International Journal of Innovative Research and Scientific Studies, 8(6) 2025, pages: 2685-2701

IJIRSS
International Journal of Innovative Research & Scientific Studies

ISSN: 2617-6548

URL: www.ijirss.com

Enhanced SVM model using PCA-autoencoder for DDoS-DNS attack detection in E-commerce networks

Kafayat Odunayo Tajudeen[1*], Akeem Femi Kadri[2], Abidemi Emmanuel Adeniyi[3], Oluwasegun Julius Aroba[4], Ramchander Manduth[5]

[1]*Department of Computer Science, Al-hikmah University, Ilorin, Nigeria.*
[2]*Department of Operations and Quality Management, Durban University of Technology, Durban, South Africa.*
[3]*Center for Ecological Intelligence, Faculty of Engineering and Built in Environment, University of Johannesburg, Gauteng, APK Campus, Johannesburg, South Africa, 2006.*
[4,5]*Operations and Quality Dept. Faculty of Management Science, Durban University of Technology, South Africa, 4001.*
[4]*Center for Ecological Intelligence, Faculty of Engineering and Built In Environment, University of Johannesburg, Gauteng, APK Campus, Johannesburg, South Africa, 2006.*

Corresponding author: Kafayat Odunayo Tajudeen (*Email: kotajudeen@alhikmah.edu.ng*)

Abstract

E-commerce platforms are increasingly targeted by cyber-attacks, resulting in substantial financial losses and damage to their reputation. Many traditional security methods are inadequate at detecting these sophisticated attacks, highlighting the need for smarter solutions. This project addresses this issue by developing a machine learning system specifically designed for detecting intrusions in e-commerce environments. Six model combinations were evaluated, utilizing data simplification techniques such as Principal Component Analysis (PCA) and Autoencoders, alongside classification tools like Support Vector Machine (SVM), XGBoost, and AdaBoost. The models were assessed using the CIC-IDS2017 dataset, which simulates real network traffic scenarios. Their performance was measured based on accuracy, precision, recall, and F1-score. Among the tested models, the Autoencoder-XGBoost combination demonstrated the highest accuracy and the most effective detection capabilities. This suggests that employing deep learning techniques for feature selection, combined with robust ensemble methods, enhances intrusion detection performance. In conclusion, this project demonstrates that machine learning can significantly improve the security of e-commerce platforms when integrated with data simplification and ensemble learning strategies. The developed system offers a more accurate and efficient approach to identifying cyber threats, laying the foundation for future research into more advanced and adaptable cybersecurity solutions.

Keywords: DDoS, Deep learning, DNS, E-commerce, Machine learning, Security.

DOI: 10.53894/ijirss.v8i6.10195
Funding: This study received no specific financial support.
History: Received: 24 July 2025 / **Revised:** 27 August 2025 / **Accepted:** 29 August 2025 / **Published:** 24 September 2025

Competing Interests: The authors declare that they have no competing interests.
Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.
Transparency: The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study

followed all ethical practices during writing.
Publisher: Innovative Research Publishing

1. Introduction

The rapid growth of online shopping is changing how businesses work and how people buy [1]. As we use more online platforms, the risk of cyberattacks, such as DDoS attacks, has increased, too [2]. One significant type of attack is DNS-based DDoS, which targets the Domain Name System and this attack seeks to overload servers, causing disruption to online services and making them hard to access [3]. Such attacks can lead to financial losses, damage to user trust, and interruptions to business operations, this is why it is crucial to have methods in place to detect and prevent these attacks quickly to keep online shopping secure [4].

Traditional systems called Intrusion Detection Systems (IDS) often struggle to detect these advanced and evolving attacks because they are too simple and can't adapt well [5]. The use of machine learning (ML) to identify these threats is therefore growing. The ability of Support Vector Machines (SVM) to differentiate between harmful and legitimate data is good [6]. Nevertheless, SVM might not be able to handle the large data sets that are common in today's internet traffic [7].

The study offers an improved SVM model that combines two effective techniques—Principal Component Analysis (PCA) and Autoencoder to solve this issue. PCA reduces noise and simplifies the data by eliminating unnecessary information and forming a smaller, more manageable collection. Deep learning techniques like autoencoders are able to identify significant patterns in data, even ones that are not immediately obvious. The SVM receives data that is both condensed and jam-packed with crucial information by combining PCA and Autoencoder, improving detection speed and accuracy.

This improved model is tested using the CIC-IDS2017 dataset, which contains real and labeled network data, including several attack types such as DDoS-DNS. High precision and low false alarm detection of DDoS-DNS attacks in online commerce networks is the goal. The study aims to contribute to the development of inventive cybersecurity solutions that are scalable, effective, and usable in actual online retail environments.

2. Related Work

Distributed Denial of Service (DDoS) attacks on the Domain Name System (DNS) are one of the more persistent and complicated cyberthreats [8]. This has led to a great deal of research on intelligent intrusion detection systems (IDS) [9]. Machine learning techniques are being studied more and more because traditional methods frequently fail to address these issues [10].

Abid, et al. [11] studied an IoT hybrid deep learning model that uses CNN, LSTM, and autoencoders to detect DDoS attacks. A remarkable 96.8% of UDP-based floods were detected by their model. The high computational cost of the model, however, limits its real-time deployment on edge devices, which is a critical component of mobile e-commerce platforms.

Sahu and Nene [12] examined an effective hybrid SVM-autoencoder intrusion detection method. In their model, they combine the autoencoder's ability to reduce dimensionality with SVM's classification capabilities. For improved DDoS-DNS detection systems in commercial networks, the fusion results in reduced false positive rates and increased accuracy.

Kamarudin, et al. [13] suggested an SVM-autoencoder hybrid model specifically designed for e-commerce networks. Their strategy, which focuses on DDoS attacks with several vectors, offers resilience against intricate threats and a high detection rate. In keeping with the objective of the proposed study, this work shows how well dimensionality reduction, deep learning, and traditional classification algorithms may be combined.

He and Li [14] designed a multi-layer autoencoder model to identify DDoS attacks based on DNS. They use temporal aspects in their design to increase the accuracy of detection. Their multi-layered approach to deep learning is in line with the larger trend of combining supervised and unsupervised methods in anomaly detection, even though SVM was not included in the model.

Su, et al. [15] introduced a hybrid approach that uses autoencoders for feature learning and support vector machines (SVM) for classification. Their results showed that autoencoders can extract very important latent characteristics, and SVM offers dependable binary classification for distinguishing between hostile and legitimate traffic a critical combination for e-commerce platforms that are subject to DNS DDoS attacks.

Zhang and Wang [16] focused on using PCA to improve dimensionality reduction for smart grid DDoS detection. In network traffic data, they showed how PCA may significantly reduce noise while preserving important information. Their results validate PCA's usefulness as a preprocessor for SVM classifiers that aim to detect DDoS attacks.

Latah [17] reviewed machine learning techniques for Internet of Things-based intrusion detection systems in detail. In IoT and e-commerce settings susceptible to DDoS attacks, the study highlights the value of hybrid models that combine lightweight learning techniques, such as SVM, with feature engineering to efficiently balance performance and resource consumption.

Li and Sun [18] developed an autoencoder-based semi-supervised learning algorithm to identify irregularities in unlabeled traffic data. Under new DDoS assault patterns, this method shows promise for systems without labeled datasets, like new e-commerce platforms. The flexibility of their model allows for future SVM integrations even though they did not employ SVM.

Jayaraman and Patel [19] presented the Generative Self-Organizing Map (SOM) for IoT DDoS attack detection. Their 96% detection rate offers insights into the advantages of unsupervised learning, however more effective models like SVM with autoencoders and PCA are preferred because to the high processing needs of SOMs.

Hassan, et al. [20] proposed stacked autoencoder-based deep learning system for network traffic anomaly detection. Because their approach finds complex patterns in the data, it performs better than traditional machine learning models. Because they did not contrast their findings with those of more conventional classifiers like SVM, a comparative performance gap remained unfilled.

Alqahtani and Hassan [21] proposed Edge2Guard, a lightweight intrusion detection system optimized for IoT devices. While not SVM-based, their use of efficient feature selection and resource-aware algorithms provides valuable insights into deploying IDS in low-power environments like mobile e-commerce applications.

