INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 5, May 2026 | ISSN: 2456-4184 | IJNRD.ORG

Intelligent Intrusion Detection System for Intranet Security Based on Machine Learning and Behaviroal Analystics

R. Prathiba (Assistant Professor)
Computer Science and Engineering
Bharath Institute Of Higher Education
And Research Chennai, India
prathiba.cse@bharathuniv.ac.in

Gande Sadvik
Computer Science and Engineering
Bharath Institute Of Higher Education
And Research Chennai,India
gandesadvik123@gmail.com

Eppakayala Manoj Kumar
Computer Science and Engineering
Bharath Institute Of Higher Education
And Research Chennai, India
manojkumarppakayala26@gmail.com

Ganagoni Pranay
Computer Science and Engineering
Bharath Institute Of Higher Education
And Research Chennai, India
gpranay292@gmail.com

Eluri Chandra Sheker
Computer Science and Engineering
Bharath Institute Of Higher Education
And Research Chennai, India
elurichandrasheker@gmail.com

***Abstract*—In today's digital age, security challenges threaten user privacy as networks face increased vulnerability to malicious attacks due to large data volumes. Intrusion Detection Systems (IDS) play a crucial role in identifying cyber-attacks and pro- tecting system resources and users. This study utilizes machine learning classifiers (MLC) to analyze the NSL-KDD dataset, optimizing by preprocessing to remove irrelevant features. System performance is assessed using four attribute subsets, comparing model accuracy across DoS, Probe, U2L, and R2L attack classes to determine the best algorithm for each class. Using Forest with 20 features successfully achieved an accuracy of up to 99% in intrusion detection.**

In the cybersecurity landscape, detecting intranet attacks remains particularly challenging as hostile tactics constantly adapt and evolve. This research introduces an innovative machine learning approach that identifies potential threats by analyzing behavioral patterns rather than relying on fixed signatures. The methodology harnesses advanced algorithms to recognize and counter intranet-based attacks by identifying anomalous behaviors that deviate from established usage patterns. By examining network traffic and system logs, our model differentiates between normal and suspicious activities, enabling it to detect and respond to threats proactively. This approach shows significant promise for strengthening intranet security through its real-time monitoring capabilities and adaptive defense systems. The solution enhances security posture by continuously analyzing behavior patterns and identifying potential threats before they can cause damage. Our empirical evaluations and comparative analyses confirm the model's effectiveness. Test results demonstrate how it successfully identifies anomalies that traditional security measures might miss, while maintaining a low rate of false positives. This technology complements existingcybersecurity frameworks rather than replacing them, providing an additional layer of protection.

***Keywords*— Machine Learning, Intrusion Detection, Behavior-Based Attacks, Cyber Security, Network Security, Intranet Attack, Zeek IDS, Feature Engineering(FE).**

I. Introduction

Despite the implementation of security measures like en- cryption and firewalls, breaches are common [1], highlighting the need for proactive Intrusion Detection Systems (IDS) that can adapt and respond in real time. IDS are crucial for identifying unauthorized access or manipulation attempts [2], monitoring network traffic for malicious activities, and serving as a vital defense mechanism.

Advancements in data analysis have shifted IDS from tra- ditional statistical methods to artificial intelligence algorithms [3], Machine learning techniques such as decision trees, K- Nearest Neighbor (KNN), XGBoost, Gradient Descent, and random forest (RF) are now employed to improve detection capabilities.

This study provides a systematic analysis of these classifiers, focusing on their effectiveness and robustness. Section 2 reviews related research and methodologies. Section 3 intro- duces our proposed architecture. Section 4 covers dataset pre- processing, feature selection, and model performance. Section 5 discusses the results, comparing various machine learning models for phishing detection. Finally, Section 6 summarizes our findings and suggests future research directions.

II. Related Work

Machine learning for intrusion detection on the NSL- KDD dataset [4], evaluates classifiers such as KNN, SVM, Logistic Regression, MLP, Na¨ive Bayes, Extra Trees, Decision Tree, and Random Forest. The study emphasizes the role of feature.

