Network Traffic Classification in SDN Networks Using PCA Integrated Boosting Algorithms

M. Muntazir Khan[1], Muhammad Ishaq[1], Zubair Ahmad Shams[2], Haseeb Ullah Jan[1], M. Ghayoor Jan[1] Hussan Fatima[3]

[1] Institute of Computer Sciences and Information Technology (ICS/IT), The University of Agriculture, Peshawar, Pakistan.

[2] Department of computer software engineering, The university of engineering and technology Mardan

[3]Faculty of Engineering and Computing, National University of Modern Languages Islamabad, Pakistan.

*** Correspondence:** M. Muntazir Khan muntazirkhan131@gmail.com

Citation| Khan. M. M, Ishaq. M, Shams. Z, A, Jan. H. U, Jan. M. G, Fatima. H, "Network Traffic Classification in SDN Networks Using PCA Integrated Boosting Algorithms.", IJIST, Vol. 07 Issue. 02 pp 856-870, May 2025

Received| April 14, 2025 **Revised**| May 14, 2025 **Accepted**| May 16, 2025 **Published**| May 17, 2025.

In recent years, internet traffic has increased as a result of the introduction of new services and apps. As a result, managing network traffic has grown more challenging. To accomplish this, several classification techniques for network traffic were proposed. Several researchers have used the most advanced deep learning and machine learning models for the suggested challenge. The suggested work can also make use of boosting methods. Boosting algorithms take advantage of the decision tree idea. They take little training time, and model training does not require a powerful system. Thus, boosting algorithms like Extreme Gradient Boosting Model (XGBM), Light Gradient Boosting Model (LGBM), Cat Boost, and Ada Boost with the integration of Principle component analysis (PCA) are used in the proposed study to classify network traffic. The results of these models are compared in terms of confusion matrix, accuracy, precision, recall, and F-Measure. The Network traffic android malware dataset, which was utilized in the proposed study, is publicly accessible online on Kaggle.com. For simulation, Python and its libraries such as sci-kit-learn, tensor flow, keras, and matplotlib are utilized. Following the simulation, the results showed that the XGBM had 90.41% accuracy, 96.39% precision, 89.72% recall, and 92.91% f-measures, while the LGBM had 89.02% accuracy, 90.04% precision, 89.8% recall, and 89.83% f-measures. 86.87% accuracy, 83.97% recall, 89.43% precision, and 86.61% f-measure were attained with Cat Boost. Following that, ada boost obtained 83.07% accuracy, 80% recall rate, 85.25 precision, and 82.58% f-measures. After the integration of the proposed boosting algorithms with PCA, we achieved a very significant enhancement in results. After the integration, it has been achieved that the accuracy rate of XGBoost has improved to 95.56%, while the recall rate is 94.39%, precision is 96.72% and the F-Measure rate has improved to 93.91%. Similarly, the performance of the light Gbm model is also improved with the integration of PCA. It achieved an accuracy rate of 93.41%, precision of 93.72%, recall of 92.39%, and f-measures of 92.91%. Following this, the performance of PCA integrated cat boost could also be seen as improved, as it achieved an accuracy rate of 94.41%, precision rate of 93.72%, recall of 92.39%, and F-measures of 93.91%. Similarly, the performance of a boost has also gained improvement by achieving an accuracy rate of 94.56%, precision rate of 94.72%, recall of 93.39%, and F-measure score of 93.91%. After all the simulations and performance evaluations, it has been achieved that the integration of PCA with the boosting algorithm is a simple trick to improve the performance of boosting algorithms. As here the performance of each model is improved to approximately 10%.

Keywords: Network, Classification, SDN, XGBM, LGBM, PCA, confusion matrix

Introduction:

Data traffic is growing exponentially in a society that is becoming more and more digital, as seen by the quick spread of smart gadgets and new technology. Consequently, network infrastructures are growing increasingly complicated and varied to effectively manage multiple devices and distribute traffic [1]. A typical production network application runs protocols and consists of several devices. The diversity of network infrastructure presents significant challenges in efficient management, resource optimization, and overall organization [2]. To address these challenges, incorporating intelligence into networks has been proposed as a solution [3]. Implementing a knowledge plane (KP) that integrates machine learning (ML) alongside deep learning (DL) approaches is one strategy that was proposed by [4]. Similarly, network programmability enabled by software-defined networking (SDN) is one of the latest developments in networking [5]. To ensure efficient network administration, Internet Service Providers (ISPs) and network managers must adapt to these changes by implementing the appropriate tools and methodologies.