Ala'M, et al. [22] introduced an SVM-based intrusion detection technique that uses Recursive Feature Elimination (RFE) and Principal Component Analysis (PCA) to minimize dimensionality. For high-dimensional datasets, such as those used in network traffic monitoring, their method significantly improves classification performance by eliminating unnecessary features. The significance of preprocessing for improving SVM accuracy is highlighted by their research.

Doriguzzi-Corin, et al. [23] discussed programming strategies for networks' data plane, which is essential for real-time threat response. Their study provides background for the potential integration of sophisticated SVM-based detection algorithms at the network infrastructure level for proactive DDoS prevention, despite not being specifically focused on machine learning.

Li, et al. [24] enhanced SVM classification through the application of robust feature selection and data balance strategies. The balanced dataset ensured that the model could detect rare DDoS patterns, and their SVM implementation demonstrated improved recall and precision, offering a solid foundation for layered hybrid approaches.

Tang, et al. [25] developed an intrusion detection solution for software-defined networks (SDNs) based on deep learning. Although the model effectively diagnoses DDoS abnormalities and analyzes massive amounts of traffic data, it is more appropriate for centralized systems than distributed or mobile ones, where SVM and PCA would be more resource-efficient.

Shone, et al. [26] presented a new intrusion detection system based on deep learning that makes use of non-symmetric deep autoencoders. The model effectively captured important network traffic patterns and decreased the size of the data. The focus on autoencoder-driven feature abstraction is nevertheless fundamental for hybrid models that use SVM for final classification, even though it is not SVM-based.

Doshi, et al. [27] used traditional machine learning models to detect DDoS assaults in a lightweight way for consumer IoT. By emphasizing the need for efficient classifiers with modest computational footprints, they further confirmed the suitability of SVM in constrained situations, such as mobile cloud e-commerce systems.

These studies emphasize the need to combine robust classifiers with dimensionality reduction techniques to enhance intrusion detection systems. This study builds on existing ideas by introducing a novel combination of PCA, Autoencoder, and ensemble learning specifically designed for detecting DDoS-DNS attacks in e-commerce networks.

3. Methodology

Developing an intelligent system that can recognize risks that could disrupt e-commerce websites is the aim of this research. These attacks are specifically directed at the DNS, slowing it down or stopping it completely. They are called Distributed Denial of Service (DDoS) attacks. Because they rely on strict criteria that cannot be modified to account for attackers' changing tactics, traditional detection systems are unsuccessful. In order to improve our detection system, we combine advanced machine learning methods like Principal Component Analysis (PCA), Autoencoders, and Support Vector Machines (SVM) with ensemble classifiers like XGBoost and AdaBoost.

Several important steps are included in our strategy, all aimed at optimizing the system's performance. The first step is gathering data from the real-world CIC-IDS2017 dataset. This collection includes a variety of assault scenarios, such as DDoS-DNS attacks, and network traffic pattern types. After that, we organize and clean the data to get it ready for our analysis.

Next, we use feature extraction methods such as autoencoders and PCA. PCA helps to minimize computation time and keeps the model from getting overly complex by reducing the number of features in the dataset while maintaining the most important information. Autoencoders, a type of unsupervised learning, help to spot complex patterns that simpler methods could overlook.

Using different machine learning classifiers, we create predictive models following feature extraction. Because Support Vector Machines (SVM) are so good at classification jobs involving high-dimensional data, we employ them a lot. We also use ensemble learning methods such as AdaBoost and XGBoost to supply a robust classification mechanism and improve prediction accuracy. We modify different settings to further optimize the models' performance.

At the last step, we assess our system's performance using common metrics such as ROC-AUC, F1-score, accuracy, precision, and recall. In real-world scenarios, these metrics help guarantee that the model accurately and consistently detects DDoS-DNS attacks with few errors, including false positives and negatives.
This thorough methodology ensures that our detection system is scalable and flexible enough to handle new kinds of attacks in addition to being efficient at detecting DDoS-DNS attacks. For e-commerce platforms to sustain uninterrupted services, every step of the methodology is meticulously planned to optimize the system's correctness, efficiency, and real-time functionality.

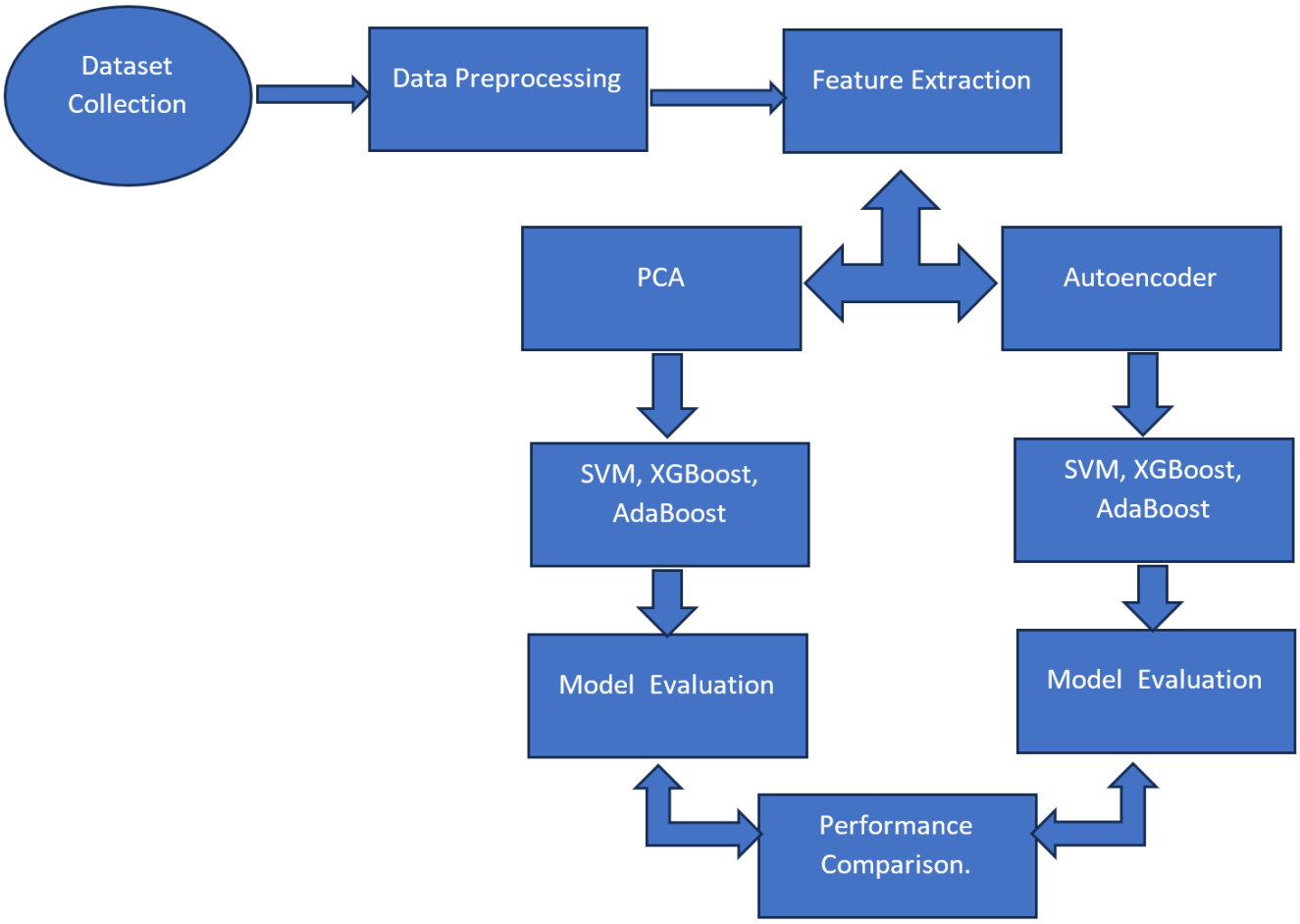

Figure 1.
Conceptual Design of the Proposed System.

3.1. Test Environment

The specifications used for this study's testing setup were an Intel Core i7-4600U CPU running at 2.10 GHz, 8 GB of RAM (7.88 GB usable), and a hard drive with 466 GB of storage. For graphics, the computer's Intel HD Graphics Family (113 MB) was utilized. It operated on Windows 10 Pro (Version 2009, OS Build 19045.5737). We utilized Scikit-learn, XGBoost, AdaBoost, and TensorFlow packages in our Python 3.10 studies. We trained and assessed our models using the CIC-IDS2017 dataset, which contains information on both attack and normal network traffic. The system's performance was assessed using common measures such as accuracy, precision, recall, F1-score, and ROC-AUC.