Extraction and selection in enhancing performance, revealing varying accuracies across attack classes and offering insights into each classifier's strengths and weaknesses.

Feature selection is crucial for improving model accuracy [5], A method using a pigeon-inspired optimizer for attribute selection was tested on UNSW-NB15, KDDCUP 99, and NSLKDD datasets.

Random Forest (RF) [6], was used to remove less important features, reducing computational costs, and Particle Swarm Optimization (PSO) improved accuracy to 99% on the NSL-KDD dataset with 10 features.

Support Vector Machines (SVM) have shown strong per- for mance in intrusion detection [7], An SVM model tested on and DoS attacks with 41 attributes and U2R attacks with 11 attributes. PCA combined with RF [11], reduced dimensionality and classified data, achieving a 96.78% classification rate compared to SVM, DT, and NB.

III. Proposed Architecture

The preprocessing and feature selection for the datasets KDDTrain+, KDDTest+, and KDDTest-21 involve a structured approach encompassing data cleaning, train-test splitting, and feature selection techniques. Initially, we load the datasets and handle any missing values, remove duplicate entries, and treat outliers. Subsequently, we combine the KDDTrain+, KD- DTest+, and KDDTest-21 datasets to create a comprehensive dataset, which is then split into 80% training and 20% testing sets.

For feature selection, we apply two techniques: Decision Tree feature selection to rank and select the top features based on importance scores, and Principal Component Analysis (PCA) to reduce dimensionality by retaining components that explain the majority of variance will train various models using the training data. To evaluate and compare the performance of these models, we will employ different metrics to determine the best model for Intrusion Detection. Subsequently, we will use the selected model to classify attempts and indicate whether they are normal or attacks as shown in Figure 1.

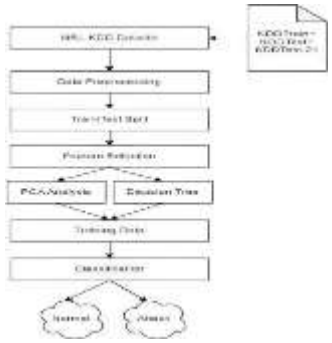

Fig. 1: Architecture of the Proposed Model

IV. Methodology

A. Dataset and Data Preprocessing

The analysis and feature selection will utilize the NSL-KDD dataset [12], which includes KDD Train, KDD Test, and KDDTest-21 files. Each file, containing 43 columns, is ana- lyzed individually using various algorithms, and performance metrics such as accuracy, recall, F-score, precision, confusion matrix, and execution time are compared.

The KDDTrain file has 125,972 rows, KDDTest has 22,543 rows, and KDDTest-21 has 11,849 rows. The target column classifies data into attack types: `normal`, `neptune`, `warezclient`, `ipsweep`, `portsweep`, `teardrop`, `nmap`, `satan`, `smurf`, `pod`, `back`, `guess_passwd`, `ftp_write`, `multihop`, `rootkit`, `buffer_overflow`, `imap`, `warezmaster`, `phf`, `land`, `loadmodule`, `spy`, and `perl`. These attack types were categorized into four classes: DoS, Probe, R2L, and U2R.

The distribution of samples for each class in the three files is as follows:

The KDDTrain+ dataset consists of 67,342 Normal in- stances, 45,927 instances of DoS attacks, 11,656 instances of Probe attacks, 995 instances of R2L attacks, and 52 instances of U2R attacks.

The KDDTest+ dataset consists of 9,711 Normal instances, 7,457 instances of DoS attacks, 2,421 instances of Probe attacks, 2,754 instances of R2L attacks, and 200 instances of U2R attacks.

The KDDTest-21 dataset comprises 2,152 Normal instances, 4,342 instances of DoS attacks, 2,402 instances of Probe attacks, 2,753 instances of R2L attacks, and 200 instances of U2R attacks.

Data preprocessing was applied to all three files, and no missing data or duplicates were identified. Each file was processed separately to train the algorithms on datasets with different distributions. Feature selection involved two tech- niques: Principal Component Analysis (PCA) to select the 20 most significant features, and a Decision Tree to identify the 8 most important features.