SDN is a new and dynamic architectural technique that divides the data plane from the control plane. The dynamic nature of contemporary applications makes this technology especially well-suited. SDN's centralized controller offers a thorough network perspective, allowing the control switches, configuration, and monitoring to regulate traffic flows [6]. The SDN controller facilitates the integration of ML models by gathering data about network flows and providing real-time programmability. This integration enhances the controller's capabilities, allowing it to perform tasks such as resource management, data analysis, traffic classification, security enforcement, and overall network optimization [7]. Although the SDN controller enables efficient flow control, its lack of direct control over end users limits its ability to achieve distributed end-to-end QoS management and mitigate security threats. However, the integration of ML techniques into SDN has garnered significant attention due to its potential to offer data-driven solutions to traditional network challenges.

Classifying network traffic into distinct classes is a difficult task that serves several purposes, such as network administration, service assessment, and network monitoring. It is also crucial for establishing Quality of Service (QoS), enforcing access restrictions, and addressing other network security considerations while enabling efficient resource allocation. One of the commonly used traffic classification techniques is the port-based approach [8]. Statistical method [9]as well as Deep Packet Inspection (DPI) [10][11]. However, with most applications now utilizing dynamic ports and encrypted network traffic, these traditional methods have become less effective. Therefore, a more adaptive and intelligent traffic categorization method must be developed to align with the evolving network environment.

Network traffic analysis and sorting now require artificial intelligence (AI) [12]. This approach collects information and uses machine learning algorithms to categorize and recognize different kinds of network traffic. By analyzing contextual data associated with network packets, traffic can be categorized into distinct groups, including encrypted traffic. This approach enables more accurate classification by considering factors such as packet size, length, frequency, and other behavioral patterns [13][14]. Consequently, machine learning and SDN algorithms offer an accurate and effective categorization approach [15]. The integration of SDN and Deep Learning (DL) techniques has recently enhanced traffic flow categorization by application type, providing effective and scalable alternatives to traditional methods [16][17]. By allocating priorities according to quality of service (QoS) & network security needs, this approach improves user experience and maximizes network performance.

Despite extensive research on traffic categorization using DL in SDN systems, several challenges still need to be addressed. These include fine-tuning hyperparameter settings, improving classification techniques, and categorizing new traffic types. Machine learning is a useful method that provides several algorithms that have shown exceptional efficacy in traffic categorization jobs.

Thus, the purpose of this study is to use the integration of PCA with a boosting algorithm to create a traffic classification model with a specific focus on categorizing traffic flows according to network assaults and application types. The following succinctly describes the primary contributions of this work:

- To develop boosting algorithms with the integration of PCA for network traffic classification.
- To ensure high efficiency of PCA integration as compared to basic boosting algorithms.

- To compare the performance of integration techniques with available boosting algorithms using matrices like accuracy, precision, recall, and F-Measures.

The structure of the paper is as follows: Related work is covered in Section 2, the suggested approach is presented in Section 3, the findings and discussion are shown in Section 4, and the conclusions and directions for further research are explained in Section 5.

Previous Work:

Researchers have been exploring statistical techniques for DL traffic categorization lately. These techniques utilize payload-independent variables including flow duration, packet length, and inter-arrival time. This section reviews and discusses previous research on traffic categorization in SDN using DL approaches. Various research studies have been conducted for network traffic classification. [18] Presented SDN-HGW, a method that improves distributed smart house administration networks. By extending control to the access network and supporting the core network controller, the method enables more efficient end-to-end network administration. This structure plays a crucial role in smart home networks. It uses network data traffic categorization to make dispersed applications visible. Three deep learning algorithms Multilayer Perceptron (MLP), Stacked Autoencoder (SAE), and Convolutional Neural Networks (CNNs) are used. An open dataset containing over 200,000 encrypted data samples from 15 different applications is used to train the classifiers. The findings of the study show how these classifiers may be used in an SDN smart home setting, allowing for distributed application awareness. According to their research, Zhang et al. [19] presented a unique categorization method. For application categorization, this method uses a hybrid deep neural network. Notably, it eliminates the need for manual feature selection by automatically extracting flow characteristics using the Stacked Autoencoder (SAE) method. Additionally, a centralized SDN controller leverages its high computational power to efficiently collect and manage vast amounts of network traffic. According to their study's findings, the accuracy of this innovative classification technique is higher than that of the conventional Support Vector Machine (SVM)-based strategy.