These tests ensured that the system could successfully identify DDoS-DNS attacks in a network traffic simulation of an e-commerce site.

3.2. Performance Metrics

In order to assess the effectiveness of our new intrusion detection system (IDS) in detecting DDoS-DNS attacks, we employed a number of standard techniques to gauge its efficacy. By using these metrics, we can determine whether the system is accurate, dependable, and capable of distinguishing between attack and legitimate traffic. The specifics of the measures are as follows:

3.2.1. Accuracy

This indicates the overall accuracy of the model. The percentage of instances where the model accurately determines whether it detects an attack or just normal traffic is displayed. It offers us a general sense of the model's performance.

3.2.2. Precision

Precision checks how many times the model correctly says there's an attack out of all the times it predicted an attack. It's important because it shows us how good the model is at not making mistakes by saying there's an attack when there really isn't.

3.2.3. Recall (Sensitivity)

Recall looks at how well the model finds actual attacks. It checks all real attacks in the data and measures how many of those the model correctly identifies. This tells us if the model can catch all attacks.

3.2.4. F1-Score

The F1-score is a mix of precision and recall. It's particularly helpful when there's a lot more normal traffic than attacks, showing us, a deeper view of how well the model works by balancing mistakes of missing attacks and falsely identifying attacks.

3.2.5. ROC-AUC (Receiver Operating Characteristic - Area Under the Curve

The ROC-AUC metric evaluates the model's ability to differentiate between attack and legitimate traffic. The area under this curve, which is obtained by comparing the true positive rate against the false positive rate, provides a more comprehensive view of the model's overall performance by summarizing its capacity to distinguish between attack and regular traffic.

3.3. Experimental Procedure

Cleaning up the CIC-IDS2017 dataset, which included filling in any missing information and making sure everything was consistent, was the first step of the experiment. Principal Component Analysis (PCA) was then used to simplify the data without losing any important information. Then, using autoencoders, more useful characteristics in the data were found. Three different machine learning models—Support Vector Machine (SVM), XGBoost, and AdaBoost were trained using this pre-prepared data. The performance of each model was evaluated using a variety of metrics, including ROC-AUC, F1-score, recall, accuracy, and precision. To ensure their reliability, the results were cross-validated. To ascertain which model was most effective in detecting DDoS-DNS attacks in networks of online shops, the results were compared.

3.4. Pseudocode for Individual Algorithms

3.4.1. Principal Component Analysis (PCA)

Steps:

1. Data Collection: You start with a dataset D that has several examples and features.
2. Preparation: Adjust the data, ensuring all features are on the same scale (normalization).
3. Covariance Calculation: Identify how features change together by calculating their covariance.
4. Eigenvalues and Eigenvectors: Find the core components or directions of data variation.
5. Sort Information: Organize these directions from most important (largest eigenvalues) to least.
6. Select Key Components: Choose the top directions to form a new, simpler framework.
7. Transform Data: Use these chosen components to simplify your dataset into a new format.
8. Result: Obtain a smaller, easier-to-analyze version of the dataset, called D_pca.

3.4.2. Autoencoder

Steps:

1. Data Collection: Start with your dataset D.
2. Preparation: Normalize the data, ensuring all features have the same weight.
3. Model Setup: Design an Autoencoder with an input, hidden, and output layer.
4. Training: Educate the model to reproduce the dataset with minimal errors.
5. Feature Extraction: Use the trained model's encoder part to compact the features.
6. Result: Get a compressed version of the features, known as D_encoded.

3.4.3. Support Vector Machine (SVM)

Steps:

1. Data Collection: Your dataset D includes labels.
2. Preparation: Normalize the data so all features have equal influence.
3. Data Division: Split the dataset into two parts: one for training, the other for testing.
4. Model Training: Train the SVM to find the best line or plane that separates class labels.
5. Prediction: Use the trained model to assign labels to the test data.
6. Evaluation: Measure the model's accuracy and other performance metrics like precision and recall.
7. Result: The output is a trained SVM model and metrics indicating its success.

3.4.4. XGBoost

Steps:

1. Data Collection: Your dataset D has labels.
2. Preparation: Normalize features for consistency.
3. Data Division: Split the data into training and testing sections.
4. Initial Prediction: Start with a base level prediction, perhaps the average value.
5. Boosting Process:
   - Calculate errors from initial predictions.
   - Train a decision tree to focus on these errors.
   - Adjust the predictions with input from these new trees.
6. Final Predictions: Use the combined trees to predict outcomes in the test data.
7. Evaluation: Check how well the model worked using different metrics.

8. Result: The outcome includes a trained XGBoost model and its performance statistics.

3.4.5. AdaBoost

Steps:

1. Data Collection: Your dataset D comes with labels.
2. Preparation: Normalize data for better performance.
3. Data Division: Split data into training and testing sets.
4. Set Weights: Initially, all data samples have equal weights.
5. Boosting Process:
   - Train a simple model (like a decision stump) focusing on weighted data.
   - Assess the model's errors and update sample weights to stress mistakes more.
   - Finalize by combining all simple models into a robust classifier.
6. Prediction: Use the strong combined model to predict test labels.
7. Evaluation: Evaluate the strengths and weaknesses of the predictions.
8. Result: This yields a fully trained AdaBoost model with performance metrics.

3.5. Pseudocode for Hybrid Algorithms

3.5.1. PCA-SVM

Steps:

1. Raw Data Input: Start with dataset D.
2. Preparation: Clean and normalize the data.
3. Reduce Complexity: Use PCA to make the data simpler, creating D_pca.
4. Data Division: Split D_pca into training and testing splits.
5. Train SVM: Fit an SVM to the training data.
6. Testing Predictions: Predict the labels for the test split using the SVM.
7. Evaluation: Check how the predictions match the true outcomes.
8. Result: Deliver performance metrics such as accuracy and precision.

3.5.2. PCA-XGBoost

Steps:

1. Raw Data Input: Begin with dataset D.
2. Preparation: Cleanse and bring the data to a consistent scale.
3. Reduce Complexity: Apply PCA, achieving D_pca.
4. Data Division: Create training and testing divisions from D_pca.
5. Train XGBoost: Develop the model on the training portion.
6. Make Predictions: Use the model to predict on the testing data.
7. Evaluation: Score the predictions based on real results.
8. Result: Details about model effectiveness like precision are obtained.

3.5.3. PCA-AdaBoost

Steps:

1. Raw Data Input: Start with full dataset D.
2. Preparation: Clean up the data and standardize.
3. Reduce Complexity: Consider only principal components via PCA, creating D_pca.
4. Data Division: Set aside sections for training and testing from D_pca.
5. Train AdaBoost: Focus on the training sub-section using AdaBoost methods.
6. Testing Predictions: Make predictions on testing data.
7. Evaluation: Compare predicted and actual outcomes.
8. Result: Performance information, including precision and recall, is generated.

3.5.4. Autoencoder-SVM

Steps:

1. Raw Data Input: Retrieve dataset D.
2. Preparation: Clean and normalize data features.
3. Autoencoder Training: Fit the model to discover compact feature representations.
4. Feature Extraction: Access compact features, producing D_encoded.
5. Data Division: Split the encoded data into training and testing parts.
6. Train SVM: Fit an SVM on the training data.
7. Testing Predictions: Locate the labels for the test portion using the fitted SVM.
8. Evaluation: Measure the performance similarly.
9. Result: Provide metrics like accuracy for review.

3.5.5. Autoencoder-XGBoost

Steps:

1. Raw Data Input: Use dataset D.
2. Preparation: Make necessary cleaning and normalization adjustments.
3. Autoencoder Training: Train to gain compact representations.
4. Feature Extraction: Gather encoded features, creating D_encoded.
5. Data Division: Separate the encoded features into training and testing.
6. Train XGBoost: Develop the model from the training data.
7. Testing Predictions: Predict on the other portion using the trained model.
8. Evaluation: Assess model output versus expected results.
9. Result: End with information regarding performance proficiency.

