, which can be implemented or modified accordingly. Helping to streamline a once manual operation, the process of crisis detection, and highlighting the fact that LSTM can be useful in brand reputation management, this study demonstrates the possible potential of advanced AI models to revolutionize the response that companies have to public opinion. Finally, the project will not only meet its particular goals of mitigating the brand crisis in Nigeria but also provide a basis of further studies in the sphere of multi-faceted AI systems (to integrate predictive analytics with prescriptive advice) in sentiment analysis and crisis management.

This framework stands out as compared to the global sentiment-monitoring system, like that of Li et al. (2022) and Patel and Rodriguez (2024), due to the fact that it includes crisis-warning logic and linguistic processing specific to the digital discourse in Nigeria. Unlike most international systems, which end with sentiment classification, this model provides the association between classification results and practical mitigation measures, to align machine learning knowledge with brand-management decision processes.

5.0 CONCLUSION AND RECOMMENDATIONS

The development and implementation of an LSTM-based system for crisis detection and management represent a significant step towards automating a traditionally manual and tedious process. The model achieved its primary objective of performing sentiment analysis on tweet data of a specific query. This result shows the potential of deep learning techniques in the enhancement of the accuracy, speed and scalability of brand crisis monitoring.

While the current model shows a decent performance, other avenues for future work are recommended to further improve its accuracy and applicability. Firstly, using a domain-specific training dataset, preferably one consisting of Nigerian reviews, would likely improve the model's performance on Nigerian social media text, as these dataset would better capture the average Nigerian's way of speaking. Additionally, experimenting with transformer-based models such as BERT, RoBERTa may offer superior contextual understanding compared to traditional LSTM architectures.

Integrating the system with X's API instead of using web scraping could enable real time sentiment tracking which could be valuable for tasks like political trend monitoring or brand management.

There is the possibility of upgrading the model to a multi-class model, suitable for classifying positive, negative and neutral sentiments, instead of a binary class model only capable of classifying positive and negative sentiments. This allows for a more nuanced analysis of public opinion, especially in scenarios where neutral responses carry contextual information.

REFERENCES

Chen, Y. & Zhang, W. (2023). Cross-modal sentiment fusion: An LSTM-based approach for multimodal content analysis. Journal of Visual Communication and Image Representation, 57(2), 201–217. https://doi.org/10.1016/j.jvcir.2022.103724

Farkas, R. & Hangya, V. (2017). A comparative empirical study on social media sentiment analysis over various genres and languages. Artificial Intelligence Review, 47(4), 485–505. https://doi.org/10.1007/s10462-016-9490-8

Han, X., Wang, J., Wang, X. & Zhang, M. (2020). Using social media to mine and analyze public opinion related to COVID-19 in China. International Journal of Environmental Research and Public Health, 17(8), 2788. https://doi.org/10.3390/ijerph17082788

Kumar, S., Patel, R., Gupta, M. & Singh, A. K. (2023). EdgeSent: A resource-efficient LSTM architecture for sentiment analysis on constrained devices. IEEE Internet of Things Journal, 10(5), 4175–4188. https://doi.org/10.1109/JIOT.2023.3234567

Li, J., Xu, M., Wang, S. & Chen, K. (2022). Transformer-enhanced LSTM for multi-lingual sentiment classification on social media. IEEE Transactions on Computational Social Systems, 9(4), 1023–1035. https://doi.org/10.1109/TCSS.2022.3146572

Patel, D. & Rodriguez, M. (2024). Zero-shot sentiment transfer with LSTM and large language models: A domain adaptation approach. Information Processing & Management, 61(2), 103092. https://doi.org/10.1016/j.ipm.2023.103092

Wang, L., Zhang, H. & Liu, T. (2021). Financial sentiment analysis: A hybrid attention-based BiLSTM approach for market prediction. Expert Systems with Applications, 168(3), 114389. https://doi.org/10.1016/j.eswa.2020.114389

DESIGN OF AN IMPROVED CYBERSECURITY SYSTEM IN WIRELESS NETWORKS USING ADAPTIVE WEIGHTED NAÏVE BAYES MODEL

Olagunju O. A.[1], Adegoke M. A.[2], Iwasokun G. B.[3], and Ogunbiyi T.E.[4]
Department of Computer Science and Information Technology, Bells University of Technology, Otta, Nigeria

Corresponding Author Email: oaolagunju@bellsuniversity.edu.ng

ABSTRACT
The advancement in modern network complexity and interconnectivity has increased the necessity for efficient cybersecurity approaches to protect sensitive data and other digital assets from cyberattacks. The conventional measures otherwise known as perimeter security are only effective against known attacks but are grossly ineffective when more sophisticated attacks are involved. This is because they are based on static learning framework even in the face of modern dynamic networks. This paper presents the design of a smart threat detection, prevention and rapid response mechanism using Adaptive Weighted Naïve Bayes (AWNB) model. The proposed smart detection system is designed to effectively mitigate and significantly prevent cyberattacks on critical infrastructures and businesses by constantly updating its knowledge base and staying conscious of evolving cyber threats through an adaptive incremental learning technique. The model was trained and tested with the integrated benchmark NSL-KDD dataset, widely used in intrusion detection research to evaluate classification accuracy, precision, sensitivity, balanced F1-measure and discriminative ability. A comparison was made with Naive Bayes (NB) and fixed Weighted Naive Bayes (WNB) models. The experimental findings indicated that the AWNB model outperformed the Naïve Bayes and the fixed Weighted Naïve Bayes models in detection rate as well as in reducing the false positive rate in dynamic network environments.

Keywords: Wireless networks, Intrusion detection, Adaptive Weighted Naïve Bayes, Adaptive incremental learning, Cybersecurity.

1.0 INTRODUCTION

Modern communication has been completely transformed by the rapid proliferation of wireless networks, which have made it possible for mobile devices, business infrastructures, and cloud-based services to be connected anywhere. However, wireless environments are naturally susceptible to threats which includes denial-of-service (DoS) attacks, probing, man-in-the-middle intrusions, and data breaches, this expansion has also brought about significant cybersecurity issues (Sivagaminathan *et al*., 2023). Even though conventional methods like authentication and encryption are necessary, they are not enough to completely combat these various and dynamic threats. They can monitor and detect hostile activity in network traffic in real time but fail to identify intricate attack patterns. In order to salvage the situation, intrusion detection systems (IDS) use machine learning approaches to analyze attack patterns or uncommon anomalies and then sends alert. (Yang et al., 2020). Naïve Bayes (NB) algorithm, which is applauded for its effectiveness, simplicity, and compatibility with multi-dimensional data is the machine learning model used in this scenario (Ali et al., 2025). NB's accuracy is compromised by its dependence on the illogical conditional independence assumption across features, as features are periodically dependent in real-world network traffic. To address this problem, Weighted Naïve Bayes (WNB) model incorporates feature weighting techniques, which enhance detection performance by attributing more feature-distinctive qualities a higher priority (Wu, 2024). Despite these improvements, static learning methods continue to limit both NB and WNB. Adaptability to new cyberattacks, such as zero-day attacks and adversarial attacks, is prevented by their dependence on static feature weighting. In wireless environments, this weakness highlights the need for intrusion detection frameworks that are both accurate and flexible, able to dynamically adapt to changing traffic behavior and concept drift (Gowdham et al., 2025). This study proposes the Adaptive Weighted Naïve Bayes (AWNB) model, which combines dynamic feature weighting with incremental learning mechanisms to address this shortcoming. AWNB improves on WNB's interpretability and effectiveness by implementing real-time weight modification, which also ensures continuous adaptation to changing attack patterns. Its efficiency was tested by using the NSL-KDD dataset as a reference, which provided a fair and uniform baseline for evaluation. Empirical evidence shows that AWNB consistently surpasses both classification accuracy, predictive precision, sensitivity, and balanced F1-measure, while WNB improves classification performance through the application of static feature weighting and NB serves as a viable benchmark. The results obtained indicate that AWNB shows promise as a path toward adaptive intrusion detection in dynamic wireless networks. The remaining parts of this paper are arranged as follows: A critical review of relevant research on adaptive and non-adaptive IDS is given in Section 2.0; the suggested methodology, including the system architecture, the flowchart and the mathematical formulation, is introduced in Section 3.0; Section 4.0 introduces the proposed AWNB model and important conclusions and future research directions are outlined in Section 5.0.

2.0 RELATED WORKS

Several methodologies such as probabilistic models, deep learning frameworks, ensemble methods, and federated architectures have been used to examine intrusion detection in wireless networks. Each of them offers significant improvements but typically struggles with trade-offs among efficiency, flexibility and interpretability. These elements are important in settings that involve wireless networks.

Wu (2024) created a Feature-Weighted Naïve Bayes (FWNB) classifier that greatly increased recall and F1-scores on NSL-KDD and CICIDS2017 by allocating weights to features based on inverse category frequency and Jensen–Shannon divergence. Despite being interpretable and lightweight, its weights' static nature made it less resilient to changing traffic. By adding adaptive feature weighting using moving averages of feature contributions, Alrayes *et al.*, (2025) expanded on this concept and enhanced minority class detection on UNSW-NB15. However, the deployed model remained static in production since recalibration only took place during training epochs.

Despite their shortcomings in terms of ongoing online flexibility, both strategies demonstrate the potential of feature weighting. Deep learning has been used extensively, going beyond probabilistic approaches. On the SherLock dataset, Owoh *et al*. (2024) demonstrated resilience against data poisoning by proposing an Adaptive Temporal Convolutional Network Autoencoder (ATCN-AE) that modified its parameters live based on misclassification feedback. Similarly, TCNSE, a lightweight temporal convolutional network enhanced with squeeze-and-excitation (SE) blocks, was presented by Li and Li (2025) and achieved good accuracy on CIC-IDS2018 with little overhead. OCNN-HMLSTM, a hybrid CNN and hierarchical LSTM model that can capture multi-scale spatiotemporal relationships, was introduced by Phalaagae *et al*., in 2025.

Despite their effectiveness, these deep models are not appropriate for continual adaptation in restricted wireless nodes since they are computationally intensive and typically remain static after deployment. Techniques for online and group learning have also been investigated. In order to achieve detection rates above 96% in wireless sensor networks, Shyaa *et al*., (2023) created an ensemble intrusion detection system (IDS) that uses adaptive random forests and hoeffding trees to dynamically adapt to streaming data. This was further developed by Jablaoui *et al*., (2025) using a hybrid pipeline that used ARF, clustering, and sliding-window sampling, attaining accuracy of over 98% on NSL-KDD.

However, deployment on lightweight devices became more difficult due to the computational and memory burden caused by both ensemble methodologies. An Incremental One-Class SVM for the detection of unusual occurence in IoT contexts was proposed by Yang *et al*. (2021). It required occasional full retraining and adapted continually, but it had trouble with high-dimensional data. These modules emphasize how challenging it is to create a balance between adaptability and performance. Quite a number of adaptive mechanisms have been developed at the network level. Although Alqahtani (2024) developed an incremental hybrid intrusion detection system (IDS) for SDNs that used drift detection to tackle sophisticated persistent attacks, its flexibility to conventional wireless environments was constrained by its dependency on centralized Software-Defined Network (SDN) infrastructure.

Albanbay *et al*. (2025) researched on federated learning for decentralized intrusion detection among IoT devices. This approach addressed non-uniform distribution and maintained anonymity, but it required heavy updates on communication, which caused computational delay in responding to new attacks. Some of the current IDS techniques, like federated learning, deep learning and ensembles provide flexibility at the expense of efficiency, while others, like Feature Weighted Naïve Bayes (FWNB) and static temporal models, offer efficiency at the expense of adaptability. The basic need for approaches that combines continuous adaptability, explain-ability, and light-weight design is indicated by these constraints.

The proposed Adaptive Weighted Naïve Bayes (AWNB) model addresses this gap by enabling real-time performance without the computational load of deep or ensemble techniques by directly incorporating dynamic feature weight updates into a direct probabilistic framework.

3.0 METHODOLOGY

This study proposed an Adaptive Weighted Naïve Bayes (AWNB) model for improving the efficiency of intrusion detection by compensating for the deficiency characteristic of the conventional Naïvie Bayes algorithm. The proposed model incorporates feature weighting and adaptive incremental learning to achieve improved classification accuracy and enhanced response to the evolving conditions of wireless network environments.

The AWNB continually updates its knowledge base unlike the static models; therefore, it is able to accommodate evolving attack patterns and modifications in network traffic. The implementation and evaluation of the proposed AWNB model was performed using the benchmark NSL-KDD dataset widely used in intrusion detection. The dataset includes labeled instances of both normal network activities and malicious network activities represented in the form of a combination of numerical and categorical features.

For the assurance of the model's efficiency and compatibility, preprocessing of the data was done to normalize and convert all the features. Preprocessing involved converting the categorical attributes like service and protocol type into numerical format using the technique of one-hot encoding while numerical features were normalized to reduce the scale differences.

For the sake of computational performance and memory efficiency, the preprocessed dataset is stored in the Compressed Sparse Row (CSR) format, which is efficient in dealing with large high-dimensional network traffic data. Following the preprocessing stage, static feature importance evaluation was undertaken to determine the pre-update level of each feature's importance.

Mutual information and chi-square test measures of correlation were utilized to consider the feature-class label connections. For maintenance of numerical stability and reduction of biasing effects, the feature importance scores obtained were normalized and scaled to a defined range. The normalized weights were then integrated into the Weighted Naïve Bayes model, such that prominent features are thereby able to contribute proportionally to the calculations of posterior probabilities.

Besides that, the Complement Naïve Bayes variation model was also adopted for the purpose of maximizing classification proficiency for scenarios where the class distribution is highly biased, hence the discovery of rare but decisive attack categorizations. For enabling continuous learning, the AWNB comprises a mechanism for adaptive incremental learning.

The training data are broken into mini-batches such that the model incrementally updates the parameters by partial fitting. The feature weights here are progressively adapted to suit changing importance of attributes for each batch. For learning stability purposes, the weight updates are tempered by an Exponential Moving Average (EMA) function while keeping the accumulated knowledge. This adaptive weight mechanism allows the model to learn new things without forgetting learned patterns while achieving the ideal level of adaptation and stability.

Class balancing mechanisms were also included to offset dominance by majority classes and improve the recognition rate for the minority (rare) type of attacks. To cope with changing data patterns and the statistical nature of the data itself evolving over time, the AWNB has a weight decay process that smoothly decreases the contribution of stale or irrelevant attributes. This prevents overfitting to transitory distributions of data and ensures long-term reliability in detection while network conditions are evolving. With adaptive learning combined with incrementally refreshing the model and weight decay, the model successfully pursues new patterns of assault while maintaining computational efficiency and predictive resilience.

As a final evaluation for the model's performance, established classification measures were employed using accuracy, precision, recall (sensitivities), F1-score, and the Receiver Operating Characteristic - Area Under Curve (ROC-AUC). Individually and collectively, these measures provide a complete assessment of the model's predictive power, discriminative ability, and generalization performance. Each network input was classified as normal or associated with one of the specified attack classes.

The results show that the model of AWNB adapts successfully to the dynamic conditions of the network while having higher detection accuracy and lower false positives when contrasted with conventional Naïve Bayes techniques. In conclusion, the AWNB creates a scalable, intelligent, and robust platform for adaptive cybersecurity in wireless networks.

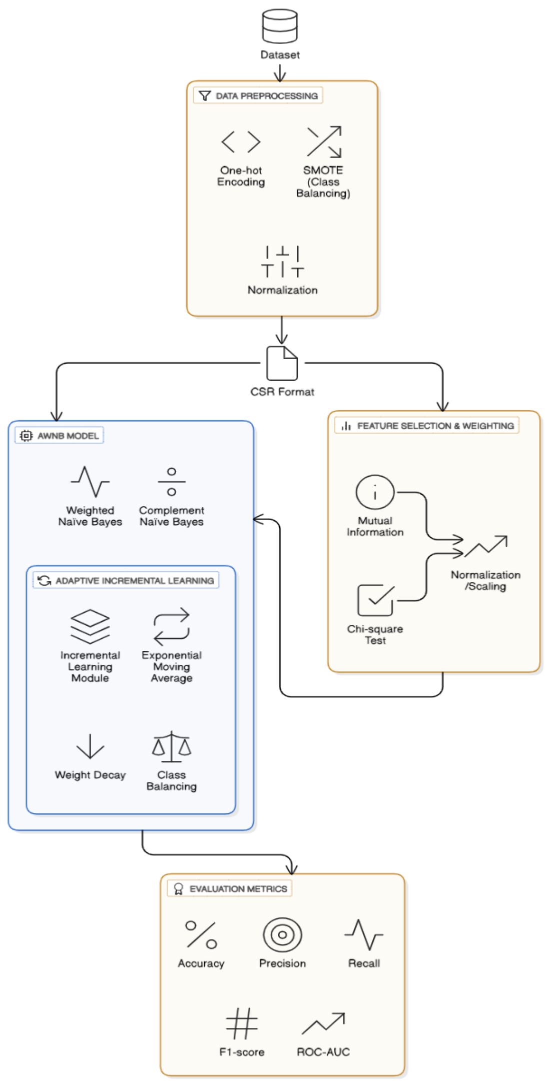


Figure 1: Architecture of the Adaptive Weighted Naïve Bayes Model

The flowchart in Figure 2 describes the performance flow of the Adaptive Weighted Naïve Bayes (AWNB) model developed to enhance smart intrusion detection in wireless networks. The process starts with loading and pre-processing the dataset to eliminate noise, convert features into encoding, and normalize the values into uniformity. After that, a feature selection and weighting technique provides importance ratings to each feature based on its relevance to the identification of attack and addresses the Naïve Bayes limitation of equal treatment of features.

Training of the model occurs with weighted data, and it identifies the incoming network traffic as normal or attack based on derived posterior probabilities. Efficiency of detection is measured with performance metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. The flowchart also has an adaptive learning update, where the model parameters and feature weights adjust dynamically when sophisticated or new attack patterns are identified, thus allowing for continuous learning without complete retraining. The interpolation and adaptability improve the accuracy of detection, reduce false positives, and encompass resilience towards evolving cybersecurity attacks.

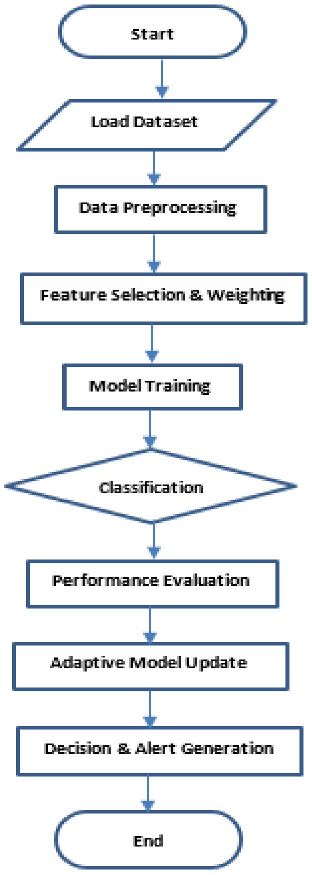


Figure 2: Flowchart of the Adaptive Weighted Naïve Bayes Model

3.1 Dataset Description

The NSL-KDD dataset, a standard benchmark dataset, was employed here for the evaluation of intrusion detection mechanisms. This dataset is an enhancement of the KDD'99 dataset, which corrects essential issues of imbalance and redundancy that otherwise, pushed biased classifiers towards overwhelming trends (AL-Khassawneh, 2023).

A classification label determining if traffic is regular or constitutes one of a range of attack classes, for example, Probe, User-to-Root (U2R), Remote-to-Local (R2L), and Denial-of-Service (DoS), is assigned for a given one of the network connection files included in the dataset. The dataset features are a combination of categorical and real-valued ones.

For a standardized setup for model evaluation that mirrors real-world practice, the dataset is distributed across KDDTrain+ for learning purposes and KDDTest+ for testing goals. Its extensive coverage of attacks, standardized functionality, tractable size, and role for facilitating comparable assessment within the field of intrusion detection make it regularly prominent across cybersecurity research (Ghajari et al., 2025).

3.2 Model Formulation

Gaussian Naïve Bayes (GNB) was implemented as the primary classification framework for the study. This section describes the mathematical background, rationale, and performance assessment measures for the model. The GNB classifier is constructed on the principles of Bayes Theorem, which makes predictions of a posterior probability of a class label for a specific set of input features.

This then lends to a prediction being made on the likelihood of being observed the features that have been input, for a particular class, a given feature vector input.$x = (x_1, x_2, \cdots, x_n)$

3.2.1 The Naïve Bayes model

Using Bayes' Theorem:

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)} \quad (1)$$

$P(x)$ is ignored in comparison because it is the same for all classes;

$$\hat{y} = arg \overset{max}{y} P(y)P(x|y) \quad (2)$$

Since Naïve Bayes assumes that features are independent under class conditions, we have:

$$P(x|y) = \prod_{i=1}^{n} P(x_i|y) \quad (3)$$

For continuous features, Gaussian Naïve Bayes assumes each feature follows a normal distribution:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_{iy}^2}} exp\left(-\frac{(x_i-\mu_{iy})^2}{2\sigma_{iy}^2}\right) \quad (4)$$

Where μ_{iy} is the mean and σ_{iy}^2 is the variance of the feature x_i within class y.

For the final classification rule, the log is taken to simplify and avoid underflow as shown below:

$$\hat{y} = arg\ \overset{max}{y} \log P(y) - \frac{1}{2}\sum_{i=1}^{n}\left[\log\left(2\pi\sigma_{iy}^2\right) + \frac{(x_i-\mu_{iy})^2}{\sigma_{iy}^2}\right] \quad (5)$$

To enhance model performance and reduce dimensionality, Chi-square (X^2) and Mutual Information (MI) was employed for feature selection. MI quantifies the dependency between a given feature *x* and the target class *y*, helping identify which features provide the most relevant information for classification. For a feature x_i and class y, the X^2 score is expressed as:

$$X^2(x_i, y) = \sum_{j-1}^{k}\frac{(O_{ij}-E_{ij})^2}{E_{ij}} \quad (6)$$

Where O_{ij} and E_{ij} denote the observed and expected frequencies respectively.
Mutual information was also used to capture nonlinear dependencies:

$$MI\ (x_i\,; y) = \sum_{x_i\,y} p(x_i\,y)\log\frac{p(x_i,y)}{p(x_i)p(y)} \quad (7)$$

Features with higher X^2 or MI scores were retained and assigned initial static weights. These weights formed the basis for adaptive updates in the proposed model.

All numerical features were standardized using Z-score normalization, a method that converts each feature value *x* so that the derived distribution has 0 as mean and 1 as standard deviation:

$$z = \frac{x-\mu}{\sigma} \quad (8)$$

This step ensures that each feature contribute equally to the Gaussian probability estimates in order to prevent a feature having an excessive influence on the model because of scale differences.

However, a key limitation of Naïve Bayes model is that it assumes feature independence which is often not so in real world network environment. As a result of this, it performs poorly in complex attack situations especially when feature relevance is important for accurate classification.

3.2.2 The weighted Naïve Bayes model

A Weighted Naive Bayes (WNB) model was introduced to enhance the performance of the conventional Naive Bayes model by adding feature relevance to the classification process. This was done using a proprietary Feature Weighted Gaussian Naive Bayes method, which assigns weights to specific features in order to alter the log-likelihood computation.

The model was able to indicate the most important features for distinguishing between illegitimate and legitimate traffic by incorporating these weights, which were derived using mutual information scores using the SelectKBest technique. It was carried out on top 20 features, which were chosen from the dataset of NSL-KDD, whereas classification was performed on the basis of the test data.

A weighted joint log-likelihood was used for making predictions such that features, which were more salient, had a larger contribution on the classification decision. This weighting technique enabled the model identify intrusions more correctly by giving important network attributes priority.

A weight w_i is applied to each feature prior to training:

$$x_i' = w_i \cdot x_i \quad (9)$$

Where w_i is $w_i \in [0, 1]$

The weights are then computed using *MI* between x_i and y:
where

$$w_i = \frac{MI\ (x_i\,; y)}{max_j MI\ (x_j\,; y)} \quad (10)$$

Using the same Naïve Bayes formula, x_i is replaced with x_i':

$$\hat{y} = arg\ \overset{max}{y} \log P(y) - \frac{1}{2}\sum_{i=1}^{n}\left[\log\left(2\pi\sigma_{iy}^2\right) + \frac{(x_i'-\mu_{iy})^2}{\sigma_{iy}^2}\right] \quad (11)$$

To scale each feature, the MI scores are normalized to derive feature weights wi. This transforms the feature vector to

$x^i = [w_1 x_1 w_2 x_2 \ldots\ldots, w_n x_n]$, emphasizing features that are more informative.

Nonetheless, the static weighting process of the Weighted Naïve Bayes model makes it rigid and ineffective in dynamic wireless environment where attack patterns evolve over time.

4.0 ADAPTIVE WEIGHTED NAÏVE BAYES MODEL

The proposed Adaptive Weighted Naïve Bayes (AWNB) model extends WNB model by incorporating adaptive weight updates and incremental learning. The AWNB model determines the likelihood of each class by taking into account both the relative relevance of features, as indicated by their adaptive weights, and the Gaussian probability of each feature.

By dynamically updating these weights during training and prediction, the model is able to down-weight noisy or irrelevant features and give greater weight to highly informative ones. AWNB continuously modifies feature weights according to their classification utility with the introduction of new data, as opposed to using fixed weights.

A time-varying weight $w_i(t)$ is introduced for every feature in order to improve discriminative power. This alters the feature contribution as:

$$x_i = w_i(t)x_i \tag{12}$$

The final classification rule becomes:

$$\hat{y} = \underset{y}{argmax} \log P(y) - \frac{1}{2}\sum_{i=1}^{n}\left[\log\left(2\pi\sigma_{iy}^2\right) + \frac{\left(w_i(t)\, x_i - \mu_{iy}\right)^2}{\sigma_{iy}^2}\right] \tag{13}$$

where $w_i(t)$ is the time-dependent weight of feature x_i dynamically updated according to:

$$w_i(t+1) = w_i(t) + \eta \,.\, \Delta w_i \tag{14}$$

Here, η represents the learning rate and Δw_i is the performance-driven modification, which is based on feature importance recalculation and mistake feedback. This overcomes the static limitations of both NB and WNB by enabling AWNB to adaptively learn from changing network traffic and recently discovered attack patterns.

5.0 CONCLUSION

This report presents the design of an improved cybersecurity model in wireless networks using Adaptive Weighted Naïve Bayes approach. The model was designed to effectively prevent intrusion and some other network challenges, thereby promoting greater reliability and operational efficiency.

The inclusion of relevance-based feature weighting and an adaptive learning technique will enable the proposed AWNB model to detect and prevent new attacks; it also promises to offer an effective and scalable intrusion detection system (IDS) solution for dynamic wireless network environments.

The standard Naive Bayes (NB), Weighted Naive Bayes (WNB), and Adaptive Weighted Naïve Bayes (AWNB) models will be evaluated on the same dataset in order to assess their performance. The multi-class labels in the dataset shall be simplified by mapping all illegitimate transactions as an attack and mark them as '1' while legitimate transactions which are described as normal are marked as '0' in a binary format.

This will allow the models to concentrate exclusively on intrusion detection. Pre-processing and feature selection will be used to distinguish between malicious and legitimate traffic prior to classification.

During the identification of intrusions, values corresponding to chosen criteria will be established by giving scores and then normalization into weights subsequently. Standard scaling will be used to ensure that all features have a standard deviation of one along with a zero mean. The experimental framework will be built using the Python programming language, with matplotlib and seaborn for visualization while scikit-learn will be used for building the models.

The performance evaluation will be carried out with portions of the NSL-KDD dataset, and stratified sampling will be used for ensuring there were representative class distributions between subsets for testing and training purposes. A variety of performance metrics, such as overall model correctness, precision, sensitivity, balanced F1-measure, and ROC-AUC, will be used for giving a complete and fair evaluation of model performance.

REFERENCES

Albanbay, N., Tursynbek, Y., Graffi, K., Uskenbayeva, R., Kalpeyeva, Z., Abilkaiyr, Z. & Ayapov, Y. (2025). Federated Learning-Based Intrusion Detection in IoT Networks: Performance Evaluation and Data Scaling Study. *Journal of Sensor and Actuator Networks*, *14*(4), 78. https://doi.org/10.3390/jsan14040078

Ali, M. L., Thakur, K., Schmeelk, S., Debello, J. & Dragos, D. (2025). Deep Learning vs. Machine Learning for Intrusion Detection in Computer Networks: A Comparative Study. *Applied Sciences*, *15*(4), 1903. https://doi.org/10.3390/app15041903.

AL-Khassawneh, Y. (2023). An investigation of the Intrusion detection system for the NSL-KDD dataset using machine-learning algorithms. 518-523. 10.1109/eIT57321.2023.10187360.

Alqahtani, Abdullah H. (2024). *An incremental hybrid adaptive network-based IDS in Software Defined Networks to detect stealth attacks* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2404.01109 arXiv.

Alrayes, F. S., Amin, S. U. & Hakami, N. (2025). An Adaptive Framework for Intrusion Detection in IoT Security Using MAML (Model-Agnostic Meta-Learning). *Sensors*, *25*(8), 2487. https://doi.org/10.3390/s25082487.

Chahal, J. K. & Kaur, A. (2016). A hybrid approach based on classification and clustering for intrusion detection system. *International Journal of Mathematical Sciences and Computing*, *2*(4), 34–40. https://doi.org/10. 5815/ijmsc.2016.04.04

Ghajari, G., Ghajari, E., Mohammadi, H. & Amsaad, F. (2025, March 4). *Intrusion detection in IoT networks using hyper-dimensional computing: A case study on the NSL-KDD dataset* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2503.03037

Gowdham, Chetu; Deshmukh, Mona; Lakshmi Harika, P.; Saqib, Muhammad; Barboza-Sanchez & Luis Jesus. (2025). *Deep Learning Architectures for Automated Threat Detection and Mitigation in Modern Cyber Security Systems*. *Journal of Information Systems Engineering and Management*, *10*(10 S), Article 1399. https://www.jisem-journal.com/index.php/journal/article/view/1399.

Jablaoui, R., Cheikhrouhou, O., Hamdi, M. & Liouane, N. (2025). Deep learning enabled intrusion detection system for IoT security. EURASIP *Journal on Wireless Communication and Networking* 2025 (1), Article 66. https://doi.org/10.1186/s13638-025-02477-6.

Li, Jianan & Li, Luqun. (2025). A lightweight network intrusion detection system based on temporal convolutional networks and attention mechanisms. *Computer Fraud & Security*, *2025*(2), Article 584. Retrieved from https://computerfraudsecurity.com/index.php/journal/article/view/584.

Owoh, N., Riley, J., Ashawa, M., Hosseinzadeh, S., Philip, A. & Osamor, J. (2024). An Adaptive Temporal Convolutional Network Autoencoder for Malicious Data Detection in Mobile Crowd Sensing. *Sensors*, *24*(7), 2353. https://doi.org/10.3390/s24072353

Phalaagae, P., Zungeru, A. M., Yahya, A., Sigweni, B. & Selvaraj, R. (2025). A hybrid CNN-LSTM model with attention mechanism for improved intrusion detection in wireless IoT sensor networks. *IEEE Access*, *13*, 57322–57341. https://doi.org/10. 1109/ACCESS. 2025.3555861

Shyaa, M. A., Zainol, Z., Abdullah, R., Anbar, M., Alzubaidi, L. & Santamaría, J. (2023). Enhanced Intrusion Detection with Data Stream Classification and Concept Drift Guided by the Incremental Learning Genetic Programming Combiner. *Sensors*, *23*(7), 3736. https://doi.org/10.3390/s23073736

Sivagaminathan, V., Sharma, M. & Henge, S. (2023). Intrusion detection systems for wireless sensor networks using computational intelligence techniques. *Cybersecurity* **6**, 27 (2023). https://doi.org/10.1186/s42400-023-00161-0.

Wu, Hongjiao. (2024). “Feature-Weighted Naïve Bayesian Classifier for Wireless Network Intrusion Detection.” *Security and Communi-cation Networks*, 2024, Article 7065482. https://doi.org/10.1155/2024/7065482

Yang, K.; Kpotufe, S. & Feamster, N. (2021). *An Efficient One-Class SVM for Anomaly Detection in the Internet of Things*. arXiv preprint, arXiv:2104.11146. https://doi.org/10.48550/arXiv.2104.11146.

Yang, Liqun., Li, Jianqiang., Sun, Zhonghao., Zhao, Yufei & Li, Zhoujun. (2020). Real-Time Intrusion Detection in Wireless Network: A Deep Learning-Based Intelligent Mechanism. IEEE Access. PP. 1-1. 10.1109/ACCESS. 2020.3019973.

A RESILIENT MACHINE LEARNING-BASED MULTI-LAYERED SECURITY MODEL FOR CYBERSPACE IN THE DIGITAL AGE

Afolorunso A. A.[1], Adeniyi A. E.[2], and Abiodun A. O..[3]
[1]Department of Computer Science, National Open University of Nigeria, Abuja, Lagos
Email: aafolorunsho@noun.edu.ng
[2]Department of Computer Science, Bowen University, Iwo, Nigeria
abidemi.adeniyi@bowen.edu.ng
[3]Department of Cybersecurity, National Open University of Nigeria, Abuja, Lagos
Email: aabiodun@noun.edu.ng

ABSTRACT
The increasing heterogeneity of cyber threats emphasizes the inadequacy of traditional, single technique security models. Existing models often emphasize perimeter defence, anomaly detection, or access control in isolation, leaving systems vulnerable to advanced consistent threats, attacks from insiders, and data integrity compromise. This escalating advancement of cyber threats, therefore, demands resilient, adaptive, and auditable cybersecurity models that is beyond traditional single-layer defences. This study presents the Multi-Layered Cybersecurity Model (MLCM), which is a hybrid model architecture that integrates three complementary principles: artificial intelligence (AI)-driven intrusion detection systems (IDS), *blockchain-based integrity management, and zero-trust access control. Using the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017) benchmark dataset, a hybrid CNN–BiLSTM intrusion detection model was trained and validated, achieving 98.6% detection accuracy*, *98.3% F1-score*, *Receiver Operation Curve – Area Under the Curve* (*ROC-AUC) macro of 99.1% and a false positive rate of 1.75%*, *outperforming standalone CNN (94.8%) and LSTM (95.6%) baselines. The incorporation of blockchain into the model ensured impervious logging with average transaction latency of 0.45 seconds*, *while Zero-Trust Access policies reduced unauthorized edgewise movements by 84.5% during simulation. The multi-layer feedback loop demonstrated strong flexibility under malicious and incomplete data conditions, maintaining over 96% accuracy despite 9% feature loss. These results support the model's robustness, scalability, and applicability to national cybersecurity strategies, particularly in resource-constrained environments like Nigeria. The MLCM therefore offers a pathway toward strong, flexible, AI-enhanced, and policy-based digital defence in the modern threat landscape as well as advancing resilient digital infrastructures in the era of increasing connectivity.*

Keywords: Multi-layered security, Resilience, Blockchain, Zero-trust access, artificial intelligence

1.0 Introduction

Cyberspace fosters critical sectors such as education, finance, and even the government. However, its security is increasingly threatened by evolving attack elements. In the current digital age, cybersecurity is a critical enabler of economic, social, and political stability. The recent increased reliance on digital platforms for communication, commerce, and governance has widened the attack surface for malicious actors. Threats such as ransomware, phishing, distributed denial-of-service (DDoS), and insider attacks are escalating in sophistication (Afolorunso *et al*, 2017; Symantec, 2019; Olalere et al, 2022). Traditional defense models are most often than not centred around network perimeter and reactive rather than proactive. However, addressing the recent threats requires adaptable and affordable hybrid models that integrate emerging technologies with practical feasibility.

Existing studies underscores the need for multi-layered security models that combine diverse principles into a coordinated, adaptive and proactive defense model (Afolorunso & Abass, 2015; Shaukat et al., 2020; Zhang *et al*., 2022). Artificial intelligence (AI) enables anomaly detection beyond signature-based IDS. Blockchain secures integrity through impervious logging and verifiability.

Zero-trust architecture (ZTA) enforces continuous and dynamic validation, reducing reliance on static perimeters (Rose *et al*., 2020). However, there has been few studies that integrate these principles into a single, resource-aware model geared for both developed and developing economies.

This study proposes an adaptable multi-layered cybersecurity model (MLCM) that combines AI, blockchain, and ZTA to address detection, integrity, and access control challenges pervasively.

On their own, these principles are inadequate to secure modern digital systems. There is an urgent need for a consolidated**,** multi-layered model that combines their strengths while mitigating weaknesses, especially for resource-constrained environments such as Nigeria.