The authors in [20], Proposed a strategy for traffic categorization in SDN using deep learning (DL) models. They preprocessed network traffic flows to generate a dataset and developed two DL classifiers: a CNN-single-layer LSTM combination and a Multi-layer Long Short-Term Memory (ML-LSTM) model. A fitting process was used to modify the model's hyperparameters. Performance investigation demonstrated the superiority of the ML-LSTM model in network packet categorization, as evidenced by the F1 score. A new application-aware traffic categorization model was presented by Chang et al. [21] for both live and offline use in an SDN testbed. The three DL algorithms used by this model MPL, CNN, and SAE are incorporated into the SDN controller. The results indicated that the model achieved a remarkable accuracy of 87.00% during online testing and exceeded 93.00% accuracy during offline training.

[22] Developed an engineered system for traffic management in SDN. Their approach is based on a classifier that was constructed with 1D-CNN and deep neural network (DNN) techniques. The dataset's unequal class distribution is addressed with the Synthetic Minority Over-Sampling Technique (SMOTE). This approach enhances service quality by assigning different priority queues based on the classification of traffic flows from various applications. According to their research, traffic data acquired during 5 and 10 s timeouts yields greater accuracy when handled by DNN and 1D CNN algorithms. Chiu et al. [23] presented the Convolutional Autoencoder Packet Classifier (CAPC), a system that uses DL to quickly classify incoming packets both in fine-grained and coarse-grained ways, differentiating between specific applications and more general categories, respectively. With the use of an autoencoder and a 1D convolutional neural network, CAPC is a packet-based deep learning model that can efficiently handle encrypted traffic, dynamic ports, and even cluster-related apps. A publicly accessible VPN dataset and a privately obtained traffic dataset are used to assess the classifier, demonstrating its remarkable performance. When classifying service traffic categories on a publicly available dataset containing 24 services, the CAPC model achieves an accuracy of over 97.00%. ByteS-GAN, a semi-supervised traffic categorization method in SDN that makes use of Generative Adversarial Networks (GAN), was created by [24]. ByteS-GAN was developed to get high accuracy even while dealing with a large number of unlabeled data points and a small number of labeled examples. To allow fine-grained traffic categorization, the conventional GAN discriminator network's structure and loss function have to be changed. The study's findings demonstrated that ByteS-GAN outperforms supervised techniques like CNN and greatly enhances the classifier's performance.

Wu et al. [25] have introduced a system for classifying encrypted network data using a deep learning algorithm. Three modules make up the framework: the first one preprocesses the network flows, the second one trains the classifier, and the third one tests the CNN model. Their results indicated that this methodology effectively classifies encrypted network data while maintaining minimal resource consumption. A framework was presented by [26] to improve the administration of smart home networks and strengthen the security of the SDN controller. This system is built on DL techniques and is called SDN Home Gate Way for Congestion (SDNHGC). SDNHGC facilitates real-time traffic analysis to optimize network capacity and resource allocation. The classifier within this framework was trained and evaluated using CNN, MLP, and SAE algorithms, leveraging data from 20 applications sourced from an open database. The outcomes show that this method outperforms other current alternatives in terms of accuracy.