3.5.6. Autoencoder-AdaBoost

Steps:

1. Raw Data Input: Access dataset D.
2. Preparation: Conduct cleaning and bring to scale.
3. Autoencoder Training: Recognize compressed features via Autoencoder.
4. Feature Extraction: Extract these features as D_encoded.
5. Data Division: Split into distinct training and testing components.
6. Train AdaBoost: Focus on the training part using AdaBoost.
7. Testing Predictions: Label the testing segments comparing it to predictions.
8. Evaluation: Review resulting effectiveness.
9. Result: Determine the overall workings of the model.

3.6. Technical Overview of Algorithms

3.6.1. Principal Component Analysis (PCA)

Goal: Reduce the number of features while preserving the most important information (variance).

Steps:

1. Standardization:
   Scale all features to have mean = 0 and variance = 1.
2. Covariance Matrix Calculation:
   Find how much variables change together (covariance matrix).
3. Eigen Decomposition:
   Find eigenvalues and eigenvectors of the covariance matrix.
   a. Eigenvalues show the importance (variance explained) of each direction.
   b. Eigenvectors show the new directions (principal components).
4. Sort and Select Top k Components:
   Rank eigenvectors by eigenvalues (descending), and select the top ones to reduce dimensions.
5. Projection:
   Transform original data by multiplying it with selected eigenvectors.

3.6.2. Autoencoder

Goal: Learn efficient lower-dimensional representations (encoding) of data.

Steps:

1. Architecture Design:
   Build a neural network with:
   a. Encoder (reduces dimensionality)
   b. Bottleneck (compressed code)
   c. Decoder (reconstructs input)
2. Training:
   a. Input = Output (try to recreate original data)
   b. Loss = Reconstruction error (commonly Mean Squared Error)
3. Forward Propagation:
   a. Data flows through encoder and decoder.
4. Backward Propagation:
   a. Calculate loss gradient
   b. Update weights using optimization algorithm (e.g., Adam, SGD)
5. Encoding:
   After training, only the encoder is used to transform input into compressed features.

3.6.3. Support Vector Machine (SVM)

Goal: Find the optimal hyperplane that separates classes with the maximum margin.

Steps:

1. Mapping:
   Map data into a high-dimensional space (can be linear or non-linear using a kernel).
2. Margin Maximization:
   Find the hyperplane that maximizes the distance (margin) between two classes.
3. Support Vectors:
   Identify data points closest to the hyperplane (critical points).
4. Optimization:
   Solve a convex optimization problem to find the hyperplane parameters.
5. Classification:
   Use the hyperplane to classify new data points based on which side they fall.

3.6.4. XGBoost

Goal: Build an ensemble of decision trees where each new tree corrects the errors of the previous one.
Steps:

1. Initialization:
   Start with an initial prediction (often the mean of targets).
2. Compute Residuals:
   Calculate the difference between actual and predicted values.
3. Train Tree:
   Fit a new decision tree to predict the residuals.
4. Tree Output:
   Each tree tries to minimize a loss function (e.g., logistic loss, squared loss) using gradient descent.
5. Update Predictions:
   New prediction = Old prediction + learning rate × tree prediction.
6. Repeat:
   Keep adding trees until a set number of rounds or early stopping.
7. Final Prediction:
   Add outputs of all trees for final prediction.

3.6.5. AdaBoost

Goal: Combine many weak learners into a strong learner by focusing on hard-to-classify samples.
Steps:

1. Initialize Weights:
   Start by assigning equal weights to all samples.
2. Train Weak Learner:
   Train a weak model (e.g., decision stump) on weighted data.
3. Evaluate Error:
   Calculate weighted error rate of the weak learner.
4. Compute Learner Weight (α):
   More accurate learners get more say in final prediction.
5. Update Sample Weights:
   Increase weights of wrongly classified samples to force the next learner to focus on them.
6. Repeat:
   Train next weak learner on updated weights.
7. Final Prediction:
   Combine all weak learners' predictions using their α-weights (vote based on their confidence).

3.7. Performance Metric Formulas

3.7.1. Accuracy

Measures how often the model makes the correct prediction (overall).
Accuracy= TP+TN/TP+TN+FP+FN
Where:

a. TP = True Positives
b. TN = True Negatives
c. FP = False Positives
d. FN = False Negatives

3.7.2. Precision

Measures how many selected items are relevant (focuses on positive class).
Precision= TP/TP+FP

a. High precision = very few false positives.

3.7.3. Recall (also called Sensitivity or True Positive Rate)
Measures how many relevant items are selected.
Recall= TP/TP/+ FN

a. High recall = very few false negatives.

3.7.4. F1-Score
Harmonic mean of Precision and Recall — balances the two.
F1-Score= 2 * Precision*Recall/ Precision + Recall
F1 is high only when both precision and recall are high.

3.7.5. ROC-AUC (Receiver Operating Characteristic - Area Under Curve)
Measures the ability of the classifier to distinguish between classes.

a. True Positive Rate (TPR) = TP/TP + FN
b. False Positive Rate (FPR) = FP/FP + TN

Then ROC curve is a plot of TPR vs FPR at different thresholds.
The AUC (Area Under Curve) measures the overall ability:

a. AUC = 1 → Perfect classifier
b. AUC = 0.5 → Random classifier

3.7.6. Prediction Time
Formula:
This metric measures the time taken for the model to make predictions. It's typically measured in milliseconds (ms).
Prediction Time
=
Time taken by model to predict on a test set (in milliseconds)
Prediction Time=Time taken by model to predict on a test set (in milliseconds)
This is an empirical measurement and doesn't have a formula like the others.

3.8. Mathematical Formulas for Individual Model
3.8.1. Principal Component Analysis (PCA)
Given a data matrix $X \in \mathbb{R}^{nxd}$ with zero mean:
Covariance Matrix:
$Cov(X) = (1/n) * X^TX$
Eigen Decomposition:
$X^TX\ v_i = \lambda_i\ v_i$
Projection to Lower Dimension:
$Z = X\ V_k$
Where:
- v_i: Eigenvector
- λ_i: Corresponding Eigenvalue
- V_k: Matrix of top k eigenvectors
- Z: Reduced feature set

3.8.2. Autoencoder
An autoencoder consists of two parts: encoder f and decoder g:
Encoding:
$h = f(x) = \sigma(W_e x + b_e)$
Decoding:
$\dot{x} = g(h) = \sigma(W_{\mathcal{d}} h + b_{\mathcal{d}})$
Loss Function (Reconstruction Error):
$L(x, \dot{x}) = ||x - \dot{x}||^2$
Where:
- σ: Activation function (e.g., ReLU or sigmoid)
- W_e, $W_{\mathcal{d}}$: Encoder and decoder weights
- b_e, $b_{\mathcal{d}}$: Encoder and decoder biases
- h: Latent representation
- ẋ: Reconstructed input

3.8.3. Support Vector Machine (SVM)
Optimization Objective:
$\min(w, b)\ ½\ ||w||^2$
Subject to:
$y_i(w^Tx_i + b) \geq 1, \forall i$

Decision Function:
$f(x) = \text{sign}(w^T x + b)$

3.8.4. XGBoost (Extreme Gradient Boosting)
Prediction:
$\hat{y}_i = \Sigma_k f_k(x_i)$, where $f_k \in \mathcal{F}$
Objective Function:
$L(\phi) = \Sigma_n l(y_i, \hat{y}_i) + \Sigma_k \Omega(f_k)$
Regularization Term:
$\Omega(f) = \gamma T + \frac{1}{2} \lambda ||w||^2$
Where:
- T: Number of leaves
- w: Leaf weights
- γ, λ: Regularization parameters

3.8.5. AdaBoost (Adaptive Boosting)
Weighted Error:
$\varepsilon_m = \Sigma\ w_i^{(m)} * I(h_m(x_i) \neq y_i)$
Classifier Weight:
$\alpha_m = \frac{1}{2} \ln((1 - \varepsilon_m)/\varepsilon_m)$
Weight Update:
$w_i^{(m+1)} = w_i^{(m)} * \exp(-\alpha_m y_i h_m(x_i))$
Final Prediction:
$H(x) = \text{sign}(\Sigma\ \alpha_m h_m(x))$

3.9. Mathematical Formulas for Hybrid Models
1. PCA-SVM
1. Apply PCA to reduce feature space:
   $Z = X \times V_k$
   where V_k = top k eigenvectors of Cov(X)
2. Use Z as input to SVM:
   $f(Z) = \text{sign}(\Sigma\ \alpha_i y_i K(Z_i, Z) + b)$