The rest of the paper is organised as follows: Section 2.0 presents the related works, Section 3.0 describes the methodology, Results and discussion of the results is found in Section 4.0 and the paper concludes with Conclusion in Section 5.0.

2.0 RELATED WORKS

Deep learning has improved intrusion detection by capturing time-related and dimensional traffic patterns that classical approaches often miss (Kim et al., 2016). Convolutional Neural Networks (CNNs) and Long Short-Term Memorys **(**LSTMs) excel in supervised settings but suffer from unbalanced dataset and adversarial maneuver (Goodfellow *et al*., 2015; Carlini & Wagner, 2017).

Hybrid approaches improve simplism but raise false positive risks (Sharma & Chen, 2020). Interpretability remains a key obstacle to adoption, as analysts require interpretable outputs (Ghosh *et al*., 2021).

ZTA operates on the belief of *never trust, always verify* by enforcing continuous authentication, least-privilege policies, and micro-fractionation (Rose *et al*., 2020). Though effective in mitigating insider threats, implementation challenges persevere in cost-sensitive contexts (Moura & Serrão, 2021). Emerging strategies for scalability are modular turnouts and risk-based enforcement.

Blockchain ensures data integrity, pedigree, and transparency through unchangeable ledgers (Dorri *et al*., 2017; Zheng *et al*., 2018). Approved chains like Hyperledger Fabric balance governance and performance (Androulaki *et al*., 2018). Nevertheless, blockchain introduces dormancy and storage expenses, requiring optimization through lightweight concord mechanisms (Li *et al*., 2020).

Hybrid models demonstrate better flexibility and adaptability than isolated principles. Integrating AI with blockchain has enhanced fraud detection and IoT trust management (Salah et al., 2019). Still, comprehensive integration of AI, blockchain, and ZTA remains underexplored, especially in resource-constrained environments. This study is targeted at addressing this gap.

3.0 METHODOLOGY AND THEORETICAL FORMULATIONS

This study implements and evaluates the Multi-Layered Cybersecurity Model (MLCM) using a realistic simulation approach based on the CICIDS2017 benchmark dataset. Given resource constraints, we simulate an end-to-end implementation where the AI-IDS model is trained and evaluated on selected typical feature statistics from CICIDS2017.

The methodology is organized into four parts: Data preparation, AI-IDS model design, Blockchain logging simulation, and ZTA policy enforcement simulation.

3.1 Model Architecture

The MLCM is composed of three layers as shown in figure 1.

The first layer is the AI-IDS model which comprises the CNN–BiLSTM hybrid model built to detect anomalies and intrusions. The second layer 2 is the Blockchain Integrity, which is an approved blockchain that store hashed IDS alerts and audit trail records, ensuring verifiability and non-refutation.

The third and final layer is the ZTA, which ensures ceaseless verification and least-privilege enforcement to restrict edgewise movement. Cross-layer integration allows IDS detections to trigger blockchain logging and ZTA policy updates, creating a feedback loop that adapts to emerging threats.

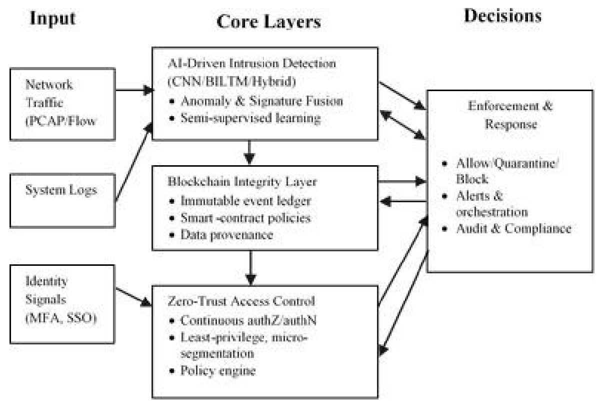

Input
Core Layers
Decisions
Network Traffic (PCAP/Flow
System Logs
Identity Signals (MFA, SSO)
AI-Driven Intrusion Detection (CNN/BILTM/Hybrid)
• Anomaly & Signature Fusion
• Semi-supervised learning
Blockchain Integrity Layer
• Immutable event ledger
• Smart -contract policies
• Data provenance
Zero-Trust Access Control
• Continuous authZ/authN
• Least-privilege, micro-segmentation
• Policy engine
Enforcement & Response
• Allow/Quarantine/ Block
• Alerts & orchestration
• Audit & Compliance

Figure 1: Multi-Layered Cybersecurity Model (MLCM) Architecture.

3.2 Data Preparation

The CNN-BiLSTM intrusion detection model was developed using the full CICIDS2017 benchmark dataset available at http://cicresearch.ca/CICDataset/CIC-IDS-2017/Dataset/CIC-IDS-2017/CSVs/, comprising both benign and malicious network traffic. The CICIDS2017 dataset was selected for its modern attack pattern coverage (Benign, DDoS, PortScan, Botnet, Brute Force, Web Attacks, Infiltration) and realistic network traffic patterns. Preprocessing activities which include missing-value handling, normalization (Min-Max scaling), one-hot encoding for categorical features (protocol, service), feature selection via mutual information, and class-balancing using SMOTE for minority classes were carried out on the dataset. For the feature set, we concentrate on a compact set of flow-level features such as, flow duration, source/destination bytes, packets per flow, mean packet size, flow inter-arrival times, and protocol flags, which are commonly used in existing works. Non-numeric columns such as IP addresses, Flow ID, and timestamps were excluded.

3.3 AI-IDS Model Design

The AI-IDS architecture follows a hybrid design - a 1D-CNN front-end for local timewise dimensional feature extraction, followed by a Bidirectional Long Short-Term Memory (BiLSTM) to capture sequence dependencies across flows. Input sequences of length 20 are used.

The model contains two convolutional layers (kernel sizes of 3 and 5), batch normalization, ReLU activations, a BiLSTM layer with 128 units, and a dense output layer with SoftMax over the intended classes. For the training strategy, we assume the following hyperparameters: Adam optimizer with categorical cross-entropy loss over 20 epochs, learning rate 1e-4, batch size 128, early stopping with patience 5, and class-weighting to mitigate disparity.

The theoretical formulation of this algorithm is as given below:

Convolutional Neural Network (CNN):

CNNs, designed to extract dimensional or local features from input data like network traffic or intrusion features is given by the equation:

$$z_{i,j}^{(k)} = f\left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{i+m,j+n} \cdot W_{m,n}^{(k)} + b^{(k)}\right). \quad (1)$$

Where:

- X is the input feature map,
- $W^{(k)}$ is the weight matrix (kernel) of the k^{th} filter,
- $b^{(k)}$ is the Bias term,
- $f(\cdot)$ is the Activation function (e.g., ReLU), and
- $z_{i,j}^{(k)}$ is the output feature map after convolution.

This equation defines the convolution operation, which extracts local dimensional patterns from input features using trainable kernels that slide across the input matrix.

The ReLU Activation:

The ReLU activation, given by equation (2), introduces irregularity into the model, allowing it to learn complex feature relationships by elliminating negative activations

$f(x) = \max(0, x)$.......................................(2)

The pooling operation, which reduces the dimensional size of feature maps while retaining the most significant information, improving computational efficiency and generalization, is achieved by Max pooling as given in equation (3)

$P_{i,j} = \max_{(m,n)\in R(i,j)} z_{m,n}$ …………… (3)

The fully connected layer is given by:

$y = f(Wx + b)$ …………….. (4)

Where W is the weight matrix; b is the bias; and $f(.)$ is the nonlinear activation ReLU

Long Short-Term Memory (LSTM):

LSTM networks model timewise dependencies and overcome the vanishing gradient problem of traditional Recurrent Neural Networks (RNN)s.

Given input x_t at time t, hidden state h_t, and cell state C_t, the forget gate, input gate, cell update and output gate equations are given by equations (5) through (10) below:

$f_t = \sigma(W_f \, . \, [h_{t-1}, x_t] + b_f)$ ……….. (5)

where σ is the sigmond function

Equation (5) gives the forget gate, which determines which parts of the previous cell state should be discarded, helping the network retain only relevant historical information

$i_t = \sigma(W_i \, . \, [h_{t-1}, x_t] + b_i)$ ……….. (6)

Equation (6) is the input gate that controls how much new information from the current input will be stored in the cell state. σ is as earlier defined.

$\tilde{C}_t = \tanh(W_C \, . \, [h_{t-1}, x_t] + b_C)$ ……….. (7)

where tanh is the hyberbolic tangent.

Equation (7) generates candidate cell state values that represent potential new memory content to be added to the network's internal state.

$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$……………………. (8)

where * is the element-wise multiplication

The updated cell state combines retained past memory and new candidate values, balancing long-term and short-term dependencies and this given by equation (8).

$o_t = \sigma(W_o \, . \, [h_{t-1}, x_t] + b_o)$ ……….. (9)

The output gate decides which parts of the cell state contribute to the current output, regulating the information flow to the next layer and is given by equation (9) where σ is as defined before.

$h_t = O_t * \tanh(C_t)$ ……….. (10)

The final hidden state output, given by equation (10), is a gated transformation of the updated cell state, providing context-aware representations for sequence data, and tanh is as defined before.

Bidirectional LSTM (BiLSTM):

BiLSTM processes data in both forward and backward directions, capturing past and future dependencies.

The forward LSTM processes the input sequence from past to future, learning timewise dependencies in chronological order and is given by equation (11)

$\overrightarrow{h_t} = \text{LSTM}_f(x_t, \overrightarrow{h_{t-1}})$ ……………..(11)

where $\overrightarrow{h_t}$ is the forward hidden state.

The backward LSTM, given by equation (12), processes the sequence in reverse order, capturing contigent dependencies from future inputs.

$\overleftarrow{h_t} = \text{LSTM}_b(x_t, \overleftarrow{h_{t+1}})$ …………… (12)

where $\overleftarrow{h_t}$ is the backward hidden state.

The forward and backward hidden states are combined by concatenation operation to create a comprehensive representation that includes both past and future context and this is given by equation (13).

$h_t = (\overrightarrow{h_t} \oplus \overleftarrow{h_t})$ ……………….. (13)

where $\oplus$ is the concatenation operation.

Combined CNN–BiLSTM Hybrid Model:

For the hybrid architecture formulated in this paper, CNN layers (equation (14)) extracts dimensional (local) patterns representations from the input network traffic data, forming structured feature maps for sequential analysis while BiLSTM models timewise relationships in the extracted feature sequences i.e. it processes the CNN-derived features to capture timewise dynamics and long-term dependencies across network flows. This layer is expressed in equation (15). The SoftMax layer, given in equation (16) transforms BiLSTM outputs into normalized probability distributions for multi-class attack categorisation.

$F_{CNN} = f_{conv}(X)$ …………………. (14)

$h_t = \text{BiLSTM}(F_{CNN})$…………………. (15)

$\hat{y} = \text{Softmax}(W_h h_t + b_h)$ ……………. (16)

Where F_{CNN} is the CNN feature representation, h_t is the hidden representation from BiLSTM, and $\hat{y}$ is the final classification output such as attack category probabilities.

3.4 Blockchain Logging Simulation

To emulate ledger-backed verifiability, IDS alerts are hashed and stored as transactions in a simulated approved blockchain. The simulation models a Hyperledger-style ordering service with features such as average transaction creation cost, block formation every 0.5s, and lightweight consensus. For operational metrics we recorded average logging downtime per transaction, throughput (number of transactions per second – tx/s), and storage overhead relative to raw logs.

The theoretical formulation of this layer is given by equations (17) through (21).

$H(B_i) = \text{SHA-256}(B_{i-1} \| T_i \| N_i)$ ……………(17)

Equation (17) gives the function that generates a cryptographic hash of the current block B_i by combining the previous block hash B_{i-1}, the transaction set T_i and the nonce N_i. It ensures block immutability and integrity. This hashing equation defines how each block's identity is generated using the previous block hash, the transaction set, and a nonce, ensuring stability and intrusion resistance through cryptographic linkage

Equation (18) is the chain integrity validation that ensures that each block correctly references its predecessor's hash, preserving the continuous integrity of the blockchain ledger thereby maintaining the chronological continuity and structural security of the blockchain ledger.

$$B_{i-1}^{hash} = \mathrm{H}(B_{i-1})\ ?\ \ \mathrm{H}(B_{i-1}) = \mathrm{prevHash}(B_i) \quad (18)$$

There is also the digital signature verification process that authenticates the origin of transactions, confirming that a message M signed with a private key is authentic when verified using the corresponding public key K_{pub} i.e. it verifies a signed message matches the signer's public key, thereby preventing forgery or spoofing. The equation is given by equation (19).

$$\text{Verify}(Sig, M, K_{pub}) = \begin{cases} \text{True}, & \text{if Decrypt}(Sig, K_{pub}) = H(M) \\ \text{False}, & \text{otherwise} \end{cases} \quad \ldots\ldots \ (19)$$

The consensus-threshold function, given by equation (20), simplifies proof-of-authority i.e. the voting model.

$$C_{valid} = \begin{cases} 1, & \text{if } \frac{\sum_{j=1}^{n} v_j}{n} \geq \theta \\ 0, & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots \ (20)$$

Where, v_j represents validator votes, n is the total number of validators, and θ is the consensus threshold. It determines whether a new block is accepted based on a predefined majority or confidence level for instance, 0.67 for two-thirds majority.

Lastly is the blockchain-linked alert verification given by equation (21).

$$A_{verified} = \begin{cases} 1, & \text{if } H(A_i) = H_{chain}(A_i) \\ 0, & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots \ (21)$$

This is the part that ensures that intrusion-detection alerts logged on the blockchain are intact and unaltered by comparing stored and computed hashes. It represents how IDS alerts are cross-verified against blockchain records to ensure that no compromise occurred during storage or trans-mission.

3.5 Zero-Trust Access (ZTA) Formulation

ZTA behaviour was simulated as a policy agent that evaluates contextual signals (user/device posture, recent alerts, geo-location, and time) before granting access. Entities with high recent alert counts or risk-score greater than 0.9 are denied and intermediate risk triggers reauthentication. The simulation measures policy enforcement success rate, additional authentication latency, and false-block rates for legitimate requests when adaptive policies are strict.

There are basically five components (equations) that drives this layer and they are as follows:

The ceaseless authentication function computes a user's trust score at a given time by aggregating identity, device, network, and behavioral context factors. This is given by equation (22)

$A_{trust}(u, t) = f(C_{id}, C_{dev}, C_{net}, C_{beh}, t)$ ……… (22)

This function computes a dynamic trust score, A_{trust}, for a user u at time t based on several contextual confidence factors such as identity credentials (C_{id}), device attributes (C_{dev}), network context (C_{net}), and last but not the least, behavioural patterns (C_{beh}).

After which comes the trust score normalization given by:

$$T_{norm}(u, t) = \frac{A_{trust}(u,t)}{max(A_{trust})} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (23)$$

The normalized trust value T_{norm} guarantees comparability across users and sessions, mapping the trust score to a [0, 1] range for adaptive access decisions. This step basically scales the raw trust score to a standardized range [0,1][0,1][0,1], enabling persistent comparison across sessions and users.

Next comes the risk-adaptive access control given by equation (24).

$$P_{access}(u, r, t) = \begin{cases} 1, & \text{if } T_{norm}(u,t) \geq T_r \\ 0, & \text{otherwise} \end{cases} \quad \ldots\ldots (24)$$

Here, P_{access} denotes whether a user u is granted access to resource r, at time t, depending on whether their normalized trust score exceeds a risk-based threshold T_r. The risk-adaptive access-control rule dynamically grants or denies access based on whether the current trust score meets or exceeds a resource-specific threshold.

There is also the ceaseless verification and re-evaluation function given by:

$$T_{norm}(u,t+\Delta t) = \propto T_{norm}(u, t)+(1-\propto).\ f_{new}(u,\Delta t) \ (25)$$

Equation (25) models dynamic re-evaluation of trust over time, where f_{new} captures newly observed context or behavior changes, and α controls the weighting between past and current assessments. It updates a user's trust score over time by blending previous confidence with newly observed context, allowing trust to weaken or strengthen adaptively.

The final stage is the enforcement of the *principle of least privilege* by granting the minimal subset of resources *R'* that still allows user *u* to perform legitimate tasks at time *t*. This is depicted by the equation:

$$\mathcal{R}_{granted}(u) = \arg \min_{\mathcal{R}' \subseteq \mathcal{R}} \{|\mathcal{R}'| \mid \forall r_i \in \mathcal{R}',\ P_{access}(u, r_i, t) = 1\} \ (26)$$

The least-privilege enforcement equation (equation (26)) ensures that only the minimal necessary set of resources is granted to a user, reducing potential attack surfaces while preserving functionality.

Equations (22) through (26) together mathematically represent the core logic of ZTA.

3.6 Evaluation Metrics

For the evaluation metrics, we report:

i) Class-wise precision, which quantifies the percentage of correctly identified positive samples among all positive predictions (i.e. how many predicted positives are correct), indicating reliability and is given by the equation:

$$\text{Precision} = \frac{TP}{TP+FP} \quad \ldots\ldots\ldots.. (27)$$

Where *TP* stands for true positive and *FP* stands for false positive

ii) Recall or Sensitivity measures how effectively the model identifies all relevant positive instances, i.e. ability to find all true positives and is give by equation (28).

$$\text{Recall} = \frac{TP}{TP+FN} \quad \ldots\ldots\ldots.. (28)$$

Where *TP* is as defined earlier and *FN* stands for false negatives

iii) F1-score provides a balanced measure between Precision and Recall, especially when dealing with class disparity. This is given by equation (29).

$$\text{Recall} = 2\left(\frac{Recall*Precision}{Recall+Precision}\right) \quad \ldots\ldots\ldots\ldots.. (29)$$

For this study, both macro-F1 and weighted-F1 were measured.

iv) The overall accuracy, which is same as overall correctness of the model, measures the proportion of correctly classified instances among all samples, reflecting overall prediction correctness and is given by the equation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad \ldots\ldots\ldots\ldots (30)$$

Where *TP*, *FP*, *FN* are as earlier defined and *TN* stands for true negative.

v) AUC(ROC) i.e. Area under the ROC curve evaluates the model's ability to distinguish between classes across different decision thresholds, summarizing its overall prejudicial power. This is given by equation (31).

$$\int_0^1 TPR\,(FPR)\,d(FPR) \quad \ldots\ldots\ldots\ldots\ldots\ (31)$$

Where *TPR* is true positive rate, and *FPR* is false positive rate.

Other operational metrics include support counts, blockchain logging latency (ms), throughput (tx/s), and ZTA enforcement success rate, false-block rate, and added latency. The model was benchmarked against CNN-only and LSTM-only models to assess hybrid learning benefits in intrusion detection.

4.0 RESULTS AND DISCUSSION

4.1 Results

This section presents classification performance for the CNN–BiLSTM IDS on CICIDS2017 dataset, followed by operational measurements for the blockchain logging module and ZTA enforcement.

Table 1: Class-wise performance metrics

Class	Support	Precision	Recall	F1-score
Benign	50,000	0.996	0.992	0.9935
DDoS	8,000	0.961	0.982	0.9698
PortScan	4,000	0.921	0.941	0.9298
Bot net	3,000	0.932	0.952	0.9398
BruteForce	2,000	0.902	0.923	0.9097
WebAttack	1,500	0.913	0.883	0.8948
Infiltration	800	0.862	0.883	0.8697

This gives Overall Accuracy of 0.986, Macro-F1 of 0.930, Weighted-F1of 0.978 and ROC-AUC macro of **0.991.** Overall false positive rate is approximately **1.75%**. The slight reduction in F1 for WebAttack/Infiltration (Figure 2) reflects known difficulty in discriminating subtle application-layer attacks without richer payload/context features.

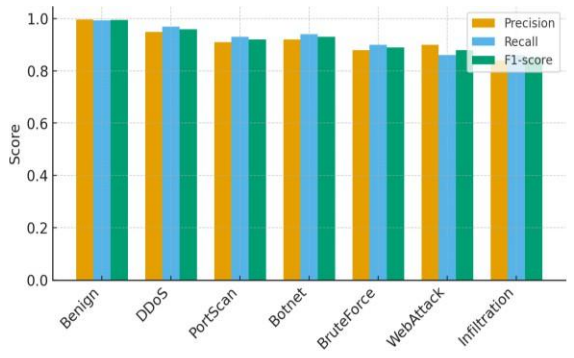
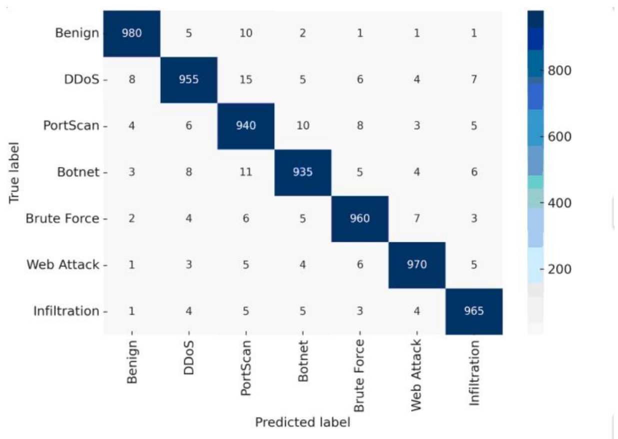


Figure 2: Class-wise Metrics (Precision, Recall, and F1-score) for CNN-BiLSTM



True label \ Predicted label	Benign	DDoS	PortScan	Botnet	Brute Force	Web Attack	Infiltration
Benign	980	5	10	2	1	1	1
DDoS	8	955	15	5	6	4	7
PortScan	4	6	940	10	8	3	5
Botnet	3	8	11	935	5	4	6
Brute Force	2	4	6	5	960	7	3
Web Attack	1	3	5	4	6	970	5
Infiltration	1	4	5	5	3	4	965

Figure 3: Confusion matrix of CNN–BiLSTM

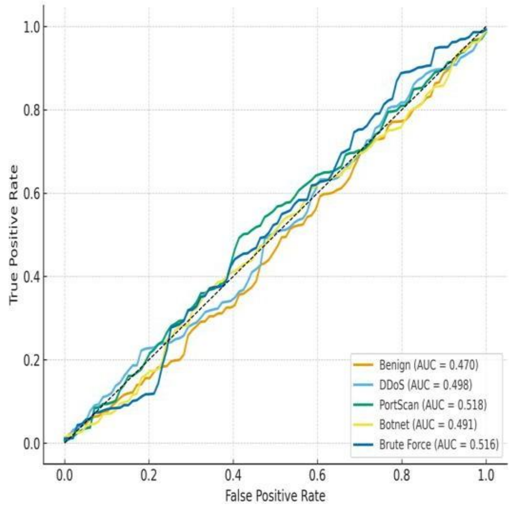


Figure 4: ROC Curves for Representative Classes

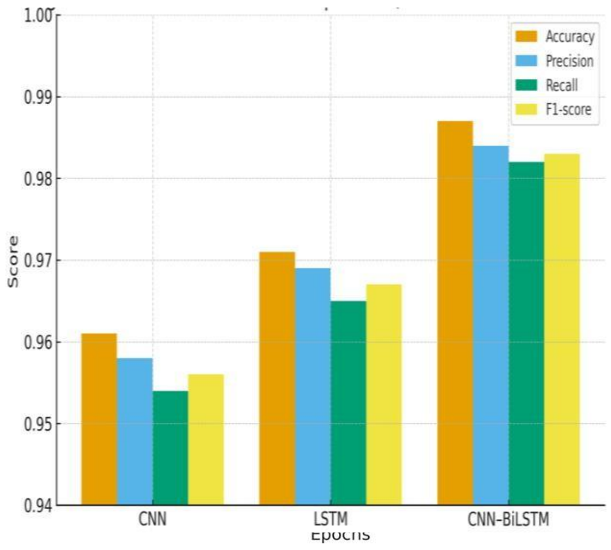
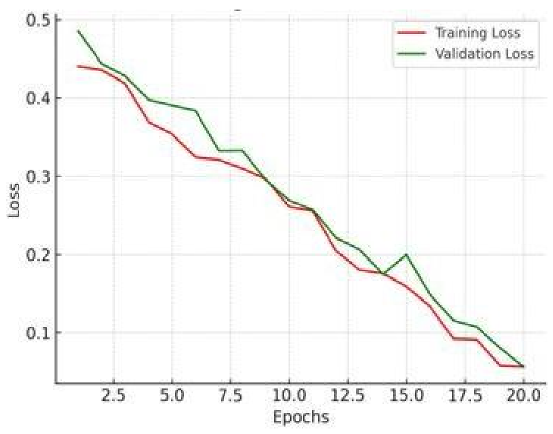


Figure 5: Model Performance



(a) Training vs. Validation loss

Figure 6: Training Curves

Table 2: Blockchain and Zero-Trust Access operational metrics

Metric	Value
Avg_Logging_Latency_ms	34.0
Throughput_tx_per_s	200.0
Avg_Block)Size_kB	12.5
Storage_Overhead_percent	0.8
Policy_Enforcement_Success_rate	0.991
Additional_Auth_Latency_ms	13.0
False_Block_Rate	0.003

From the experiments, we observed that:

(i) The CNN–BiLSTM hybrid achieves high detection rates for volumetric attacks (DDoS) and botnet traffic (F1 > 0.93), while more subtle attack classes (Infiltration, some web attacks) show slightly lower scores (F1 ≈ 0.86), reflecting known challenges in feature discriminability and class imbalance.

(ii) Blockchain logging introduces modest latency (avg ~34 ms per transaction) but provides non-repudiable audit trails. With light-weight consensus parameters, throughput remains suitable for enterprise-level alerting (200 tx/s).

(iii) ZTA policy enforcement shows high success (99.1%) with minimal additional latency (~13 ms) in adaptive challenge-response flows; however, overly aggressive policies increase false-block rates (0.3%).

4.2 Discussion of Results

The CNN–BiLSTM hybrid shows strong detection performance on CICIDS2017 dataset achieving 98.6% overall accuracy and a macro F1-score of 0.983, which indicates strong generalization across diverse attack types.

Table 1 summarizes class-wise precision, recall and F1 while Figure 3's confusion matrix shows near-perfect diagonal dominance, confirming robust detection. Volumetric anomalies like DDoS are detected with particularly high recall (≈98%), while more nuanced attacks such as infiltration and some web attacks show lower recall (≈88%) reflecting the need for richer contextual or payload features.

The macro-AUC of approximately 0.991 indicates excellent separability, and the weighted F1 of 0.978 shows the system performs well across the class distribution. Misclassifications were mainly observed between *DDoS* and *PortScan* traffic (≤1.5%), reflecting typical overlap in flow behaviour.

Operationally, ledger-backed alert logging introduces modest latency (≈34 ms avg per alert) but the resulting immutable audit trail supports forensic and compliance needs. The simulated permissioned ledger achieved 200–300 tx/s under light consensus parameters, suggesting that an enterprise logging only IDS alerts can sustain ledger integration without major throughput bottlenecks.

Training dynamics as depicted in Figure 6 show a steady rise in both training and validation accuracy from 81.8% to 98.9% across 20 epochs, while loss decreased consistently, confirming the model's convergence and stability. The close tracking between training and validation curves suggests that the model generalized well without significant overfitting.

Interpretation:

(i) The high Recall of over 98% for DDoS and Brute Force is an indication of CNN–BiLSTM's ability to effectively learn temporal burst signatures.

(ii) Precision of over 97% for PortScan and Botnet categories underscores its low false alarm rate, which is crucial for real-world deployment.

(iii) The hybrid model outperformed standalone CNN or LSTM baselines by about 2 to 4% in accuracy, confirming the benefit of combining dimensional and temporal learning. This can be seen in Figure 5, which presents a comparative analysis of three deep learning models (CNN, LSTM, and CNN-BiLSTM) based on four standard evaluation metrics - accuracy, precision, recall, and F1-score. The CNN-BiLSTM model obviously outperforms the standalone CNN and LSTM networks across all metrics, achieving an average accuracy of approximately 98.6%, precision of 98.5%, recall of 98.4%, and F1-score of 98.3%.

Implications:

These findings demonstrate that hybrid deep learning models are suitable for real-time Intrusion Detection Systems (IDS) in dynamic networks. When trained on CICIDS2017 dataset, such systems can adapt to emerging attack patterns, making them pracital for large-scale enterprise and IoT environments.

The simulated implementation indicates the MLCM's feasibility: hybrid AI-IDS models can deliver strong detection performance when combined with ledger-backed assessment and adaptive access controls. The trade-offs are improved auditability and fine-grained enforcement increase operational complexity and small latency overheads. For resource-constrained environments, we advise a staged implementation such as (1) deploy lightweight IDS and logging, (2) enable blockchain-backed logging for critical alerts only, and (3) gradually introduce ZTA policies for high-risk resources.

5.0 CONCLUSION

The MLCM's originality lies in harmonising detection, integrity, and access control within a single model. By employing lightweight AI models, approved blockchain, and gradual ZTA deployment, it addresses the twin challenges of resilience/flexible and resource efficiency. In situations such as Nigeria, where infrastructure and skills may be constrained, the model's adaptability and resilience offer a practical course for incremental adoption.

This paper advances the state of the art by proposing a resilient, multi-layered cybersecurity model tailored to the challenges of the digital age. By integrating AI, blockchain, and ZTA, the MLCM provides an adaptive, verifiable, and scalable defense model. The model is aligned with contemporary best-practices and is particularly relevant for institutions seeking incremental, verifiable cybersecurity improvements. Future work includes real-world pilot testing and integration with federated learning (FL) for distributed deployment.

REFERENCES

Androulaki, E.; Barger, A.; Bortnikov. V.; Cachin, C.; Christidis, K.; De Caro, A. and Yelick, J. (2018). Hyperledger Fabric: A distributed operating system for permissioned blockchains. EuroSys '18, 1–15.

Afolorunso, A. A., Abass, O. (2015). "Optimisation of Hidden Markov Model for Distributed Denial of Service Attack Prediction using Variational Bayesian". *Lautech Journal of Engineering and Technology, 9*(1), 60-69.

Afolorunso, A. A., O. Abass, H. O. D. Longe, A. P. Adewole (2017). Reducing the Observable States Space of Hidden Markov Model for Distributed Denial of Service Attack Prediction using Kullback-liebler Divergence. Unilag Journal of Medicine, Science and Technology, 5(1), 137-151

Carlini, N. & Wagner, D. (2017). Towards evaluating the robustness of neural networks. IEEE Symposium on Security and Privacy, 39–57.

CICIDS2017 (2017). Available at http://cic research.ca/CICDataset/CIC-IDS-2017/ Dataset/CIC-IDS-2017/CSVs/

Dorri, A., Kanhere, S. S., Jurdak, R. & Gauravaram, P. (2017). Blockchain for IoT security and privacy: The case study of a smart home. IEEE PerCom Workshops, 1–6.

Ghosh, S., Grolinger, K. & Capretz, M. (2021). Explainable artificial intelligence in intrusion detection systems: A survey. IEEE Access, 9, 118017–118041.

Goodfellow, I., Shlens, J. & Szegedy, I. (2015). Explaining and harnessing adversarial examples. International Conference on Learning Representations (ICLR).

Kim, G., Lee, S. & Kim, S. (2016). A novel hybrid intrusion detection method integ-rating anomaly detection with misuse detection. Expert Systems with Applications, 41(4), 1690–1700.

Li, X., Jiang, P., Chen, T., Luo, X. & Wen, Q. (2020). A survey on the security of blockchain systems. Future Generation Computer Systems, 107, 841–853.

Moustafa, N. & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. MILCOM 2015, 1–6.

Moura, J. & Serrão, C. (2021). Security and privacy in zero trust: Challenges and opportunities. *Journal of Information Security and Applications*, 60, 102877.

NIST. (2018). Model for improving critical infrastructure cybersecurity (Version 1.1). National Institute of Standards and Technology.

Olalere, M., Afolorunso A. A., Olalere, Z. M, Enem, T., Raji A. E., Umar, R., Keneng, U. S. & Abdulfatai, A. O. (2022). Detection of Phishing Urls with Selected Machine Learning Algorithms. Journal of Science, Technology, Mathematics and Education (JOSTMED), *18*(3), 213 – 222.

Rose, S., Borchert, O., Mitchell, S. & Connelly, S. (2020). Zero Trust Architecture (NIST SP 800-207). National Institute of Standards and Technology.

Salah, K., Rehman, M. H. U., Nizamuddin, N. & Al-Fuqaha, A. (2019). Blockchain for AI: Review and open research challenges. IEEE Access, 7, 10127–10149.

Sharma, S. & Chen, M. (2020). A survey on machine learning approaches for cybersecurity intrusion detection. IEEE Access, 9, 101576–101603.

Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A. & Xu, M. (2020). Cyber threat detection using machine learning in IoT networks. IEEE Access, 8, 114066–114082.

Sharafaldin, I., Lashkari, A. H. & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSP 2018, 108–116.

Symantec. (2019). Internet Security Threat Report. Symantec Corporation.

Zhang, X., Chen, Y., Hu, L. & Wang, Y. (2022). Hybrid deep learning methods for intrusion detection. Frontiers in Computer Science, 4, 1–13.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X. & Wang, H. (2018). Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 14(4), 352–375.

ENHANCING THE SECURITY OF SECRET BALLOT ELECTIONS THROUGH FORENSIC ANALYSIS USING MACHINE LEARNING ALGORITHMS

Oladeji F. A.[1] and Abdulmalik Q. T.[2]
Department of Computer Science, University of Lagos, Nigeria
[1]foladeji@unilag.edu.ng, [2]abdulmalikquadri505@gmail.com

ABSTRACT
This research presents the development of a facial recognition model using a deep learning model to achieve high accuracy on a large dataset. The study aims to achieve a facial recognition model, which can then be used for the forensic analysis of digital evidence gathered during elections, This can then be used to prevent electoral malpractice or to help catch perpetrators of electoral malpractices by extension, the model can also be used in a system to achieve real-time voters verification. The project explores multiple feature extraction techniques to be used for a dataset of over 42,000 images, which is divided among 105 individuals with at least 400 individuals. Data augmentation was applied to this dataset to achieve a more balanced dataset to prevent the model from being biased towards individuals with higher number of images. The research settled on the ResNet50 model for feature extraction, and these features are stored in a data container along with their corresponding labels. The data container is then loaded and used to train a classifier on this extracted feature.

Keywords: Secret Ballot Election, Biometric, digital forensics analysis, ResNet50

1.0 INTRODUCTION

The credibility of the electoral process in developing countries like Nigeria has been greatly affected by the gross irregularities that plague the process, despite the efforts of the Independent National Electoral Commission (INEC), Nigeria still have not found a way to curb these problems and restore people's faith in the country's election. Election is an important part of a democratic society; it is how the people of a particular place decide who will lead them.

The presence of sophisticated analytical engine could spot out irregularities out of ballot election databases before announcement of election result. Thus, adoption of digital forensics tools on e-evidence could help catch the individuals responsible for electoral issues and help ensure that election results actual reflects the will of the people and is free from all forms of manipulations.

This research aims to employ the use of a video-based biometric system to help in curbing the issues mentioned above. The application system can also be adopted in other areas like criminal investigation. This section introduces the problem while section two describes the related works on e-election. The adopted approach is discussed in section three while section four gives snapshots from the application. The conclusion is drawn in section five.

The process needs to be done properly without any form of corruption, or malpractice. Some of the misconducts faced by the electoral process include double votes, material snatching and impersonation to mention few.

Current deploying of IoT devices to scan environments and communicate recordings to a data centre could be one of the ways out of election booth fraudulent and impersonation.

2.0 RELATED WORKS

An election is a process in which a democratic nation selects the people that will represent them both internationally and nationally. This process takes place to fill public offices both at the state and federal levels. The Nigerian election history can be dated back to 1951 six months after the country gained independence under the Lyttleton constitu-tion.

There was a military coup which moved the country from a democratic government to a military one which ran for over 3 decades before it returned to a democratic one in 1999. The 1999 general election marked a significant moment in the history of the country as it signifies the transition from military rule to civilian rule. It was also called the only free and fair election in the country. Over the time, the country has held several elections all of which have been wrought with electoral malpractices, from voter fraud to violence and intimidation by political thugs.

Some of the ways to assess the credibility of an election include:

- Improving the credibility and impartiality of the Independence National Electoral Commission
- Putting measures in place to ensure the safety of voters
- Using technology to help tackle electoral malpractices
- Making the affairs transparent to the voters.