Anh et al. [27] introduced an explanatory technique that incorporates a genetic algorithm in an attempt to address the interpretability issue in DL-based traffic categorization. They used the evolutionary algorithm to generate the best feature selection masks and developed a traffic classifier based on the ResNet model. This creative method produced a remarkable accuracy percentage of about 97.24%. Notably, by determining the dominance rate of each characteristic and quantifying its relevance, the scientists shed light on the fundamental mechanics of the classifier and offered insightful information on how it functions for different Internet services. To address the problem of traffic categorization in SDN systems, Jang et al. [28] developed a novel strategy that makes use of a vibrational autoencoder (VAE). Their goal was to guarantee service quality and efficiently classify various Internet service classes. This was accomplished by training the VAE with six statistical features, which allowed latent feature probabilities for flows inside each service class to be extracted. By contrasting its latent feature distribution with the previously learned distributions, query traffic was classified. The trial findings demonstrated that this approach achieved an average accuracy of 89.00%, surpassing traditional statistics-based and machine learning-based methods in effectiveness.

The authors of the literature review propose traffic classification models utilizing deep learning (DL) approaches, which were assessed based on the number of input features, classification accuracy, and the distinct classes identified in the model's output. Despite the models' high accuracy, issues with multiclass imbalance, training time, computational cost, and dataset availability still exist. Additionally, idea drift a problem in network traffic classification occurs as a result of the exponential expansion of applications and the ongoing evolution of network threat fingerprints. Temporal location variations lead to changes in the environment and data distribution.

Identifying every application and network attack is both challenging and impractical. Therefore, the current study proposes developing a machine learning-based PCA-integrated boosting classifier that categorizes applications and the most common network attacks based on specific traffic flow attributes. This idea allows for the classification of various attacks and applications into classes. Assigning QoS rules or limits to network traffic is made easier by this more efficient classification of applications as well as attack traffic flows than by categorizing them separately.

Material and Methods:

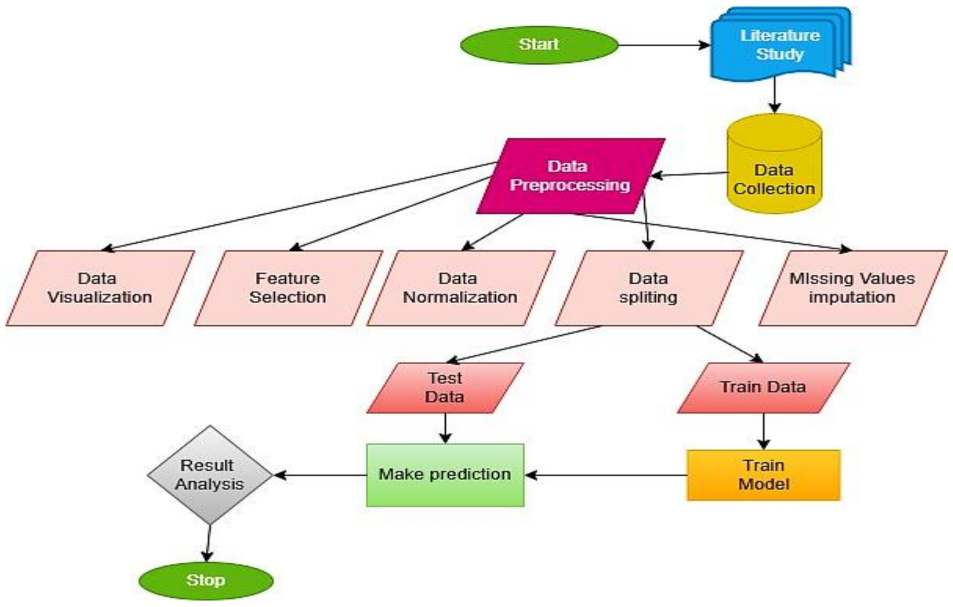
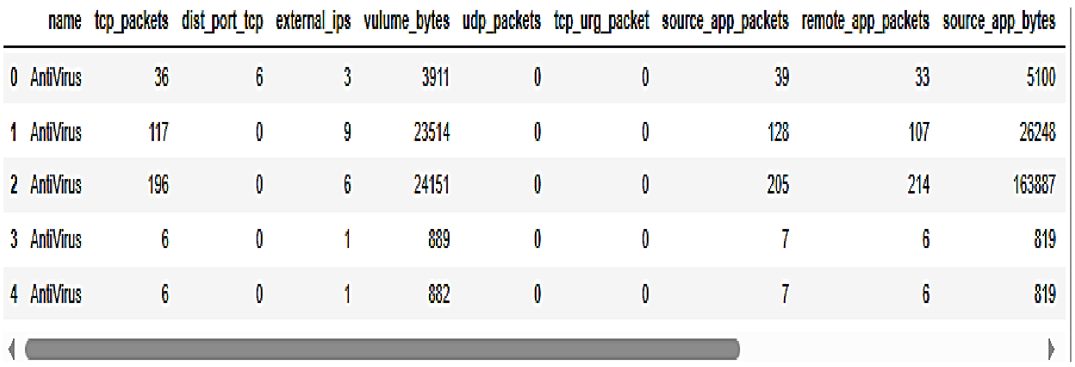