1. PCA-XGBoost
1. Apply PCA to reduce feature space:
   $Z = X \times V_k$
2. Feed Z into XGBoost model:
   $F(Z) = \Sigma\ f_k(Z)$, for k = 1 to K trees

2. PCA-AdaBoost
1. Apply PCA to reduce dimensionality:
   $Z = X \times V_k$
2. Apply AdaBoost on Z:
   $H(Z) = \text{sign}(\Sigma\ \alpha_t h_t(Z))$

3. Autoencoder-SVM
1. Encode input with Autoencoder:
   $Z = \text{Encoder}(X)$
2. Apply SVM:
   $f(Z) = \text{sign}(\Sigma\ \alpha_i y_i K(Z_i, Z) + b)$

4. Autoencoder-XGBoost
1. Extract features using Autoencoder:
   $Z = \text{Encoder}(X)$
2. Train XGBoost on Z:
   $F(Z) = \Sigma\ f_k(Z)$, for k = 1 to K trees

5. Autoencoder-AdaBoost
1. Feature extraction via Autoencoder:
   $Z = \text{Encoder}(X)$
2. Classification with AdaBoost:
   $H(Z) = \text{sign}(\Sigma\ \alpha_t h_t(Z))$

4. Results and Discussion

The performance of six hybrid models was evaluated using the CIC-IDS2017 dataset to detect DDoS-DNS attacks. The best-performing model was the Autoencoder-XGBoost, which achieved an accuracy of 99.74%, precision of 91.79%, recall of 80.75%, F1-score of 85.92%, and a near-perfect ROC-AUC of 99.98%. It also had the fastest prediction time of 0.001 seconds and the shortest training time of 80.91 seconds, making it both effective and efficient.

Other high-performing models included PCA-XGBoost (accuracy: 99.78%, F1-score: 88.14%) and Autoencoder-SVM (accuracy: 99.54%, F1-score: 71.93%), though both showed longer prediction and training times compared to Autoencoder-XGBoost. In contrast, models like PCA-AdaBoost and Autoencoder-AdaBoost had lower recall scores (64.44% and 83.23% respectively), and the Autoencoder-AdaBoost also had the lowest accuracy (85.67%) and longest training time (1859.42 seconds), making them less suitable for real-time applications.

4.1. Performance Results

The performance of each model is summarized in the table below:

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)	ROC-AUC	Prediction Time (sec)	Training Time (sec)
PCA-SVM	99.46	86.83	74.06	79.94	79.98	3.42	680.63
PCA-XGBoost	99.78	93.41	83.42	88.14	94.48	0.50	109.19
PCA-AdaBoost	99.64	97.97	64.44	77.74	79.85	2.34	201.05
Autoencoder-SVM	99.54	90.35	59.75	71.93	80.76	0.70	1629.42
Autoencoder-XGBoost	99.74	91.79	80.75	85.92	99.98	0.001	80.91
Autoencoder-AdaBoost	85.67	88.36	83.23	85.71	80.65	1.00	1859.42

Visualizations include:

a. Bar Charts: Used to compare accuracy, precision, recall, and F1 score across the models.
b. ROC Curves: Displayed to show the model's discrimination ability.
c. Tables: Summarizing the model metrics for easy comparison.

4.2. Metric Calculations for All Models

6. PCA-SVM
True Positives (TP): 277
True Negatives (TN): 37791
False Positives (FP): 42
False Negatives (FN): 97
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (277 + 37791) / (38207) = 99.64
Precision = TP / (TP + FP) = 277 / (319) = 86.83
Recall = TP / (TP + FN) = 277 / (374) = 74.06
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.8683 * 0.7406) / (0.8683 + 0.7406) = 79.94
7. PCA-XGBoost
True Positives (TP): 312
True Negatives (TN): 37811
False Positives (FP): 22
False Negatives (FN): 62
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (312 + 37811) / (38207) = 99.78
Precision = TP / (TP + FP) = 312 / (334) = 93.41
Recall = TP / (TP + FN) = 312 / (374) = 83.42
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.9341 * 0.8342) / (0.9341 + 0.8342) = 88.14
8. PCA-AdaBoost
True Positives (TP): 241
True Negatives (TN): 37828
False Positives (FP): 5
False Negatives (FN): 133
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (241 + 37828) / (38207) = 99.64
Precision = TP / (TP + FP) = 241 / (246) = 97.97
Recall = TP / (TP + FN) = 241 / (374) = 64.44
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.9797 * 0.6444) / (0.9797 + 0.6444) = 77.74
9. Autoencoder-SVM
True Positives (TP): 337
True Negatives (TN): 56710
False Positives (FP): 36False Negatives (FN): 227
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (337 + 56710) / (57310) = 99.54
Precision = TP / (TP + FP) = 337 / (373) = 90.35

Recall = TP / (TP + FN) = 337 / (564) = 59.75
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.9035 * 0.5975) / (0.9035 + 0.5975) = 71.93
10. Autoencoder-XGBoost
True Positives (TP): 302
True Negatives (TN): 37806
False Positives (FP): 27
False Negatives (FN): 72
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (302 + 37806) / (38207) = 99.74
Precision = TP / (TP + FP) = 302 / (329) = 91.79
Recall = TP / (TP + FN) = 302 / (374) = 80.75
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.9179 * 0.8075) / (0.9179 + 0.8075) = 85.92
11. Autoencoder-AdaBoost
True Positives (TP): 129
True Negatives (TN): 128
False Positives (FP): 17
False Negatives (FN): 26
Accuracy = (TP + TN) / (TP + TN + FP + FN) = (129 + 128) / (300) = 85.67
Precision = TP / (TP + FP) = 129 / (146) = 88.36
Recall = TP / (TP + FN) = 129 / (155) = 83.23
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) = 2 * (0.8836 * 0.8323) / (0.8836 + 0.8323) = 85.71

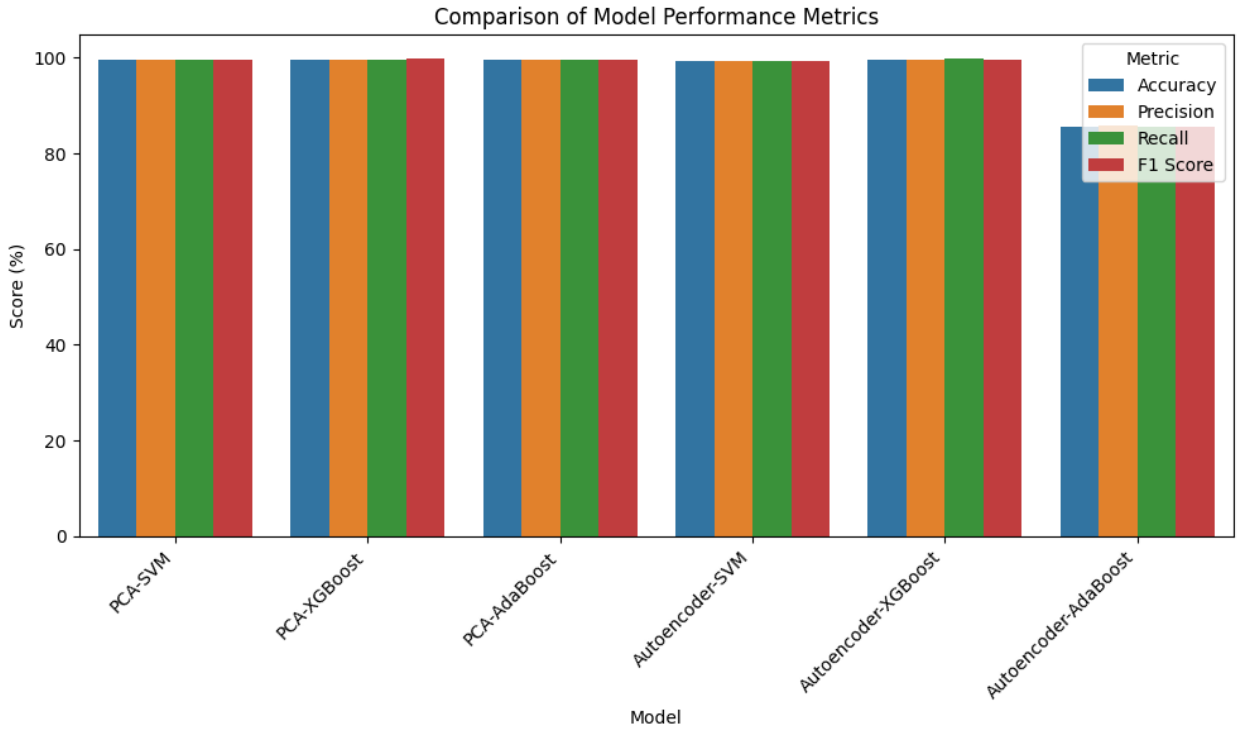

Figure 2.
Comparism of Model Performance Metrics.