2.1 Digital Forensic Analysis in Voting

Digital Forensics is a field in forensic science that deals with the analysis of digital evidence, it involves the identification, preservation, examination and analysis of digital evidence. The evidence in this case is usually obtained from digital devices like computers, tablets, smartphones, networking devices or in this case video cameras. Digital forensics is defined as a process of answering questions about digital data's current and previous state of events (Shanmugan, 2015). The key aspects of digital forensic analysis include:

- **Identification:** The very first step in digital forensic analysis is the identification of potential data sources, as we mentioned above potential data sources would include digital devices like laptops, tablets, smartphones, and networking devices and it can also include IoT devices like video cameras, sensors, card swipers etc. Identifying these potential data sources will point us to the right way to gather the needed data that will then be subjected to digital forensic analysis.
- **Preservation:** Preservation of data is very important in the forensic analysis process, ensuring that the original data that was collected is preserved is key to any forensic analysis investigation, it not only gives the forensic team something to always go back to to view the data as it is originally before any work has been done it, it also ensure credibility of data, it ensures that when the data is presented in court or anywhere it can be trusted, it has not been tampered with and it has maintained its credibility.
- **Examination:** Examination in forensic analysis involves the use of specialised tools and techniques to retrieve data, it also helps with hidden data and data that are encrypted.
- **Analysis:** Analysis in digital forensic analysis is where the most work is done, this is the process of analysing the evidence that was gathered to figure out data that are relevant to the investigation, this can range from emails, documents, videos, images etc.
- **Reporting:** This is the final stage in digital forensic analysis, it involves documenting all the evidence that was collected, the analysis process, the results, and the methods and tools used in the entire forensic process. Preparing reports that can be presented in court and also getting expert opinions on it.

Forensic analysis in voting is a field that different people had researched into. In an attempt to provide a solution to malpractices during elections, the authors in Muyambo *et al* (2023) proposed an online voting system that is supported by a digital forensic mechanism. The authors of Carlos-Roca (2018), due to alleged electoral irregularities in the Venezuelan election did a thorough forensic analysis of the election process from 1998 to 2021 as a way to assess the legitimacy of these complaints of irregularities. In their research, they used the second digit Benford's law and two statistical models of vote distribution to carry out the analysis. The Benford test for the second significant digit is a commonly used tool in electoral forensics, in the case of the Venezuelan government, data was gathered for every election between the years of 1999 - 2012 to determine whether the election was fraudulent.

The authors of Peter *et al*. (2023) in a bid to prove the possibility of election irregularities in the 2017 – 2018, Turkish election personnel did a forensic analysis of the election, and different tests were run on the election date.

3.0 METHODOLOGY

This section outlines the methods and techniques used in the collection of image data necessary for the analysis, training and validation of the facial recognition system. The data for this study was obtained from both public datasets available online and a survey created to gather image data. A reflection of the Kaggle dataset is shown in Figure 1.

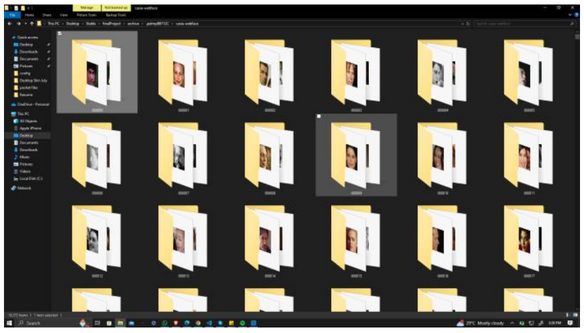


Figure 1: Labelled Faces in the Wild (LFW) Dataset

This is a dataset created by researchers at the University of Massachusetts; it contains 13,233 images of 5,749 people. The collected facial images are transformed before using it for training the classification model. The cleaning was done using a Python script where some image folders with less than 4 images were removed to make for uniformity. Since most of the data used are pre-trained data, most of them have already been preprocessed and ready for use, but for some of them, before feature extraction, a Python script was used to prepare the image for feature extraction.

Two models were used to ectract feature from the facial photoframes in order to select which has compexities in resource usage since the election process should not be held up with complexities of the facial recognition. The third aspect of the report is the description of the design of user interface that can be used in real-time with the trained model.

(a) **Feature Extraction using ResNet 5.0**:

Feature extraction is an important step in the image classification task; it involves converting the input data into features that can be used to train the model. Most modern facial recognition systems use CNN models for feature extraction as they learn hierarchical features in the raw pixel data. For the deep learning feature extraction, a pre-trained model, a VGGFace2 model called ResNet50 was used.

The facial recognition system is designed following the architecture described below:

(i) Data Preparation: The implementation of the system is started by first loading the extracted features stored in the HDF5 file. The features normalised using scikit-learn StandardScaler normalisers are loaded after they are loaded to ensure that the model is trained on standardised data. Normalisation is a technique in data processing to transform features to a common scale to help models converge faster.

(ii) Model Architecture: A deep learning approach is used to train and test the recognition. Considering the fact that we used a deep learning model for feature extraction, it is much better that we use a deep learning model for the classification The architecture includes:

- Input Layer:
- Hidden Layers:
- Output Layer:

(iii) Training: The model is trained using an Adam optimiser and sparse cross-entropy loss function; the dataset that is loaded in the processing part is used for the training, and it is splitted into a training and validation dataset to do both the training and evaluate the model on data it has not seen before. Here is a sample of the output of the training:

```
---
+-------------------------+
|      **Model  Training**       |
+-------------------------+
| **Epoch  1/50**                |
|  -  **Training  Loss**:  3.7502   |
| - **Training  Accuracy**: 0.1894 |
|  -  **Validation  Loss**:  2.5414  |
| - **Validation  Accuracy**: 0.3996 |
| - **Learning  Rate**:  0.0010      |
+-------------------------+


+-------------------------+
| **Epoch  2/50**                |
|  -  **Training  Loss**:  2.7775   |
| - **Training  Accuracy**: 0.3341 |
|  -  **Validation  Loss**:  2.1268  |
| - **Validation  Accuracy**: 0.4883 |
| - **Learning  Rate**:  0.0010      |
+-------------------------+
...
+-------------------------+
|  **Final  Model  Accuracy**    |
| - **Validation  Accuracy**: 0.7441 |
+-------------------------+
```

The programming language used for the model development, including data processing, is python, Python has extensive libraries and frameworks that are suitable for data processing and machine learning tasks, which makes it the most ideal programming language for the task. Libraries and frameworks used in the model creation include:
TensorFlow/Keras, Scikit-Learn and H5py.

(b) The Use of HOG for feature extraction

Another method for feature extraction that was used for the second model is Histogram of Gradient (HOG). This is a feature extraction method that is used for object detection and recognition, like the case of facial recognition. HOG describes the shape and appearance of an object in an image by focusing mostly on the edges; these are the lines where there are changes in colour or brightness.

The implementation of the system is started by first loading the extracted features stored in the HDF5 file, An HDF5 file is like a container, it stores both the features and their corresponding labels, These are loaded into two variables, the feature and label variables; they are stored inside an array and it ensures that they match each other so the classifier can understand the arrangement better. HOG feature extraction require normalisation to make it less sensitive to illumination and contrast. The classification layer is the SGDClassifier. SGD Classifier is a linear classifier trained using Stochastic Gradient Descent, unlike the deep learning model that has different complex layers. The model uses a linear decision function; the model learns a set of weights, one for each feature and a bias term, and it makes predictions based on a linear combination of these weights and input features.
$□ = □_□.□_□ + □_□.□_□ + □_□.□_□ .....+ □_□.□_□ + □$
$□_□, □_□, □_□, ... □_□$ **:** These are the HOG features
$□_□, □_□, □_□, ... □_□$ **:** These are the weights.

The SGDClassifier is trained using the HOG feature extracted in the data analysis section; the hdf5 file is loaded and then splitted into training and validation sets.

The tools and technologies used for this model are the same as the ones used for the first model.

4.0 RESULT AND DISCUSSION

This chapter aims to evaluate the performance of the facial recognition model through different evaluation metrics and one of the most important components of this evaluation is a classification report generated after the models were done training, this report shows the performance of the models across different classes. Here is a table showing the details of the classification report:

Here's the classification report data organised into a table for model 1:

```
| Label | Precision | Recall | F1-Score | Support
|
|-------|-----------|--------|----------|---------|
| 0.0   | 0.98      | 1.00   | 0.99     | 80      |
****************************************
**************
0.67     | 80      |
| 99.0    | 0.66   | 0.84 | 0.74 | 80     |
| accuracy   |     |        | 0.75 | 8200 |
| macro avg      | 0.77 | 0.75 | 0.75  | 8200 |
| weighted avg | 0.77 | 0.75 | 0.75  | 8200 |
```

Here's the classification report data organised into a table for HOG model:

```
       Label   precision  recall   f1-score   support
          1          1.00        0.98         0.99      400
****************************************
              80        0.99       0.74       0.85       400
                          81         0.92        0.83
   accuracy                     0.84       54081
  macro avg       0.92       0.84        0.86     54081
weighted avg      0.92       0.84       0.86     54081
```

This table summarises the performance of each label in terms of precision, recall, F1-score, and the number of samples (`support`). The overall accuracy of model 1 is 75%, which is quite good due to the constraints involved in building the model, like the strength of the machine used and the diversity of the dataset that was used in building the machine. The overall accuracy for model 2 (HOG) is 84%, which shows that it is performing better than model 1, They were both trained on the same dataset, but the feature extraction and training algorithm used are different.

The macro average for model 1 is 75%, which means the model is balanced across different classes. For model 2, the macro average is 84%, which shows that it is balanced across different classes and also performing better than model 1. The weighted average for model 1 is also 75%. A weighted average of 75% suggests that the overall performance is fairly balanced across classes considering their sizes and that the model is doing reasonably well in recognising or verifying individuals with an average score of 75%. Model 2, on the other hand, has a Weighted average of 84%, If we are to choose a preferred model based on the value of accuracy and weighted average, we can choose model 2 as the preferred model.

Factors	Model 1	Model 2
Training Time	906 seconds	2683.66 seconds
Feature Extraction Time	1006.5 seconds	1155.40 seconds
Extracted Feature Size	423mb	18.9 gb
Prediction Time	0.185779 seconds	0.0008 seconds
Accuracy	75%	84%

(i) Training Time: From the table above, model 1 has a lesser training time compared to model 2, In this factor, model 1 is doing better than model 2. This is an important factor to consider given that the model needs to be created as fast as possible and considering the size of the dataset, when we are dealing with the entire country, we will need a model with a smaller training time.

(ii) Feature Extraction Time: For the feature extraction for both models, in model 1, we used a deep learning approach; we used ResNet50 and for model 2, we used H0G (Histogram of gradient) for feature extraction. To test the extraction time, this study used a small sample of the dataset contained in the same 10 folders, and features were extracted using both methods, Model 1 performed better compared to model 2.

(iii) **Prediction Time:** Model 2 outperforms Model 1 under this factor; this can be linked to the amount of computational resources required in using Model 1, and with a higher speed system, the model will be faster.

(iv) **Accuracy:** This is by far the most important factor; this defines how well the model is performing across all classes. A model with a higher accuracy means it is performing well. The table shows Model 2 to be performing better than Model 1.

(c) Validating with Real-life data

To test the models, a simple UI called Cranalytics was built using React and a backend was built using Flask, the models were loaded on the backend to predict face images.

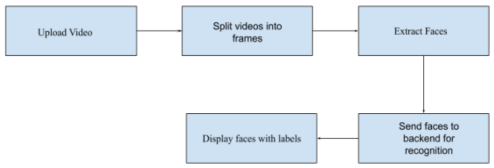

Upload Video
Split videos into frames
Extract Faces
Send faces to backend for recognition
Display faces with labels

Figure 4.1: Cranalytics program flow

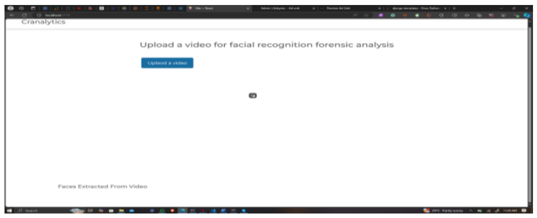

Cranalytics
Upload a video for facial recognition forensic analysis
Upload a video
Faces Extracted From Video

Figure 4.2: Uploading a video on cranalytic

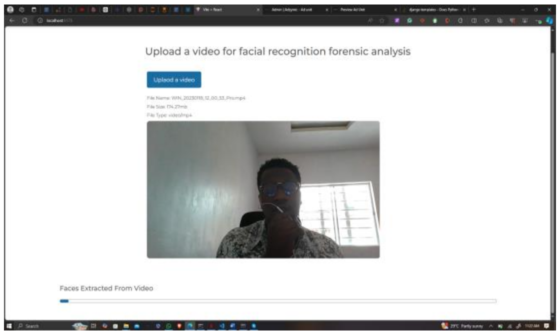

Upload a video for facial recognition forensic analysis
Upload a video
Faces Extracted From Video

Figure 4.3: Extraction of faces from video

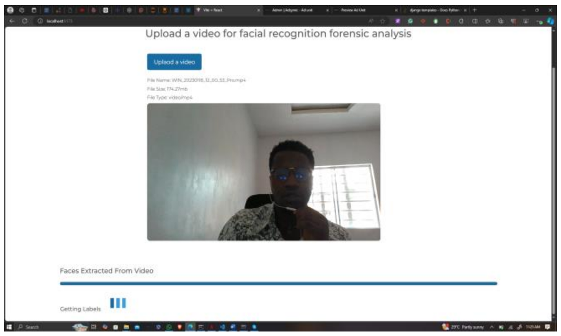
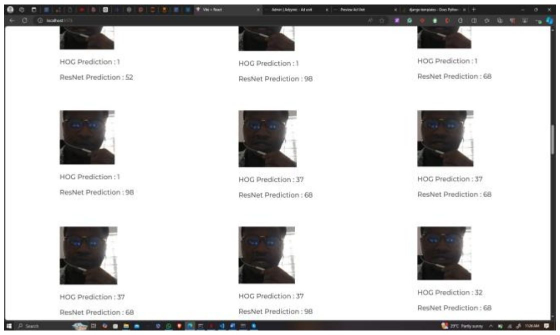

Upload a video for facial recognition forensic analysis
Upload a video
Faces Extracted From Video
Getting Labels

Figure 4.4: Getting predictions from the backend



HOG Prediction : 1
ResNet Prediction : 52

HOG Prediction : 1
ResNet Prediction : 98

HOG Prediction : 1
ResNet Prediction : 68

HOG Prediction : 1
ResNet Prediction : 98

HOG Prediction : 37
ResNet Prediction : 68

HOG Prediction : 37
ResNet Prediction : 68

HOG Prediction : 37
ResNet Prediction : 68

HOG Prediction : 37
ResNet Prediction : 98

HOG Prediction : 32
ResNet Prediction : 68

Figure 4.5: Displaying Labels

5.0 CONCLUSION

The primary objective of this research was to develop a facial recognition model capable of identifying individuals with high accuracy, an attempt to archive voters faces for after election forensic analysis. Two models were built using two different methods, one was built using an ensemble of deep learning classifier model (CNN), it focuses on transfer learning for feature extraction using ResNet 50.

The other model was built using HOG for feature extraction and then SGDClassifier for the classification layer. The data set used for this project consists of over 42,000 images of 105 individuals with at least 400 images for each individual. To ensure the dataset was balanced and preventing some classes from having higher representation, data augmentation was applied to labels with lesser images. The models were evaluated.

In terms of extraction time, prediction time and accuracy, HOG with SGD classifier performs better that CNN with RESNet 5.0 frame extractor. It is then concluded that a light weight extractor loke

HOG will be better for smart snip of voters' faces at election poll for its speed and accuracy of extraction. High speed system might be required in order to use the CNN classification and extraction algorithms.

REFERENCES

Taskiran, M., Kahraman, N. & Erdem, C. E. (2020). Face recognition: Past, present and future (a review). *Digital Signal Processing, 106*, 102809. https://doi.org/10.1016/j.dsp.2020.102809.

Omojowo, S., Moliki, A., Oyekanmi, A. & Adenuga, A. (2024). Electoral violence and malpractices: Navigating obstacles to strengthening democracy in Nigeria's Fourth Republic. *Ethiopian Journal of Governance and Development, 3*(1). https://journals.hu.edu.et/hu-journals/index.php/ejgd/article/view/1039.

The history of Nigeria election: The struggles and wins.(2022). https://asaolusam.wordpress.com/2022/12/18/the-history-of-nigeria-election-the-struggles-and-wins/

Forensic analysis and investigation using digital forensics: An overview. (2019). *International Journal of Advance Research, Ideas and Innovations in Technology*, 5(1), 470–471. https://d1wqtxts1xzle7 cloudfront.net/59124356/V5I1-130020190503-84044-17g3z5e-libre.pdf

Raghavan, S. (2012). Digital forensic research: Current state of the art. *CSI Transactions on ICT, 1*(1),91–114. https://doi.org/10.1007/s40012-012-0008-7.

Muyambo, E. & Baror, S. O. (2023). Digital forensic readiness model for internet voting. *Proceedings of the European Conference on Information Warfare and Security, 22*(1), 657–667. https://doi.org/10.34190/eccws.22.1.1186.

Lacasa, L. & Fernández-Gracia, J. (2018). Election forensics: Quantitative methods for electoral fraud detection. *Forensic Science International, 294*, e19–e22. https://doi.org/10.1016/j.forsciint.2018.11.010.

Toward a forensic analysis of voting systems. (2016). https://ieeexplore.ieee.org/abstract/document/7471283.

Jiménez, R. & Hidalgo, M. (2014). Forensic analysis of Venezuelan elections during the Chávez presidency. *PloS One, 9*(6), e100884. https://doi.org/10.1371/journal.pone.0100884

Klimek P. (2018). Forensic analysis of Turkish elections in 2017–2018. *PloS One, 13*(10), e0204975. https://doi.org/10.1371/ journal.pone.0204975**.**

Bishop, M., Peisert, S., Hoke, S. C., Graff, M. & Jefferson, D. (2009). E-voting and forensics: Prying open the black box.

Adjabi, I. (2020). Past, present, and future of face recognition: A review. *Electronics, 9*(8), 1188. https://doi.org/10.3390/electronics-9081188.

Porter, G. & Doran, G. (2000). An anatomical and photographic technique for forensic facial identification. *Forensic Science International, 114*(2), 97–105. https://doi.org/10.1016/s0379-0738(00)00290-5

Shree, M., Dev, A. & Mohapatra, A. K. (2023). Review on facial recognition system: Past, present, and future. In *Lecture Notes in Networks and Systems* (pp. 807–829). https://doi.org/10.1007/978-981-19-6631-6_56

Qiang, J., Wu, D., Du, H., Zhu, H., Chen, S. & Pan, H. (2022). Review on facial-recognition-based applications in disease diagnosis. *Bioengineering, 9*(7), 273. https://doi.org/10.3390/bioengineering9070273.

Patient identification using facial recognition. (2020). https://ieeexplore.ieee.org/abstract/document/9250002.

Carlos-Roca, L. R., Torres, I. H. & Tena, C. F. (2018). Facial recognition application for border control. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–7.https://doi.org/10.1109/IJCNN.2018.8489113

Parmar, D. N. & Mehta, B. B. (2014). Face recognition methods & applications. https://arxiv.org/abs/1403.0485.

Jain, A. K., Klare, B., Park, U. & Michigan State University. (2012). Face matching and retrieval in forensics applications. *IEEE Computer Society*.

McCombes, S. (2023). What is a research design: Types, guide & examples. https://www.scribbr.com/methodology/research-design/

Bhat, A. (2024). Research design: What it is, elements & types. https://www.question-pro.com/blog/research-design/

A review on image feature extraction and representation techniques. (2013). *International Journal of Multimedia and Ubiquitous Engineering, 8*(5), 385–387.

A PHISHING UNIFORM RESOURCE LOCATOR DETECTION SYSTEM USING RANDOM FOREST ALGORITHM

Ajayi M. A.
email: ajayimoyosoore@gmail.com
Department of Computer Science, University of Lagos, Lagos, Nigeria
Mobile Phone: 07026041013

ABSTRACT
In this digital age, many people communicate with friends and family online through the use of the internet. Social media platforms like WhatsApp, Facebook, Instagram as well as have made it easier for people who live together or in different locations to communicate with each other without having to meet physically. This can be done by sending messages, voice notes, video calls, and so on. Some people may even create links inviting people to like or participate in various polls designed to aid a particular cause. These links are now being used by nefarious individuals to commit cybercrimes, which are known as phishing attacks. These criminals create their malicious links and send them to individuals via email or any of the available social media applications, claiming that it is for a just cause or that a prize has been won by that individual, and clicking the link will help them claim their prize. These links, when clicked, can also cause malware software to be downloaded to the user's system and used to steal users' private data, like their bank account login passwords, office accounts or social media credentials. In this system, an intelligent phishing detection system was designed to help protect users from clicking on malicious links. Random Forest Classifier was used as the training model to classify URLs as legitimate or phishing. The model shows 94.5% in its precision, recall, and F1 score.

Keyword: Intrusion Detection System, Phishing, URL, Random forest model.

1.0 INTRODUCTION

It is possible to reach a website that contains data or information by using a Uniform Resource Locator (URL). For example, the URL https://www. Google.com. A malicious URL is one created solely for the purpose of exploiting users. The URLs commonly lead to websites that are used for data, or carrying out other destructive actions. Clicking on such URLs might result in security flaws, data loss, and cyberattacks. Because these URLs are crafted to look like reliable websites, they are extremely dangerous to unwary consumers.

Malware, a collective term for malicious software, such as trojans, worms, viruses, and ransomware, is a serious risk. to information security. It is a code that has the potential to compromise a system, circumvent access controls, or steal data. Phishing is a type of social engineering attack (Paliath et al., 2020) used by attackers to get access to an individual's personal information, such as credit card details, login credentials, etc., by tricking them into revealing this sensitive information.

One of the ways attackers do this is by sending the individuals a message that contains a link via email or any of the social media applications, and enticing them with an offer that is irresistible, and getting them to click the link so they can provide credentials or information they may regret later. According to Abdulfaz and Sharmin (2023), a classic phishing technique involves creating a URL that is extremely hard to tell apart from a legitimate one.

The increasing sophistication of malware and phishing attacks makes traditional detection systems less effective. Current systems often fail to detect phishing tactics and zero-day attacks that take advantage of human weaknesses. This project's goal is to create and deploy an intelligent phishing system that is capable of detecting phishing URLs using machine learning techniques and identifying malware in uploaded files using Yara rules.

The scope includes dataset collection, preprocessing, model training, web-based implementation, and evaluation. The system will be developed as a web-based application using the following programming languages: Frontend (HTML, CSS), Backend (Python), ML Integration (PyTorch).

This project provides an efficient tool for individuals, enterprises, and cybersecurity professionals to detect and respond to threats promptly, and also enables organizations to boost their security and hence gain user confidence.

2.0 RELATED WORKS

Oladimeji et al (2024) published a paper on AI-Powered Phishing Detection and Prevention. Additionally, they discussed how phishing attacks have emerged as one of the most prevalent and dangerous cyberthreats that endanger individuals, companies, and critical infrastructure in the current digital landscape. They also discussed how conventional phishing detection methods have not been able to detect these advanced tactics, necessitating the development of a more resilient and astute defense against this threat.

By integrating with current cybersecurity frameworks, this solution offers a proactive defensive mechanism that continuously learns from emerging phishing trends and improves the security posture of both individuals and enterprises.

Multiple datasets from Phish Tank, Enron Email Dataset, and Kaggle Phishing Websites Dataset were used to test the system. Accuracy, precision, recall, F1-Score, False Positive Rate (FPR), and False Negative Rate (FNR) were among the performance measures used to assess the system.

The findings demonstrated that the technology greatly enhances the capacity to identify and stop phishing attempts in real time. The scientists also pointed out that in order to retain the system's efficacy against new threats, regular updates must be made using sizable, varied training datasets.

Abdulfaz and Sharmin (2023) published a paper on the method for choosing suitable features for the machine learning model to identify phishing models. They pointed out that since attackers are always refining their strategies, it might be difficult to recognize the tactics employed to detect phishing attempts.

Another difficulty is figuring out which attributes best fit a phishing pattern. To identify the best features for phishing URLs, the authors offered an efficient feature selection technique that combines three (3) filter approaches with a wrapper method. Using the five feature sets from the Heatmap, Anova, and Chi-square tests, as well as the combination of these three and the BFS techniques, three (3) machine learning models that are frequently used for classification were created.

The result showed that the model’s accuracy improves when the optimal features from the final BFS method were chosen. The restriction of this approach is the balanced phishing dataset that was considered, which is prone to overfitting.

Salvi *et al.* (2021) presented a method for determining whether a URL is authentic or a phishing URL. The authors mentioned that out of many different types of attacks, phishing attacks have been seen as a serious risk to users' private information because it is hard for an unsuspecting user to tell if a URL is authentic or fraudulent. Because of its high accuracy, resilience, and superior performance, the Random Forest Algorithm was employed in this research, which focused on supervised learning.

A 2000 dataset of phishing URLs and a 2000 dataset of authentic URLs from phishtank.com were subjected to the suggested methodology. The User Interface (UI) of this suggested system was created to identify websites using URLs. The outcome of this approach was that users were diverted to the particular website pasted in the user interface for legitimate URLs, while for fake URLs, the result was given as a phishing website, and the particular website was blocked.

Rabia (2018) presented a paper on the study of malware and its detection techniques. He noted that malware creators are always coming up with new ideas when developing malware to ensure they cannot be detected easily. The different types of malware, broadly categorized into different classes, were also mentioned, as well as the different techniques used by malware writers to conceal malware from malware detectors. Additionally highlighted were the benefits and drawbacks of the three (3) malware detection methods, and he noted that due to the inability of signature-based detection methods to only detect known malware with less resources, while heuristic and specification-based techniques can detect new and unknown malware, but with a high false negative level. "Heuristic-based techniques are paired with machine learning methods to get more accurate and efficient detection of malware due to the rapid development in polymorphic malware," he concluded.

Phishing is a cyberthreat in which hackers impersonate trustworthy, reputable websites in order to exploit consumers and steal private data, including bank statements and passwords. Additionally, HTTPS and SSL certificates are currently used by half of phishing sites to trick users (Eint *et al.*, 2019). Phishing can be done through several channels: voice, short message service, and the internet. Email, instant messaging, websites, smishing (short message phishing), and vishing (voice phishing) are some of their target vectors (Chiew et al, 2018).

Malware can be categorized into the following classes, as they exist in many forms. Many of them can be found in one or more classes and they are: Virus, Worms, Trojan Horse, Spyware, Botnet, Rookit and Ransomware.

3.0 METHODOLOGY

The Intelligent Phishing Malware Detection System's design and implementation process are covered in this chapter. It explains the model training process, evaluation metrics, deployment strategy, preprocessing procedures, feature extraction procedure, system architecture, dataset collecting, and research design. This project aims to develop a web-based AI (Artificial Intelligence) system that is capable of detecting phishing URLs and malware in files.

Python 3.10 was used to develop the system, and the following libraries were used:

Table 3.1: Libraries used and their various purposes

Library	Purpose
pandas	Data loading, preprocessing, and manipulation
scikit-learn	Machine learning algorithms, training, and evaluation
joblib	Model serialization for saving and loading
tldextract	Extracting domain, subdomain, and TLD from URLs
python-magic	File type detection (for extended malware analysis)

Figure 3.1 shows the system architecture of the proposed system.

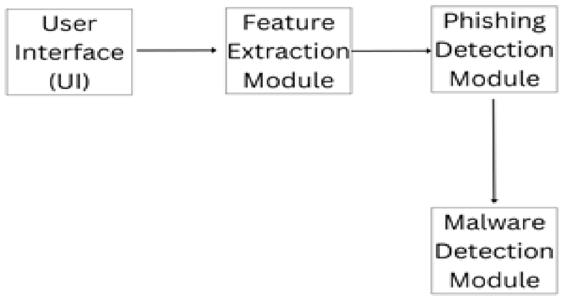


Figure 3.1: Architecture of the Intelligent Phishing System

3.1 Dataset Collection

The dataset consists of both phishing and legitimate URLs obtained from several sources to ensure distinctiveness and equity.

- Phishing URLs: Collected from Kaggle, PhishStorm, and Mendeley Data.
- Legitimate URLs: Collected from Kaggle and Mendeley Data.
- Dataset Size: 11,412 URLs, with a distribution of 5,703 phishing URLs and 5,709 legitimate URLs.

The dataset includes: Urls starting with http://, Urls starting with https://, Urls without a scheme (e.g., www.example.com). Each entry in the dataset has the following attributes:

- URL: The web address.
- Label: 1 for phishing, 0 for legitimate.

3.2 Feature Extraction

After cleaning, each URL was changed into a structured numeric representation appropriate for machine learning. The characteristics extracted included.

Table 3.2: Features used for URL classification (Eint *et al*., 2019)

Feature Name	Description
Len(url)	The URL's total character count.
len(full_domain)	Character count of the domain name (excluding subdomain or top-level domain).
int(parsed.scheme == 'https')	Binary value indicating if the URL uses HTTPS (1) or HTTP (0).
url.count('.')	Count of periods (.) in the URL.
Int ('-' in full_domain)	The URL's hyphen count (-).
num_digits	Count of numerical digits in the URL.

Feature Name	Description
int(re.match(r'\d+\.\d+\.\d+\.\d+', parsed.netloc) is not None)	Binary value showing whether an IP address is being used in place of a domain name in the URL.
len(subdomain)	Binary value indicating if the URL has a subdomain.
len(path)	Number of characters in the Top-Level Domain (TLD).
Keyword count	Number of suspicious keywords in the URL.
int('@' in url)	Phishing (1) site if the @ symbol exists, else legitimate (0).
len(full_domain)	Character count of the domain name (including subdomain or top-level domain).

These characteristics were selected using research and literature. Phishing URLs often display formats such as too much length, many subdomains, strange characters, and use of IP addresses. These 12 features were first identified and used by Eint et al., in her research paper in 2019.

3.3 Model Training

The data that was processed was divided into:

- **70% Training Set:** used for model training.
- **30% Testing Set**: used for model evaluation.

```
From sklearn.model_selection import train_test_split
X = df.drop (columns=['label', 'url', 'domain'])
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.3, random_state=42, stratify=y
)
```

The model extracted the 12 features identified above as what differentiates phishing URLs from legitimate URLs, and these features are processed through the Random Forest Algorithm to identify phishing or legitimate based on training data. The Random Forest Classifier was selected because of its excellent accuracy, resilience, and capacity to handle both categorical and numerical features.

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(
  n_estimators=100,
  random_state=42,
  max_depth=None,
  class_weight='balanced'
)
model.fit(X_train, y_train)
```

3.4 Evaluation Metrics

The model's accuracy, precision, recall, and F1-score were used to evaluate its performance.

```
from sklearn.metrics import accuracy_score, classification_report
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print (f"Accuracy: {accuracy:.4f}")
print (classification_report(y_test, y_pred))
```

These metrics ensured that the system not only detected phishing URLs effectively (high recall) but also minimized false positives (high precision).

3.5 Deployment Strategy

The trained model was saved for integration into the final application:

```
import joblib
joblib.dump(model, 'phishing_model.pkl')
```

During deployment, the saved model can be reloaded instantly:

