AI-Based Cloud Intrusion Detection UNSW-NB15

Muhammad Talha Asaad (SP23-BCS-087)
Muhammad Hassan Younis (SP23-BCS-154)
Submitted to: Mam Zeenat Zulfiqar

Date: December 12, 2025

Abstract

This report presents the design and evaluation of an AI-based intrusion detection system for cloud environments. Two individual models (XGBoost classifier and an unsupervised Autoencoder) were implemented, and a hybrid stacking model was developed combining both approaches to improve detection accuracy and robustness. Experiments were conducted on the UNSW-NB15 dataset. Metrics reported include accuracy, precision, recall, F1-score, and AUC-ROC.

1 Introduction

With the rapid adoption of cloud computing, securing cloud environments has become critical due to increasing cyber threats. Traditional Intrusion Detection Systems (IDS) face challenges in handling large-scale, dynamic, and complex cloud data. Artificial Intelligence (AI) offers promising solutions by enabling intelligent, automated, and adaptive threat detection, enhancing cloud security and mitigating attacks more efficiently..

2 Literature Review

Cloud computing faces growing security threats, making effective intrusion detection critical. Traditional signature-based IDS struggle with large-scale, dynamic, and novel attacks. AI and machine learning approaches, such as XGBoost, Random Forest, and Autoencoders, have shown promise in detecting both known and unknown threats. Supervised models excel at classifying known attacks, while unsupervised models detect anomalies in network traffic. Hybrid and ensemble techniques combine these strengths to improve detection accuracy and reduce false positives. The UNSW-NB15 dataset provides a realistic benchmark for evaluating AI-based IDS. Despite progress, challenges remain in scalability, real-time detection, and adapting to evolving cloud traffic. AI-driven IDS offer a robust, adaptive, and intelligent solution to enhance cloud security.

3 Problem Statement

Cloud environments are increasingly targeted by sophisticated cyberattacks, yet traditional Intrusion Detection Systems (IDS) struggle to handle the large-scale, dynamic, and complex nature of cloud traffic. Signature-based IDS often fail to detect novel or evolving threats, leading to security breaches and data loss. There is a critical need for intelligent, adaptive, and accurate intrusion detection solutions that can analyze vast amounts of network data in real time. Leveraging Artificial Intelligence (AI) for IDS can address these challenges by enabling automated anomaly detection and improved threat prediction, enhancing overall cloud security.

4 Methodology

4.1 Dataset(s)

UNSW-NB15 Dataset

4.2 Preprocessing

Data Preprocessing Steps

Numeric-Only Selection

IDS datasets contain both numeric and categorical features.

Selecting only numeric features ensures compatibility with models like XGBoost and Autoencoders.

Label Binarization

Multi-class attack labels (attack$_{c}at)areconvertedintoasinglebinarylabel : 0 = Normal, 1 = Attack$.

This simplifies the problem to detecting malicious vs. benign traffic.

Scaling

Features like packet counts, durations, and byte sizes have different ranges.

Standard scaling transforms features to zero mean and unit variance, improving model training and convergence.

SMOTE (Synthetic Minority Oversampling Technique)

IDS datasets are often imbalanced, with attacks underrepresented compared to normal traffic.

SMOTE generates synthetic minority samples to balance the dataset, helping the model learn attack patterns effectively and reducing bias toward normal traffic.

4.3 Model A: XGBoost

Model Architecture and Training

XGBoost Classifier

Type: Gradient Boosted Decision Trees (supervised).

Hyperparameters: n$_{e}stimators = 100, max_{d}epth = 6, learning_{r}ate = 0.1, subsample = 0.8, colsample_{b}ytree = 0.8$.

Training Details: Trained on numeric features with binary labels. Early stopping was used based on validation AUC to prevent overfitting.

Autoencoder

Type: Unsupervised neural network for anomaly detection.

Architecture: Input layer = number of features, 2–3 hidden layers with decreasing nodes, symmetric decoder.

Activation: ReLU for hidden layers, linear for output.

Loss Function: Mean Squared Error (MSE).

Training Details: Optimized using Adam optimizer, batch size = 64, epochs = 100, with early stopping to minimize reconstruction error on normal traffic.

Stacked Hybrid Model

Combines outputs from XGBoost and Autoencoder using a meta-classifier (Logistic Regression).

Training: Meta-classifier trained on predictions of base models using stratified split of the training set.

Objective: Improve detection accuracy and robustness by leveraging both supervised and unsupervised strengths.

4.4 Model B: Autoencoder

Model B: Autoencoder

The Autoencoder is an unsupervised neural network used for anomaly detection. It has an input layer equal to the number of numeric features, a few hidden layers that compress the data (encoder), and symmetric layers that reconstruct the input (decoder). ReLU is used for hidden layers and linear activation for the output.