Figure 1. step by step research flow diagram for the proposed study

This section outlines the materials and methods used to conduct the proposed research study. It details the suggested models, data collection, and preprocessing steps like data exploration, normalization, and correlation is discussed. Moreover, boosting algorithms like XGBoost, light GBM, ada boost and cat boost along with principal component analysis (PCA), are thoroughly explained in this section. Further performance evaluation matrices used in the proposed study are discussed as results evaluation in this section. Figure 1 displays the step-by-step research flow diagram of this study.

Data Collection:



	name	tcp_packets	dist_port_tcp	external_ips	vulume_bytes	udp_packets	tcp_urg_packet	source_app_packets	remote_app_packets	source_app_bytes
0	AntiVirus	36	6	3	3911	0	0	39	33	5100
1	AntiVirus	117	0	9	23514	0	0	128	107	26248
2	AntiVirus	196	0	6	24151	0	0	205	214	163887
3	AntiVirus	6	0	1	889	0	0	7	6	819
4	AntiVirus	6	0	1	882	0	0	7	6	819

Figure 2. Dataset head

Proper imputation of missing values in categorical datasets also require smart algorithms[30].The dataset used in the proposed study was obtained from the publicly available repository Kaggle.com. The dataset is named as "Network traffic android malware", and is located on "https://www.kaggle.com/datasets/xwolf12/network-traffic-android-malware". In this research, we preprocessed the files to get network characteristics. This dataset relies on another dataset (DroidCollector), which provides entire network traffic as pcap files. Figure 2 displays the first five records of the proposed dataset.

Table 1. dataset description

	Name	Tcp Packets	Dist Port tcp	External ips	Volume bytes	Udp packets	Tcp urg packate	Source app packets
0	Antivirus	36	6	3	3911	0	0	39
1	Antivirus	117	0	9	23514	0	0	126
2	Antivirus	196	0	6	24151	0	0	205
3	Antivirus	6	0	1	889	0	0	7
4	Antivirus	6	0	1	882	0	0	7

Data Exploration:

The graphical display of information and data that makes patterns, trends, and relationships simple to understand is known as data visualization. Using plots, graphs, and charts including bar charts, scatter plots, histograms, and heat maps, it turns unstructured data into insightful information. By simplifying complicated datasets and supporting anomaly identification, feature significance analysis, and machine learning model assessment, effective visualization improves decision-making. Since inadequate representation can cause misunderstandings, selecting the appropriate visualization method is essential. Data visualization, using programs like Matplotlib, Sea Born, as well as Tableau, is essential to scientific research, corporate intelligence, and analytics. The heat map visualization of the suggested dataset is presented in Figure 3 below.

The suggested dataset's data visualization is shown in Figure 3. Most histograms show data points clustered near zero, with a few extreme values creating long tails, indicating highly skewed distributions. This suggests the presence of outliers or a significant imbalance in feature values. Extreme levels for features like volume bytes, source_app_bytes, as well as remote_app_bytes show significant fluctuations in network traffic. Furthermore, features like tcp_urg_packet and udp_packets appear to have relatively low values, suggesting they are infrequent in the dataset. To improve data distribution and model performance, methods such as log transformation, normalization, or outlier treatment (clipping) are used.

Handling Correlation:

Correlation measures the relationship between two variables by showing how changes in one correspond to changes in the other. It ranges from -1 to 1, where 0 indicates no correlation, -1 represents a perfect negative correlation (one increases as the other decreases), and 1 signifies a perfect positive correlation (both increase together). While weak correlations indicate low reliance, high correlations between features in machine learning can cause redundancy and multicollinearity, which can impact model performance.