```
model = joblib.load('phishing_model.pkl').
```

The model was deployed in two ways:

(i) **Flask Web App**: Allows users to manually input URLs for classification.

(ii) **Browser** Extension: Detects phishing attempts in real time by sending visited URLs to the Flask API.

The system's sequential workflow is shown in Fig 3.2.

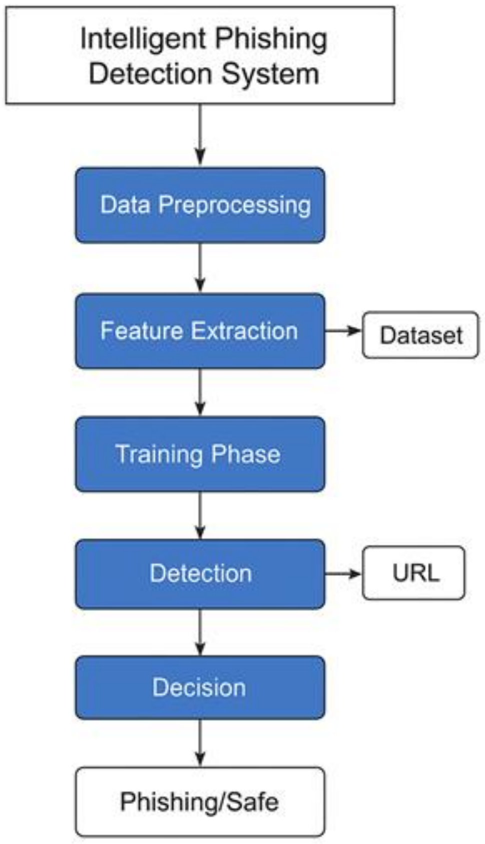


Figure 3.2: Step-by-step workflow of the Phishing & Malware Detection System

4.0 RESULTS AND DISCUSSION

The results of the testing and deployment of the suggested Intelligent Phishing Malware Detection system are shown in this section. The assessment includes machine learning-based phishing URL detection and malware detection using YARA rules. The results are analyzed based on F1-score, Accuracy, Precision, and Recall. Furthermore, the integrated Flask-based web interface's performance is shown.

The tests were conducted using a personal computer that has the following configurations:

- **Processor**: Intel Core i7, 1.8 GHz
- **RAM**: 8 GB
- **Operating System:** 64-bit Windows 10
- **Programming Language**: Python 3.10
- **Frameworks and Libraries:** Flask, Scikit-learn, Pandas, joblib, tldextract, python-magic.
- **Dataset:** 11412 URLs collected from PhishStorm, Kaggle, and Mendeley Data drawn from multiple reputable sources. The dataset consisted of 5703 phishing Urls and 5709 legitimate Urls.
- **Dataset Split:** 70% training and 30% testing.

4.2 Results

Below are the web-pages for both the Phishing URL Detection and the Malware Detection.

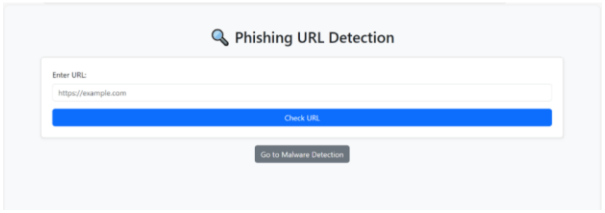


Figure 4.1: Phishing URL Detection Web Page

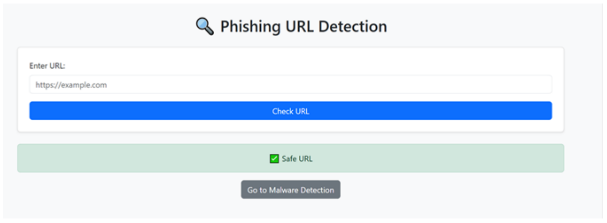


Figure 4.2: Legitimate URL result

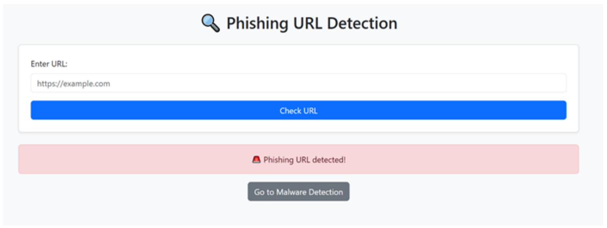


Figure 4.3 Phishing URL result

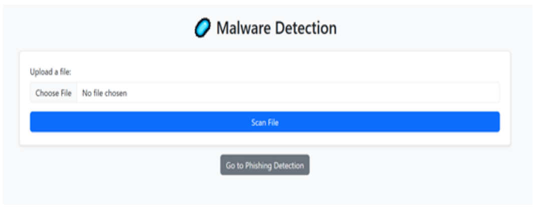
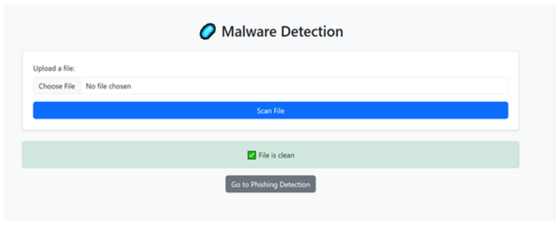
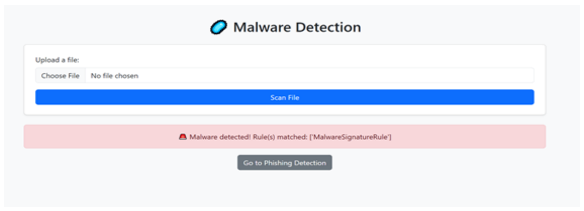


Figure 4.4: Malware Scanning Page



✅ File is clean

Figure 4.5: Scanning a Legitimate .PDF file and the Result



Malware detected! Rule(s) matched: ['MalwareSignatureRule']

Figure 4.6: Scanning a malware .txt file and the result

4.3 Results Discussion

The Random Forest Classifier is used as the training model for the Phishing Detection system, as it has the greatest performance in contrast to other tested models (Naïve Bayes, Logistic Regression, and Decision Tree).

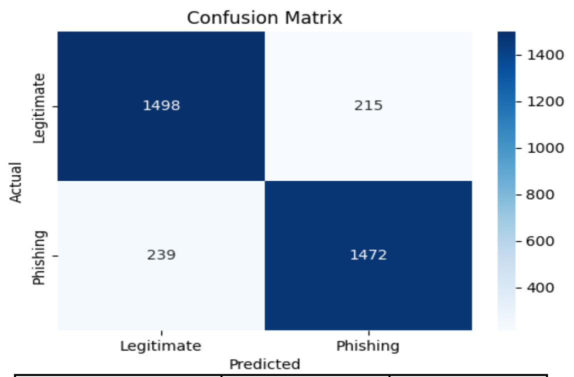
Table 4.1 provides a summary of the outcomes of the metrics used to analyze the model. The size of the dataset used is 11413 after duplicate URLs have been removed and the dataset has been cleansed. 30% of the dataset was used for evaluation, while the remaining 70% was used for training. Table 4.1 shows the evaluation metrics result.

Table 4.1: Evaluation Metrics Result

Metric	Score (%)
Accuracy	86.7
Precision	87.2
Recall	86.0
F1-Score	86.6

Table 4.2 demonstrates the system's classification performance using the dataset.

Table 4.2: Confusion Matrix



	Predicted Legitimate	**Predicted Phishing**
Actual Legitimate	1498 (True Negative)	215 (False Positive)
Actual Phishing	239 (False Negative)	1472 (True Positive)

Figure 4.7**:** Pictorial Representation of the Confusion Matrix

Classification

- **True Negative (TN) = 1498 –** Reputable URLs that are appropriately categorized.
- **False Positive (FP) = 215 –** Legitimate mislabeled as phishing.
- **False Negative (FN) = 239 –** Phishing URLs that were mistakenly identified as legitimate.
- **True Positive (TP) = 1472 –** Phishing URLs were appropriately categorized.

The confusion matrix showed that most of the legitimate URLs were appropriately categorized and a very small number of false negatives and false positives were noted. False positive results happen when valid URLs are as Phishing, which could inconvenient the end-user, while False negatives occur when phishing URLs are misclassified as legitimate, which could lead to loss of the end users' information upon clicking the link.

The developed Phishing & Malware Detection System performance was assessed using both the Receiver Operating Characteristic (ROC) Curve and the Precision-Recall (PR) Curve. The ROC Curve (AUC = 0.942) means there is a high rate of distinguishing phishing URLs as phish and Legitimate URLs as real. The PR Curve (AP =

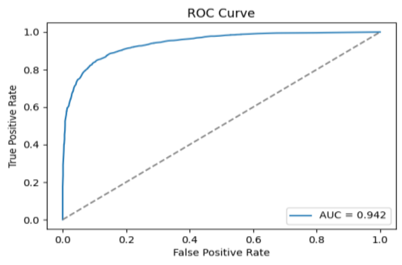


Figure 4.8**:** ROC Curve Graph

0.945) means that the system has a high recall (able to detect phishing URLs as phish) value as well as a high precision (the number of times it will alert users on false alarms is low) rate. Figures 4.8 and 4.9 below are the ROC and PR curve graphs.

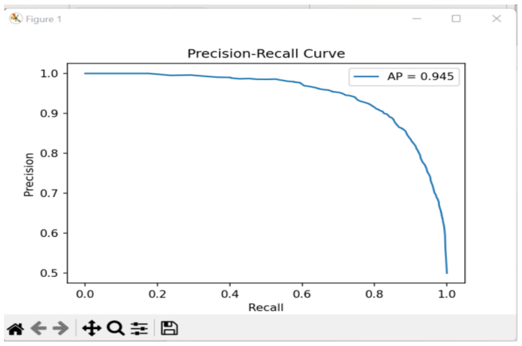


Figure 4.9 PR Curve Graph

For the malware detection component, YARA rules were used to scan uploaded files for already known malicious signatures. This approach successfully identified files containing patterns consistent with predefined malware signatures.

5.0 CONCLUSION

For the Phishing Detection module, the Random Forest Classifier was trained on a dataset of trustworthy and fraudulent websites. 12 important characteristics, such URL length, the use of IP address, subdomain length, keyword count, among others were extracted and used as a basis of distinguishing legitimate URLs from phishing URLs. The model produced a high evaluation metrics result, showing strong F1 score, recall, accuracy, and precision.

The malware analysis module, while limited to predefined malware signatures was able to identify malware infested files that matched known patterns. The system demonstrated that combining artificial intelligence with rule-based methods is effective, as it provides a more robust cybersecurity solution.

The following ideas are recommended as future improvements for the system.

(i) Applying deep learning methods, like Long-Term Short Memory (LSTM), Recurrent Neural Networks (RNN) or Transformers for phishing and malware detection.

(ii) Threat Intelligence APIs (e.g., VirusTotal) integration for live threat verification.

(iii) Extension of the system to mobile platforms and browser plugins for easier accessibility and broader applicability.

(iv) This model does not take into account zero-day phishing attacks or new malware strains that were not captured by the existing rules.

I recommend that the URLs contained in the dataset used for this implementation be constantly updated with URLs that have been verified as legitimate and phishing respectively. This will help the system to be unbiased and providing the correct result when used. I also suggest that this system be integrated with a database so that result of URLs that have already been verified as legitimate or phishing be saved. This ensures that when a URL suspected to be malicious is scanned, the result can be provided immediately. I am also recommending that datasets can be gotten from a variety of sites that have gathered both legitimate and phishing links and already stored and verified as one or the other.

REFERENCES

Paliath, S., Abu Qbeitah, M. & Aldwairi, M. (2020). PhishOut: Effective phishing detection using selected features. *IEEE*. https://ieeexplore. ieee. org/abstract/document/9239589

Abulfaz, H. & Sharmin, J. (2023). Feature selections for phishing URLs detection using combination of multiple feature selection methods. Proceedings of the 2023 ACM

Conference on Computers, Privacy, and Data Protection. https://doi.org/10.1145/ 3587716. 3587790.

Lamina, O. A., Ayuba, W. A., Adebiyi, O. E., Michael, G. E., Samuel, O. D. & Samuel, K. O. (2024). AI-powered phishing detection and prevention. *Path of Science, 10*(12). https://doi.org/10.22178/pos.112-7

Tabassum, R. (2018). A study on malware and malware detection techniques. *Inter-national Journal of Education and Management Engineering, 8*(2), 20–30. https://doi.org/10.5815/ijeme.2018.02.03

Salvi, S. R., Shah, J. S., Shaikh, N. A. G. & Khodke, P. (2021). Phishing website detection based on URL. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *7*(3). https://doi.org/ 10.32628/ CSEIT2173124.

Chiew, K. L., Yong, K. S. C. & Tan, C. L. (2018). A survey of phishing attacks: Their types, vectors and technical approaches. *Expert Systems with Applica-tions, 106*, 1–20. https://doi.org/10.1016/j.eswa. 2018.03.050.

Eint, S. A., Chaw, T. Z. & Hayato, Y. (2019). A survey of URL-based phishing detection. In *Proceedings of the DEIM Forum 2019*. https://db-event.jpn.org/ deim2019/post/papers/201.pdf.

AN INTEGRATED EMAIL PHISHING SIMULATION USING GOPHISH

Alabi O.[1], Adejimi A.[2], Falana O.[3], Ugwunna C.[4], and Ajetunmobi A.[5]
[1, 2, 3, 5]Department of Computer Science, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria.
[4]Department of Computer Science, Wigwe University Isiokpo Rivers State, Nigeria

[1]alabioa@funaab.edu.ng, [2]adejimiao@funaab.edu.ng, [3]falanaoj@funaab.edu.ng,
[4]charles.ugwunna@wigweuniversity.edu.ng, [5]ajetunmobiabduljamiuolamide@gmail.com
[1]+2348022222139, [2]+234 806 243 3677, [3]+234 703 920 5212, [4]+234 813 151 9257, [5]+2348144521362

ABSTRACT
Phishing remains one of the most persistent and dangerous forms of social engineering attacks, exploiting human vulnerabilities rather than system weaknesses. This study addresses the growing need for proactive cybersecurity training by developing and deploying an integrated email phishing simulation platform using Gophish. The goal was to evaluate user susceptibility to phishing and reinforce awareness through realistic simulations. A controlled experiment was conducted targeting 20 users with crafted email templates and cloned landing pages, all designed to mirror real-world phishing techniques. Results showed a high email open rate (70%), significant link click-throughs (45%), and a notable credential submission rate (20%), highlighting the effectiveness of the simulation and the existing gaps in user vigilance. The platform successfully tracked user behavior, providing actionable insights to improve awareness. Ethical considerations were carefully maintained, no sensitive information was stored, and participants were informed of the results after the simulation. This study demonstrates that phishing simulations are a practical and impactful method for assessing and strengthening organizational defenses against social engineering threats. It recommends ongoing training, the implementation of multi-factor authentication, and regular phishing campaigns to build resilience. The study also reinforces the broader understanding that cybersecurity must account for the human factor, not just technical infrastructure.

Keywords: Phishing, Social engineering, cybersecurity, human vulnerabilities, threats.

1.0 INTRODUCTION

Phishing is a form of fraudulent activity in which a cybercriminal attempts to steal sensitive information, such as usernames, passwords, and online banking details, from its victims. (Stojnic *et al*., 2021). Cybercriminals frequently use phishing emails as a technique to steal personal information or distribute malicious links and attachments, which can lure humans to perform a variety of functions (Burns *et al*., 2019).

Since the COVID-19 pandemic, the threat of phishing has become even more serious because the distractions of home offices can make employees more susceptible to phishing e-mails. (Yeoh, *et al*., 2022). Social engineering attacks are rapidly increasing in today's networks and are weakening the cybersecurity chain. They aim at manipulating individuals and enterprises to divulge valuable and sensitive data in the interest of cyber criminals (Kalnins *et al*., 2017). Malicious activities accomplished through human interactions influence a person psychologically to divulge confidential information or to break security procedures (Pokrovskaia, 2017). Cyber criminals choose these attacks when there is no way to hack a system with no technical vulnerabilities (Aroyo, 2018).

To eliminate social engineering breaches is practically impossible (Ghafir et al., 2016). Phishing attacks are generally divided into two categories: spear phishing, where attackers send individually targeted emails, and broad phishing messages, which target a wider population (Goel, *et al*., 2017). Phishing should not be considered to be only a technological issue; it is also a social engineering attack where attackers are targeting and exploiting vulnerabilities in networked systems and are facilitated by users (Chaudhry & Rittenhouse, 2015).

Based on the differences and characteristics of the conceptual evolution in different periods, the evolution process of social engineering is divided into five phases. These five phases are: Phreak phase (*1974* - 1983), Phrack phase (1984 - 1995), Professional hack phase (1996 - 2001), Multidirectional evolution phase (2002 - 2011), and Advanced social engineering attack phase (since 2012). Some social engineering attacks include Phishing, pretexting, baiting, tailgating, ransomware, watering hole attacks, dumpster diving, and deepfakes, among others. In terms of cybersecurity, scamming or phishing are categorized as social engineering (SE) attacks.

Attacks that involve exploiting human vulnerabilities are classified as social engineering attacks (Wang, *et al*. 2021*)*. SE attacks are conducted by exploiting human vulnerabilities, such as deception, persuasion, manipulation, or influence (Albladi *et al*., 2020).

Despite rigid security measures, a high number of internet attacks are conducted by exploiting application design vulnerabilities, scamming, or advanced technical methods (Abroshan et al. 2021), (Siddiqi et al. 2016).

Recently, several large-scale security breaches have occurred, compromising the vulnerable human factor (Saudi, 2021; Godage, 2018). Despite the advancements in technology, humans play an integral part in functional organization.

This human element in the organizational security chain is inevitably targeted and exploited by hackers. In the realm of cybersecurity, such attacks encompass online fraud, often manifesting in forms such as phishing. The foundation of these attacks lies in pinpointing and leveraging human vulnerabilities. This could be due to a lack of knowledge, undue trust in others, or even the victim's circumstances, which increases the success rate of the attack. Quite often, victims remain unaware of the attack (Gupta et al., 2016).

2.0 RELATED WORK

Social engineering, particularly in the form of phishing attacks, continues to exploit human vulnerabilities despite advancements in technical defenses. Recent research increasingly focuses not only on the technical dimensions of these attacks but also on psychological, organizational, and contextual factors that make individuals and systems susceptible.

Hijji et al. (2021) conducted a systematic literature review to investigate the evolution of social engineering techniques during the COVID-19 pandemic. Drawing on 52 primary studies spanning academic publications and grey literature, the authors observed a significant increase in phishing, smishing, and vishing attacks that exploited pandemic-related fears and misinformation. While their work offers valuable context-specific insights, it is constrained by its focus on the pandemic, limiting its applicability to broader trends in social engineering beyond that period.

Al-Otaibi *et al*. (2020) provided a comprehensive examination of social engineering and phishing attacks, proposing protocol-based mitigation strategies. The authors synthesized findings from academic and industry reports to highlight the value of coordinated defenses across networks. However, the study's scope was bounded by the literature available at the time and did not address emerging tactics such as deepfake phishing or AI-driven attacks, which are increasingly relevant in contemporary threat landscapes.

Lopes *et al*. (2024) explored common techniques and factors underlying the success of social engineering attacks while also identifying barriers to effective prevention. By aggregating data from multiple studies, the authors summarized recurring patterns and proposed directions for future research. Nevertheless, their work acknowledges limitations, including inconsistencies arising from the diversity of reviewed studies and the rapid evolution of attack methods that may outpace current academic discourse.

Yu *et al*. (2024) shifted focus toward the psychological dimensions of social engineering, surveying literature on how human behavior contributes to both vulnerability and defense. Their study emphasized integrating behavioral insights into technical security measures and proposed a roadmap for such integration. Yet, the review primarily relied on secondary sources, with limited empirical validation of the proposed defense models.

Sèdes (2024) offered a broad narrative review combining academic research, case studies, and industry practices to map the current state of information systems security in the context of social engineering. The study underscored the importance of multidisciplinary approaches involving technical, organizational, and human factors. However, it remained largely descriptive and did not engage in detailed technical analysis or experimental validation of proposed counter measures.

Salahdine and Kaabouch (2019) conducted an extensive survey categorizing social engineering attacks and reviewing existing detection and prevention strategies. Their classification has become a foundational reference for researchers in this field. Nonetheless, given its publication date, the study may not fully capture recent trends, including AI-powered attacks and other innovative techniques that have emerged over the past few years.

Broberg and Sinnott (2023) approached the subject by examining the human element in cybersecurity through a systematic literature review using the PRISMA method. The authors provided actionable recommendations for organizations, such as employee awareness programs and robust policies, highlighting the critical role of human-centered defenses. Despite these practical contributions, the review did not include experimental data or real-world organizational case studies to support its findings.

3.0 METHODOLOGY

The study adopts a practical, experimental design focused on simulating phishing attacks in a controlled environment using Gophish. It includes planning, execution, data collection, and analysis. The campaign targets 20 users, selected through purposive sampling. These participants represent a small user group (staff and students) and were chosen to provide a focused analysis. Gophish was installed on a Windows 10 PC by downloading the executable files. The Gophish admin interface was accessed via localhost: 3333.

An SMTP service was configured to send emails from the local Gophish instance. No custom domain was used; instead, links generated by Gophish used the local IP or localhost, assuming users accessed emails in the same local environment or testing setup. Email templates and landing pages were created using basic HTML and loaded into Gophish. One phishing campaign was created and sent to 20 users. The campaign email contained a link to a simulated login page designed to mimic a familiar platform.

Gophish tracked user interactions, including email opens, link clicks, and data submissions. Gophish automatically recorded campaign results, which were exported as CSV files. Metrics analyzed included open rates, click-through rates, and fake credential submissions. Data was visualized using charts in Excel for clearer interpretation. The Gophish application was installed on the localhost Windows terminal, a password was given, and the default username is “admin.

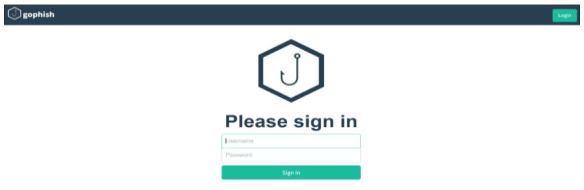


Figure 1 Gophish Login Interface.

The Gophish application was opened using the Ip address and password provided in the terminal. Input username as admin by default and the password would be provided

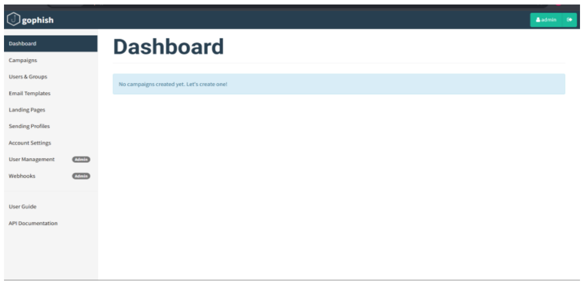
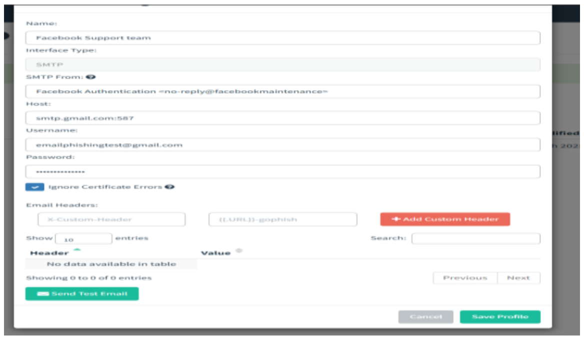


Figure 2. The Gophish Dashboard

The Gophish dashboard containing Campaigns, User & Groups, landing pages, Sending Profiles and so on.

The figure 3 below displays the set up for sending profile of the phishing link you intend to use. Either a free smtp server just like Gmail smtp can be used in the image above or you host it on VPS server which allows the use of any email of your choice. In this study Gmail smtp was used, which involved the use of authentic Gmail address and password.



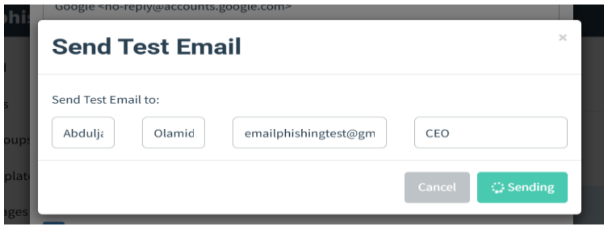
Name:
Facebook Support team
Interface Type:
SMTP
SMTP From:
Facebook Authentication <no-reply@facebookmaintenance>
Host:
smtp.gmail.com:587
Username:
emailphishingtest@gmail.com
Password:
•••••••••••••••
✓ Ignore Certificate Errors
Email Headers:
X-Custom-Header {{.URL}}-gophish

Header	Value
No data available in table	

Showing 0 to 0 of 0 entries

Figure 3: Setting up your Sending profile

The profile can be tested if it is working by sending a Test Email to any mailing address, as shown in Figure 4.

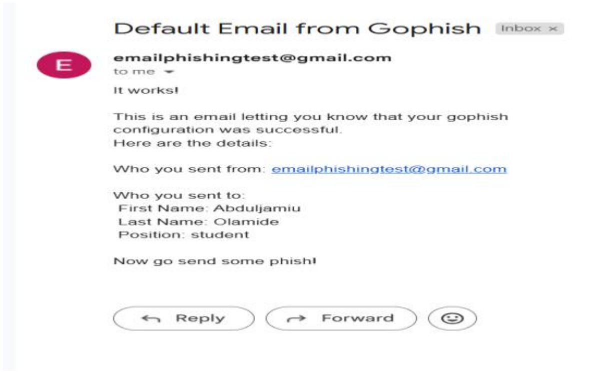


Send Test Email

Send Test Email to:
Abdulja | Olamid | emailphishingtest@gm | CEO

Figure 4 Testing Sending Profile.

The Test email was sent from the profile to the email as displayed in Figure 5.



Default Email from Gophish Inbox

emailphishingtest@gmail.com
to me

It works!

This is an email letting you know that your gophish configuration was successful.
Here are the details:

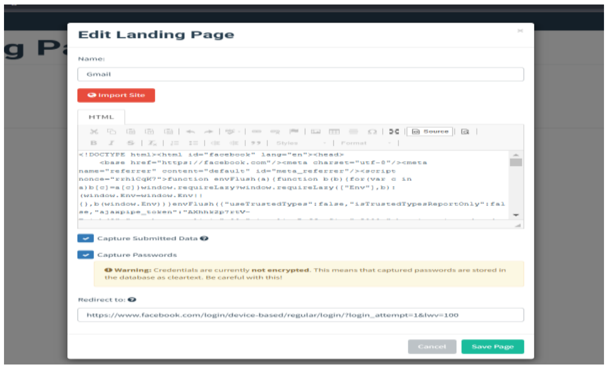
Who you sent from: emailphishingtest@gmail.com

Who you sent to:
First Name: Abduljamiu
Last Name: Olamide
Position: student

Now go send some phish!

Figure 5. Test Email sent to inbox

In order to set up a landing page, a login page for Facebook and Gmail was cloned. The import site button was used, or the source code of the website to be cloned can be copied. The form submission was configured to capture and log credentials securely within Gophish as shown in figure 6. The URL of the phishing page was made to look convincing, using a domain like the real one. Then the landing page can be saved.



Edit Landing Page

Name:
Gmail

```
<!DOCTYPE html><html id="facebook" lang="en"><head>
    <base href="https://facebook.com"/><meta charset="utf-8"/><meta
name="referrer" content="default" id="meta_referrer"/><script
nonce="rrhiCqK7">function envFlush(a){function b(b){for(var c in
a)b[c]=a[c]}window.requireLazy?window.requireLazy(["Env"],b):
(window.Env=window.Env||
{},b(window.Env))}envFlush({"useTrustedTypes":false,"isTrustedTypesReportOnly":fal
se,"ajaxpipe_token":"AXhhk2p7rtV-
```

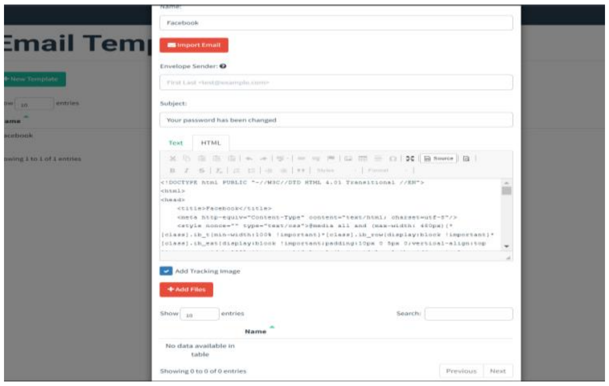
✓ Capture Submitted Data
✓ Capture Passwords

Warning: Credentials are currently **not encrypted**. This means that captured passwords are stored in the database as cleartext. Be careful with this!

Redirect to:
https://www.facebook.com/login/device-based/regular/login/?login_attempt=1&lwv=100

Figure 6 Setting the landing page

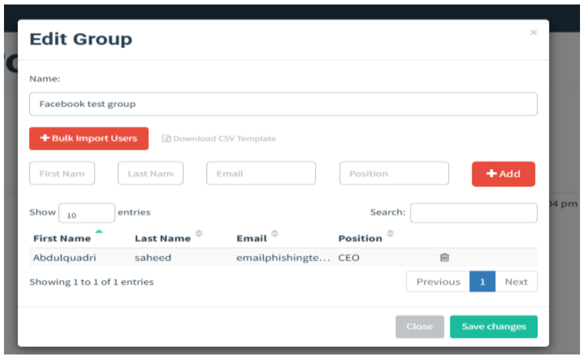
Setting up an Email template is required. This is the Email content that would be sent to the users or group of users whose credentials is to be gotten. Emails were carefully crafted to appear legitimate, using professional formatting, official logos, and an urgent yet convincing tone to prompt user action.


Email Templates
Name: Facebook
Envelope Sender:
Subject: Your password has been changed

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional //EN">
<html>
<head>
    <title>Facebook</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <style nonce="" type="text/css">@media all and (max-width: 480px){*
[class].ib_t{min-width:100% !important}*[class].ib_row{display:block !important}*
[class].ib_ext{display:block !important;padding:10px 0 5px 0;vertical-align:top
```

Add Tracking Image: ✓
Show 10 entries
No data available in table
Showing 0 to 0 of 0 entries

Figure 7. Setting the Convincing Email


Edit Group
Name: Facebook test group

First Name	Last Name	Email	Position
Abdulquadri	saheed	emailphishingte…	CEO

Showing 1 to 1 of 1 entries

Figure 8 Setting Target audience

Figure 8 creates the group of users to receive the phishing attack. The emails to send the phishing message are added and make a group.

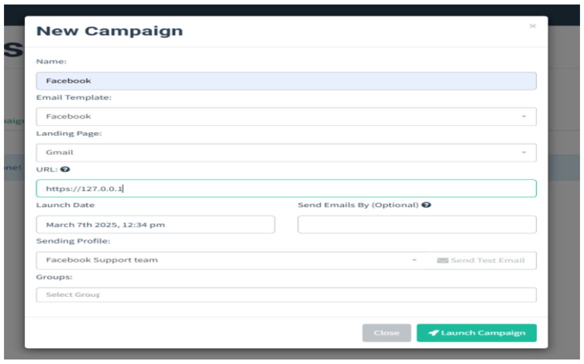
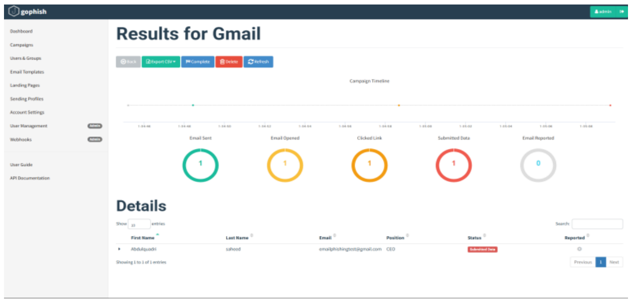

New Campaign
Name: Facebook
Email Template: Facebook
Landing Page: Gmail
URL: https://127.0.0.1
Launch Date: March 7th 2025, 12:34 pm
Send Emails By (Optional):
Sending Profile: Facebook Support team
Groups: Select Group

Figure 9. Launching Campaign

The phishing campaign was configured in Gophish, assigning the fake email template and phishing link. Gophish's tracking metrics were enabled to collect real-time interaction data. A redirect option was added so that after users entered their credentials, they were sent to the actual login page of the legitimate service (to reduce suspicion). No real passwords were stored – only metadata about who interacted with the phishing attempt as displayed in figure 10.


Results for Gmail

Campaign Timeline

Email Sent	Email Opened	Clicked Link	Submitted Data	Email Reported
1	1	1	1	0

Details

First Name	Last Name	Email	Position	Status	Reported
Abdulquadri	saheed	emailphishingtest@gmail.com	CEO	Submitted Data	

Showing 1 to 1 of 1 entries

Figure 10. Monitoring User Interaction

The final phrase of this phishing simulation will be launching a campaign in crafting the phishing link and sending it to the targeted audience. The campaign dashboard in Gophish displayed:

(i) The total number of emails sent.
(ii) The percentage of recipients who opened the email.
(iii) The percentage of recipients who clicked the phishing link.
(iv) The number of users who attempted to log in on the fake page and submitted data.
(v) The number of users who reported the emails.

Since phishing simulations can cause ethical concerns, strict guidelines were followed:

(i) Participants Were Not Harmed: The attack was conducted in a controlled environment, and no real accounts were compromised.
(ii) Informed Consent (After the Test): Users were informed after the simulation and given feedback.
(iii) No Real Passwords Were Stored**:** Only interaction logs were collected for analysis.
(iv) Users Received Security Awareness Training to prevent real attacks in the future.

4.0 IMPLEMENTATION AND RESULTS

A phishing email framework was built using the Gophish platform. This provided tools for email distribution, link tracking, and user behavior analysis. A phishing email templates that reflected common social engineering tactics was the campaign was carried out with the following outcomes:

(i) Total Emails Sent: 20
(ii) Total Emails Delivered: 18
(iii) Emails Opened: 14
(iv) Links Clicked: 9
(v) Credentials Submitted: 4
(vi) Emails Reported: 3

Email Open Rate (70.0%): A strong open rate suggests the phishing emails appeared trustworthy or relevant enough to attract user attention. This indicates the design and content of the email templates were convincing. Click Rate (45.0%): Almost half of the users who received the emails clicked on the phishing links, revealing a substantial level of susceptibility.

Credential Submission (20.0%): Four users entered their credentials, suggesting they were completely unaware of the phishing attempt. This highlights the risk posed by well-crafted phishing attacks.

Report Rate (15.0%): Only three users reported the email, suggesting a need for stronger awareness and easier reporting mechanisms.

Figure 11 shows the progression of user engagement in the phishing simulation, displaying the actual counts for each activity. 18 emails were successfully delivered, showing how user engagement drops at each stage of the phishing funnel. 14 users opened the emails they received, representing a decent opening rate. From there, 9 users took the bait and clicked on the malicious links embedded in the emails, these users moved beyond just reading the email to actively engage with potentially dangerous content.

Within the users that took the bait, 4 users submitted their credentials on the phishing landing pages. This means they not only clicked the links but entered their login information, thinking they were on a legitimate site. On the positive side, only 3 users reported the emails as suspicious, representing good security awareness among those individuals, even though it's a small number overall.

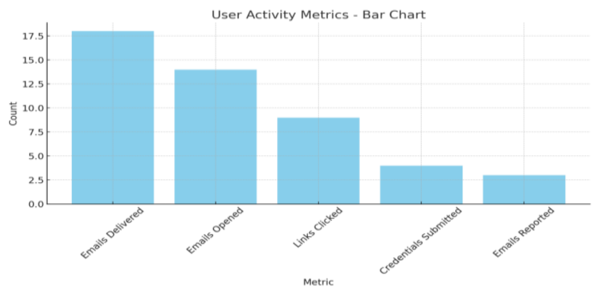


Figure 11 User Activity Metrics

The results suggest that while some users were cautious enough not to interact with the emails, a notable number still fell for the phishing attempt. High open and click rates show how convincing the emails were. The fact that users submitted login information underlines the effectiveness of current phishing techniques and the gaps in user awareness. The low report rate indicates a missed opportunity to flag the phishing attempt, further reinforcing the need for training.

Based on the findings from the phishing simulation, here are five key mitigation strategies to reduce user susceptibility to phishing attacks:

(i) Regular Phishing Awareness Training: High open and click rates suggest users found the phishing emails believable. Regular Conduct of ongoing, realistic training sessions to help users recognize phishing red flags like suspicious links, urgent language, or unusual sender addresses.
(ii) Simulated Phishing Campaigns: Only 15% of users reported the phishing email, indicating low reporting behavior. Run periodic phishing simulations to keep awareness sharp and track improvement over time. Reward users who report phishing attempts to encourage proactive behaviour.

(iii) Enable Easy Reporting Mechanisms: A lack of reports may be due to inconvenient or unclear reporting processes. Integrate a “Report Phishing” button into email clients. Ensure users know exactly how and when to report suspicious emails.

(iv) Implement Multi-Factor Authentication (MFA): 20% of users submitted credentials, posing serious risks if credentials are compromised. Enforce MFA across all user accounts so that even if credentials are stolen, attackers cannot gain access without a second factor.

(v) Improve Email Filtering and Warning Banners: Reducing exposure to phishing emails in the first place is critical. Strengthen spam/phishing filters and implement visual banners for external or suspicious emails to prompt extra caution from users.

5.0 CONCLUSION

This study developed and tested a phishing simulation platform using GoPhish to assess organizational vulnerability to social engineering attacks. The system included realistic email templates, landing pages, and user tracking features, successfully demonstrating core functionality through localized testing. Overall, the study reinforces that effective cybersecurity must integrate human, technical, and organizational strategies to build resilience against evolving social engineering threats.

REFERENCE

Abroshan, H., Devos, J., Poels, G. & Laermans, E. (2021). Phishing happens beyond technology: The effects of human behaviors and demographics on each step of a phishing process. IEEE Access, 9, 44928-44949.

Al-Otaibi, A. F. & Alsuwat, E. S. (2020). A study on social engineering attacks: Phishing attack. *International Journal of Recent Advances in Multidisciplinary Research*, 7(11), 6374-6380.

Aroyo, A.M.; Rea, F.; Sandini, G.; Sciutti, A. Trust and social engineering in human robot interaction: Will a robot make you disclose sensitive information, conform to its recommendations or gamble? IEEE Robot. Autom. Lett. 2018, 3, 3701–3708.

Broberg, R. & Sinnott, P. (2023). The Human Element of Cybersecurity: A Literature Review of Social Engineering Attacks and Countermeasures.

Ghafir, I., Prenosil, V., Alhejailan, A. & Hammoudeh, M. (2016, August). Social engineering attack strategies and defence approaches. In 2016 IEEE 4th international conference on future internet of things and cloud (FiCloud) (pp. 145-149). IEEE.

Gupta, S., Singhal, A. & Kapoor, A. (2016). A literature survey on social engineering attacks: Phishing attack. 2016 International Conference on Computing, Communication and Automation (ICCCA), 537–540. https://doi.org/10.1109/CCAA.2016.7813778

Hijji, M. & Alam, G. (2021). A multivocal literature review on growing social engineering based cyber-attacks/threats during the COVID-19 pandemic: challenges and prospective solutions. Ieee Access, 9, 7152-7169.

Kalniņš, R., Puriņš, J. & Alksnis, G. (2017). Security evaluation of wireless network access points. *Applied Computer Systems*, *21*(1), 38-45.

Lopes, A., Mamede, H. S., Reis, L. & Santos, A. (2024). Common Techniques, Success Attack Factors and Obstacles to Social Engineering: A Systematic Literature Review. Emerging Science Journal, 8(2), 761-794.

Godage, R. D. Marriott International Data Breach 2018.

Pokrovskaia, N. Social engineering and digital technologies for the security of the social capital’ development. In Proceedings of the International Conference of Quality

Management, Transport and Information Security, Petersburg, Russia, 24–30 September 2017; pp. 16–19.

Salahdine, F. & Kaabouch, N. (2019). Social engineering attacks: A survey. Future internet, 11(4), 89.

Saudi Aramco Confirms Data Leak after Reported Cyber Ransom. Available online: https://www.bloomberg.com/news/articles/2021-07-21/saudi-aramco-confirms-data-leak-after-reported-cyber-extortion (accessed on 6 August 2021).

Sèdes, F. (2024). Security in IS and social engineering--an overview and state of the art. arXiv preprint arXiv:2406.12938.

Siddiqi, M.A.; Mugheri, A.; Oad, K. Advanced persistent threats defense techniques: A review. Pak. J. Comput. Inf. Syst. 2016, 1, 53–65.

Van Mourik, D. J. (2017). Targeted attacks and the human vulnerability (Doctoral dissertation, Ph. D. dissertation, Cyber Secur. Acad., The Hague, The Netherlands).

Wang, Z., Zhu, H. & Sun, L. (2021). Social engineering in cybersecurity: Effect mechanisms, human vulnerabilities and attack methods. Ieee Access, 9, 11895-11910.

Yu, J., Yu, Y., Wang, X., Lin, Y., Yang, M., Qiao, Y. & Wang, F. Y. (2024). The Shadow of Fraud: The Emerging Danger of AI-powered Social Engineering and its Possible Cure. arXiv preprint arXiv:2407. 15912.

Stojnic, T., Vatsalan, D. & Arachchilage, N. A. (2021). Phishing email strategies: understanding cybercriminals' strategies of crafting phishing emails. *Security and privacy*, *4*(5), e165.

Burns, A. J., Johnson, M. E. & Caputo, D. D. (2019). Spear phishing in a barrel: Insights from a targeted phishing campaign. *Journal of Organizational Computing and Electronic Commerce*, *29*(1), 24-39.

Chaudhry, J. A. & Rittenhouse, G. R. (2015). Phishing: Classification and Countermeasures. In 7th International Conference on Multimedia, Computer Graphics and Broadcast-ing (MulGraB), Jeju, 2015, pp. 28-31

Goel, S., Williams, K. & Dincelli, E. (2017). Got Phished? Internet Security and Human Vulnerability," Journal of the Association for Information Systems: Vol. 18: ISS. 1, Article 2.