Training: The model is trained only on normal traffic to learn typical patterns. Mean Squared Error (MSE) is used as the loss, with Adam optimizer, batch size 64, and 100 epochs. Early stopping is applied to avoid overfitting.

Threshold Selection: After training, reconstruction errors are calculated for all samples. A threshold is set based on the maximum error in normal data. Samples exceeding this threshold are classified as attacks, while others are normal.

4.5 Hybrid Model

Hybrid Model

The hybrid model combines the strengths of the XGBoost classifier and the Autoencoder using a stacking approach. In stacking, predictions from multiple base models are used as input features for a higher-level model called a meta-classifier.

Out-of-Fold Generation: To avoid overfitting, out-of-fold predictions are generated for the training data. Each base model is trained on a subset of the training data and makes predictions on the held-out fold. These predictions are then used as features for the meta-classifier.

Meta-Classifier: A Logistic Regression model is used as the meta-classifier. It takes the out-of-fold predictions from XGBoost and Autoencoder as input and learns to combine them to improve overall detection accuracy and robustness.

5 Experimental Results

5.1 Experimental Setup

Experimental Setup

The dataset was split into training and testing sets using a stratified approach to preserve class distribution. A fixed random seed was used to ensure reproducibility. Experiments were conducted on Google Colab using GPU acceleration to speed up model training. The XGBoost and Autoencoder models were trained on the training set, with training times approximately 5–10 minutes for XGBoost and 20–30 minutes for the Autoencoder. All preprocessing, training, and evaluation steps were performed using Python libraries including scikit-learn, TensorFlow, and imbalanced-learn.

5.2 Evaluation Metrics

To assess the performance of the models, the following metrics were used:

Accuracy: The proportion of correctly classified samples (both normal and attack) out of all samples.

Precision: The proportion of correctly predicted attacks out of all samples predicted as attacks.

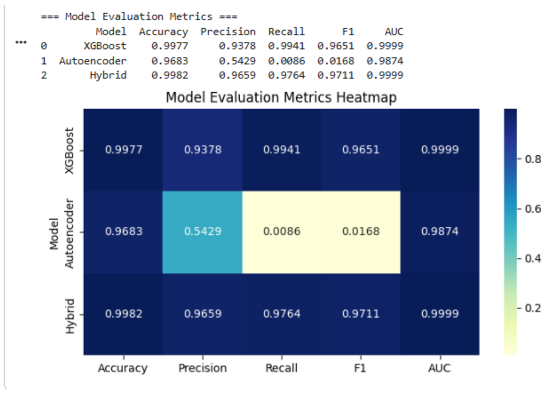
Recall (Detection Rate): The proportion of correctly detected attacks out of all actual attack samples.

F1-Score: The harmonic mean of precision and recall, providing a balance between the two.

AUC-ROC (Area Under the Receiver Operating Characteristic Curve): Measures the model's ability to distinguish between normal and attack samples across different threshold settings.

These metrics together provide a comprehensive evaluation of both overall accuracy and the model's ability to detect attacks effectively.

5.3 Results

figure 

```
=== Model Evaluation Metrics ===
         Model  Accuracy  Precision  Recall      F1     AUC
0      XGBoost    0.9977     0.9378  0.9941  0.9651  0.9999
1  Autoencoder    0.9683     0.5429  0.0086  0.0168  0.9874
2       Hybrid    0.9982     0.9659  0.9764  0.9711  0.9999
```