B. Feature Selection based on Decision Tree

To begin, the Decision Tree algorithm will be utilized to assess all features and identify the top 8 most influential ones as shown in Figure 2. Initially, the algorithm identified these features based on their impact on the outcome column.

Features selected by the Decision Tree: num shells, dst _ host rerror rate, level, srv diff host rate, service, dst host same src port rate, logged in, is guest login

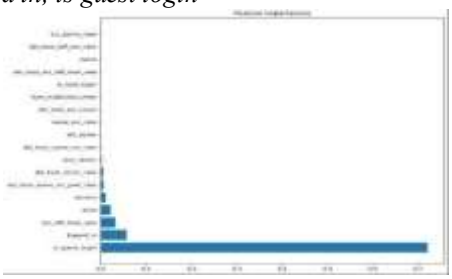

Fig. 2: This figure illustrates the features upon which the outcome column is dependent and selected by the Decision Tree.

C. Feature Selection based on PCA Analysis

The NSL-KDD dataset preprocessing involved several steps. First, features were categorized as either categorical or numer- ical. Categorical features included "is host login," "protocol type," "service," "flag," "land," "logged in," "is guest login," "level," and "outcome, "while the remaining features were numerical.

Numerical features were normalized to improve machine learning performance, and categorical features ("protocol type," "service," "flag") were one-hot encoded into binary variables. This transformed the dataset from 43 columns to 124 columns while retaining the same 125,972 rows.

To reduce computational overhead, Principal Component Analysis (PCA) was applied to select the top 20 principal components, ensuring the dataset was ready for machine

learning tasks. After feature selection, the datasets were reshaped as fol- lows: KDDTrain+ Dataset: The Decision Tree approach re- sulted in 125,972 rows and 8 columns, while PCA produced 125,972 rows and 20 columns. KDDTest+ Dataset: The De- cision Tree approach resulted in 22,543 rows and 8 columns, while PCA produced 22,543 rows and 20 columns. KDDTest- 21 Dataset: The Decision Tree approach resulted in 11,849 rows and 8 columns, while PCA produced 11,849 rows and 20 columns. The model's performance was then evaluated using metrics such as accuracy, recall, F-score, precision, confusion matrix, and execution time for both the training and testing phases.

D. Results

After preprocessing and feature selection using PCA and Decision Tree methods, the data was split into 80% train- ing and 20% testing sets. Various machine learning models were then applied, including Logistic Regression (LR), K-Nearest Neighbors (KNN), Naive Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), XGBoost (XGB), and Gradient Boosting (GB).

The models were evaluated using accuracy, precision, recall, and execution time during both the training and testing phases. Confusion matrices were generated to analyze true and false positives/negatives, providing a comprehensive performance comparison across all models.

E. Performance Assessment

The model's performance will be evaluated on 3 datasets, focusing on accuracy, precision, recall, and training execution time. Tables 1 to 4 highlight the accuracy,

Precision, Recall, and Execution Time of all models across two distinct feature selection methods over separated files.

KDDTrain+ Dataset Evaluation:

Accuracy	*Decision Tree*	*PCA Analysis*
LR	*80.05%*	*51.25%*
KNN	*98.47%*	*99.53%*
NB	*77.21%*	*48.47%*
SVM	*82.25%*	*96.76%*
DT	*99.04%*	*99.96%*
RF	*99.04%*	*99.98%*
XGB	*98.33%*	*99.96%*
GB	*97.89%*	*99.81%*

TABLE 1: Evaluation of Model Accuracy KDDTrain+

Precision	*Decision Tree*	*PCA Analysis*
LR	*81.71%*	*38.83%*
KNN	*98.49%*	*99.53%*
NB	*83.91%*	*72.16%*
SVM	*84.39%*	*96.75%*
DT	*99.05%*	*99.99%*
RF	*99.05%*	*99.97%*
XGB	98.36%	99.96%
GB	97.89%	99.81%