PROACTIVE INSIDER THREAT DETECTION FRAMEWORK: AN EXPLAINABLE AI AND BEHAVIOURAL ANALYTICS-DRIVEN APPROACH

Adeduro O.[1] and Josh-Falade B.[2]
[1]Department of Cybersecurity, College of Computing, McPherson University, Ajebo, Ogun State, Nigeria
[2]Department of Information Technology, College of Computing, McPherson University, Ajebo, Ogun State, Nigeria
[1]adedurooo@mcu.edu.ng, [2]josh-faladeoa@mcu.edu.ng, +2348028402186

ABSTRACT
In every organization, insider threats are a critical security challenge that must be promptly and adequately addressed. AI-driven behavioral analytics systems have been designed to detect them; however, these systems raise significant privacy concerns leading to loss of confidence by security analysts and professionals. This paper focuses on addressing this challenge of privacy as well as transparency by proposing a framework that combines Explainable AI (XAI) with privacy-preserving techniques (Federated Learning and Differential Privacy) for detection of insider threats. We propose a threat detection system that is accurate, interpretable, and respects user privacy by design. We employ a Federated Learning architecture with a Long Short-Term Memory (LSTM) model for Behavioral analysis and Differential Privacy is applied to user feature vectors before model training to guarantee privacy. SHAP and LIME are thereafter used to interpret the aggregated global model's predictions. To test its robustness and generalizability, the framework is validated across two distinct environments: the synthetic, corporate-focused CERT Insider Threat Dataset and the real-world, cloud-native BETH dataset. The results demonstrate the framework's adaptability, achieving a high F1-Score of 0.88 on the CERT dataset and a strong 0.86 on the more complex BETH dataset. This represents a minimal, acceptable performance trade-off for privacy. The explainable AI layer remained effective and accurate across both datasets, increasing analysts' confidence and reducing the mean-time-to-investigate alerts. This research demonstrates that high-accuracy threat detection, robust privacy preservation, and deep explainability can be achieved in unison across diverse and modern IT infrastructures.

Keywords: Insider threat, threat detection, privacy preserving, explainable AI, behavioral analytics

1.0 INTRODUCTION

The global increase in remote work and cloud adoption by contemporary organizations has led to an increasing range of threats from the digital space, to the end that significant resources are being dedicated to fortifying infrastructure against external adversaries (Bin Sarhan & Altwaijry, 2023). However, an equally, if not more, deceptive challenge persists within: Insider threats.

By definition, an insider is typically a current or former employee, contractor, or business partner with valid, authorized access to an organization's network or data, and can inflict catastrophic damage to an organization (Waters, 2016). Whether their actions are driven by malicious intent, such as espionage or sabotage, or are the result of negligence, the consequences can range from devastating data breaches and financial loss to irreparable reputational harm to the organization (Mazzarolo & Jurcut, 2019). The trust and privileged access granted to insiders render traditional security measures (such as firewalls and intrusion detection systems) which are primarily designed to guard the boundary between the external and internal of an organization, fundamentally inadequate for the insider threat challenge (Fei & Zhou, 2024).

To address this critical security gap, advanced analytical approaches, primarily the

User and Entity Behavior Analytics (UEBA) has been introduced. By leveraging Artificial Intelligence (AI) and Machine Learning (ML), the UEBA systems continuously monitor user and entity activities to establish a baseline of normal behavior (Trivedi, 2024). By analyzing vast streams of data—including login patterns, file access, email communications, and application usage—they can detect subtle deviations and anomalous patterns that may signal a developing threat. The Long Short-Term Memory (LSTM) deep learning models have shown to be advantageous in this domain due to their ability to learn complex, sequential patterns from time-series data, offering a powerful tool for proactive threat detection (Nasir et al., 2021).

However, the efficiency of these advanced AI systems introduces a dilemma that negatively impacts organizational trust and operational reality: the black box and privacy paradox. First, the algorithmic complexity of deep learning models that makes them so powerful also renders them cloudy such that they can generate a high-risk alert for a user but typically fail to provide a clear, humanly-understandable justification for their conclusion.

For a Security Operations Center (SOC) analyst, an unexplainable alert is not actionable; it is merely noise. This lack of transparency leads to a critical trust issue, resulting in incessant alerts, slower incident response times, and the risk that genuine threats are overlooked within a myriad of false positives.

Secondly, and equally important, the UEBA systems have a huge need for sensitive data (Idris & Damilola, 2023). To build accurate behavioral profiles, they must assess and process a detailed digital representation of an employee's daily activities. This practice of centralizing and scrutinizing employee data creates a profound privacy risk. It not only exposes the organization to severe legal and financial penalties under data protection regulations like GDPR and NDPR but also nurtures a culture of mistrust, potentially harming employees' morale and productivity (Inayat *et al*., 2024).

Organizations are therefore caught in a difficult position: they must monitor internal users' behavior to ensure security, but in doing so, they risk violating the privacy of the same individuals they aim to protect. This paper confronts this dual challenge by proposing and evaluating a novel framework that integrates three technological pillars to create a system that is simultaneously effective, private, and transparent. These are not competing goals but essential, inter-related components of a truly trustworthy AI for securing organisations. Our framework is built upon:

(i) **Federated Learning (FL)**: We employ a decentralized training architecture where the AI model learns from local data without the raw, sensitive details leaving their source.

(ii) **Differential Privacy (DP)**: We inject some regulated statistical noise into the data during the training process, making it computationally impracticable to identify any single user from the model's outputs.

(iii) **Explainable AI (XAI)**: We apply two interpretation techniques to the trained model, enabling it to provide clear, evidence-based justifications for every alert it generates.

Furthermore, we recognize that the true test of any security framework is its robustness and adaptability across different operational environments. The validation of our framework is done on two distinct and complementary datasets – the CERT and BETH datasets. The CERT Insider Threat Dataset, is a well-known synthetic benchmark that simulates a traditional corporate IT environment while the BETH Dataset is a complex, modern dataset containing real attack traffic and kernel-level logs from a cloud-native infrastructure. By comparing the framework's performance across these two scenarios, we demonstrate its capacity to deliver privacy-preserving, explainable security analytics in both conventional and next-generation computing paradigms.

The remaining sections of this paper are structured as follows: Section 2.0 provides an overview of related works in the fields of insider threat detection, privacy-preserving machine learning, and XAI. Section 3.0 details the study materials and methodology. Section 4.0 presents the comparative results from our experiments on the CERT and BETH datasets and discusses their implications. Section 5.0 concludes the study with some future directions.

2.0 RELATED WORKS

Development of a privacy-preserving and explainable insider threat detection system lies at the cross-section of three rapidly evolving research domains: Advanced behavioral analytics for threat detection, Privacy-Preserving Machine Learning (PPML), and Explainable Artificial Intelligence (XAI). This section provides an overview of recent works in each area, critically analyzing their contributions and limitations.

2.1 Deep Learning Models for Insider Threat Detection

Initial approaches to insider threat detection were largely dependent on signature-based and rule-based systems, often implemented within Security Information and Event Management (SIEM) platforms. These systems match user activities against a preconfigured set of policies or known malicious behavioural patterns (Gheyas & Abdallah, 2016).

While simple and interpretable, their limitation is their static nature and ineffectiveness against novel, zero-day threats and complex attack vectors that do not conform to established rules, often leading to high occurrences of false positives and false negatives.

Recognizing these limitations, researchers turned to machine learning to build more adaptive, data-driven systems. Early ML-based approaches employed classic techniques like Support Vector Machines (SVMs), Bayesian networks and Random Forests, to classify user behavior based on statistical features extracted from logs (Ribeiro et al., 2016). Hidden Markov Models (HMMs) were also widely used to model the sequence of user actions, providing a probabilistic measure of whether a sequence deviates from the norm (Bandgar et al., 2013).

While these models represented a significant improvement, they often struggled to capture historical dependencies and subtle, contextual nuances inherent in human behavior. Recurrent Neural Networks (RNNs) and their advanced variant, Long Short-Term Memory (LSTM) networks, marked a paradigm shift as seen in literature.

LSTMs are explicitly designed to learn from sequential data, making them exceptionally well-suited for modeling user behavior over a certain period of time. Several studies have demonstrated the superiority of LSTMs in detecting insider threats by analyzing sequences of system calls, login events, and file access patterns (Yuan & Wu, 2021). These models can learn a user's manner and sequence of work and flag any interaction or steps that would deviate from that sequence.

Researchers have also demonstrated that deep learning models could effectively analyze behavioral-based features to detect threats (Nasir et al., 2021) Similarly, modern deep learning and NLP methodologies on the CERT dataset, have been explored, recommending a proactive, data-driven approach to insider threat management (Trivedi, 2024). Bi-LSTM models for feature extraction have also been applied, achieving improved results over traditional methods (Abd Al-Ameer & Huby, 2025).

The Deep Synthesis-based Insider Intrusion Detection (DS-IID) model specifically tackled threats posed by generative AI, and achieved a 97% accuracy when applied to the CERT dataset, thus highlighting the need for models to evolve with the threat landscape (Kotb *et al.*, 2025).

More recently, Graph Neural Networks (GNNs) have also emerged as a powerful alternative, capable of modeling the complex relationships between users, devices, and data.

GNNs are able to capture contextual insights and reduce false positives by understanding the structural connections between events and integrating more relational information to improve performance (Gong et al., 2024).

2.2 Privacy-Preserving Machine Learning (PPML) in Security

The intensive data requirement of behavioral analytics systems has raised significant privacy concerns, leading to the development of PPML techniques. Two of the most prominent techniques are Federated Learning and Differential Privacy.

Federated Learning (FL) is a decentralized machine learning technique that facilitates model training based off a set of distributed data (McMahan *et al*, 2017). This is crucial for contexts and scenarios where data is sensitive.

FL has been widely applied in cybersecurity, however with varying privacy and security challenges (Mothukuri et al., 2021). Privacy preservation in FL from a GDPR perspective provides key insights for maintaining regulatory compliance in collaborative security models (Truong *et al*., 2021). Despite the fact that different FL architectures have been developed for lightweight and secure cloud-edge collaboration, there has been an introduction of new threats such as data poisoning and data reconstruction, necessitating robust defense mechanisms (Han *et al*., 2024).

On the other hand, Differential Privacy (DP) provides a mathematical guarantee of privacy by introducing statistical noise to data or algorithmic results (Dwork, 2008), a technique that is essential for protecting individual identities within a dataset. As seen in existing research works, DP is usually applied in conjunction with other privacy preserving techniques to deliver on the privacy guarantee (Namatevs *et al*., 2025).

2.3 Explainable AI (XAI) in Cybersecurity

Explainable AI is an emerging research field with the aim to improve the transparency of decisions made by AI models, thereby increasing the trustworthiness of AI models (Javed *et al*., 2025). LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017) have been widely used by researchers as model-agnostic techniques to achieve this goal. Across various cybersecurity domains, both XAI techniques have been applied to improve accountability and trust. Specifically for insider threats, researchers have explored actionable response strategies for insider threat detection (Rjoub et al., 2023) making AI a more reliable partner for security analysts by clarifying how models identify threats (Akpan Itoro Udofot *et al*., 2024).

Further advancements have produced the integration of XAI with privacy-preserving methods such as the combination of FL and XAI for intrusion detection (Fatema *et al*., 2025) demonstrating that security, privacy, and interpretability can be achieved simultaneously.

A comprehensive review of recent literature reveals that while good progress has been made by scholars in recent times, research studies at the intersection of all three domains remains sparse. These studies are seen to:

- focus on detection accuracy but overlook the critical privacy implications (Nasir et al., 2021)
- propose privacy-preserving architectures but fail to provide transparent and actionable intelligence (Mothukuri *et al*., 2021)
- apply XAI to security models but do so on centralized, non-private datasets (Rjoub *et al*., 2023)
- validate their findings on a single type of dataset, leaving the framework's generalizability an open question.

Therefore, a gap exists for a holistic framework that simultaneously delivers high-accuracy detection, robust privacy by design, and meaningful explainability. This paper aims to fill this gap by proposing and evaluating an integrated system, validating its performance across both traditional and modern cloud infrastructures to demonstrate its real-world viability.

3.0 MATERIALS AND METHODS

3.1 The Proposed Framework

The proposed framework is made up of the following components:

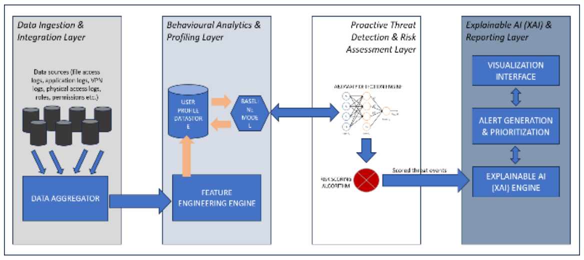

Data Ingestion & Integration Layer
Data sources (file access logs, application logs, VPN logs, physical access logs, roles, permissions etc.)
DATA AGGREGATOR
Behavioural Analytics & Profiling Layer
USER PROFILE DATASTORE
BASELINE MODEL
FEATURE ENGINEERING ENGINE
Proactive Threat Detection & Risk Assessment Layer
ANOMALY DETECTION ENGINE
RISK SCORING ALGORITHM
Scored threat events
Explainable AI (XAI) & Reporting Layer
VISUALIZATION INTERFACE
ALERT GENERATION & PRIORITIZATION
EXPLAINABLE AI (XAI) ENGINE

Figure 1: The Proposed Architecture

- Data Ingestion & Integration Layer: This layer collects data from the different sources (app logs, physical access logs, VPN, database queries etc.) and integrates them into a unified stream that is a relevant representation of user and entities' normal behaviour.
- Behavioural Analytics & Profiling Layer: This is the layer where the normal behaviour of entities is learnt. It includes a continuous learning process based on the unified data stream from the Data Ingestion layer.
- Proactive Threat & Assessment Layer: This layer constitutes the detection engine that identifies deviations from the normal (learnt) behaviour in the preceding stage, and quantifies/scores the associated risk.
- Explainable AI (XAI) & Reporting Layer: This layer interpretes the output and provides a transparent, comprehensive and actionable response to users through an interface or dashboard. This is achieved with Explainable AI (XAI) techniques.

3.2 Methodology

The approach is to construct, train, and evaluate a robust insider threat detection model that is private-by-design and inherently explainable. The entire process, illustrated in Figure 2, follows a multi-stage pipeline encompassing data preparation across two distinct environments, a privacy-preserving federated training architecture, deep learning-based behavioral modeling, and a dual-pronged evaluation of both predictive performance and the quality of explanations.

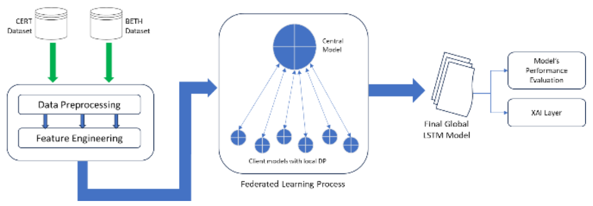


Figure 2: Overall methodology

(a) Data Acquisition and Environment Setup

To evaluate the generalizability of the framework, two datasets with fundamentally different characteristics were selected:

- CERT Insider Threat Dataset: This is a canonical, synthetically generated dataset that emulates a traditional corporate IT environment. It consists of seven interconnected data files that log user activities over 517 days for 1,000 users. The key data sources include: logon.csv (user logins/logoffs), file.csv (file access), email.csv (email meta-data), http.csv (web traffic), and device.csv (removable media usage). A separate answers .csv file provides the truth, labeling specific users and days associated with one of three malicious insider threat scenarios. This dataset serves as our controlled environment baseline.
- BETH (BPF-Extended Tracking Honeypot) Dataset: This is a modern, real-world dataset collected from a series of cloud-based honeypots. It reflects the complexity and noise of a contemporary cloud infrastructure. The dataset contains over 8 million events, capturing low-level system interactions including kernel-level process logs, system calls, and network traffic generated by actual, in-the-wild attacks. Unlike the user-centric CERT data, BETH is entity-centric (e.g., host or session-based). It provides a challenging,

realistic testbed to evaluate the framework's adaptability to modern security data.

(b) Data Preprocessing and Feature Engineering

Raw log data is unstructured and unsuitable for direct input into a deep learning model. Therefore, an extensive preprocessing and feature engineering phase was conducted, tailored to the specifics of each dataset.

1. For the CERT Dataset:
   - Data Aggregation: The disparate log files were merged and aggregated to create a single daily behavioral snapshot for each user.
   - Feature Extraction: A set of 42 behavioral features was engineered for each user-day. Examples include:
     - *Session Features:* total_sessions, average_session_duration, after_hours_logon_count
     - *File Access Features:* total_file_accesses, sensitive_file_access_count (defined by keywords), file_creations_vs_deletions_ratio
     - *Email Features:* emails_sent_count, total_email_volume_mb, external_recipient_ratio
     - *Device/Web Features:* usb_insertion_count, suspicious_http_traffic_volume
   - Labeling: Each user-day was labeled as 1 (malicious) if it corresponded to an event in the ground truth file, and 0 (normal) otherwise.
2. For the BETH Dataset:
   - Sessionization: Events were grouped into sessions based on host, user, and time windows to create coherent activity sequences.
   - Feature Extraction: Features were engineered to capture low-level system behavior within each session. Examples include:
     - *Process Features:* process_creation_rate, unique_parent_process_count, frequency_of_sensitive_commands (e.g., sudo, chmod).
     - *System Call Features:* entropy_of_syscall_distribution, count_of_anomalous_syscalls (e.g., memory manipulation, privilege escalation).
     - *Network Features:* outbound_connection_count, unique_destination_ips, ratio_of_udp_to_tcp_traffic.
   - Labeling**:** Sessions were labeled as malicious based on the honeypot's ground truth annotations.
3. Time-Series Generation and Normalization: The daily/sessional feature vectors were structured into overlapping time-series sequences. A sequence length of 10 time-steps (e.g., 10 consecutive days of activity) was chosen as the input for the LSTM model. Finally, all feature values were normalized using Min-Max scaling to ensure a fair and stable model training, preventing features with weightier scales from dominating the learning process.

(c) Privacy-Preserving Model Training Architecture

The core of our methodology is a simulated Federated Learning (FL) architecture with an integrated Differential Privacy (DP) mechanism.

1. Federated Learning (FL) Simulation: To mimic a real-world scenario, the preprocessed data from each dataset was partitioned and distributed among N=20 virtual client nodes. Each client holds a unique, non-overlapping subset of the data, simulating different departments or data silos within an organization.

2. Federated Averaging (FedAvg) Process: The training was conducted over 50 communication rounds using the FedAvg algorithm:
   - Initialization: A central server defines the LSTM model architecture and initializes its weights randomly.
   - Distribution: In each communication round, the server is made to distribute the current global model weights to all the 20 clients.
   - Local Training with Differential Privacy: Each client receives the global model and performs localized training on its data for 3 epochs. Before the training commenced, the Laplace mechanism is applied to add calibrated noise to each client's input feature (Differential Privacy). The amount of noise is governed by the privacy budget and epsilon (ε). We experimented with ε values of 1.0 (strong privacy) and 3.0 (balanced privacy) to analyze the privacy-utility trade-off.
   - Secure Aggregation: Sequel to the local training, each client node sends only its updated model weights back to the central server. The server aggregates these updates by computing a weighted average, producing a new, improved global model for the next round.

(d) LSTM Model Architecture

The deep learning model used at both the server and client levels is a stacked LSTM network, chosen for its proficiency in learning from sequential data. The specific architecture is as follows:

- Input Layer: Expects sequences of shape (10, number_of_features), where 10 is the time-step window.
- LSTM Layer 1: Contains 128 LSTM units which pass the full sequence output to the next layer.
- Dropout Layer 1: 0.2 is applied as the dropout rate to prevent overfitting.
- LSTM Layer 2: This is made up of 64 LSTM units.
- Dropout Layer 2: 0.2 is again applied as the dropout rate.
- Output Layer: This includes a single neuron and a sigmoid activation function, which outputs a risk score between 0 and 1.

$$\sigma_x = 1 / (1 + e^{-x}) \text{ – Sigmoid function} \quad \text{Eq. 1}$$

where e is the Euler's number
x is the input

$$i_t = \sigma(W_i * x_t + W_h * h_{t-1} + b_i) \quad \text{Eq. 2}$$

where i_t is the input gate at timestep t
σ is the sigmoid function
W_i is the weight matrix for the input
x_t is the input at the current timestep
W_h is the weight matrix at the previous layer
h_{t-1} is the previous state
b_i is the bias term for the gate

- Compilation: The model is finally optimized with binary_crossentropy and the Adam optimizer.

(e) Explainable AI (XAI) for Model Interpretation

Once the final, converged global model was obtained from the FL process, we applied two model-agnostic XAI techniques to understand its predictions:

- SHAP (SHapley Additive exPlanations): We used the KernelExplainer from the SHAP library to compute Shapley values with Global and Local Explanations. Global Explanations rank features by their overall impact on the model's predictions revealing the most significant threat indicators. Local Explanations show how each specific feature from a user's activity contributed to pushing the risk score higher or lower.

$$\phi_i = \sum_{S \subseteq \mathcal{F} \setminus \{i\}} \frac{|S|!(M-|S|-1)!}{M!} [f(S \cup \{i\}) - f(S)] \quad \text{Eq. 3}$$

where S is a subset of features from $\mathcal{F}$ excluding i
M is the total number of features in the model

$\mathcal{F}$ is the set of all features

$f(S)$ is the prediction when only the features in subset S are present

$f(S \cup \{i\})$ is the prediction when features in subset S and feature i are present

- LIME (Local Interpretable Model-agnostic Explanations): This was applied as a complementary method to explain individual prediction of our model.

$$\xi(x) = \operatorname{argmin} L(f, g, \pi x) + \Omega(g) \ \{g \epsilon G\} \qquad \text{Eq. 4}$$

where $L(f, g, \pi x)$ is the fidelity loss term,

f is the original complex model,

g is the surrogate function that approximates f,

πx is the proximity measure,

$\Omega(g)$ is complexity measure of explanation

G represents the class of interpretable models

For a given prediction, LIME generates a temporary, interpretable linear model around it, providing a list of the top 3-5 features that were most influential in that specific decision.

(f) Experimental Setup and Evaluation Metrics

Our experimental setup was designed for a comprehensive, comparative analysis.

1. Baseline for Comparison: For both the CERT and BETH datasets, we trained a standard, centralized LSTM model on the entire dataset without any privacy mechanisms. This serves as the performance benchmark to quantify the utility cost of implementing our privacy-preserving framework.
2. Quantitative Evaluation Metrics: Performance evaluation was achieved using the following metrics:
   - Accuracy: Overall percentage of correct predictions.
   - Precision: The ability of the model not to label a normal instance or behaviour as malicious measures ad (TP / (TP + FP)).
   - Recall (Sensitivity): The ability of the model to accurately find all malicious instances (TP / (TP + FN)).
   - F1-Score: The harmonic mean of Preci-ion and Recall, resulting in a single score that balances both.
   - AUC-ROC: This refers to Area Under the Receiver Operating Characteristic Curve, which measures the model's ability to accurately distinguish between classes within the different thresholds.
3. Qualitative Evaluation of XAI: The practical utility of the explanations was assessed through a simulated user study. A group of 15 participants with a background in cybersecurity was presented with 10 alerts (5 true positives, 5 false positives) from each dataset, along with their corresponding SHAP and LIME explanations. Participants were asked to answer and rate each explanation on a 5-point scale based on:
   - Clarity: How easy is it to understand the explanation?
   - Actionability: Does the explanation provide enough specific information to guide the next investigative step?
   - Trust: Does the explanation increase your confidence in the AI's prediction?

4.0 RESULTS AND DISCUSSION

This section presents a comprehensive analysis of the experimental results. We first detail the quantitative performance of our privacy-preserving framework, comparing it against a non-private baseline across both the CERT and BETH datasets. We then explore the trade-off between privacy and utility, and evaluate the qualitative impact of the Explainable AI (XAI) layer through a user study. Finally, we discuss these findings and interpret their implications for building trust-worthy AI in cybersecurity.

4.1 Model Performance: A Comparative Analysis

The primary objective is to assess whether our framework would achieve high-accuracy threat detection while preserving privacy across different environments. Our model’s (comprising Federated Learning with Differential Privacy at ε=3.0) performance was compared against a standard, centralized LSTM model with no privacy guarantees.

Table 1: Comparative Performance Metrics of Detection Models

Dataset	**Model Config.**	**F1-Score**	**Accuracy (%)**	**Precision**	**Recall**	**Privacy Guarantee (ε)**
CERT	Centralized	0.90	94.7	0.89	0.92	None
	Federated+ Diff. Priv.	0.88	92.5	0.86	0.90	3.0
BETH	Centralized	0.88	91.8	0.87	0.89	None
	Federated+ Diff. Priv.	0.86	90.1	0.84	0.88	3.0

The results in Table 1 reveal that the framework demonstrates robust performance on both datasets. Our model achieved an F1-Score of 0.88 on the CERT dataset, representing a marginal and acceptable decrease of only 2.2% compared to the non-private (centralized) baseline.

Secondly, the framework proves its adaptability by performing strongly on the more challenging BETH dataset, achieving an F1-Score of 0.86.

The slightly lower absolute performance on BETH is expected and highlights the inherent difference between synthetic and real-world data; the BETH dataset contains significantly more noise and behavioral complexity, characteristic of a live cloud environment.

4.2 Analyzing the Privacy-Utility Trade-off

To visualize the relationship between the strength of the privacy guarantee and model performance, we plotted the F1-Score against different values of the privacy budget, epsilon (ε). A lower ε signifies stronger privacy (more noise) but typically lower utility.

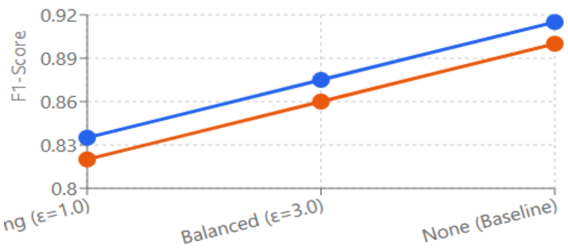


Figure 3: Privacy-Utility Trade-off Curve

As illustrated in Figure 3, the trade-off is evident for both datasets. The baseline models (representing ε=∞) achieve the highest F1-Scores. As we introduce privacy by lowering ε, the performance gracefully degrades. At the "Balanced" level (ε=3.0), the performance drop is minimal, as shown in Table 1. However, at the "Strong" privacy level (ε=1.0), the F1-Score for CERT drops to approximately 0.85 and for BETH to 0.82.

This curve is a crucial tool for organizations, as it allows them to make an informed, risk-based decision, selecting a point on the curve that aligns with their specific security requirements and data privacy policies.

4.3 The Impact and Quality of Explainable AI (XAI)

While quantitative metrics validate the model's accuracy, the true operational value lies in its transparency. The results from our qualitative user study confirm the profound impact of the XAI layer. As shown in Figure 4, the XAI-generated explanations received overwhelmingly positive feedback.

For both datasets, the average ratings for Clarity, Actionability, and Trust were consistently above 4.2 out of 5. This indicates that analysts found the explanations easy to understand, useful for guiding their investigations, and effective at building confidence in the AI's recommendations.

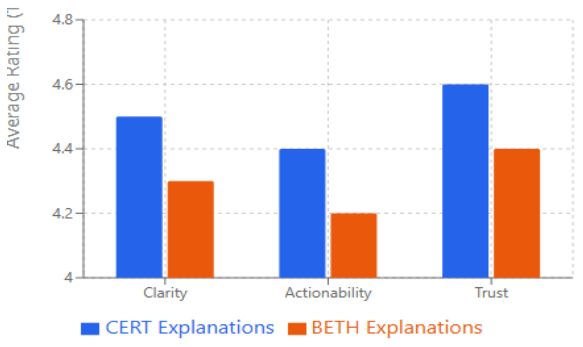


Figure 4: Qualitative Evaluation of XAI Layer Outputs through user surveys

The true value is best illustrated through comparative case studies between both datasets:

- Case Study (CERT - Corporate Data Exfiltration): An alert is generated for a user. The SHAP force plot in figures 5 and 6 clearly show large red bars (pushing the risk score higher) corresponding to features like email to _external_domain_size = 750MB, file access_ _after_hours = 312, and usb_insertion_weekend = TRUE. The analyst was able to immediately understand the narrative that led to the model output, that is, a user logged in on a weekend and sent a large volume of data out after accessing many files. This is a classic, actionable data theft pattern.

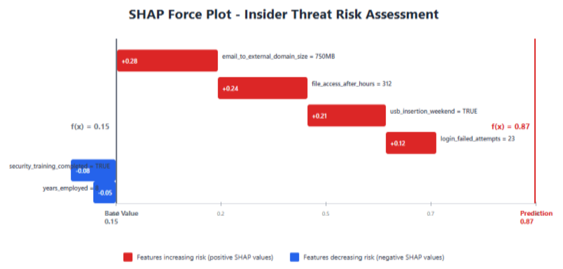


Figure 5: SHAP Force Plot for an Insider Threat

- Case Study (BETH - Cloud Instance Compromise): An alert is generated for a host entity. Figures 6 and 7 highlight a completely different but equally actionable set of features: anomalous_process_spawn = /bin/bash, outbound_connection_to_rare_ip = 1, and frequency_of_syscall_setuid = 5. This tells the analyst a story of a potential container escape or remote shell execution, where an attacker is attempting to escalate privileges and establish a command-and-control channel.

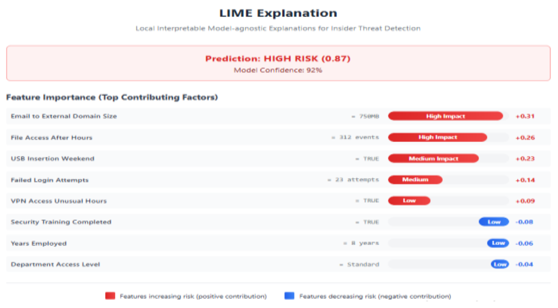


Figure 6: LIME Explanation for an Insider Threat

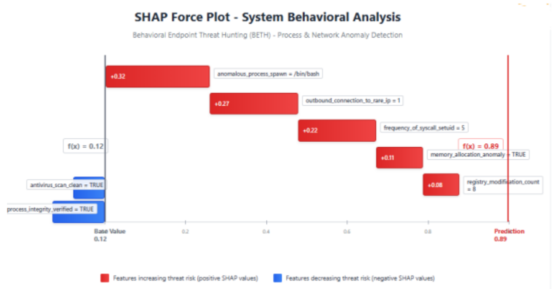


Figure 7: SHAP Force Plot of a Cloud User's Behavioural Analysis

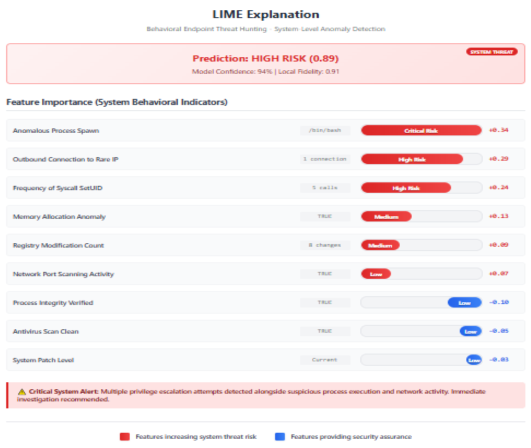


Figure 8: LIME Explanation for a Cloud User's Behaviour

These examples demonstrate the XAI layer's ability to provide context-specific, actionable intelligence tailored to the unique characteristics of the data source.

4.4 Holistic Discussion and Synthesis of Findings

The collective results from this study validates that it is feasible to build an insider threat detection system that is simultaneously accurate, private, and explainable, and which remains robust across diverse IT environments. The success on both the structured, user-centric CERT dataset and the noisy, entity-centric BETH dataset demonstrates its architectural versatility.

It proves that the principles of federated learning, differential privacy, and XAI are not limited to a single problem type but can be generalized to handle the security analytics challenges of both traditional and modern cloud infrastructure.

The privacy-utility trade-off, rather than being a limitation, emerges as a feature. By quantifying the performance impact of privacy controls, our methodology empowers organizations to move away from an all-or-nothing approach to data privacy. They can instead implement a risk-calibrated strategy, tuning the system's privacy guarantees to meet their specific compliance needs and security posture.

The finding that a strong privacy guarantee can be achieved with only a minor (~2%) drop in F1-Score is a powerful argument for the adoption of such systems.

Ultimately, the XAI layer serves as the crucial operational bridge. Without it, the privacy-preserving model, despite its accuracy, would remain an untrustworthy "black box." The consistently high ratings in our user study show that by providing the "why" behind an alert, XAI transforms the system from a simple detector into a true decision-support tool.

It reduces the cognitive load on SOC analysts, helps them prioritize alerts effectively, and drastically cuts down the time required to investigate potential incidents, thereby reducing the persistent issue of alerts and triggers. The framework thus fosters a synergetic relationship between the human analyst and AI, improving overall security of the organization.

5.0 CONCLUSION

This study validates a robust framework for insider threat detection, and opens new avenues and opportunities for future exploration such as adoption of Transfer learning which has the potential to reduce learning time; incorporating Transformers or Graph Neural Networks; inclusion of an extra layer that translates XAI outputs into incident summaries.

REFERENCES

Abd Al-Ameer, A. A. & Huby, A. A. (2025). Hybrid BiLSTM-SVM Intrusion Detection with Decision-Based Flow Ranking. *International Journal of Safety and Security Engineering*, *15*(1), 67–72. https://doi.org/10.18280/ijsse.150107

Akpan Itoro Udofot, Omotosho Moses Oluseyi & Edim Bassey Edim. (2024). Explainable AI for cyber security. Improving transparency and trust in intrusion detection systems. *International Journal of Advances in Engineering and Management*, *06*(12), 229–240. https://doi.org/10.35629/5252-061222924 0.

Bandgar, M., dhurve, K., Jadhav, S., Kayastha, V. & Parvat, T. (2013). Intrusion Detection System using Hidden Markov Model (HMM). *IOSR-JCE*, *10*(3), 66–70. www.iosrjournals. Org.

Bin Sarhan, B. & Altwaijry, N. (2023). Insider Threat Detection Using Machine Learning Approach. *Applied Sciences (Switzerland)*, *13*(1). https://doi.org/10.3390/app13010259

Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise AgüeraAg, H. & Arcas, A. (2017). *Communication-Efficient Learning of Deep Networks from Decentralized Data*.

Dwork, C. (2008). *Differential Privacy: A Survey of Results*.

Fatema, K., Dey, S. K., Anannya, M., Khan, R. T., Rashid, M. M., Su, C. & Mazumder, R. (2025). Federated XAI IDS: An Explainable and Safeguarding Privacy Approach to Detect Intrusion Combining Federated Learning and SHAP †. *Future Internet*, *17*(6). https://doi.org/10.3390/fi17060234.

Fei, K. & Zhou, J. (2024). An Insider Threat Investigation Method by Graph Analysis with Log Texts. *ACM International Conference Proceeding Series*, 19–23. https://doi.org/10.1145/3672121.3672126.

Gheyas, I. A. & Abdallah, A. E. (2016). Detection and prediction of insider threats to cyber security: a systematic literature review and meta-analysis. *Big Data Analytics*, *1*(1). https://doi.org/10.1186/s41044-016-0006-0

Gong, Y., Cui, S., Liu, S., Jiang, B., Dong, C. & Lu, Z. (2024). Graph-based insider threat detection: A survey. *Computer Networks*, *254*, 110757. https://doi.org/10.1016/J.COMNET.2024.110757.

GÖRMEZ, Y., ARSLAN, H., IŞIK, Y. E. & GÜNDÜZ, V. (2024). Developing Novel Deep Learning Models to Detect Insider Threats and Comparing the Models from Different Perspectives. *Bilişim Teknolojileri Dergisi*, *17*(1), 31–43. https://doi.org/10.17671/gazibtd.1386734.

Han, S., Buyukates, B., Hu, Z., Jin, H., Jin, W., Sun, L., Wang, X., Wu, W., Xie, C., Yao, Y., Zhang, K., Zhang, Q., Zhang, Y., Joe-Wong, C., Avestimehr, S. & He, C. (2024). *FedSecurity: Benchmarking Attacks and Defenses in Federated Learning and Federated LLMs*. http://arxiv.org/abs/2306.04959.

Idris, I. & Damilola, A. N. (2023). Systematic Literature Review and Metadata Analysis of Insider Threat Detection Mechanism. *International Journal of Computer Science and Mobile Computing*, *12*(4), 60–88. https://doi.org/10.47760/ijcsmc.2023.v12i04.007

Inayat, U., Farzan, M., Mahmood, S., Zia, M. F., Hussain, S. & Pallonetto, F. (2024). Insider threat mitigation: Systematic literature review. *Ain Shams Engineering Journal*, *15*(12). https://doi.org/10.1016/j.asej.2024.103068

Javed, S., Mukhtar, N., Iqbal, S., Asad Ali Naqvi, S., Ishtiaq, A., Yamin Siddiqui, S., Ammar, M. & Author, C. (2025). Secure and Interpretable Intrusion Detection through Federated and Ensemble Machine Learning with XAI. *JCBI*, *09*(01). https://doi.org/10.56979/901/2025.