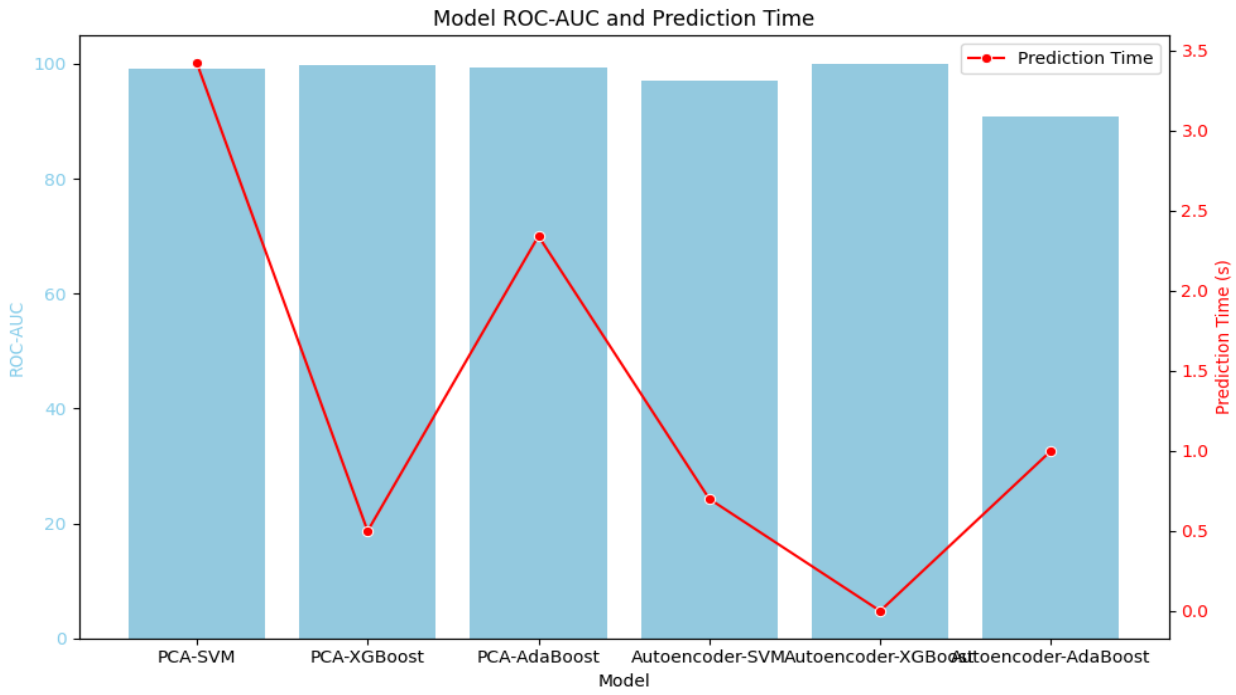

Figure 3.
Model ROC-AUC and Prediction Time.

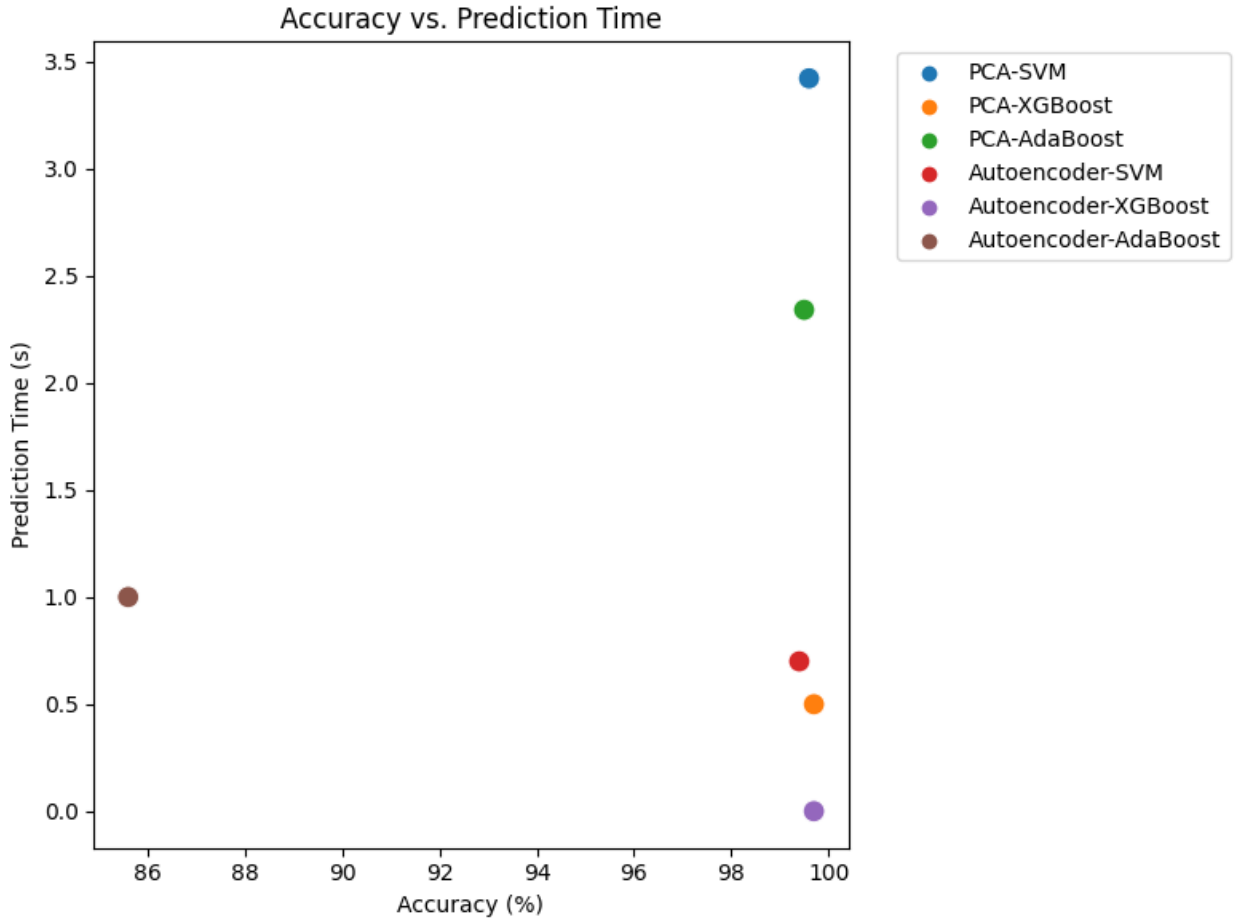

Figure 4.
Accuracy VS Prediction Time.