Recall	*Decision Tree*	*PCA Analysis*
LR	*80.05%*	*51.25%*
KNN	*98.47%*	*99.53%*
NB	*77.21%*	*48.47%*
SVM	*82.25%*	*96.76%*
DT	*99.04%*	*99.93%*
RF	*99.04%*	*99.96%*
XGB	*98.33%*	*99.96%*
GB	*97.89%*	*99.81%*

TABLE 2: Evaluation of Model Precision KDDTrain+

TABLE 3: Evaluation Model Recall KDDTrain+

Execution	*Decision Tree*	*PCA Analysis*
LR	*4.24 s*	*9.24 s*
KNN	*0.84 s*	*0.01 s*
NB	*0.05 s*	*0.09 s*
SVM	*1.59 s*	*66.91 s*
DT	*0.15 s*	*7.03 s*
RF	*6.74 s*	*102.5 s*
XGB	*1.12 s*	*1.60 s*
GB	*46.37 s*	*1024 s*

TABLE 4: Evaluation Execution time KDDTrain+

KDDTest+ Dataset Evaluation: *The evaluation of models on the KDDTest+ dataset shows notable variations across accuracy, precision, recall, and execution time.*

Logistic Regression (LR) achieved moderate accuracy, rang- ing from 75.76% to 76.81%, while K-Nearest Neighbors (KNN) excelled with accuracy near 98%. Naive Bayes (NB) had lower accuracy, especially with PCA (41.09%). Support Vector Machines (SVM) showed a significant boost with PCA, increasing from 74.77% to 90.41%. Decision Tree (DT) and Random Forest (RF) both achieved over 99% accuracy, with RF slightly outperforming DT, particularly with PCA.

Precision mirrored accuracy trends, with KNN, DT, RF, and XGB achieving near-perfect precision. LR and NB had lower precision, especially with PCA. SVM showed substantial improvement, jumping to 90.22% with PCA.

KDDTest-21 Dataset Evaluation: *The evaluation of mod- els on the KDDTest-21 dataset revealed varied performance in terms of accuracy, precision, recall, and execution time.*

Logistic Regression (LR) showed modest accuracy between 57.84% and 62.10%, while K-Nearest Neighbors (KNN) performed

significantly better, achieving around 97%. Naive Bayes (NB) had lower accuracy, ranging from 43.37% to 53.08%, and Support Vector Machines (SVM) improved no- tably with PCA, rising from 61.34% to 84.22%. Both Decision Tree (DT) and Random Forest (RF) delivered high accuracy, with RF reaching 99.96% using PCA. Gradient Boosting (GB) and XGBoost (XGB) also performed well, achieving accuracy close to 99.96%.

Precision closely mirrored accuracy, with KNN and SVM maintaining high values, particularly with PCA. DT, RF, and XGB achieved near-perfect precision, around 99.94%-99.99%. Recall followed a similar trend, with DT, RF, and XGB continuing to show strong performance.

Execution times varied widely. LR took significantly longer without PCA (167.9 seconds vs. 5.58 seconds with PCA), while KNN and NB were quick, especially with PCA. SVM, XGB, DT, and RF showed moderate times, and GB's execution time increased substantially with PCA (86.6 seconds compared to 5.78 seconds).

V. Discussion

A. Model Performance and Selection

The evaluation results show clear differences in model performance across accuracy, precision, recall, and execution time. PCA models consistently outperformed those with Decision Tree feature selection, particularly in accuracy and precision, with Random Forest, Decision Tree, and XGBoost models achieving near-perfect results.

Recall followed a similar trend, with PCA-based models showing better performance. However, execution time favored Decision Tree-based models, which were generally faster, es- pecially for complex models like Random Forest and Gradient Boosting. This highlights a trade-off between achieving higher accuracy and maintaining computational efficiency.

B. System Implementation and User Interaction

After conducting the analysis, a software application was meticulously developed using Flask in Python, seamlessly integrated with HTML, CSS, and JavaScript. This interactive platform empowers users to select from a refined set of high-performing models via a dropdown menu.

The available models include Random Forest utilizing PCA, Random Forest with Decision Tree features, Decision Tree leveraging PCA, Decision Tree using Decision Tree features, KNN with PCA, and KNN using Decision Tree features. Depending on the model chosen, users are directed to input either 20 specific features for PCA-based models or 8 specified features for Decision Tree-based models, as outlined earlier.