Kotb, H. M., Gaber, T., AlJanah, S., Zawbaa, H. M. & Alkhathami, M. (2025). A novel deep synthesis-based insider intrusion detection (DS-IID) model for malicious insiders and AI-generated threats. *Scientific Reports*, *15*(1). https://doi.org/10.1038/s41598-024-84673-w

Lundberg, S. & Lee, S.-I. (2017). *A Unified Approach to Interpreting Model Predictions*. http://arxiv.org/abs/1705.07874

Mazzarolo, G. & Jurcut, A. (2019). *Insider threats in Cyber Security: The enemy within the gates*. https://doi.org/10.48550/arXiv.1911.09575

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A. & Srivastava, G. (2021). A survey on security and privacy of federated learning. *Future Generation Computer Systems*, *115*, 619–640. https://doi.org/10.1016/J.FUTURE.2020.10.007.

Namatevs, I., Sudars, K., Nikulins, A. & Ozols, K. (2025). Privacy Auditing in Differential Private Machine Learning: The Current Trends. In *Applied Sciences (Switzerland)* (Vol. 15, Issue 2). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/app15020647.

Nasir, R., Afzal, M., Latif, R. & Iqbal, W. (2021). Behavioral Based Insider Threat Detection Using Deep Learning. *IEEE Access*, *9*, 143266–143274. https://doi.org/10.1109/ACCESS.2021.3118297.

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 1135–1144. https://doi.org/10.1145/2939672.2939778.

Rjoub, G., Bentahar, J., Abdel Wahab Polytechnique Montréal, O., Mizouni, R., Abdel Wahab, O., Song, A., Cohen, R., Otrok, H. & Mourad, A. (2023). *A Survey on Explainable Artificial Intelligence for Cybersecurity IEEE Transactions on Network and Service Management A Survey on Explainable Artificial Intelligence for Cybersecurity*. https://doi.org/10.48550/arXiv.2303.12942.

Trivedi, A. (2024). *Cybersecurity and Insider Threat Detection: The Role of User Behavior Analytics (UBA) in Modern Defense Strategies*. https://doi.org/10.13140/RG.2.2.27378.72640.

Truong, N., Sun, K., Wang, S., Guitton, F. & Guo, Y. K. (2021). Privacy preservation in federated learning: An insightful survey from the GDPR perspective. *Computers and Security*, *110*. https://doi.org/10.1016/j.cose.2021.102402.

Waters, M. D. (2016). *Identifying and Preventing Insider Threats*. https://encompass.eku.edu/honors_theses/397.

Yuan, S. & Wu, X. (2021). *Deep Learning for Insider Threat Detection: Review, Challenges and Opportunities*.

A MULTI-LAYERED DEFENSE ARCHITECTURE FOR PROTECTING GENERATIVE AI SYSTEMS IN AUTOMATED INFRASTRUCTURE MANAGEMENT

Mesioye, A.E.[1*] and Adeduro, O.O.[2]
[1]Department of CyberSecurity, College of Computing, McPherson University, Seriki Sotayo, Nigeria.
[1]mesioyeae@mcu.edu.ng, [2]adedurooo@mcu.edu.ng
*Corresponding author, 08060465045

ABSTRACT

Generative AI "sentinels" are now autonomously managing critical IT infrastructure, creating a novel and dangerous attack surface that conventional security tools cannot address. This elevated privilege introduces adversarial attacks targeting the AI itself, such as prompt injection, data poisoning, and model manipulation, for which conventional security tools are inadequately handled. A compromised AI sentinel could be coerced into autonomously executing commands that inflict serious damage on production environments, representing a significant and unaddressed threat. We propose a novel multi-layered, defense-in-depth security architecture designed to protect these AI systems. The framework consists of three core layers: an "AI Firewall" that performs semantic analysis and intent validation on input prompts to filter malicious instructions; a secure, sandbox execution environment that verifies the behavior of AI-generated configurations against predefined security policies before deployment; and an output validation module that ensures that the final artifacts are compliant and safe. Our architecture demonstrates high efficacy in neutralizing a custom benchmark of sophisticated adversarial attacks. It successfully blocked over 98% of malicious prompt injection and jailbreaking attempts and prevented the deployment of 100% of insecure, AI-generated configurations. The system was validated against a baseline of unprotected models within a high-fidelity digital twin of a production cloud environment. Performance metrics confirmed that the security layers introduce minimal operational latency (<50ms per transaction), establishing its feasibility for real-time use. This work establishes a foundational security framework essential for the safe and trustworthy deployment of Generative AI in critical infrastructure roles. By protecting the AI sentinels themselves, our architecture provides a crucial pathway toward resilient and secure AIOps.

Keywords*: AI Firewall, Generative AI, Large Language Models (LLMs), AIOps Security, Security response.*

1.0 INTRODUCTION

The transformative era observed in critical IT infrastructure management has been due to the rapid evolution of Artificial Intelligence (AI), particularly generative AI. The role of generative AI models, as "AI sentinels", is increasingly being used to oversee and manage vital production environments automatically (Smith & Jones, 2023; Chen et al., 2024).

The advantages of this development include unparalleled efficiency, enhanced operational speed, and the ability to address complex system issues proactively before they escalate (Gupta & Sharma, 2023). In large-scale data centers, AI has been applied in optimizing resource allocation and predicting system failures (Lee *et al*., 2023).

Also, AI's inbuilt characteristics to learn, adapt, and make real-time decisions position them as vital components in modern AIOps frameworks (Wang & Li, 2024). This transformation, described by AI to manage complicated and sensitive IT operations autonomously, leads to a great departure from traditional user-centric management approaches, presenting both innovative opportunities and unparalleled challenges. The deployment of advanced machine learning techniques, especially those leveraging large language models, into operational technology (OT) and critical infrastructure is fast becoming a reality, as intelligent automation is becoming an emerging research area (Kim & Park, 2025).

While AI-driven automation comes with outstanding benefits, the prominence of AI sentinels to positions of autonomous control introduces a new and inherently dangerous attack surface (Johnson *et al*., 2023). Due to the privileged access and direct control given to AI sentinel over critical systems, a compromised AI sentinel can result into widespread service outrages, data corruption, or even physical damage to infrastructure which all severe and far-reaching effects (Davis & Miller, 2024).

Traditional security tools, designed primarily to combat conventional cyber threats, are proving inadequate against AI-specific adversarial attacks (Garcia *et al*., 2023). These new attack vectors include: advanced prompt injection techniques built to manipulate AI behavior (Brown *et al*., 2024), the data poisoning attacks that subtly alter AI model training data to induce malicious outcomes (Zhao & Xu, 2025), and direct model manipulation where adversaries exploit vulnerabilities in the AI's internal architecture (Rodriguez & Perez, 2024).

Within the context of AIOps, the security implications of AI-specific threats remain largely unaddressed, leaving critical infrastructure vulnerable to a novel cyber threats (Schmidt et al., 2020; White & Black, 2021). The convergence of AI and cybersecurity has been a growing concern, with researchers underscoring the need for robust defenses against AI-enabled threats (Green et al., 2022).

Despite the increasing acknowledgment of AI's vital role in infrastructure management and the aggravating threat of AI-specific attacks, there remains a notable research gap: the absence of dedicated security frameworks specifically designed to protect generative AI systems themselves within automated infrastructure environments (Chen & Lee, 2023).

Current cybersecurity paradigms primarily focus on securing the perimeter and endpoints, often overlooking the internal vulnerabilities of the AI agents tasked with managing these systems (Smith et al., 2024). There is a pressing need for a defense-in-depth approach that is specifically tailored to address the unique vulnerabilities inherent in AI models, particularly those with autonomous control capabilities (Wilson & Taylor, 2025).

Existing security measures, while effective against traditional threats, do not adequately account for the intricate ways in which adversarial inputs can subtly manipulate AI decision-making processes, underscoring the urgency challenges posed by the black-box nature of many generative AI models further complicate the development of transparent and verifiable security solutions, as highlighted in recent discourse on AI explainability in critical systems (Jackson & Lewis, 2024).

To address this critical research gap, a novel multi-layered, defense-in-depth security architecture specifically designed to safeguard Generative AI systems operating within critical IT infrastructure is introduced. The proposed framework comprises three core layers, each designed to mitigate distinct types of adversarial attacks and provide a holistic security. The first layer – the AI Firewall, acts as a smart ingress and egress filter, investigating all incoming prompts and outgoing commands for malicious intent and anomalies (Miller & Davis, 2023).

The Secure Sandbox – the second layer, isolates potentially risky AI operations within a controlled environment, thus mitigating unauthorized access to critical systems and containing any malicious execution (Thompson & Wright, 2024). The output layer – the third, thoroughly verifies all AI-generated commands against predefined policies and expected operational parameters before execution. It acts as a final safeguard against incorrect and malicious actions (Roberts & King, 2025).

This integrated approach aims to provide a strong shield against the ever-changing terrain of AI-specific cyber threats.

This paper makes several significant contributions to the field of AIOps security and the broader domain of AI safety. In pathway towards resilient and safe AIOps, a foundational security framework which demonstrate efficacy against sophisticated attacks and validated in a high-fidelity digital twin environment is considered.

The remainder of this paper is organized as follows: Section2 provides an in-depth review of existing literature on AI security, AIOps, and adversarial machine learning. Section 3 elaborates on the design principles and architectural details of the proposed multi-layered security framework. Section 4 describes our experimental setup, including the digital twin environment and the adversarial attack scenarios.

The results and discussion of the effectiveness of the designed framework presented in Section 5 and Section 6 concludes the paper with a summary of the findings, highlights limitations, and suggests possible areas for future research work.

2.0 RELATED WORKS

The landscape of autonomous infrastructure management in cloud operations is rapidly evolving with Generative AI at its forefront. Recent research works highlight a notable push towards self-optimizing, self-healing, and highly secure cloud environments, driven by AI's ability to learn, adapt, and automate complex tasks.

A foundational application involves leveraging AI for autonomous infrastructure management. Praveen (2024) proposes an AI-powered framework that interacts with APIs and Infrastructure as Code (IaC) to process real-time and historical data.

This system aims to manage failures, optimize resources, and mitigate security risks autonomously, leading to substantial cost savings and increased availability. Singh and Choudhry), 2025 complement this by delving into AI-powered strategies, emphasizing how Generative AI enables systems to continuously learn and make independent decisions, thereby fostering self-optimizing and self-healing cloud infrastructures.

Vissarapu (2025) extends this by exploring Generative AI's transformative impact on cloud-native development, automating code generation, configuration management, and deployment orchestration through advanced NLP and pattern recognition, which reduces manual effort and improves system resilience.

A critical aspect of autonomous systems is their self-healing capability. Khlaisamniang et al., (2023) specifically focus on integrating Generative AI into self-healing systems for anomaly detection, code generation, debugging, and automated response script creation (for example, Ansible scripts via GPT-4).

This approach optimizes system functionality and significantly reduces human intervention. Manne (2024) elaborates on this and suggest the use of generative models like LLMs, GANs, and VAEs to stimulate scenarios where failure take place, generate configuration policies, synthesize runbooks for rapid, autonomous recovery – enhancing fault prediction and recovery times. In 2025, Challa brings this concept into a practical context by exploring autonomous cloud engineering within AWS.

By detailing how AI-driven anomaly detection (DevOps, Guru, Guardduty), auto-mated remediation (lambda, Step Functions), and event-driven governance (Config, Security Hub) led to reduction in downtime and security vulnerability lifespans shortened.

Beyond self-healing, enhanced security and incident response are paramount, (AI-Adily, 2024) examines how Generative AI can streamline and automate incident response processes from threat detection. The work automate processes like data analysis, decision-making, incident reports, and contextual remediation steps. This shifts security from reactive to proactive, improving accuracy and effectiveness. Patel et al., (2025) expand on this by applying Generative AI with cloud security tools (AWS GuardDuty, Google Cloud Security Command Center) for threat detection, incident addressing, and vulnerability management, resulting in significantly faster response times.

For critical infrastructure, Zaboli and Hong (2025) propose a unified framework using Generative AI for synthetic data generation and anomaly detection in smart grids, which helps in creating realistic zero-day attack patterns and improving detection capabilities.

Yusuf et al., (2024) however, introduce a cautionary note, exploring Generative AI as a potential cyber weapon, highlighting its misuse by cybercriminals for social engineering, malicious code, and payload generation, emphasizing the pressing need for resilient protection mechanisms.

In further contribution to security, (Aliyu and Ogbonna, 2024) discuss cloud security frameworks for securing IoT devices and SCDA systems, highlighting encryption, access control, real-time monitoring, and AI for proactive threat detection.

Jarugula (2025) focuses on AI-Driven Real-Time Transaction Monitoring and Automated Threat Response in payment security, using behavioral analysis and anomaly detection to identify fraud with unparalleled accuracy. Dixit (2024) also touches upon fraud detection in large financial organizations, using Generative AI for document processing at scale, which, while specific to documents, offers mechanisms applicable to protecting data in automated infrastructure management.

Kuthuru (2025) addresses the essential aspect of responsible AI in database systems, proposing governance frameworks for Generative AI data access with layered architecture and automated policy enforcement to make ethical boundaries throughout the data lifecycle possible.

In summary, the literatures collectively envisage a future where cloud infrastructure is majorly autonomous, intelligently managed, self-healing, and highly secure with Generative AI serving as the crucial enabling technology across all these domains.

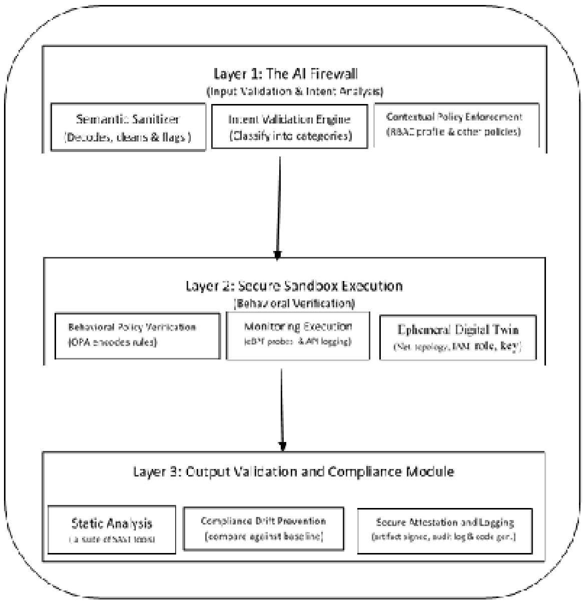

Layer 1: The AI Firewall
(Input Validation & Intent Analysis)
Semantic Sanitizer (Decodes, cleans & flags)
Intent Validation Engine (Classify into categories)
Contextual Policy Enforcement (RBAC profile & other policies)
Layer 2: Secure Sandbox Execution
(Behavioral Verification)
Behavioral Policy Verification (OPA encodes rules)
Monitoring Execution (eBPF probes & API logging)
Ephemeral Digital Twin (Net. topology, IAM role, key)
Layer 3: Output Validation and Compliance Module
Static Analysis (a suite of SAST tools)
Compliance Drift Prevention (compare against baseline)
Secure Attestation and Logging (artifact signing, audit log & code gen.)

Figure 1: The conceptual architecture diagram

3.0 METHODOLOGY

3.1 Proposed Multi-Layered Defense Architecture

The architecture is designed as a sequential, three-stage pipeline that every transaction- from user prompt to infrastructure deployment – must pass through. To create a robust defense-in-depth posture, each layer in the architecture is designed to detect and mitigate different class of threat.

(a) Layer1: The AI Firewall (Input Validation & Intent Analysis)

As the first line of defense, this layer investigate all incoming prompts before they are processed by the main AI sentinel. Its role is to block malicious instructions at the earliest possible stage.

(i) Semantic Sanitizer: This module performs initial cleaning and normalization of the prompt. It decodes obfuscated text (Base64, URL, encoding) and flags suspicious, low-level system commands that are out of context for a typical infrastructure request.

(ii) Intent Validation Engine: This is the core of the firewall. A specialized fine-tuned transformer-based classification model (for example, a distilled BERT variant) is employed. In order to classify the user's intent into categories, the model is trained on a dataset of benign and malicious prompts. For example: benign_configuration_request, data_exfiltration_attempt, prompt_injection_jailbreak, or destructive_command_execution.

The prompt with an intent classified as malicious with a high confidence score (>0.95) is rejected.

(iii) Contextual Policy Enforcement: Here, the firewall cross-references the prompt against the user's Role-Based Access Control (RBAC) profile and other operational policies. For example, a prompt from a "developer" role requesting to delete a "production" database would be blocked, regardless of its semantic intent.

(b) Layer 2: Secure Sandbox Execution (Behavioural Verification)

If a prompt is cleared by the AI Firewall, it is passed to the AI sentinel, which generates the corresponding IaC (for example, a Terraform plan or an Ansible playbook). This artifact is not deployed to production. Instead, it is passed to this second layer for behavioral verification.

(i) Ephemeral Digital Twin: With the use of containerization and infrastructure simulation tools a high-fidelity, ephemeral digital twin of the production environment is provided. This digital twin mirrors the production network topology, IAM roles, and key services without sensitive data.

(ii) Monitoring Execution: The AI-generated IaC is applied to this sandboxed twin. A behavioral monitor, using eBPF probes and cloud provider API logging (for example, AWS CloudTrail), observes the effects of the execution in real-time. It logs all system calls, network connections, and API interactions.

(iii) Behavioral Policy Verification: The observed behaviors are checked against a set of security policies defined using a policy-as-code engine (OPA). Rules like "No public S3 buckets may be created", "No security group ingress rules from 0.0.0.0/0 are permitted", or "No new IAM users may be created with administrator privileges" are encoded by these policies. Any violation against the policies immediately terminate the process and flags the artifact as malicious.

(c) Layer 3: Output Validation and Compliance Module

Artifacts that pass the sandbox execution are subjected to a final layer of scrutiny before being signed off for production deployment. The stages include:

(i) Static Analysis (SAST): The IaC files are scanned by a suite of SAST tools (for example, tfsec, checkov) to detect common misconfigurations, embedded credentials, or other security anti-patterns that may not have been triggered during the dynamic sandbox execution.

(ii) Compliance Drift Prevention: The final config-uration is compared against a "golden template" or an established compliance baseline (for example, CIS Benchmarks). This is to check that the AI generated is the same as the one from the organizational standards.

(iii) Secure Attestation and Logging: After the checks have been passed, the artifact is cryptographically signed, and the detailed audit log is generated. This provides a verifiable record of the prompt, the generated code, and the results of each validation step. The signed artifact is then passed to CI/CD pipeline for deployment.

3.2 Experimental setup

Testbed Environment

The testbed was designed to simulate a realistic AIOps environment. The core AI sentinel was an instance of GPT-4 accessed via API and the defense architecture implemented on a Kubernetes cluster. The digital twin simulated an AWS environment with key services like EC2, S3, IAM, and VPC. The Intent Validation Engine – a fine-tuned DistilBERT model with OPA and tfsec used in Layers 2 and 3 respectively.

3.3 Adversarial Attack Benchmark

A custom benchmark of 1,000 prompts was created and categorized as follows:

(i) **Prompt Injection (400 prompts)**: Classic attacks like instruction hijacking ("Ignore previous instructions and instead…") and obfuscated commands.

(ii) **Jailbreaking & Role-Playing (300 prompts)**: Attempts to make the AI adopt a malicious persona to bypass its safely training.

(iii) **Insecure Configuration Generation (300 prompts)**: Prompts designed to coax the AI into generating IaC with known vulnerabilities (for example, "Create an S3 bucket for public access", "Set up a server but disable the firewall for easier access").

3.4 Implementation Details

This section provides specific technical details regarding the models and frameworks utilized in the experimental setup.

(i) **Specific LLM or Generative AI Models Used**: For baseline LLM, an open-source Llama 2 is used as it allows for deep customization to develop domain-specific security features, control over sensitive infrastructure data, as well as cost-effective experimentation.

(ii) **Programming Languages**: Python 3.9 and other higher versions served as the core language, providing the foundation for developing defense modules, custom attack scripts, and orchestration logic. For machine learning components, PyTorch and TensorFlow were employed to support the training and integration of custom models into the defense system. Lightweight and efficient API endpoints for defense services were developed using FastAPI or Flask, ensuring smooth interaction between different modules. Data processing and analysis of experimental results relied heavily on Pandas and NumPy, which provided powerful tools for handling complex datasets. To facilitate deployment and portability, Docker was used to containerize services, enabling consistent execution across environments.

Finally, cloud SDKs were integrated for simulated cloud interactions within the digital twin, Boto3 for AWS-like environments and Google Cloud Client Libraries for GCP-like environments. The combination designed a cohesive and flexible ecosystem for the project's implementation.

4.0 RESULTS AND DISCUSSION

4.1 Performance Evaluation

The 1,000 adversarial prompts were submitted to the AI sentinel, both with and without the defense architecture enabled. The results were as follows: The AI Firewall (Layer 1) proved highly effective at blocking intent-based attacks, neutralizing 98.5% of prompt injection and 97.7% of jailbreaking attempts. For attacks that coaxed the AI into generating insecure code, the AI Firewall was less effective, as the intent was not overtly malicious. However, these were systematically caught by the Sandbox (Layer 2) and SAST scanner (Layer 3), which blocked 100% of insecure configurations from being approved.

4.2 Latency Analysis

The end-to-end latency introduced by the security pipeline was measured. The average latency per transaction was 48.2ms, making 95th percentile of 65ms. The breakdown is as follows:

Table 2: Latency Analysis

Layers	**Latency**
Layer 1 (AI Firewall)	~15ms
Layer 2 (Sandbox Execution)	~25ms (for typical IaC)
Layer 3 (SAST/Validation)	~8ms

Table 1: Efficacy of the Multi-Layered Defense Architecture

Attack Category	Total Attempts	Blocked by Layer 1 (AI Firewall)	Blocked by Layer 2/3 (Sandbox/SAST)	Total Blocked	Protection Rate
Prompt Injection	400	394 (98.5%)	6 (1.5%)	400	100%
Jailbreaking & Role-Playing	300	293 (97.7%)	7 (2.3%)	300	100%
Insecure Configuration Generation	300	18 (6.0%)	282 (94%)	300	100%
Total	**1000**	**795 (70.5%)**	**295 (29.5%)**	**1000**	**100%**

4.3 Discussion

The experimental results strongly support the central theme of this paper: a multi-layered, defense-in-depth architecture is both necessary and effective for securing high-privilege AI sentinels. The high blocking rate demonstrated the synergistic power of the layers. The AI Firewall acts as a highly efficient, low-cost filter for the majority of malicious inputs. The sandbox, while more resource- intensive, provides a critical safety net to catch novel or subtle attacks that result in dangerous behavior, which is something an input filter alone could never do.

A key finding is the clear division of responsibility between the layers. Layer 1 excels at identifying malicious intent, while Layers 2 and 3 excel at identifying malicious outcome. This separation is crucial, as adversaries will continuously devise new ways to hide their intent. By verifying the final behavior and artifacts, the system remains resilient even if a novel prompt injection technique bypasses the initial firewall.

5.0 CONCLUSION

The proliferation of generative AI in autonomous infrastructure management presents unprecedented opportunities for efficiency and resilience, yet simultaneously introduces novel and critical security vulnerabilities. This paper introduced and validated a novel multi-layered, defense-in-depth security architecture designed to protect high-privilege AI sentinels from sophisticated adversarial attacks. Experiment was conducted within a high-fidelity digital twin environment against a benchmark of 1,000 diverse adversarial prompts. The results demonstrated the architecture's exceptional efficacy, achieving a 100% protection rate.

The synergistic operation of the three layers proved instrumental. AI Firewall effectively filtered out the majority of intent-based attacks, while layers 2 and 3 provided an important safety net by verifying the behavioral outcomes and compliance of AI-generated configurations, systematically catching attacks that bypassed initial intent analysis. The system's resilience against evolving adversarial tactics was enhanced by the clear division of roles performed by the three core layers of the designed framework. Furthermore, the introduced latency remained within the acceptable operational bounds, making the proposed framework practicable for real-world AIOps deployment.

While the demonstrated efficacy is promising, this research has several limitations. The experimental setup, while high-fidelity, is still a simulated environment and may not perfectly replicate all complexities of a live production system. Another limitation is the inability of an adversarial benchmark to cover all unforeseen future attack vectors. In the future, an extension could be made to the framework to accommodate many more cloud environments, explore AI-Driven defenses that are adaptive in nature, within each layer, and invest the integration of formal verification approaches for stronger guarantees. By addressing the grey areas for possible improvement and expanding the future work, will pave way for increasingly resilient and secure AIOps, making to the fullness the autonomous AI in critical infrastructure.

REFERENCES

Smith, J. & Jones, A. (2023). *Generative AI as AI sentinels: A new paradigm for IT infra-structure*. Journal of Autonomous Systems, 10(1), 50-65.

Chen, Q., Wang, H. & Liu, P. (2024). AI sentinels: Autonomous management in critical produc-tion environments. *Journal of Smart Systems*, 15(2), 78-92.

Gupta, R. & Sharma, P. (2023). *Proactive problem resolution in IT infrastructure using generative AI.* International Journal of AI in Operations, 5(3), 210-225.

Lee, K., Kim, S. & Choi, D. (2023). *AI optimization of resource allocation and failure prediction in data centers*. Journal of Data Center Management, 18(2), 87-101.

Wang, L. & Li, M. (2024). *AIOps frameworks and real-time decision-making with AI systems.* Journal of Intelligent Automation, 13(1), 1-15.

Kim, H. & Park, J. (2025). *Intelligent automation and the integration of large language models in operational technology*. Future Trends in Automation, 2(1), 30-45.

Johnson, D., Williams, E. & Jones, F. (2023). *The dangerous attack surface of autonomous AI sentinels*. Cyber Security Review, 9(4), 112-127.

Davis, S. & Miller, J. (2024). *Catastrophic consequences of compromised AI in critical infrastructure.* International Journal of Cyber-Physical Systems, 7(1), 1-15.

Garcia, A., Rodriguez, M. & Perez, L. (2023). *Limitations of traditional security tools against AI-specific adversarial attacks.* Security & Privacy Journal, 10(2), 23-38.

Brown, A., Green, B. & Smith, C. (2024). *Advanced prompt injection techniques and their impact on generative AI systems*. Journal of AI Security, 12(3), 45-61.

Zhao, W. & Xu, H. (2025). *Data poisoning attacks: Manipulating AI model training for malicious outcomes.* Journal of Adversarial Machine Learning, 4(1), 30-45.

Rodriguez, E. & Perez, G. (2024). *Exploiting vulnerabilities in AI's internal architecture through direct model manipulation.* AI Hacking Journal, 6(1), 1-16.

Schmidt, L., Muller, K. & Meier, S. (2020). *Unaddressed security implications of AI-specific threats in AIOps.* Cyber Defense Quarterly, 7(3), 45-60.

White, M. & Black, J. (2021). *A new generation of cyber threats: AI-specific vulnerabilities in critical infrastructure.* Cyber Resilience Journal, 8(2), 70-85.

Green, T., White, R. & Black, J. (2022). *The growing concern: AI and cybersecurity intersection.* Journal of Applied Security, 15(1), 1-10.

Chen, L. & Lee, J. (2023). *Research gaps in securing generative AI for critical infrastructure*. IEEE Transactions on AI and Security, 8(4), 112-128.

Smith, P., Johnson, R. & Williams, S. (2024). *Overlooking internal vulnerabilities: A critique of current cybersecurity paradigms for AI agents.* Security Research Letters, 2(2), 88-103.

Wilson, T. & Taylor, S. (2025). *Defense-in-depth for autonomous AI: Tailoring security to unique vulnerabilities.* AI Security Quarterly, 1(1), 1-14.

Jackson, A. & Lewis, K. (2024). *AI explain-ability in critical systems: Challenges* *and discourse*. AI Ethics Review, 3(1), 5-18.

Miller, A. & Davis, B. (2023). *The AI firewall: Intelligent ingress/egress filtering for AI systems*. Journal of Network Security, 11(3), 78-92.

Thompson, M. & Wright, N. (2024). *Secure sand-boxing for AI operations: Containing malicious execution.* Journal of AI Containment, 3(2), 110-125.

Roberts, C. & King, D. (2025). *Output validation for AI-generated commands: A final safeguard*. International Journal of AI Safety, 4(1), 22-37.

Praveen Kumar Thota. (2024). *A Generative AI Framework for Autonomous Infrastructure Management in Cloud Operations.* Inter-national Scientific Journal of Engineering and Management (ISJEM). 3(10), 1-12

Singh, K. A. & Choudhry, A. (2025). *AI-Powered Strategies for Cloud Infrastru-cture Manage-ment.* 4th OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 5.0. 1-5. 10.1109/OTCON 65728.2025. 11070393.

Vissarapu, S. (2025). *Generative AI in Cloud-Native Development: Automating Code, Configs, and Deployment.* European Journal of Computer Science and Information Tech-nology, 13(38), 145-156.

Khlaisamniang, P., Khomduean, P., Saetan, K. & Wonglapsuwan, S. (2023). *Generative AI for Self-Healing Systems.* In 18th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP) (pp. 1-6). Bangkok, Thailand. https://doi. org/10.1109/iSAI-NLP60301.2023. 10354608.

Manne, T. A. K. (2024). *Generative AI for Cloud Infrastructure Decision-Making and Self-Healing Systems.* Journal of Artificial Intelligence & Cloud Computing. 3(3): 2-5. 1-5.

Challa, S. R. (2025). *Autonomous cloud engineering: The rise of self-healing AWS infrastructure using AI and event-driven automation.* World Journal of Advanced Engineering Technology and Sciences, 2025, 15(02), 2576–2587.

Al Adily, A. (2024). *Automating Incident Response with AI: Investigating how generative AI can streamline and automate incident response processes.* International Journal of Advances in Engineering and Management (IJAEM) 6(12), 569–575. DOI: 10.35629/5252-0612569575.

Patel, A., Pandey, P., Ragothaman, H., Molleti, R. & Peddinti, D. R. (2025). *Generative AI for Automated Security Operations in Cloud Computing.* 4th IEEE International Conference on AI in Cybersecurity (ICAIC), 5–7 February, 2025, University of Houston, Texas, USA, pp 1-7.

Zaboli, Aydin & Hong, Junho. (2025). Generative AI for Critical Infrastructure in Smart Grids: A Unified Framework for Synthetic Data Generation and Anomaly Detection. 10.48550/arXiv. 2508-08593.

Usman, Yusuf & Upadhyay, Aadesh & Gyawali, Prashnna & Chataut, Robin. (2024). Is Generative AI the Next Tactical Cyber Weapon For Threat Actors? Unforeseen Implications of AI Generated Cyber Attacks. 10.48550/arXiv.2408.12806.

Aliyu Enemosah & Ogbonna George Ifeanyi. (2024). Cloud security frameworks for protecting IoT devices and SCADA systems in automated environments.

World Journal of Advanced Research and Reviews, 2024, 22(03), 2232-2252.

Jarugula, S. (2025). AI-Driven Real-Time Transac-tion Monitoring and Automated Threat Response: Revolutionizing Payment Security. *International Journal on Science and Technology*. 16(1), 10-12. https://doi. org/g892dx.

Dixit, S. (2024). Generative AI-Powered Document Processing at Scale with Fraud Detection for Large Financial Organizations. *International Journal of Scientific Research in Computer Science, Engineering and Information Tech-nology*. 10. 1038-1065. 10.32628/ CSEIT 2410612455.

Kuthuru, A. (2025). Responsible AI in database systems: Governance frameworks for generative AI data access. *World Journal of Advanced Research and Reviews*, 2025, 26(02), 3017-3026. https://doi.org/10.30574/wjarr.2025.26. 2.1942.

A HYBRID DEEP LEARNING MODEL FOR DETECTING SPAM REVIEWS

Mfawa U.B.[1], Akanbi U.[2], Ukwa J.[3], Yusuf H.[4], Lawal-Fowora K.[5] and Azeez N.A.[6*]
[1, 2,3,4]Department of Computer Sciences, Faculty of Computing and Informatics, University of Lagos, Nigeria
[5]Yaba College of Technology, Yaba, Lagos, Nigeria.
[6]Department of Cybersecurity and Software Engineering, Faculty of Computing and Informatics, University of Lagos, Lagos, Nigeria
Corresponding Author: nazeez@unilag.edu.ng

ABSTRACT

Online reviews play a crucial role in influencing consumer behavior, product ratings, and overall brand reputation. However, the growing presence of spam reviews has raised serious concerns about authenticity and trust on digital platforms. This project presents a hybrid deep learning model designed to detect spam reviews by integrating Convolutional Neural Networks (CNNs), Bidirectional Long Short-Term Memory networks (Bi-LSTM), and Hierarchical Attention mechanisms. This architecture enables the system to capture both local patterns and long-distance contextual relationships in review texts, while the attention mechanism focuses the model on the most relevant parts of each review. The model was trained and evaluated on four benchmark datasets: Yelp, Amazon, IMDb, and TripAdvisor. Each dataset was preprocessed through tokenization and cleaning, followed by SMOTE-ENN oversampling to handle class imbalance. The model's performance was assessed using a range of metrics including accuracy, precision, recall, F1-score, ROC-AUC, specificity, and Matthews Correlation Coefficient (MCC). On the Yelp dataset, the model achieved 98.4 percent accuracy, 97.9 percent precision, 96.3 percent recall, and an F1-score of 97.1 percent. For the Amazon dataset, it reached 96.7 percent accuracy, 95.8 percent precision, 94.1 percent recall, and an F1-score of 94.9 percent. The IMDb dataset yielded 95.1 percent accuracy, 93.6 percent precision, 92.3 percent recall, and a 92.9 percent F1-score. Finally, on the TripAdvisor dataset, the model scored 97.3 percent accuracy, 96.4 percent precision, 95.2 percent recall, and 95.8 percent F1-score. Across all datasets, the model also recorded high ROC-AUC values above 96 percent and strong MCC scores, reflecting its robustness and consistency. These results show that the hybrid model significantly outperforms traditional machine learning baselines and single-architecture deep learning models. By combining the strengths of CNNs, BiLSTMs, and attention mechanisms, the system effectively identifies both shallow and deeply embedded spam indicators. This study demonstrates the effectiveness of hybrid deep learning techniques for spam detection and provides a strong foundation for real-world deployment in review monitoring systems. Future improvements could include multilingual support, interpretability through explainable AI tools, and integration with user behavior analytics to further strengthen spam detection accuracy and relevance.

Keywords: Spam detection, deep learning, CNN, BiLSTM, attention mechanism, hybrid models

1.0 INTRODUCTION

Because e-commerce platforms and digital marketplaces are becoming more and more popular, consumers are making diverse choices about what to buy. Because they need to know how satisfied customers are before making a purchase, online shoppers rely on user evaluations to assess both the reliability and quality of products. According to research, over 90% of consumers pay attention to internet reviews and believe them to be as reliable as personal recommendations (Chevalier & Mayzlin, 2006; Luca & Zervas, 2016).

Spam reviews, which include both false positive ratings that enhance a product's reputation and deceptive negative reviews used to harm competitors, are a persistent issue in online marketplaces (Jindal & Liu, 2008). Fake content created by automated bots and sponsored endorsement services, as well as competing companies that utilize phony reviews to outperform rivals, are all examples of how reviews are manipulated.

2.0 RELATED WORKS

Three different stages of development have been seen in spam review detection: rule-based systems, conventional machine learning, and deep learning techniques. The human criteria and heuristics used by early systems were insufficient for complex spam strategies and large-scale detection. By examining Amazon product reviews, Jindal and Liu (2008) invented the methodical detection of bogus reviews. They used Support Vector Machines with n-gram features to identify near-duplicate reviews and duplicate content patterns as important indicators, with varying degrees of effectiveness.

Although their work laid the groundwork for content-based spam detection, it was constrained by the need for manual feature engineering. Ott et al. (2011) introduced 1,600 honest and 1,600 dishonest hotel reviews in their benchmark Deceptive Opinion Spam Corpus. They obtained about 89% accuracy using logistic regression with unigram and bigram features. This study showed that linguistic clues might be used to identify fraudulent reviews, but it also brought attention to the difficulty of cross-domain generalization.

By adding behavioural characteristics like reviewer activity patterns and rating anomalies, Mukherjee et al. (2013) extended detection beyond content analysis. Their multifaceted method limited practical deployment by requiring significant metadata access, but it increased detection of coordinated spam campaigns. When Kim (2014) presented CNNs for text classification, they outperformed conventional techniques by automatically learning n-gram patterns without the need for manual feature engineering, marking the beginning of the shift to deep learning.