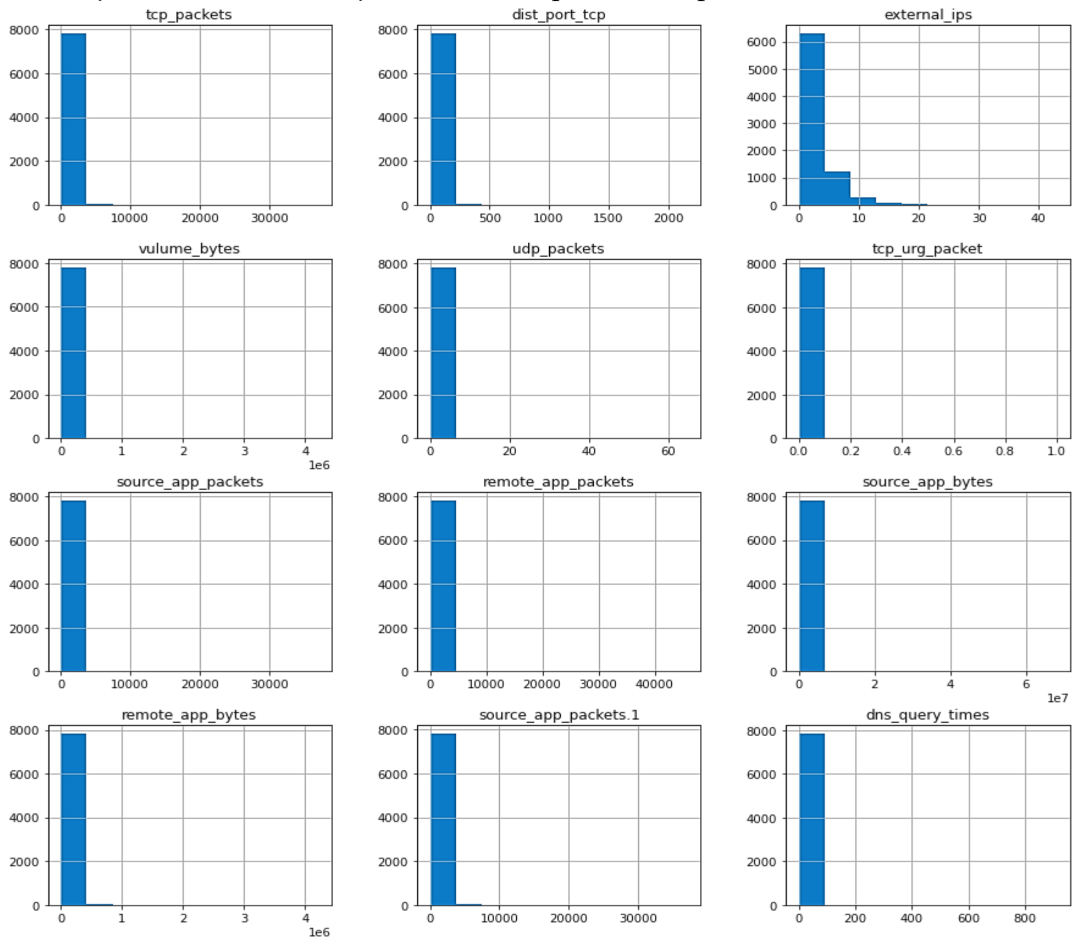


Figure 3. Graphical visualization of the dataset

To increase model efficiency and interpretability, correlations must be found and managed using methods like feature selection, PCA, or Variance Inflation Factor (VIF). Figure 4 below shows the suggested dataset's correlation heat map.