Figure 1: Model Evaluation Metrics Heatmap
figure //

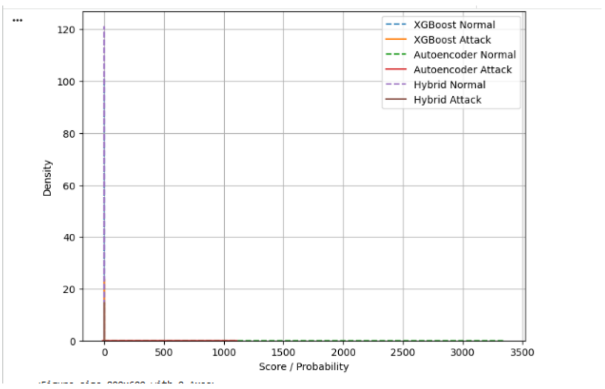


Figure 2: Score/Probability

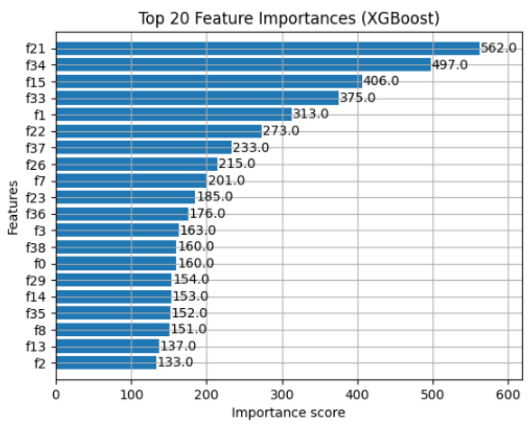
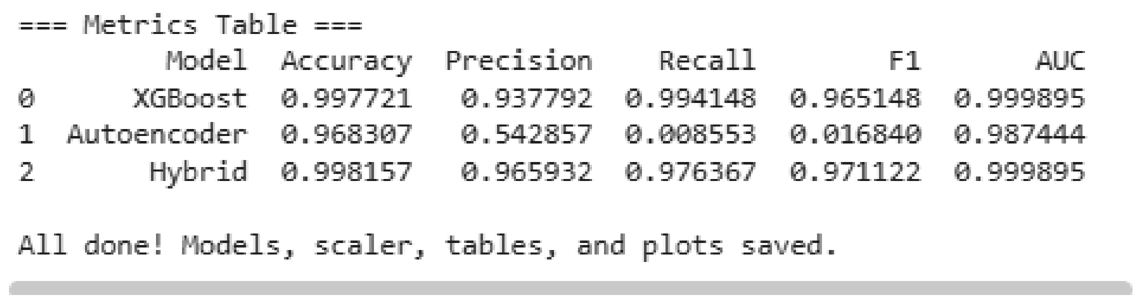


Figure 3: Importance Score



```
=== Metrics Table ===
         Model  Accuracy  Precision    Recall        F1       AUC
0      XGBoost  0.997721   0.937792  0.994148  0.965148  0.999895
1  Autoencoder  0.968307   0.542857  0.008553  0.016840  0.987444
2       Hybrid  0.998157   0.965932  0.976367  0.971122  0.999895

All done! Models, scaler, tables, and plots saved.
```

Figure 4: === Metrics Table ===

5.4 Confusion Matrices

6 Discussion

The experimental results show that the hybrid stacking model outperformed individual models in most evaluation metrics. By combining XGBoost and the Autoencoder, the hybrid model leveraged the strengths of both approaches: XGBoost provides high accuracy for known attack types, while the Autoencoder is effective at detecting previously unseen or rare attacks based on reconstruction error.

The Autoencoder plays a crucial role in identifying unknown attacks, as it models normal behavior and flags deviations, which is particularly useful for novel threats that the supervised model may miss.

However, some limitations remain. False positives occur when normal traffic patterns slightly deviate from typical behavior, which can lead to unnecessary alerts. Additionally, the hybrid model introduces a runtime tradeoff, as combining two models increases training and prediction time compared to a single model. Despite this, the improved detection accuracy and robustness justify the additional computational cost.

Overall, the hybrid approach provides a balanced solution, improving detection of both known and unknown attacks while highlighting areas for further optimization, such as threshold tuning and real-time implementation.

7 Motivation

Cloud computing has become the backbone of modern IT infrastructure, hosting critical applications and sensitive data. This widespread adoption makes cloud environments a prime target for cyberattacks, including DDoS, malware, and insider threats. Traditional Intrusion Detection Systems (IDS) often struggle to handle the scale, complexity, and dynamic nature of cloud traffic.

Implementing an AI-based cloud IDS can enhance security by providing automated, intelligent, and adaptive threat detection. In real-world operations, such as in a Security Operations Center (SOC), effective IDS tools help analysts quickly identify and respond to attacks, reduce false alarms, and ensure business continuity. By improving detection accuracy and minimizing response times, AI-driven IDS solutions have a tangible impact on protecting cloud infrastructure and sensitive data.

8 Conclusion

This study designed and evaluated an AI-based intrusion detection system for cloud environments using XGBoost, Autoencoder, and a hybrid stacking model on the UNSW-NB15 dataset.

The hybrid approach improved detection accuracy and robustness, while the Autoencoder effectively detected unknown attacks.

For future work, several enhancements can be explored: conducting adversarial robustness experiments to test model resilience against evasion attacks, implementing online learning to handle concept drift in dynamic cloud traffic, applying explainable AI techniques such as SHAP to interpret model decisions, and investigating federated learning for privacy-preserving collaborative IDS across multiple cloud environments.

References

1. Moustafa, N., Slay, J. (2015). UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). 2015 Military Communications and Information Systems Conference (MilCIS), IEEE. https://research.unsw.edu.au/projects/unsw-nb15-dataset.

2. Sharafaldin, I., Lashkari, A. H., Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. ICISSP. https://www.unb.ca/cic/datasets/ 2017.html

A Code

Link: https://colab.research.google.com/drive/1CPGTponPtdugLcR5kOnrFRQ6yod75P2h