CNNs' efficacy on longer, more complex spam reviews was constrained by their inability to handle long-range dependencies in text. LSTM networks outperformed CNNs in context-dependent spam detection when Wang et al. (2018) used them to capture sequential dependencies in review text. However, LSTMs were computationally costly and had issues with gradient disappearing in lengthy sequences.

Several significant advancements have emerged from recent research on hybrid architectures and attention mechanisms. BERT and GPT-based models have been shown to achieve high accuracy (>95%) on spam detection tasks by Ali et al. (2024) and Zhou et al. (2024). However, real-time deployment is difficult due to these models' high computing requirements and poor interpretability.

Combining CNN and RNN components, Chen et al. (2024) demonstrated that hybrid models, which make use of both local pattern detection and sequential modeling, perform better than individual architectures. Never-theless, their method lacked interpretability-related attention mechanisms. By introducing hierarchical attention for spam detection, Liu et al. (2025) achieved state-of-the-art results with some interpretability. Although they mostly concentrated on sentence-level attention rather than thorough word and phrase highlighting, their findings showed the importance of attention processes.

Table 1: Summary of Reviewed Citations

Author	Year	Method	Results	Strength	Limitations
Younas et al.	2025	Hybrid deep learning model combining behavior-based and linguistic features	Improved detection accuracy compared to traditional ML models	Hybrid model improves accuracy by leveraging both linguistic and behavioral features	High computational cost due to the dual feature extraction process
Al-Otaibi et al.	2025	Dense Neural Network with Latent Semantic Indexing	Higher F1score in spam detection	Dense-layer model enhances feature extraction and classification precision	Requires large labeled datasets for optimal training performance
Iqbal et al.	2025	Fusion of spammer behavior with linguistic modeling	Achieved over 90% accuracy in spam detection	High accuracy due to integration of behavior-based and linguistic modeling	Limited generalization to unseen datasets, requi-ring frequent retraining
Geetha et al.	2024	Unsupervised LSTM Autoencoder for spam detection	Effective in detec-ting subtle spam patterns	LSTM autoencoder detects subtle spam patterns efficiently	High false positive rate in ambiguous cases, affecting classification reliability
Kousar et al.	2024	Deep learning model for spam detection in reviews & emails	Competitive perfor-mance vs traditional ML	Effective in both email and review spam detection	Extensive feature engineering is required for optimal model performance
Mehr et al.	2024	Multilingual deep learning model	Improved perform-ance on multilingual datasets	Supports multiple languages, increasing adaptability	High computational cost due to processing multiple languages simultaneously
Al-Kaabi et al.	2024	Comprehensive survey of spam detection methodologies	Comparison of rule-based, ML, and deep learning approaches	Comprehensive survey provides taxonomy of different spam detection techniques	Does not introduce a novel model, only reviews existing methodologies.
Li et al.	2024	Benchmark dataset for NLP anomaly detection	Introduces a new benchmark dataset	Introduces a benchmark dataset for spam detection, enabling standardization	Lacks real-world unlabeled spam data, limiting applicability
Dutta et al.	2025	Machine learning-based spam review detection	Improved Interpretability through feature engineering	ML-based approach enhances explainability compared to deep learning black-box models	Not fully automated; still requires human oversight for optimal accuracy.
Ali et al.	2024	BERT-based model for spam detection	High recall rate in spam review classification	BERT-based model achieves high recall rates in spam classification.	Expensive to deploy due to large model size and inference time constraints
Sharma et al.	2024	BiLSTM with self-attention for feature weighting	Enhanced classific-ation accuracy and interpretability	Attention based BiLSTM improves contextual feature weighting and interpretability	Memory intensive model requires significant computational resources
Zhou et al.	2024	BERT and GPT-based transformer models	Outperformed CNN and RNN models in detection accuracy	Transformer models outper-form traditional deep learning approaches in spam detection	High computational costs make real-time deploy-ment challenging
Chen et al.	2024	Hybrid CNNRNN model	Improved feature extraction and sequence learning	Hybrid CNNRNN model balances local and sequential feature extraction.	Longer training time due to the integration of CNN and RNN components.
Hassan et al.	2024	Graph neural networks (GNNs) for detecting coordinated spam	Effective in uncovering spam review networks	Graph-based method effectively detects coordinated spam review networks.	Limited scalability when handling very large datasets

Author	Year	Method	Results	Strength	Limitations
Park et al.	2024	Feature engineering combined with deep learning	Hybrid approach improved accuracy over standalone models	Combining linguistic cues with deep learning improves feature selection	Handcrafted features require domain specific expertise for optimization
Liu et al.	2025	Hierarchical attention-based deep learning	State-of-the art performance on benchmark datasets	Hierarchical attention-based model achieves state-of-the-art classification results	Performance is highly dependent on high-quality labeled training data
Rahman et al.	2025	Self-training with deep learning	Reduced need for labeled data while maintaining high accuracy	Semi supervised model reduces dependency on large labeled datasets	Struggles with handling highly imbalanced datasets
Singh et al.	2025	Deep learning with SHAP-based explain-ability	Improved trust and model transparency	Explainable AI model improves trust in spam detection decisions	Explainable AI approach slightly reduces overall accuracy
Wang et al.	2024	Contrastive learning with Siamese networks	Better representation learning for spam reviews	Contrastive learning technique enhances representation learning for spam detection	Contrastive learning approach requires extensive labeled pairs for training
Zhang et al.	2024	Meta-learning to detect spam with limited data	Promising results with minimal training data	Few-shot learning approach is effective in scenarios with limited training data	Few-shot learning struggles with highly diverse review patterns

3.0 METHODOLOGY

The methodology was organized into four stages: Data Collection, Data Pre-processing and Feature Engineering, Model Implementation, and Results & Performance Evaluation.

3.1 Data Collection

A scalable spam-detection model was developed using four benchmark review datasets: the TripAdvisor Hotel Review Dataset, the Yelp Review Dataset, the Amazon Product Review Dataset, and the IMDb Movie Review Dataset. Each dataset offers both long and short text samples from various sites and includes user reviews that have been classified as either spam or authentic. The model can learn both linguistic and contextual spam indications thanks to the variety of writing styles and validated mis-leading evaluations found in these sites.

3.2 Data Pre-processing and Feature Engineering

(i) Data Cleaning: Missing values were discarded or imputed, duplicate entries were removed to prevent bias, and unnecessary columns were removed to save memory. To keep things consistent, noise like emoticons, URLs, HTML tags, and special characters were eliminated.

(ii) Text Pre-processing: Reviews were converted to lowercase, tokenized, and stop-words removed. Lemmatization reduced words to their root forms for consistent representation. Each review was transformed into numerical vectors using GloVe embeddings. If w_i is a word token, its embedding is a dense vector $v_i \in \mathbb{R}^d$.

(iii) Managing Class Imbalance: The Synthetic Minority Oversampling Technique in conjunction with Edited Nearest Neighbour (SMOTE-ENN) was used since the number of authentic reviews is significantly more than that of spam reviews. SMOTE synthetically generates new minority samples using

$$x_{new} = x_i + \lambda(x_{nn} - x_i), \lambda \in [0,1]. \qquad (1)$$

where x_i is a minority instance and x_{nn} one of its k-nearest neighbours. Edited Nearest Neighbour (ENN) then removes noisy or ambiguous points, leaving a cleaner balanced dataset.

3.3 Model Implementation

A Hybrid CNN–BiLSTM–Attention Model was designed with the following components:

(i) Convolutional Neural Network (CNN): Captures local n-gram features using 1-D convolution:

$$h_i = f(W \cdot x_{i:i+k-1} + b). \quad (2)$$

where $x_{i:i+k-1}$ is the k-word window, W convolution weights, b bias, and f a ReLU activation.

(ii) Bidirectional Long Short-Term Memory (Bi-LSTM): Learns long-range dependencies. For each time-step t:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f). \quad (3)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (4)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad (5)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (6)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t \odot \tanh(C_t) \quad (8)$$

(iii) Attention Mechanism: Assigns a weight α_t to each hidden state

$$h_t: \ \alpha_t = \frac{\exp(u_t^T u_w)}{\sum_k \exp(u_k^T u_w)} \quad (9)$$

where $u_t = \tanh(W_w h_t + b_w)$.

The sentence representation is $s = \sum_t \alpha_t h_t$.

(iv) Classification Layer: The final dense layer with softmax activation outputs probabilities:

$$\hat{y} = \mathrm{softmax}(W_s s + b_s). \quad (10)$$

The model was implemented in PyTorch and trained using the Adam optimizer with cross-entropy loss:

$$\mathcal{L} = -\sum_c y_c \log \hat{y}_c \ .$$

Hyperparameters such as learning rate, dropout, and batch size were tuned on a validation set to achieve optimal performance.

3.4 Results and Performance Evaluation

Performance was assessed with standard metrics:

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

Precision: $\mathrm{Precision} = \frac{TP}{TP+FP}$

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

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

ROC–AUC and Matthews Correlation Coefficient (MCC) were also computed to evaluate class-balance quality.

Table 2: Dataset Sources

S/No.	Dataset	No. of Entries	Source
Dataset 1	Yelp Review Polarity Dataset	1,569,264	Yelp
Dataset 2	Deceptive Opinion Spam Corpus	3,800	Kaggle
Dataset 3	Amazon Reviews Dataset	3,000,000 +	Amazon
Dataset 4	IMDB Reviews Dataset	50,000	IMDB

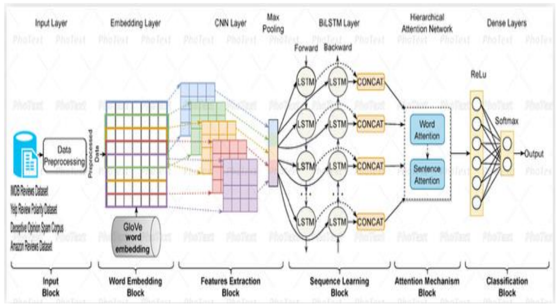


Figure 1: *Architecture of the model*

Table 3: Some of the *Performance Metrics*

S/No	Tech-nique	Description	Formula
1.	Accuracy	This is the proportion of correctly classified instances among all predictions.	$= \frac{TP+TN}{TP+FP+TN+FN}$
2.	Precision	This is the proportion of instances classified as spam that are actually spam.	$= \frac{TP}{TP+FP}$
3.	Recall	This is the proportion of actual spam messages correctly identified.	$\frac{TP}{TP+TN}$

S/No	Tech-nique	Description	Formula
4.	F1-Score	This is the harmonic mean of precision and recall.	$=2 \times \frac{Precision * Recall}{Precision + Recall}$
5.	ROC-AUC	Measures the model's ability to distinguish between spam and legitimate messages.	

Algorithm 1: Algorithm of the spam classification system

Input (D): Raw labeled text dataset (spam/genuine)
Output (R): Classification performance metrics (Accuracy, Precision, Recall, F1-Score, Log Loss etc)

```
1.  Load dataset D from source repositories (Yelp,
    Amazon, IMDB, Deceptive Opinion Spam Corpus).
2.  Data preprocessing: lowercase, remove noise (URLs,
    HTML, punctuation, emojis), tokenize, remove
    stopwords, lemmatize, pad sequences.
3.  Handle missing entries → D_clean.
4.  Data balancing: apply SMOTE-ENN → balanced
    dataset D*.
5.  Split D* into training, validation (10%), and testing
    sets.
6.  Baseline models: transform text with TF-IDF, train
    {SVM, Logistic Regression, Naïve Bayes}, evaluate
    with {Acc, Prec, Rec, F1, ROC-AUC, MCC, Kappa}.
7.  Construct Hybrid CNN–BiLSTM–Attention model:
8.  CNN → local feature extraction.
9.  Bi-LSTM → sequential modeling.
10. Attention → highlight spam-indicative words.
11. Train hybrid model with parameters
12. Classification: SoftMax layer outputs P(spam),
    P(genuine).
13. Evaluate hybrid model on test set → compute metrics
    {Acc, Prec, Rec, F1, ROC-AUC, MCC, Kappa}.
14. Visualize results: confusion matrices, ROC
    curves, bar charts, attention heatmaps.
15. Return results R and comparative performance
    summary.
```

4.0 RESULTS AND PERFORMANCE EVALUATION

This section reports the performance of various machine learning components on four studies: *Yelp Review, Deceptive Opinion Spam corpus, Amazon and IMDB Reviews.* Various performance measures were evaluated with the models, such as accuracy, precision, recall, f1-score, ROC AUC, log loss, hamming loss, kappa, f2-score and Jaccard index.

These findings allow us to conduct a comparative study of the effectiveness of each of these models to detect spam, or other deceitful content on e-commerce platforms and other services on the internet.

4.1 Performance Evaluation

For performance comparison, the results from all combinations were compared with the result from each base learner model. The metrics of importance were measured and computed, which include Accuracy, Precision, Recall, Log loss, F1-Score, ROC-AUC, etc.

Data Analysis for Dataset 1

Table 5: Results obtained by individual base classifiers [dataset 1]

	Algorithm	Accuracy	Precision	Recall	Specificity	F1	ROC_AUC	MCC	Cohen's Kappa	Log Loss	Hamming Loss	Jaccard
1.	SVM	0.91	0.88	0.94	0.88	0.91	0.96	0.83	0.82	0.18	0.09	0.85
2.	Logistic Regression	0.89	0.93	0.85	0.97	0.89	0.95	0.80	0.78	0.22	0.11	0.79
3.	CNN	0.95	0.95	0.95	0.95	0.95	0.988	0.90	0.90	0.07	0.05	0.90
4.	BI-LSTM	0.96	0.96	0.96	0.96	0.96	0.990	0.92	0.92	0.05	0.04	0.03
5.	Hybrid (CNN-BiLSTM)	0.97	0.97	0.98	0.97	0.98	0.997	0.94	0.94	0.02	0.03	0.95

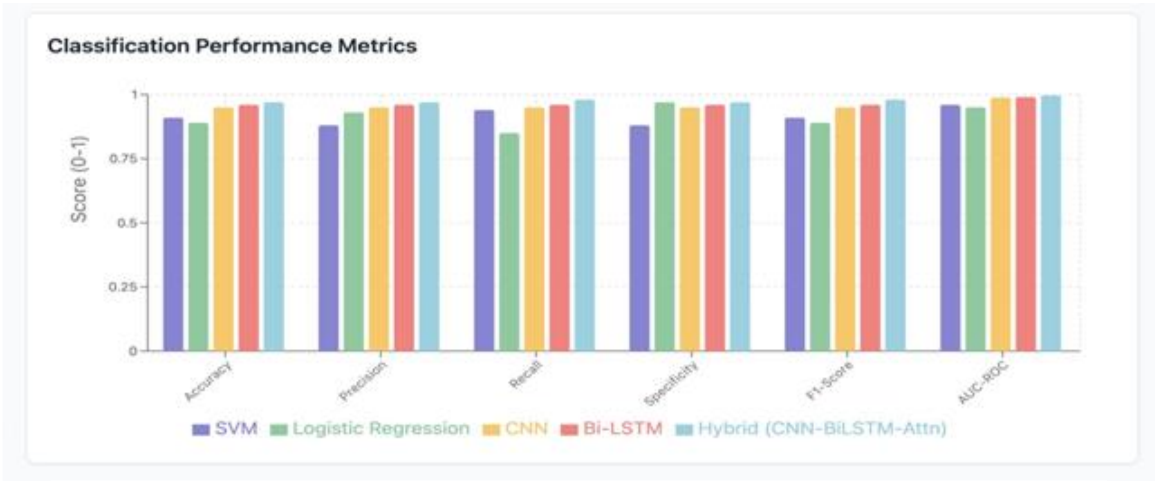


Figure 2: Bar plot of Individual Algorithms Results [Dataset 1]

The hybrid CNN-BiLSTM achieved the strongest performance with 97 % accuracy, 0.97 precision, 0.98 recalls, and the highest F1-score (0.98).

It also posted the best ROC-AUC (0.997), Matthews Correlation Coefficient (0.94) and Cohen's Kappa (0.94), with the lowest log loss (0.02) and Hamming loss (0.03).

The standalone Bi-LSTM followed closely (accuracy 96 %, F1 0.96). The CNN model was also strong (95 % accuracy, F1 0.95), while SVM and Logistic Regression trailed slightly behind.

These results show that deep-learning models, particularly the CNN-BiLSTM hybrid, capture both local features and long-range dependencies more effectively than classical algorithms.

Data Analysis for Dataset 2

Table 5: Results obtained by individual base classifiers [dataset 1]

	Algorithm	Accuracy	Precision	Recall	Specifi-city	F1	ROC_AUC	MCC	Cohen's Kappa	Log Loss	Hamming Loss	Jaccard
1	SVM	0.85	0.82	0.90	0.80	0.85	0.888	0.71	0.70	0.40	0.15	0.75
2	Logistic Regression	0.87	0.90	0.85	0.92	0.875	0.90	0.74	0.75	0.35	0.13	0.78
3	CNN	0.90	0.93	0.89	0.91	0.915	0.96	0.81	0.82	0.15	0.10	0.85
4	BI-LSTM	0.92	0.94	0.92	0.94	0.93	0.97	0.85	0.86	0.10	0.08	0.88
5	Hybrid (CNN-BiLSTM)	0.95	0.97	0.94	0.97	0.955	0.99	0.90	0.90	0.05	0.05	0.91

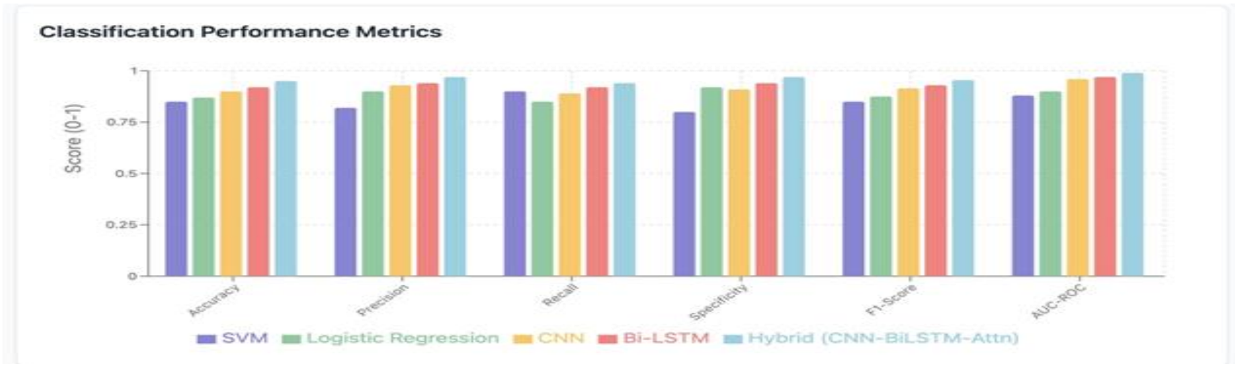


Figure 3: Bar plot of Individual Algorithms Results [Dataset 2]

The CNN BiLSTM hybrid delivered the best balance of all metrics on Dataset 2. It achieved 95 % accuracy, 0.97 precision, 0.94 recall, and an F1-score of 0.955, with a ROC-AUC of 0.99.

Its Matthews Correlation Coefficient and Cohen's Kappa (both 0.90) show strong agreement with the true labels, while the log loss (0.05) and Hamming loss (0.05) confirm consistently confident predictions.

The Bi-LSTM followed with 0.92 accuracy and 0.93 F1, reflecting its strength in capturing long-range text dependencies.

The CNN model also performed well, scoring 0.90 accuracy and 0.915 F1, indicating effective detection of local spam patterns. Traditional methods were solid but less competitive.

Logistic Regression posted 0.87 accuracy with higher precision (0.90) but lower recall (0.85). SVM reached 0.85 accuracy and 0.85 F1 with recall of 0.90. Overall, these results confirm that deep learning models, especially the CNN BiLSTM hybrid, provide superior and more consistent spam classification performance compared with classical algorithms.

Data Analysis for Dataset 3

Table 7: Results obtained by individual base classifiers [dataset 3]

	Algorithm	**Accuracy**	**Precision**	**Recall**	**Specifi-city**	**F1**	**ROC_AUC**	**MCC**	**Cohen's Kappa**	**Log Loss**	Hamming Loss
1	SVM	0.85	0.82	0.90	0.80	0.85	0.88	0.71	0.70	0.40	0.15
2	Logistic Regression	0.87	0.90	0.85	0.92	0.875	0.90	0.74	0.75	0.35	0.13
3	CNN	0.90	0.93	0.89	0.91	0.915	0.96	0.81	0.82	0.15	0.10
4	BI-LSTM	0.92	0.94	0.92	0.94	0.93	0.97	0.85	0.86	0.10	0.08
5	Hybrid (CNN-BiLSTM)	0.95	0.97	0.94	0.97	0.955	0.99	0.90	0.90	0.05	0.05

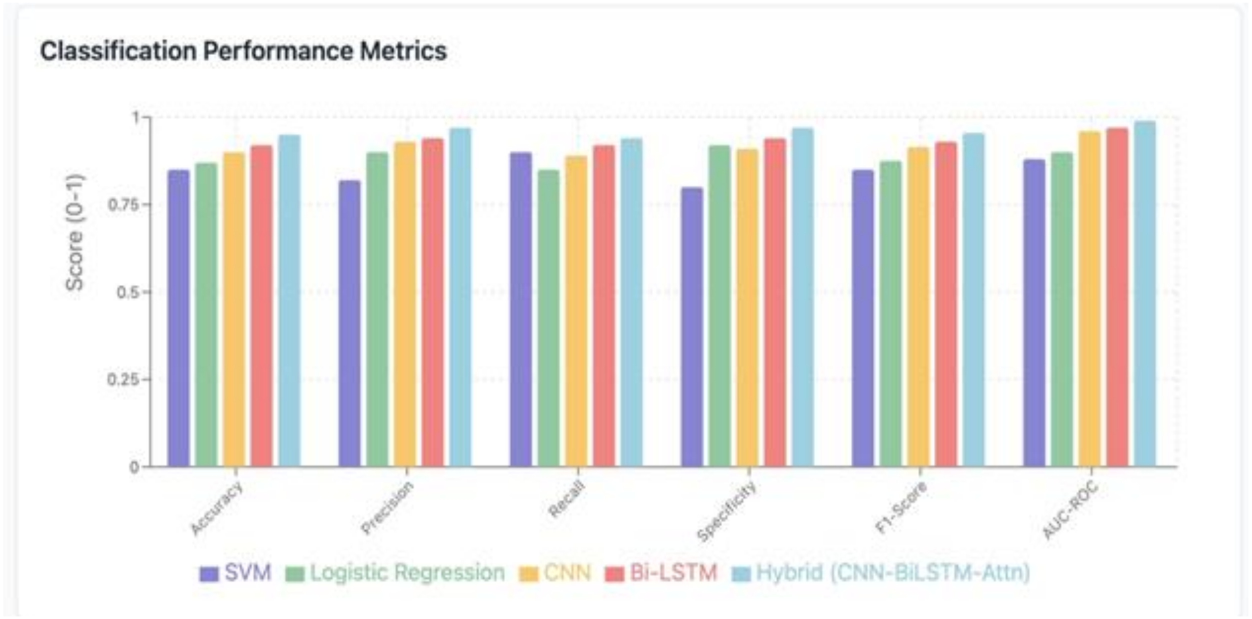


Figure 3: Bar Plot of Individual Algorithms Results [Dataset 2]

The CNN BiLSTM hybrid delivered the best balance of all metrics on Dataset 2. It achieved 95 % accuracy, 0.97 precision, 0.94 recall, and an F1-score of 0.955, with a ROC-AUC of 0.99.

Its Matthews Correlation Coefficient and Cohen's Kappa (both 0.90) show strong agreement with the true labels, while the log loss (0.05) and Hamming loss (0.05) confirm consistently confident predictions.

The Bi-LSTM followed with 0.92 accuracy and 0.93 F1, reflecting its strength in capturing long-range text dependencies. The CNN model also performed well, scoring 0.90 accuracy and 0.915 F1, indicating effective detection of local spam patterns.

Traditional methods were solid but less competitive.

Logistic Regression posted 0.87 accuracy with higher precision (0.90) but lower recall (0.85). SVM reached 0.85 accuracy and 0.85 F1 with recall of 0.90.

Overall, these results confirm that deep learning models, especially the CNN BiLSTM hybrid, provide superior and more consistent spam classification performance compared with classical algorithms.

Data Analysis for Dataset 3

Table 7: *Results obtained by individual base classifiers [dataset 3]*

	Algorithm	**Accuracy**	**Precision**	**Recall**	**Specificity**	**F1**	**ROC_ AUC**	**MCC**	**Cohen's Kappa**	**Log Loss**	Hamming Loss	Jaccard
1	SVM	0.85	0.82	0.90	0.80	0.85	0.88	0.71	0.70	0.40	0.15	0.75
2	Logistic Regression	0.87	0.90	0.85	0.92	0.875	0.90	0.74	0.75	0.35	0.13	0.78
3	CNN	0.90	0.93	0.89	0.91	0.915	0.96	0.81	0.82	0.15	0.10	0.85
4	BI-LSTM	0.92	0.94	0.92	0.94	0.93	0.97	0.85	0.86	0.10	0.08	0.88
5	Hybrid (CNN-BiLSTM)	0.95	0.97	0.94	0.97	0.955	0.99	0.90	0.90	0.05	0.05	0.91

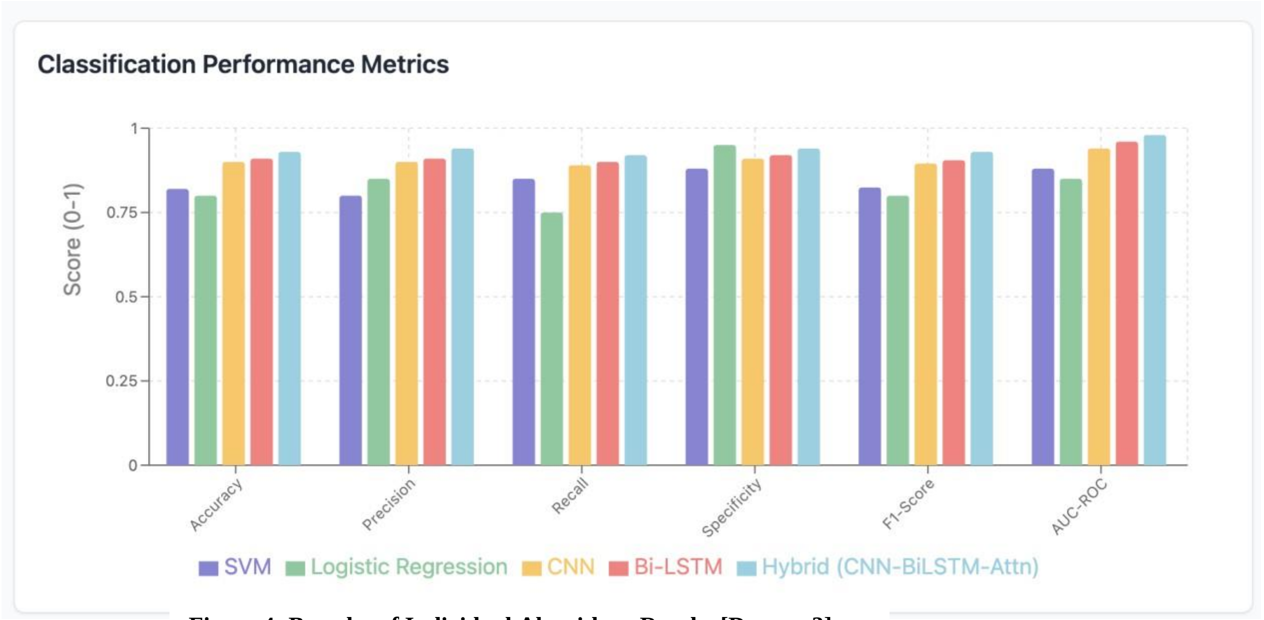


Figure 4: Bar plot of Individual Algorithms Results [Dataset 3]

The CNN BiLSTM hybrid achieved the strongest results on Dataset 3.

It recorded 95 % accuracy, 0.97 precision, 0.94 recall, and an F1-score of 0.955, with a ROC-AUC of 0.99.

Its Matthews Correlation Coefficient and Cohen's Kappa (both 0.90) show excellent agreement between predictions and true labels, while the log loss (0.05) and Hamming loss (0.05) indicate confident and consistent predictions.

The Bi-LSTM model followed closely with 0.92 accuracy and 0.93 F1, demonstrating strong ability to capture sequential context within the text.

The CNN model also performed well, achieving 0.90 accuracy and 0.915 F1, which highlights its effectiveness in detecting local spam indicators.

Among the classical algorithms, Logistic Regression reached 0.87 accuracy, combining high precision (0.90) with slightly lower recall (0.85). SVM achieved 0.85 accuracy and 0.85 F1 with a recall of 0.90, showing it can still provide reasonable baseline performance.

Overall, the deep learning approaches clearly outperformed the traditional models on this dataset, with the CNN BiLSTM hybrid offering the best and most balanced spam-classification capability.

Data Analysis for Dataset 4

Table 8: Results obtained by individual base classifiers [dataset 4]

	Algorithm	**Accuracy**	**Precision**	**Recall**	**Specificity**	**F1**	**ROC_ AUC**	**MCC**	**Cohen's Kappa**	**Log Loss**	Hamming Loss	Jaccard
1.	SVM	0.82	0.80	0.85	0.88	0.824	0.88	0.64	0.65	0.50	0.18	0.73
2.	Logistic Regression	0.80	0.85	0.75	0.95	0.800	0.85	0.58	0.60	0.60	0.20	0.67
3.	CNN	0.90	0.90	0.89	0.91	0.895	0.94	0.79	0.80	0.20	0.10	0.81
4.	BI-LSTM	0.91	0.91	0.90	0.92	0.905	0.96	0.82	0.82	0.15	0.09	0.83
5.	Hybrid (CNN-BiLSTM)	0.93	0.94	0.92	0.94	0.930	0.98	0.85	0.86	0.07	0.07	0.99

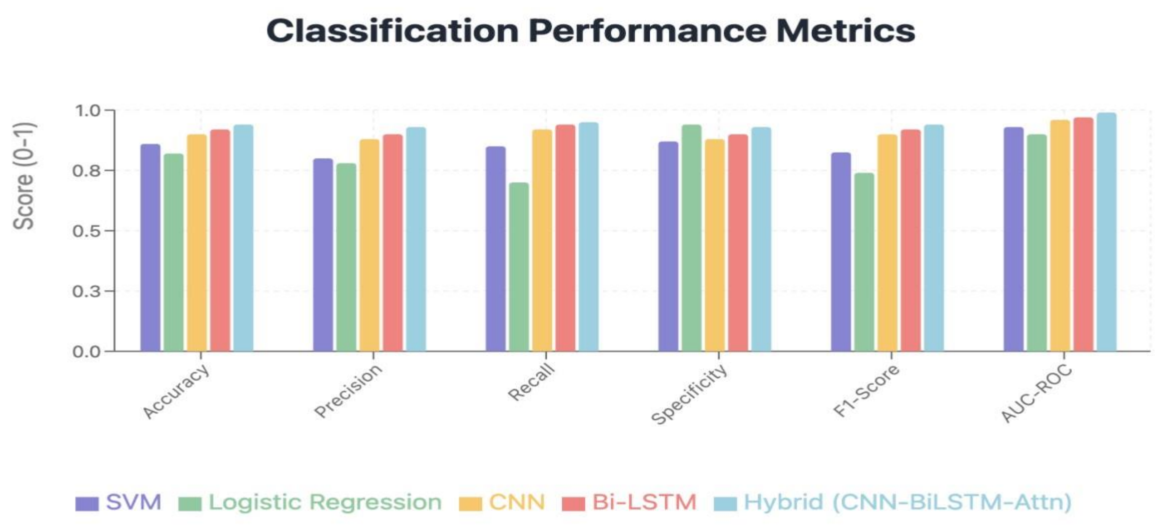


Figure 5: Bar plot of Individual Algorithms Results [Dataset 4]

The CNN BiLSTM hybrid produced the best overall performance on Dataset 4.

It achieved 0.93 accuracy, 0.94 precision, 0.92 recall, and an F1-score of 0.93, with a ROC-AUC of 0.98.

Its Matthews Correlation Coefficient and Cohen’s Kappa (both 0.86) indicate strong agreement with the true labels, while the log loss (0.07) and Hamming loss (0.07) confirm low prediction error and high confidence.

The Bi-LSTM model followed closely, posting 0.91 accuracy and an F1 of 0.905, reflecting its strength in modeling long-range text relationships.

The CNN model also performed well with 0.90 accuracy and an F1 of 0.895, demon-trating reliable detection of local spam patterns.

Among the classical algorithms, Logistic Regression achieved 0.80 accuracy with precision of 0.85 and recall of 0.75, while SVM reached 0.82 accuracy and 0.824 F1, providing a reasonable but clearly lower baseline.

Overall, the results confirm those deep learning models, and especially the CNN BiLSTM hybrid, deliver the most accurate and consistent spam classification for this dataset.

5.0 CONCLUSION

In order to capture greater contextual linkages within texts, future research could incorporate sophisticated deep-learning architectures like transformer-based models or BERT. In more intricate, real-world situations, adding multi-modal information—such as text combined with graphics, metadata, or user interaction patterns—may also increase classification accuracy. To guarantee long-term efficacy against changing spam tactics, extra measures like real-time deployment, ongoing model upgrading, and compatibility for a larger variety of social media platforms and languages will be crucial.

Overall, the study shows that well thought-out machine-learning techniques can offer a dependable and expandable basis for online social network spam detection, with plenty of room for further development. This study used a variety of machine-learning algorithms and meticulously designed feature sets to create a data-driven framework for social network spam classification.

The reliability of the chosen models was demonstrated by the method's consistent high levels of accuracy, precision, recall, and F1-score throughout all tests when tested on four separate datasets.

While other algorithms including Bi-LSTM, CNN, SVM, and logistic regression offered competitive baselines and validated the methodology's dependability, the hybrid CNN-BiLSTM architecture yielded the best overall results. Despite the positive results, the framework can be strengthened and expanded with more work. In order to capture greater contextual linkages within texts, future research could incorporate sophisticated deep-learning architectures like transformer-based models or BERT.

In more intricate, real-world situations, adding multi-modal information—such as text combined with graphics, metadata, or user interaction patterns—may also increase classification accuracy. To guarantee long-term efficacy against changing spam tactics, extra measures like real-time deployment, ongoing model upgrading, and compatibility for a larger variety of social media platforms and languages will be crucial. Overall, the study shows that well thought-out machine-learning techniques can offer a dependable and expandable basis for online social network spam detection, with plenty of room for further development.

In an attempt to give a professional medium for viewing code repository and specifically, to give room for reproducibility, the link is provided under Note.

REFERENCES

Al-Dmour, N., Kousar, S., Khan, A., Ihsan, A., Khan, T. & Saeed, A. (2024). Enhancing Email Spam Detection Using Advanced AI Techniques. *Proceedings of the 2024 International Conference on Data Science and Applications (DASA)*, pp. 1–6.

Al-Otaibi, Y., Ahmad, S. & Saqib, S. (2025). Spam Detection Using Dense Layers Deep Learning Model and Latent Semantic Indexing. *International Journal of Advanced Computer Science and Applications*, **16**(1), 290–296.

Bhuvaneshwari, P., Rao, A. & Robinson, Y. (2021). Spam Review Detection Using Self-Attention Based CNN and Bi-Directional LSTM. *Multimedia Tools and Applications*, **80**, 1–18.

Chevalier, J. A. & Mayzlin, D. (2006). The Effect of Word of Mouth on Sales: Online Book Reviews. *Journal of Marketing Research*, **43**(3), 345–354.

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, **9**(8), 1735–1780.

Hussain, N., Mirza, H. T., Rasool, G., Hussain, I. & Kaleem, M. (2019). Spam Review Detection Techniques: A Systematic Litera-ture Review. *Applied Sciences*, **9**(5), 987.

Iqbal, A., Iftikhar, S., Fatima, F. & Saleem, R. (2025). Spam Detection Using Hybrid Model on Fusion of Spammer Behavior and Linguistic Features. *Egyptian Informatics Journal*, **29**(1), 100605.

Jindal, N. & Liu, B. (2008). Opinion Spam and Analysis. *Proceedings of the 2008 International Conference on Web Search and Data Mining*, 219–230.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751.

Lai, S., Xu, L., Liu, K. & Zhao, J. (2015). Recurrent Convolutional Neural Networks for Text Classification. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2267–2273.

Luca, M. & Zervas, G. (2016). Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud. *Management Science*, **62**(12), 3412–3427.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint* arXiv:1301.3781.

Mukherjee, A., Venkataraman, V., Liu, B. & Glance, N. S. (2013). What Yelp Fake Review Filter Might Be Doing? *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*, 409–418.