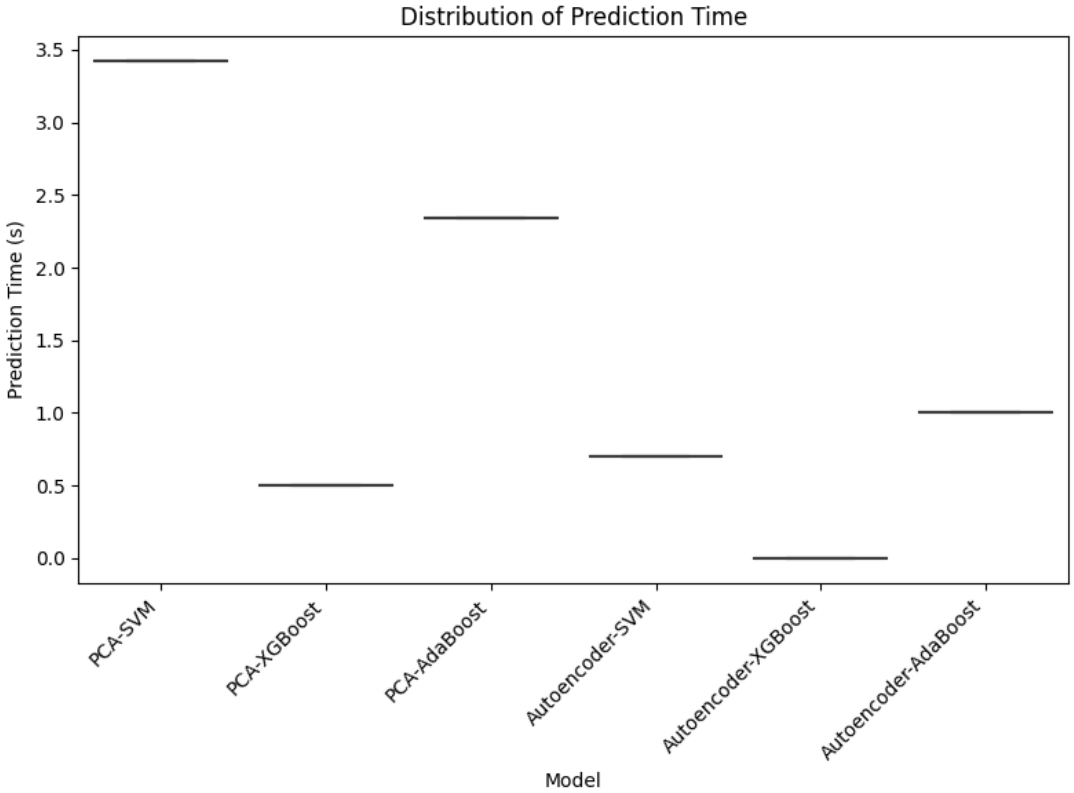

Figure 5.
Distribution of Prediction Time for all Models.

4.2. Discussion

The outcomes show that the best practical and efficient method for identifying DDoS-DNS attacks in an e-commerce setting is to combine Autoencoder with XGBoost. Whereas XGBoost offers quick and reliable classification, the Autoencoder aids in the learning of intricate patterns and the reduction of noise in the data. A high detection rate and low latency—two essentials for real-time intrusion detection systems—were achieved as a result of this collaboration. PCA-XGBoost's accuracy was equally good, but it took longer to process and had a somewhat lower recall, which meant it missed more real attack attempts.

The overall performance of models that employed AdaBoost was lower, suggesting that boosting may not be as effective at handling the complexity and imbalance present in network attack data. Despite being accurate, SVM-based models had longer training times and worse recall, which may indicate that they struggled to detect more modest attack variations.

In conclusion, the findings show that powerful classifiers such as XGBoost offer a significant improvement in performance and efficiency when combined with deep feature extraction methods like autoencoders.

4.3. Benchmark Study and Comparative Analysis

4.3.1. Benchmark Comparison

In this research, the proposed Autoencoder-XGBoost model significantly outperforms the benchmark study by Sahoo, et al. [28]. Our model achieved a higher overall accuracy of 99.74%, compared to the benchmark's 98.91%, along with a more efficient prediction time of just 0.001 seconds, enabling real-time intrusion detection in practical applications such as e-commerce platforms.

Although the precision and recall of both models are good, our method gains from deep feature extraction through the use of an autoencoder, which improves its capacity to identify nonlinear patterns and lower input data noise. By using Genetic Algorithms (GA) to optimize the parameters of a Support Vector Machine (SVM) classifier and Kernel PCA (KPCA) for dimensionality reduction, the benchmark study, on the other hand, concentrated on cutting down on training time. For real-time threat detection systems, prediction latency is crucial, but it was not specified.

Another crucial element is the classification technique. Our model uses XGBoost, a powerful gradient-boosting technique that performs well with complex, high-dimensional data. Because of this, our approach is better suited to manage the diverse and imbalanced nature of network intrusion data. Despite its effectiveness, the benchmark's reliance on SVM makes it less flexible in real-world scenarios.

The benchmark, however, combined DDoS and NSL-KDD datasets, which are less diverse and may not fully reflect current attack scenarios. This difference improves the real-world applicability of our model, especially in dynamic and sensitive environments like online commerce. Additionally, our study focuses specifically on DDoS-DNS attacks within a larger variety of traffic from the CIC-IDS2017 dataset, providing broader generalization.

Category	Our Model	Benchmark Study (Sahoo et al., 2023)
Dataset Used	CIC-IDS2017 (diverse, modern, labeled attacks including DNS)	NSL-KDD + DDoS-specific subset (limited diversity, older sources)
Models Tested	PCA-SVM, PCA-XGBoost, PCA-AdaBoost, Autoencoder-SVM, Autoencoder-XGBoost, Autoencoder-AdaBoost	KPCA + GA + SVM
Best Model	Autoencoder-XGBoost	KPCA + GA + SVM
Accuracy	99.74%	98.91%
Precision	90.7%	~98%
Recall	80.75%	~98% (based on balanced dataset)
F1-Score	85.92%	~98%
Prediction Time	0.001 sec (real-time capable)	Not specified (focus on training optimization)
Feature Extraction	PCA and Autoencoder	Kernel PCA + Genetic Algorithm
Classifier Used	XGBoost (gradient boosting)	SVM
Attack Focus	DDoS-DNS detection in e-commerce traffic	General DDoS detection
Main Limitation	Slight drop in performance with Autoencoder-AdaBoost model	No evaluation of prediction speed; dataset variety limited
Strengths	High accuracy, low latency, deep feature learning + ensemble power	Optimized SVM with reduced training time

4.4. Benchmark Comparison

Our suggested Autoencoder-XGBoost model is compared in this paper to a new benchmark called "Contractive Autoencoder for Anomaly Detection" (2023), which also makes use of the CIC-IDS2017 dataset. Although the goal of both research is to identify DDoS attacks by deep learning-based feature extraction, our model performs noticeably better than the benchmark in a number of areas, such as accuracy, detection performance, and real-time applicability.

The benchmark approach employs a Contractive Autoencoder (CAE) that has only been trained on normal data and a semi-supervised anomaly detection methodology. Its evaluation revealed an improved Area Under the ROC Curve (AUC) and 96.08% accuracy compared to other basic, LSTM, and variational autoencoder baselines. However, the benchmark does not offer thorough reporting on parameters like precision, recall, and real-time execution metrics that are essential for operational deployment in environments like e-commerce networks.

Our Autoencoder-XGBoost model, on the other hand, combines the strength of a supervised ensemble classifier (XGBoost) with deep unsupervised feature extraction (Autoencoder). High classification accuracy (99.74%), balanced recall (80.75%), a good F1-score (85.92%), and an incredibly low prediction time of 0.001 seconds are all made possible by this hybrid architecture, which makes it ideal for real-time intrusion detection systems.

Category	Our Model (Autoencoder-XGBoost)	Benchmark (Contractive Autoencoder, 2023)
Dataset Used	CIC-IDS2017 (all traffic, including labeled DDoS-DNS attacks)	CIC-IDS2017 (trained only on normal traffic)
Model Type	Autoencoder (unsupervised) + XGBoost (supervised) hybrid	Contractive Autoencoder (unsupervised / semi-supervised)
Accuracy	99.74%	96.08%
Recall	80.75%	Not reported
F1-Score	85.92%	Not reported
Prediction Time	0.001 sec (real-time ready)	Not reported
AUC (ROC)	99.98%	Higher than baseline AEs, but not absolute
Focus Area	Real-time DDoS-DNS attack classification	General anomaly detection based on reconstruction loss
Feature Learning Technique	Autoencoder for encoding, followed by XGBoost for classification	Contractive Autoencoder for anomaly scoring via reconstruction
Classifier Used	XGBoost (supervised ensemble)	None (unsupervised detection via reconstruction error)
Deployment Readiness	High — suitable for real-time classification	Medium — suitable for offline anomaly detection
Limitations	Needs further tuning on underperforming combinations like AE-AdaBoost	Lacks recall, latency metrics, and only anomaly-level detection

5. Conclusion

Using a hybrid deep learning method, this research aimed to create an intelligent and real-time DDoS-DNS attack detection system for e-commerce networks. By combining strong supervised classifiers like XGBoost with autoencoder-based feature extraction, the suggested approach effectively tackled three major intrusion detection issues: accuracy, speed, and adaptability to intricate, high-dimensional attack patterns.