When the prediction process is initiated, the application swiftly evaluates whether the input data indicates a normal occurrence, an attempted attack, or an intrusion detection scenario (Dos, Probe, R2L, U2R). This robust approach ensures structured deployment of machine learning models for intrusion detection, enhancing user engagement and enabling real-time decision-making capabilities.

VI. Conclusion and Future Work

This study evaluated various machine learning models for intrusion detection using the NSL-KDD dataset. We applied PCA and Decision Tree feature selection, comparing models based on accuracy, precision, recall, and execution time.

Our results showed that decision tree-based models, such as Random Forest and Gradient Boosting, consistently out-performed others in terms of accuracy and recall. SVM also demonstrated strong performance but with longer execution times, highlighting a trade-off between precision and compu- tational efficiency.

A software tool will detect intrusions using Decision Tree (8 features) and PCA (20 features). It will classify results into normal or attack types based on selected features for accurate detection.

A. Future Directions for Research

Future research could prioritize increasing the sample size for each class to achieve a balanced dataset, potentially improving model accuracy. Advanced feature reduction tech- niques like LASSO and Recursive Feature Elimination may also streamline the feature set. Sophisticated feature engineer- ing approaches could extract more discriminative information from network traffic data. Additionally, evaluating deep learn- ing models such as (CNNs) and (RNNs) could address the complexity of intrusion detection tasks. Finally, transitioning the best models into realtime systems.

References

[1] *GS. T. Siddiqui, S. Alam, M. Shuaib, and A. Gupta, "Cloud Computing Security using Blockchain," Journal of Emerging Technologies and Innovative Research, vol. 6, no. 6, pp. 791–794, 2019.*

[2] *F. Masoodi, S. Alam, and S. T. Siddiqui, "Security and privacy threats, attacks and countermeasures in Internet of Things," Int. J. Netw. Secur. Appl., pp. 67–77, 2019.*

[3] *S. Alam, M. Shuaib, and A. Samad, "A Collaborative Study of Intrusion Detection and Prevention Techniques in Cloud Computing," in Lecture Notes in Networks and Systems, vol. 55, pp. 231–240, 2019. doi: 10.1007/97898113-2324-9 23.*

[4] *F. Masoodi, "Machine learning for classification analysis of intrusion detection on NSL-KDD dataset," Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 12, no. 10, pp. 2286-2293, 2021.*

[5] *H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer," Expert Systems with Applications, vol. 148, p. 113249, 2020.*

[6] *N. Kunhare, R. Tiwari, and J. Dhar, "Particle swarm optimization and feature selection for intrusion detection system," Sa¯dhana¯, 2020. doi: 10.1007/s12046-0201308-5.*

[7] *A. Bachar, N. El Makhfi, and O. El Bannay, "Towards a behav- ioral network intrusion detection system based on the SVM model," in 2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology, IRASET 2020. doi: 10.1109/IRASET48871.2020.9092094.*

[8] *L. Lv, W. Wang, Z. Zhang, and X. Liu, "A novel intrusion detection system based on an optimal hybrid kernel extreme*

learning machine," Knowledge-Based Systems, vol. 195, p. 105648, 2020. doi: 10.1016/j.knosys.2020.105648.

[9] *I. Sumaiya Thaseen, B. Poorva, and P. S. Ushasree, "Network Intrusion Detection using Machine Learning Techniques," in International Confer- ence on Emerging Trends in Information Technology and Engineering, IcETITE 2020. doi: 10.1109/ic-ETITE47903.2020.148.*

[10] *R. A. Ghazy, E. S. M. El-Rabaie, M. I. Dessouky, N. A. ElFishawy, and F. E. A. El-Samie, "Feature Selection Ranking and Subset-Based Techniques with Different Classifiers for Intrusion Detection," Wireless Personal Communications, vol. 111, no. 1, pp. 375–393, 2020. doi: 10.1007/s11277-019-06864-3.*