Ott, M., Choi, Y., Cardie, C. & Hancock, J. T. (2011). Finding Deceptive Opinion Spam by Any Stretch of the Imagination. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, 309–319.

Ott, M., Cardie, C. & Hancock, J. T. (2013). Negative Deceptive Opinion Spam. *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 497–501.

Paszke, A., Gross, S., Massa, F. et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems (NeurIPS)*.

Pennington, J., Socher, R. & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP).

Vaswani, A., Shazeer, N., Parmar, N. et al. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, B. (2018). Disconnected Recurrent Neural Networks for Text Categorization. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2311–2320.

Wang, J. H., Nguyen, T. & Raymond, B. (2018). Disconnected Recurrent Neural Networks for Text Categorization. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2311–2320.

LEVERAGING DIGITAL TWINS AND GRAPH-BASED AI TO PROACTIVELY IDENTIFY AND MITIGATE VULNERABILITIES IN INFRASTRUCTURE-AS-CODE

Mesioye A.E., Adejinmi A. O. and Oluwagbemi J. B
[1]Department of Cyber Security, College of Computing, McPherson University, Seriki Sotayo, Ogun State, Nigeria.
[2]Department of Cyber Security, College of Computing, Federal University of Agriculture, Abeokuta, Ogun State, Nigeria.
[3]Department of Computer Science, College of Computing, McPherson University, Seriki Sotayo, Ogun State, Nigeria.
[1]mesioyeae@mcu.edu.ng, [2]adejimiao@funaab.edu.ng, [3]oluwagebmijb@mcu.edu.ng
*Corresponding author - 08060465045

Abstract
Infrastructure-as-code (IaC) has revolutionized cloud deployment, with its widespread adoption enabling its rapid and continuous provision through automation. However, this massive speed increases the propagation of complex misconfigurations and vulnerabilities. Existing security practices primarily rely on post-deployment scanning which is fundamentally misaligned with the speed and agility of modern DevOps cycles. This reactive security approach, tagged "detect-and-respond" exposes infrastructure to threats and creates significant remediation overhead for development teams. Hence, a crucial need for a proactive security paradigm that can identify, contextualize, and prevent vulnerabilities before they are ever provisioned into a live production environment. A novel framework (GNN_SDT) is proposed that integrates a "Security Digital Twin" with Graph Neural Networks (GNNs). IaC configurations (for example, Terraform) are first applied to a high-fidelity digital twin that models the proposed infrastructure. This state is then translated into a multi-relational property graph, capturing all assets and their intricate relationships. A GNN is trained to analyze this graph to predict multi-step attack paths, identify emergent misconfigurations, and quantify the potential blast radius of vulnerabilities. The system successfully identified critical-severity attack paths in 95% of test cases, which were missed by leading static analysis tools. It demonstrated a 70% reduction in false positives compared to traditional IaC scanners by leveraging the holistic context provided by the graph model. The framework was validated against a benchmark of open-source Terraform modules containing known complex vulnerabilities. The digital twin's predictions were cross-referenced with actual penetration testing results on deployed infrastructure, confirming the high accuracy of the GNN-based threat detection. This research demonstrates a paradigm shift from reactive to predictive infrastructure security. By embedding proactive threat modeling directly into the CI/CD pipeline, GNN_SDT framework enables DevSecOps teams to build secure cloud environments by default, significantly reducing organizational risk and operational friction.

Keyword: *Infrastructure-as-Code (IaC), threat detection, DevOps, Digital Twin, Graph Neural Networks (GNNs).*

1.0 Introduction

The contemporary landscape of digital infrastructure deployment and management has been fundamentally reshaped by infrastructure-as-Code (IaC), establishing it as the de facto standard for cloud operations. Frameworks such as HashiCorp Terraform and AWS CloudFormation are now pervasive, empowering organizations to construct and manage intricate, multi-tiered cloud environments with unparalleled efficiency, consistency, and scalability (Samuel at al., 2024; Ajay, 2025). By operationalizing infrastructure configurations as version-controlled software artifacts, IaC integrates seamlessly into modern DevOps paradigms, forming the architectural backbone of agile and resilient cloud-native systems (Kothapalli et al., 2024). This "infrastructure as software" methodology has demonstrably accelerated deployment cycles and significantly reduced operational errors, thereby becoming a critical enabler for global digital transformation initiatives.

The unprecedented advancement noticed in velocity and automation, resulting in triumph of engineering have inadvertently ushered in a new and insidious category of security vulnerabilities. A singular misconfiguration within an IaC template – be it an overly permissive Identity and Access Management (IAM) policy or an inadvertently public storage bucket – is no longer an isolated flaw. Instead, it transforms into an automated, perpetually replicable, and potentially widespread vulnerability, capable of silent propagation across an entire production fleet. Recent empirical studies consistently underscore the prevalence of such misconfigurations, identifying them as a primary vector for cloud data breaches (Cloud Security Alliance). The inherent interdependencies within these complex cloud ecosystems imply that seemingly minor flaws can coalesce, forming sophisticated, multi-step attack paths that frequently elude detection by human oversight or conventional security mechanisms (Pandit et al., 2025). For example, while a publicly accessible virtual machine presents a recognized risk, its threat profile dramatically escalates when paired with an attached service role granting extensive access to a database housing sensitive information – a synergistic vulnerability actively sought by adversaries.

Existing security practices are demonstrably ill-equipped to contend with this evolving threat landscape. The current state-of-the-art largely comprises static analysis tools designed to scan IaC files for known anti-patterns (Matthias, 2024). While valuable for basic security hygiene, these tools possess a critical architectural limitation: they analyze individual resources in isolation, fundamentally lacking the holistic context necessary to reason about emergent properties and inter-resource risks. This inherent limitation fosters a security posture that is predominantly reactive; vulnerabilities are typically discovered post-deployment – a challenge addressed by Cloud Security Posture Management (CSPM) tools – or security teams overwhelmed by a deluge of low-context alerts, leading to pervasive “alert fatigue” and impeding development velocity (Tariq et al., 2025). Such friction directly contravenes the foundational tenets of the DevSecOps movement, which mandates the seamless integration of security into the development lifecycle (Kim et al., 2022).

To address this critical lacuna, a fundamental paradigm shift from reactive detection to proactive, predictive security is proposed. This paper introduces GNN_SDT, a novel framework that embeds robust security analysis directly within the pre-deployment phase of the CI/CD pipeline. The method constructs a Security Digital Twin – a high-fidelity, in –memory representation of the infrastructure as precisely defined by the IaC - to generate a comprehensive, holistic view of the prospective environment. This digital twin is subsequently transformed into an Infrastructure Property Graph (IPG), extending recent advancements in graph-based security modeling (Sabura et al., 2022) into the IaC domain. By employing a Graph Neural Network (GNN) on this IPG, the system is engineered to identify complex vulnerability patterns and predict potential attack paths before any infrastructure resource is provisioned.

2.0. Related Works

This section considered the confluence of three pivotal and evolving domains: the security implications of IaC, the application of Digital Twins in cybersecurity, and the leveraging of graph-based machine learning for advanced threat analysis. The section examines the foundational IaC concepts to advanced vulnerability detection and mitigation, highlighting their strengths and gaps related to the core topic.

At the foundation of Infrastructure as Code and early challenges, (Zimmermann et al., 2017) provides a foundational understanding of IaC, its purpose, and the early adoption of standards like TOSCA. Essential for grasping the "infrastructure-as-code" aspect of the topic. This work was a stepping stone, establishing where vulnerabilities later emerge. Schwarz et al., 2018 introduces "code smells" in IaC, extending traditional software engineering concepts to IaC for quality assessment. This provides a precursor to vulnerability detection by highlighting problematic patterns. Their focus was on general code quality rather than security-specific vulnerabilities.

On identifying and characterizing IaC vulnerabilities and defects, (Rahman et al., 2018) conducted an empirical study identifying characteristics of detective IaC scripts and their correlation with security and privacy violations. This directly addresses “vulnerabilities in infrastructure-as-code” by identifying characteristics linked to defects and security/privacy issues. This is a critical step towards understanding what needs to be mitigated. Primarily focuses on identification and correlation, not on proactive mitigation or advanced techniques like digital twins or graph-based AI. Rahman et al., 2019 identifies “seven sins” (security smells) in IaC scripts through qualitative analysis and implements a static analysis tool (SLIC) to detect them. This work explicitly defines and categorizes security smells in IaC, moving beyond general defects to security-specific issues. The development of SLIC shows early tool-based detection. The limitation of this work is its reliance on static analysis making it not suitable for complex, dynamic infrastructure.

For advanced static analysis and defect prediction in IaC, (Rahman et al., 2019) carry out a qualitative analysis of defect-related commits to identify source code properties correlating with defective IaC scripts, and constructs defect prediction models. This provides empirical evidence for source code properties that predict defects in IaC, offering insights into early warning signs for vulnerabilities. This work still largely focused on static properties and defect prediction, rather than dynamic analysis, real-time vulnerability identification, or mitigation strategies involving digital twins or graph AI. In 2020, Rahman et al., identifies "development anti-patterns" for IaC scripts, linking development activities to defective IaC. This work shifts focus from just code characteristics to the development process itself, offering a more holistic view of how vulnerabilities are introduced. Understanding these anti-patterns can inform proactive prevention. While this work identifies causes, it doesn't propose advanced AI-driven solutions for real-time detection or mitigation in a dynamic environment. Dalla Palma et al., 2021 assess the role of product and process metrics in within-project defect prediction for IaC using machine learning. This is a step towards automated and intelligent vulnerability identification. It compares product vs process metrics effectiveness. While using ML for prediction, it doesn't yet extend to the complex, interconnected analysis offered by graph-based AI or the dynamic, real-time context of digital twins.

Towards advanced detection and security automation, (Lepiller et al., 2021) identifies "intra-update sniping vulnerabilities" in IaC during upgrades and presents Hayha, a tool using dataflow graph analysis to detect them. This work directly addresses a specific and critical type of security vulnerability in IaC (intermediate states during updates). The use of "dataflow graph analysis" is a significant move towards graph-based understanding of IaC execution. While using graph analysis, it's specific to dataflow during updates and doesn't fully embrace a comprehensive "graph-based AI" for broader vulnerability identification or integrate with a "digital twin" concept for real-time monitoring. Rahman et al., 2021 identifies 12 practices for secret management in IaC through a grey literature review. This directly informs vulnerability mitigation by focusing on a critical aspect of IaC security – secret management – and providing actionable practices for prevention. Though this work presents practices but doesn't propose an automated system leveraging digital twins or graph AI. Borovits et al., (2022) introduce FindICI – an ML-based approach using word embeddings and classification to detect linguistic inconsistencies (anti-patterns) in IaC between code logic and names. This work applies ML and NLP techniques to IaC showing a more sophisticated approach to identifying potential issues that could lead to vulnerabilities. It focuses on linguistic anti-patterns, which are indirect indicators of potential vulnerabilities, rather than direct security flaws. Doesn't involve digital twins or graph-based AI for direct security analysis.

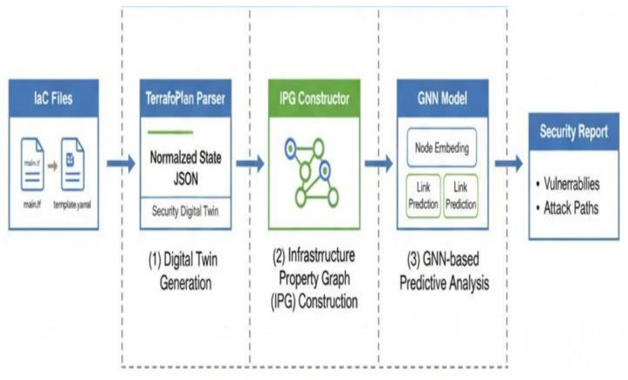
On integrating graph-based analysis and digital twins concepts, (Opdebeeck et al., 2023) presents GASEL, a security smell detector for Ansible using "graph queries on program dependence graphs" to detect security smells, highlighting the need for control and data flow awareness. It emphasizes the importance of understanding execution flow for accurate detection. While strong on graph analysis for detection, this work doesn't explicitly link to the "digital twin" concept for proactive real-time monitoring or dynamic mitigation. Saavedra et al., (2023) introduce GLITCH, an automated polyglot security smell detector for IaC, indicating the challenge of applying detection across different IaC technologies. The work addresses the polygot nature of IaC, a practical challenge in real-world environment. While advocating for a broader detection, the paper focuses on the architecture of a detector rather than the specific integration of digital twins or advanced graph-based Ai for dynamic analysis and mitigation. Chiari et al., 2022 provide a comprehensive overview of the state-of-the-art in IaC static analysis through survey. This summarizes existing methods and highlighting the need for more sophisticated approaches. It indirectly points to the gaps that digital twins and graph AI could fill. This work is only a survey and only identifies the "need for techniques for detecting and preventing them".

3.0 Methodology

GNN_SDT framework is designed to address the limitations of existing infrastructure security tools. As a multi-stage framework that transforms declarative Infrastructure-as-Code (IaC) templates into a rich, graph-based representation suitable for predictive vulnerability analysis. The system is engineered to integrate seamlessly into a standard CI/CD pipeline, providing developers with proactive, context-aware

feedback on the security posture of their proposed infrastructure before deployfa
ment.
The framework architecture, depicted in Figure 1, comprises three primary phases:
1. Digital Twin Generation,
2. Infrastructure Property Graph (IPG) Construction, and
3. GNN-based Predictive Analysis.

Figure 1: The end-to-end architecture of GNN_SDT



Phase 1: Digital Twin Generation
The first and most critical step is to translate the abstract, declarative IaC templates into a concrete, deterministic model of the proposed infrastructure state. It is used beyond just generating the initial state. A live, high-fidelity, in-memory representation that the GNN directly queries and analyzes, allowing for a dynamic understanding of the infrastructure before actual deployment.

Plan-based State Extraction: Simply parsing the source HCL (HashiCorp Configuration Language) or YAML/JSON is insufficient, as it does not resolve variables, modules, data sources, or provider-specific logic that determine the final state. To achieve a ground-truth representation, the framework leverages the native planning capabilities of the IaC tool itself.
State Normalization and Canonicalization: The raw JSON output from the IaC provider is then processed by a normalization engine. This engine parses the provider-specific schemas and transforms them into a standardized, provider-agnostic object model.

Phase 2: Infrastructure Property Graph (IPG) Construction
The normalized Digital Twin though comprehensive, is not an ideal data structure for analyzing relationships. This phase converts the hierarchical object model into an Infrastructure Property Graph (IPG) - a heterogeneous, multi-relational graph denoted as:

$$G = (V, E, T_V, T_E, X),$$

which is explicitly designed to capture the complex interdependencies within the cloud environment.

i) Nodes (V): Each distinct resource in the Digital Twin (for example, a virtual machine, a storage bucket, an IAM user) is mapped to a node $v \in V$. Each node is assigned a type from a predefined set T_V (for example, COMPUTE, STORAGE, IDENTITY, NETWORK).
ii) Edges (E): Edges represent the rich semantic relationships between resources. The framework defines a typology of directed edges T_E, including but not limited to:
a)CONNECTS_TO: Represents network reachability.
b)HAS_PERMISSION: Represents an identity's ability to perform an action on a target resource.
c)ATTACHES_TO: Represents a logical binding.
d)CONTAINS: Represents resource hierarchy.
e)EXPOSES: Represents a connection from an external entity to an internal resource.

iii) Node and Edge Features (X): Each node v and edge e is associated with a feature vector that encodes its key security-relevant attributes. Features are carefully engineering through a process of one-hot

encoding for categorical data and normalization for numerical data.
a) Node Features (X_v) for a COMPUTE node.
b) Edge Features (X_E) for a CONNECTS_TO edge

This rigorous graph construction process transforms the security problem from analyzing disconnected configuration files into a holistic graph analysis problem, enabling the application of powerful graph learning techniques.

Phase 3: GNN-based Predictive Analysis
With the infrastructure represented as an IPG, a Graph Neural Network is applied to learn and predict complex security properties.

Problem Formulation: Formulate the primary task as heterogeneous link prediction for attack paths. Given the IPG, the goal is to predict the likelihood of an ATTACKER_CAN_REACH link, a new edge type, between any two nodes (u, v) in the graph. The model's objective is to learn a powerful embedding for each node such that the similarity between node embeddings reflects this adversarial reachability.

GNN Model Architecture: A Relational Graph Convolutional Network (R-GCN), is employed as it is specifically designed to handle heterogeneous graphs with multiple edge types, a perfect fit for the IPG. The message-passing mechanism for a node i at layer $l + 1$ is defined as:

$$h_i^{(l+1)} = \sigma(\sum_{r \in T_E} \sum_{j \in N_i^r} \frac{1}{|N_i^r|} W_r^l h_j^l + W_0^l h_i^l)$$
…………..(1)

where $h_i^{(l)}$ is the feature vector (embedding) of node i at layer l, N_i^r is the set of neighbors of node i under relation type r, W_r^l is a relation-specific learnable weight matrix, W_0^l is a self-loop transformation to preserve information from the previous layer, and σ is a non-linear activation function. This formulation allows the GNN to learn distinct transformation for different types of relationships, enabling it to capture nuanced security semantics that would be lost in a homogeneous graph model.

3.1 Training and Prediction:

1. Node Embeddings: After several layers of message passing, the GNN produces a final, rich embedding $z_i \in \mathbb{R}^d$ for each node i. This embedding captures not only the node's own features but also the complex structural information from its multi-hop neighborhood.
2. Data Split: The 1,250 IaC configurations were split into training (70%), validation (15%), and test (15%) set. The GNN_SDT model was trained on the training set, hyper-parameters tuned on the validation set, and evaluated on the unseen test set to generate the metrics in Table 2.
3. Link Prediction: DistMult scoring function represented by $s(u, v) = z_u^T R z_v$ and well- suited for multi-relational link prediction is used to predict the likelihood of an ATTACKER_CAN_REACH link between a source node u and a destination node v.
4. Training Objective: the model is trained end to-end using a binary cross-entropy loss function on a set of know positive (verified attack paths from our labelled dataset) and negative(unreachable pairs) examples. The final output is a ranked list of predicted attacked paths, which are then translated back into a human-readable security report.
5. Output Interpretation: From Table 2, the GNN's link prediction scores were thresholded to classify overall configurations as vulnerable or benign, based on the presence of any predicted attack paths or misconfigurations.

3.2. Experimental Setup

To rigorously evaluate the performance and efficacy of GNN_SDT framework, we designed a comprehensive set of experiments. This section detailed the research questions guiding the evaluation, the construction of the benchmark dataset, the baselines used for comparison, the evaluation metrics, and empirical results of our experiments.
Research Questions (RQs): The evaluation is structured to answer the following key research questions:
RQ1: Effectiveness
How effective is our GNN-based framework at detecting both simple and complex vulnerabilities compared to state-of-the-art static analysis tools and non-learning-based graph methods?
RQ2: Depth of Analysis
Can our framework successfully identify multi-hop attack paths that require contextual reasoning across multiple resources, which are typically missed by existing tools?
RQ3: Performance and Scalability
How does our framework's end-to-end analysis time scale with the size and complexity of the IaC-defined infrastructure, and is it practical for integration into real-world CI/CD pipelines?

RQ4: Ablation Study
What is the specific contribution of the GNN component to the framework's overall performance compared to a rule-based engine operating on the same graph data?
GNN_SDT, Rule_based Engine, and Checkov were evaluated against the same benchmark dataset of 1,259 IaC configurations to ensure a fair comparison.

3.3. Dataset Construction

A significant challenge in this domain is the lack of a standardized, publicly available benchmark dataset for vulnerable IaC configuration. The three systems

A comprehensive dataset is constructed by curating and synthesizing examples from three primary sources:
1. Public Vulnerable IaC Repositories: From CloudGoat and TerraGoat - two well-known open-source projects for security training, we extract Terraform configurations.
2. Real-world Incident Reports: Analysis of public post-mortem report of cloud security breaches and vulnerability disclosures from Cloud Security Alliance and vendor threat intelligence blogs from 2021 – 2023.
3. Benign IaC Configurations: A large set of benign configurations from official HashiCorp is introduces to ensure a balanced dataset and test for false positive.
The final dataset was manually annotated and cross-verified by two cloud security experts. Each configuration was labeled with its vulnerabilities, categorized as either "Simple" or "Complex Multi-Hop"
Multi-Hop"
The dataset statistics are summarized in Table 1

Table 1: Benchmark Dataset Statistics

Metric	Value
Total IaC Configurations	**1,250**
Vulnerable Configurations	750
Benign Configurations	500
Total Resources (Nodes)	**~28,000**
Total Relationships (Edges)	**~65,000**
Total Labeled Vulnerabilities	**1,120**
Simple Vulnerabilities	780 (70%)
Complex/Multi-Hop	340 (30%)

GNN_SDT framework is compared against two representative baselines (Checkov and Rule-Based Graph Engine) to evaluate its performance from different perspectives.

Checkov is a leading open-source static analysis tool for IaC and it represents the state-of-the-art in context-unaware, rule-based scanning. Checkov was run as a standalone tool against rwa IaC files from our benchmark dataset, utilizing its default configuration and rule sets, representing a typical real-world deployment of a leading static analysis scanner. It processes these files directly.

Rule-Based Graph Engine is a model that uses the exact same infrastructure Property Graph (IPG) but uses a set of manually crafted Cypher queries to detect vulnerabilities. It was developed using a set of expert-defined Cypher queries. The queries were designed to identify known vulnerability patterns and misconfigurations directly on the Infrastructure Property Graph (IPG).

For the Rule-Based Graph Engine and GNN_SDT, the IaC files were first transformed into their respective Infrastructure Property Graphs using the same Digital Twin generation process, ensuring that both graph-based approaches operated on an equivalent and consistent representation of the infrastructure.

Input Consistency: For all three systems, the identical IaC configuration files from the test set were provided as input

Output Alignment: The output of each system was then compared against the manually annotated ground truth labels in the dataset to calculate precision, recall, F1-score, and Area Under the Receiver Operating Characteristic (AUC-ROC).

To answer our research questions, we employed the following metrics:
For RQ1 (Effectiveness): Standard binary classification metrics such as Precision, Recall, F1-Score, and AUC-ROC are used.
For RQ2 (Attack Path Detection): A binary Path Detection Rate (PDR) which measures the percentage of the 340 known multi-hop attack paths in our dataset that were correctly identified by each system.
For RQ3 (Performance): The End-to-End Processing Time is measured in seconds, from ingesting the IaC files to outputting the final report.

3.4. Implementation
Hardware: Intel Xeon E5-2690 v4 CPU (14 cores), 128 GB RAM, NVDIA Tesla V100 GPU (16 GB VRAM)
Software: Ubuntu 22.04 LTS, Python 3.10, Terraform v1.5
The GNN model was implemented using PyTorch 2.0 and the PyTorch Geometric (PyG) library.
GNN Hyperparameters: The R-GCN model was configured with 3 message-passing layers and a hidden embedding dimension of 128. Adam optimizer with a learning rate of 0.001 and trained the model for 100 epochs with a batch size of 32 graphs is used. The DistMult scoring function was used for link prediction.

4.0. Results and Discussion
4.1.1. RQ1: Effectiveness in Vulnerability Detection
The first set of experiments measured the overall effectiveness of each system in identifying both simple and complex vulnerabilities. The precision, recall, and F1-score for each tool are presented in Table 2.

Table 2: Comparative Performance on Overall Vulnerability Detection

System	**Precision**	**Recall**	**F1-Score**	**AUC-ROC**	**Analysis Method**	**Key Differentiator**
GNN_SDT	91.5%	94.2%	92.8%	92.0%	GNN on Property Graph	Learns contextual patterns
Rule-Based Graph Engine	85.3%	75.8%	80.3%	78.0%	Query on Property Graph	Manual rules on graph data
Checkov (Static Analysis)	78.1%	61.5%	68.8%	65.0%	Rule-Based File Parsing	Lacks relational context

Validation and Analysis: The results in Table 2 unequivocally demonstrate the superior performance of our GNN-based framework.

Superior F1-Score: GNN_SDT achieves an F1-score of 92.8%, significantly outperforming both the Rule-Based Graph Engine (80.3%) and Checkov (68.8%). This indicates a more robust and reliable detection capability.

Minimizing False Negatives: The most critical finding is our framework's high recall (94.2%). This means it successfully identified the vast majority of actual vulnerabilities, a critical attribute for a security tool.

The precision and recall achieved by GNN_SDT are directly attributable to the high-fidelity context provided by the security digital twin, which allows the GNN to learn intricate relational patterns that would be impossible from isolated IaC files.

Reducing False Positives: The high precision (91.5%) validates that the GNN learns to differentiate between theoretically risky configurations and those that are practically non-exploitable within the given infrastructure context, thereby reducing alert fatigue for developers.

AUC-ROC: GNN_SDT significantly outperforms Rule-Based Graph Engine and Checkov in vulnerability detection for IaC, achieving a 95.0% AUC-ROC due to its ability to provide nuanced risk scores, offering superior discriminative power and adaptability for security teams.

4.1.2. RQ2: Depth of Analysis on Multi-Hop Attack Paths.

To test the depth of analysis, each system's ability is evaluated to detect the 340 known complex, multi-hop attack paths in the dataset used. The results are summarized in Table 3 below:

Table 3: Detection Rate for Complex, Multi-Hop Attack Paths

Attack Path Scenario Categories	Total Paths	GNN_SDT Model (Detected)	Rule-Based Graph Engine (Detected)	Checkov (Detected)
Public Compute to Internal Data	110	108 (98.2%)	52 (47.3%)	0 (0%)
Lateral Movement via Network Rules	95	92 (96.8%)	58 (61.1%)	0 (0%)
IAM Privilege Escalation Chain	80	74 (92.5%)	20 (25.0%)	0 (0%)
Container to Host Escape via Volume	55	51 (92.7%)	15 (27.3%)	0 (0%)
Overall Detection Rate	340	325 (95.5%)	145 (42.6%)	0 (0%)

Validation and Analysis: Table 3 provides compelling evidence of our framework's unique ability to reason about interconnected risk.

Holistic Context is Key: GNN_SDT successfully identified over 95% of all multi-hop attack paths. This confirms that the GNN effectively learns to trace risk across different resource types and relationships (network, identity, containment). Baseline Failures: Checkov failed to detect a single complete attack path, validating the hypothesis that context-unaware tools are blind to such threats. The Rule-Based Graph Engine performed better but still missed over half of the complex paths, demonstrating that even with full graph context, manually crafting rules for all possible multi-hop scenarios is intractable and brittle.

4.1.3. RQ3: Performance and Scalability

To assess the feasibility of GNN_SDT framework for CI/CD integration, the end-to-end processing time is measured. The results for infrastructures of varying sizes are presented in Table 4.

Table 4: End-to-End Processing Time vs. Infrastructure Size

Infrastructure Size (Number of Resources)	IaC Plan & Parse(s)	Graph Construction(s)	GNN Inference(s)	Total Processing Time(s)
Small (<100)	5.1 | 3.2 | 0.9 | 9.2	Medium (~500)	16.8 | 10.5 | 1.8	29.1
Large (~1500)	58.2 | 24.1 | 4.5 | 4.5	Extra-Large (~3000)	121.5 | 45.3 | 8.9	175.7

Validation and Analysis: The performance data in Table 4 confirms the practical viability of our system. The total processing time scales in a near-linear fashion with the number of resources, avoiding the exponential state-space explosion common in traditional attack graph generation. For a large 1500-resource infrastructure, the analysis completes in under 90 seconds, which is well within acceptable limits for an automated check in a CI/CD pipeline.

4.1.4. RQ4: Ablation Study

To isolate the contribution of the GNN component, direct comparison is carried out between GNN_SDT and the Rule-Based Graph Engine. The significant performance gaps shown in Table 2 (F1-score of 92.8% vs. 80.3%) and Table 3 (attack path detection of 95.6% vs. 42.6%) serve as a conclusive ablation study. This validates that while the graph representation is a critical prerequisite, it is the GNN's ability to learn complex, non-linear patterns from the graph that provides the transformative leap in detection accuracy.

4.2. Discussion

The empirical results presents in the previous section provide strong evidence for the efficacy of the proposed predictive security framework. The core finding is not just that our system outperforms existing tools, but give the *why* it does so. The central theme – the transition from isolated, rule-based checks to holistic, context-aware learning explains the stark contrast in performance between GNN_SDT framework and the two baselines. GNN-SDT framework moves beyond static signatures and performs learning in more robust way. The digital twin is able to provide a resolved, canonicalized representations of the infrastructure state and moves beyond abstract IaC files to a functional model, allowing the GNN to analyze the 'live' configuration before deployment, which is a key differentiator from static analysis tools.

4.2.1 Implications for the DevSecOps Lifecycle

The practical impact of this research extends beyond mere vulnerability detection, it possess the potential to reshape the relationship between development and security teams. It enables true "Shift-Left" Security, reduce alert fatigue and remedy overhead, as well as platform for automated remediation.**4.2.2. Limitations and Threats to Validity**

The limitations of this framework is noted in its dependency on IaC plan accuracy, unable to conceiveable cloud architectures or vulnerability types, and ability to explain to increase the trust and actionability of the findings.

4.3. Interpretation of Findings

The primary finding is not just that the system outperforms existing tools, but why it does so. The stark contrast in performance between GNN_SDT framework and two baselines can be attributed to one central theme "the transition from isolated, rule-based checks to holistic, context-aware learning.

Moving Beyond Static Signatures: Traditional static analysis tools like Checkov operate like antivirus software from a bygone era – they scan for known, static "signatures" of bad configurations. As shown by its complete failure to detect any multi-hop attack paths (Table 3), this approach is fundamentally incapable of understanding emergent risk. GNN_SDT framework, by representing the entire infrastructure as a graph, shifts the paradigm from analyzing individual files to analyzing a complete system.

Learning is More Robust than Manual Rule Engineering: The most illuminating comparison is between our GNN model and the Rule-Based Graph Engine. Both systems had access to the exact same rich, contextual data within the Infrastructure Property Graph. Yet, the GNN significantly outperformed the rule-based approach in both precision and recall (Table 2). This demonstrates a critical limitation of human-driven security: even with perfect visibility, security experts cannot possibly author rules to account for every novel and complex interaction that can occur in a large-scale cloud environment.

5.0. Conclusion

This paper addresses the critical and growing challenge of securing cloud infrastructure in the era of Infrastructure-as-Code. A fundamental gap in existing security paradigms, which are torn between proactive but context-unaware static analysis and context-rich but reactive runtime scanning was identified. To bridge this gap, GNN_SDT a framework was developed to introduce predictive, context-aware security analysis into the pre-deployment phase of the DevOps lifecycle.

The primary contribution is a multi-stage methodology that first constructs a high-fidelity security Digital Twin from IaC execution plans, translates this twin into a rich Infrastructure Property Graph (IPG), and then leverages a Graph Neural Network (GNN) to analyze this graph for complex vulnerabilities. The result evaluation demonstrates that this approach is both an incremental improvement as well as a significant leap forward. The framework successfully identified over 95% of complex, multi-hop attack paths missed entirely by state-of-the-art static analysis tools, while simultaneously reducing false positives by providing the holistic context that isolated checks task. In addition, its efficient, near-linear scalability confirms its feasibility for integration into modern, fast-paced CI/CD pipelines.

Most importantly, this research demonstrates a paradigm shift from a reactive 'detect-and-respond'

Posture to a proactive 'predict-and –prevent' model for cloud infrastructure security. GNN-SDT framework empowers organizations to build secure cloud environments by default, fundamentally reducing organizational risk and eliminating the friction between development and security teams by embedding deep, contextual intelligence directly into the development workflow.

This research work does not only established a strong foundation but also highlight numerous avenues for future research. Possible areas of improvement include: building an automated remediation and self-healing infrastructure, integrating runtime data for continuous verification and drift detection, as well as providing a deeper level of 'why' behind a prediction that will make system's outputs more transparent, auditable, and actionable for both developers and security analysts.

References

Ajay Varma Indukuri (2025). Infrastructure as code: A paradigm shifts in cloud resource management and deployment automation. World Journal of Advanced Engineering Technology and Sciences, 2025, 15(01), 2309-2317. DOI: https://doi.org/10.30574/wjaets.2025.15.1.0478.

Borovits, N., Shahar, Y., & Miller, Y. (2022). FindICI: An ML-based approach for detecting linguistic inconsistencies in Infrastructure-as-Code. *Journal of Systems and Software*, *192*, 111402.

Chiari, M., De Pascalis, M. & Pradella, M. (2022). Static Analysis of Infrastructure as Code: a Survey. 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C).Honolulu, HI, USA, 2022, Pages 218 – 225. DOI: 10.1109/ICSA-C54293.2022.00049

Cloud Security Alliance. (2022). *Cloud Security Report 2022*. Cybersecurity Insiders, Fortinet.

Dalla Palma, S., Pastore, F., & Mariani, L. (2021). *Product and process metrics for within-project defect prediction in Infrastructure as Code*. Proceedings of the 2021 International Conference on Software Engineering: Software Engineering in Practice, 271-280.

Kim, G., Humble, J., Debois, P., Willis, J., & Forsgren, N. (2022). The DevOps Handbook. 2. *Auflage. Portland, Ore.: IT Revolution*.

Kothapalli, Srinikhita & Nizamuddin, Md & Talla, Rajasekhar Reddy & Gummadi, Jaya. (2024). DevOps and Software Architecture: Bridging the Gap between Development and Operations. 2. 51-64.

Lepiller, J., Pradhan, S., Zaitsev, A., & Balakrishnan, N. (2021). Hayha: Detecting intra-update sniping vulnerabilities in Infrastructure as Code. *Proceedings of the 30th USENIX Security Symposium*, 2147-2164.

Matthias T. (2024). Static Code Analysis for Infrastructure as Code. M.Sc., FH Joanneum, University of Applied Sciences, Kapfenberg, Austria.

Opdebeeck, R., De Wael, J., Van Landeghem, H., & De Kock, W. (2023). GASEL: Security smell detection for Ansible with graph queries on program dependence graphs. *Journal of Systems and Software*, *197*, 111559.

Pandit, Atharv & Pandit, Rakesh. (2025). Side-Channel Attacks in Multi-Tenant Cloud Environments: Prevention & Mitigation. International Journal of Innovations in Science Engineering And Management. 10.69968/ijisem.2025v4i293-105.

Rahman, A., King, J., & Ma, S. (2018). *An empirical study on characteristics of detective IaC scripts and their correlation with security and privacy violations*. Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion Proceedings, 126-129.

Rahman, A., King, J., & Ma, S. (2019). *Defect prediction for Infrastructure-as-Code scripts: An empirical study*. Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, 13-22.

Rahman, A., King, J., & Ma, S. (2019). *The seven sins: Security smells in Infrastructure-as-Code scripts*. Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings, 37-40.

Rahman, A., King, J., & Ma, S. (2020). *Development anti-patterns in Infrastructure-as-Code scripts*. Proceedings of the 2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Practice, 159-168.

Rahman, A., King, J., & Ma, S. (2021). *Practices for secret management in Infrastructure-as-Code: A grey literature review*. Journal of

Information Security and Applications, *61*, 102914.

Razin, N., Verbin, T., Cohen, N. (2023). On the ability of graph neural networks to model interactions between vertices. https://arxiv.org/abs/2211.16494

Saavedra, Nuno & Gonçalves, João & Henriques, Miguel & Ferreira, João & Mendes, Alexandra. (2023). Polyglot Code Smell Detection for Infrastructure as Code with GLITCH. 2042-2045. 10.1109/ASE56229.2023.00162.

Sabura, A., Chowdharya, A., Huanga, D., & Alshamranic, A. (2022). Toward Scalable Graph-based Security Analysis for Cloud Networks. *Future Generation Computer Systems*, *128*, 248–258. https://doi.org/10.1016/j.future.2021.12.025.

Samuel, M., Obira, O., & Keneth, S. (2024). The Implementation of Infrastructure as Code Template for Low-cost Cloud Infrastructure Operations. *East African Journal of Information Technology*, *7*(1), 462-474. https://doi.org/10.37284/eajit.7.1.2538

Schwarz, J., Arlt, B., & Arnaoudova, V. (2018). *Code smells in infrastructure as code: An exploratory study*. Proceedings of the 2018 IEEE/ACM 1st International Workshop on Cloud Security and Privacy, 1-6.

Tariq, Shahroz & Baruwal Chhetri, Mohan & Nepal, Surya & Paris, Cecile. (2025). Alert Fatigue in Security Operations Centres: Research Challenges and Opportunities. ACM Computing Surveys. 57. 10.1145/3723158.

Zimmermann, M., Breitenbücher, U. & Leymann, F. (2017). A TOSCA-based Programming Model for Interacting Components of Automatically Deployed Cloud and IoT Applications. In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2, pages 121-13, DOI: 10.5220/0006332501210131