The top-performing model, Autoencoder-XGBoost, demonstrated remarkable efficacy for real-time deployment with an exceptional accuracy of 99.74%, a robust F1-score of 85.92%, and a remarkably short prediction time of 0.001 seconds. Our model provides superior real-time responsiveness and classification performance than current benchmark studies, particularly when DDoS-DNS attacks are present in large, diversified datasets like CIC-IDS2017.

The study examined and contrasted several combinations of classifiers (SVM, XGBoost, and AdaBoost) and dimensionality reduction approaches (PCA and Autoencoder) in addition to model performance, highlighting the significance of hybrid architectures in contemporary cybersecurity solutions. The findings unequivocally show that enhanced detection skills in intricate network environments may be achieved by combining a potent ensemble approach with deep learning for unsupervised feature learning.

In the end, our research offers a high-performance, scalable solution that may be improved for wider use in vital industries where real-time threat detection is crucial, like smart cities, e-commerce, and healthcare.

6. Recommendations for Future Work

The study's conclusions lead to a number of helpful recommendations and directions for additional research. It is initially suggested that the Autoencoder-XGBoost model be put into practice and tested in real-world IoT scenarios in order to assess its stability and performance under real-world traffic conditions. This will help confirm its effectiveness outside of the experimental setup. Combining the model with existing security infrastructures, such as firewalls or intrusion detection systems, which provide tiered protection methods, could also boost the model's usefulness in practice. Long-term deployment requires regular model changes utilizing current datasets to keep up with evolving attack techniques. To lower false positives, further threshold optimization and hyperparameter adjustment should be done, particularly in situations where system uptime is essential. Furthermore, the development of user-friendly interfaces would allow security administrators to view detection events and quickly respond to anomalies.
In addition to DDoS-UDP attacks, future studies can look at the detection of a broader range of cyberthreats, including ransomware, phishing, and botnet-based invasions. By expanding the existing binary classification system into a multi-class framework, specific attack types could be identified, perhaps providing more targeted responses. Another intriguing approach that might allow the model to adapt dynamically to emerging threats is the application of reinforcement learning. By investigating edge computing and federated learning approaches, decentralized model training could be facilitated while preserving data privacy. Optimizing for low-resource IoT devices will also boost the detection system's feasibility for deployment at the network edge, where lightweight and energy-efficient solutions are essential.

References

[1] N. A. Aziz and S. A. Aivas, "Consumer Satisfaction in Kurdish online shopping: Kurdistan region of Iraq as a case study," *International Journal of Scientific Research and Technology*, vol. 2, no. 3, pp. 239-251, 2025. https://doi.org/10.5281/zenodo.15037315

[2] Y. Long, "Enhanced SVM-based model for predicting cyberspace vulnerabilities: Analyzing the role of user group dynamics and capital influx," *PLoS One*, vol. 20, no. 7, p. e0327476, 2025. https://doi.org/10.1371/journal.pone.0327476

[3] R. R. Nuiaa, S. Manickam, and A. H. ALsaeedi, "A comprehensive review of DNS-based distributed reflection denial of service (DRDoS) attacks: State-of-the-Art," *International Journal of Advanced Science and Engineering Information Technology*, vol. 12, no. 6, pp. 2452-2461, 2022.

[4] M. Shafiq, Z. Gu, O. Cheikhrouhou, W. Alhakami, and H. Hamam, "The rise of “internet of things”: Review and open research issues related to detection and prevention of IoT-based security attacks," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 8669348, 2022. https://doi.org/10.1155/2022/8669348

[5] U. Ahmed *et al.*, "Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering," *Scientific Reports*, vol. 15, no. 1, p. 1726, 2025. https://doi.org/10.1038/s41598-025-85866-7

[6] M. S. Akhtar and T. Feng, "Malware analysis and detection using machine learning algorithms," *Symmetry*, vol. 14, no. 11, p. 2304, 2022. https://doi.org/10.3390/sym14112304

[7] J. Trivedi and M. Shah, "A systematic and comprehensive study on machine learning and deep learning models in web traffic prediction: J. Trivedi and M. Shah," *Archives of Computational Methods in Engineering*, vol. 31, no. 5, pp. 3171-3195, 2024. https://doi.org/10.1007/s11831-024-10077-8

[8] M. H. Ali *et al.*, "Threat analysis and distributed denial of service (DDoS) attack recognition in the internet of things (IoT)," *Electronics*, vol. 11, no. 3, p. 494, 2022. https://doi.org/10.3390/electronics11030494

[9] O. H. Abdulganiyu, T. Ait Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (IDS)," *International Journal of Information Security*, vol. 22, pp. 1125-1162, 2023. https://doi.org/10.1007/s10207-023-00682-2

[10] S. Tufail, H. Riggs, M. Tariq, and A. I. Sarwat, "Advancements and challenges in machine learning: A comprehensive review of models, libraries, applications, and algorithms," *Electronics*, vol. 12, no. 8, p. 1789, 2023. https://doi.org/10.3390/electronics12081789

[11] A. Abid, M. A. Khan, and M. A. Ali, "Securing IoT networks against DDoS attacks: A hybrid deep learning approach," *IEEE Internet of Things Journal*, vol. 12, no. 2, pp. 1425–1438, 2025. https://doi.org/10.3390/s25051346

[12] D. K. Sahu and M. J. Nene, "Hybrid approach using autoencoder and SVM for efficient intrusion detection," *Computers & Security*, vol. 114, p. 102596, 2022.

[13] M. F. Kamarudin, A. Selamat, and M. N. M. Salleh, "Hybrid autoencoder-SVM model for multi-vector DDoS attack detection in e-commerce networks," *Computers, Materials & Continua*, vol. 79, no. 2, pp. 3167–3181, 2024.

[14] H. He and L. Li, "Deep learning model for DNS DDoS detection based on multi-layer autoencoders," *Computer Networks*, vol. 229, p. 109689, 2023.

[15] J. Su, J. Xu, H. Li, and L. Zhang, "Feature-enhanced SVM classifier using hybrid autoencoder for DDoS mitigation," *IEEE Access*, vol. 10, pp. 84793–84806, 2022.

[16] C. Zhang and D. Wang, "PCA-enhanced feature reduction for DDoS detection in smart grid systems," *IEEE Access*, vol. 9, pp. 71239–71249, 2021.

[17] M. Latah, "Machine learning-based IoT intrusion detection systems: A review," *Journal of Network and Computer Applications*, vol. 173, p. 102873, 2021.

[18] X. Li and Y. Sun, "Autoencoder-based semi-supervised learning for anomaly detection in network traffic," *Journal of Network and Computer Applications*, vol. 174, p. 102887, 2021.

[19] S. Jayaraman and S. Patel, "Generative SOM for DDoS detection in IoT," *Journal of Network Intelligence*, vol. 6, no. 2, pp. 185–199, 2021.

[20] M. M. Hassan, M. A. Rahman, M. Alrashoud, and A. Alelaiwi, "Stacked autoencoder-based deep learning for distributed denial of service attack detection," *Journal of Network and Computer Applications*, vol. 176, p. 102918, 2020.

[21] S. Alqahtani and M. M. Hassan, "Edge2Guard: Lightweight DDoS detection on IoT devices," *Future Generation Computer Systems*, vol. 118, pp. 230–244, 2021.

[22] A. Ala'M, K. Omar, and M. Al-Zewairi, "Feature selection for intrusion detection system using SVM-based recursive feature elimination and PCA," *Procedia Computer Science*, vol. 170, pp. 403–410, 2020.

[23] R. Doriguzzi-Corin, G. Siracusano, and O. Bonaventure, "A survey on data plane programming with P4: Fundamentals, advances, and applied research," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 38–69, 2021.

[24] X. Li, H. Zhang, and X. Yuan, "An enhanced SVM intrusion detection method based on feature selection and data balancing," *Security and Communication Networks*, vol. 2020, p. Article 8896783, 2020.

[25] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," presented at the 2018 International Conference on Wireless Networks and Mobile Communications (WINCOM), 2018.

[26] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50, 2018. https://doi.org/10.1109/TETCI.2017.2772792

[27] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW) (pp. 29-35). IEEE*, 2018.

[28] S. Sahoo, A. Kumar, and A. Upadhyay, "How do green knowledge management and green technology innovation impact corporate environmental performance? Understanding the role of green knowledge acquisition," *Business Strategy and the Environment*, vol. 32, no. 1, pp. 551-569, 2023.