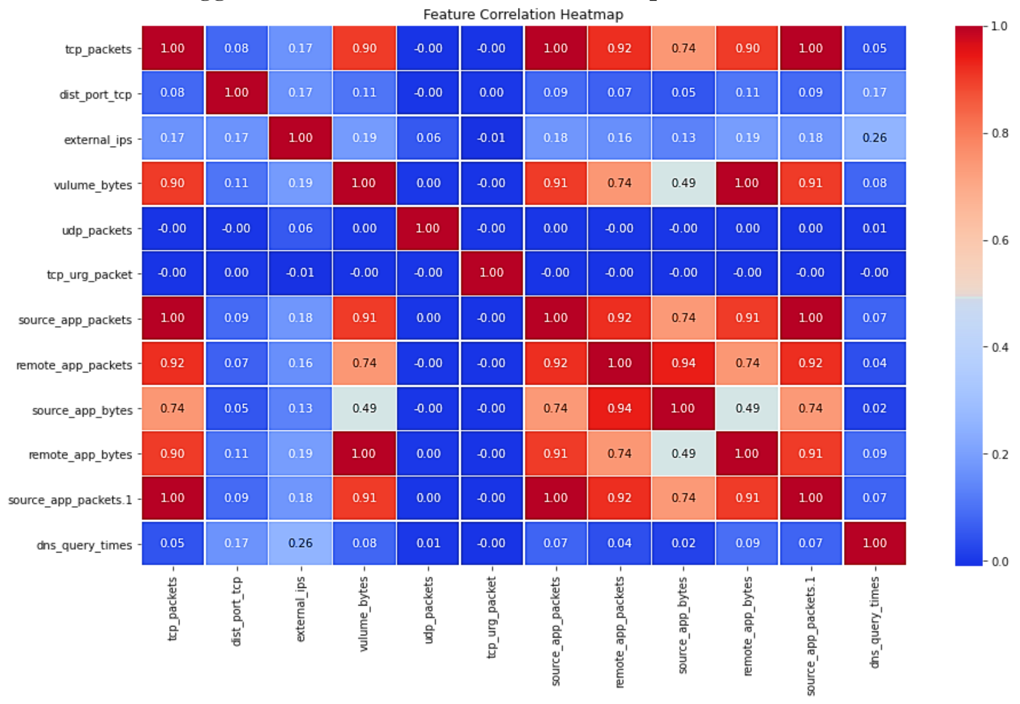


Figure 4. Correlation matrix of the proposed dataset

This heat map highlights several feature correlation-related problems that may affect the effectiveness of machine learning models. Highly correlated features, such as tcp_packets, source_app_packets, and remote_app_packets, contribute to redundancy and multicollinearity, as their correlations approach high values. To address this, Principal Component Analysis (PCA) is applied to remove redundant features while preserving essential information. Furthermore, features with weak correlations, such as udp_packets and tcp_urg_packet, show little to no association with other variables, suggesting they may have minimal impact on prediction performance. To improve model efficiency, these features are removed. The existence of characteristics with significant correlation (such as source_app_bytes & remote_app_bytes with volume_bytes) is another problem since it might result in some redundancy. Lasso regression is a more effective method for managing these relationships. Resolving these problems will increase interpretability, avoid overfitting, and boost model efficiency.

Normalization:

To enhance the performance of machine learning models, normalization is a data preparation technique that involves scaling numerical values into a predefined range, often between 0 and 1 or -1 and 1. Avoiding the dominance of variables with bigger magnitudes guarantees that features contribute evenly to the learning process. Z-score normalization, which standardizes data according to mean and standard deviation, and Min-Max scaling, which modifies data within a predetermined range, are examples of common normalization approaches [29]. The proposed study employs Z-score normalization to scale the dataset, ensuring values are standardized while maintaining a mean of 0 and a standard deviation of 1.

Gradient Boosting Models:

An ensemble approach for forward learning, Gradient Boosting Models (for Regression and Classification) builds regression trees sequentially on all of the dataset's properties in a fully distributed manner, with each tree being generated in parallel. The idea behind GBM is that good forecasting results can be obtained with ever-finer approximations. One popular forward machine learning ensemble technique is the GBM (Regression & Classification) approach. The idea of GBM is to build atop weak trees one after the other, with each tree improving and learning from the one before it. GBM builds regression trees sequentially, with each tree being built in parallel, on each of the variables in the dataset. In decision trees, gradient boosting turns weak classifiers into powerful learners. The existing trees are unaffected when new trees are added to boost. When a new tree is added to the Gradient Boosting Machine (GBM), the original dataset remains unchanged, but the tree is trained on a modified version to improve predictions. Training a decision tree and giving each observation equal weight is the first step of the Gradient Boosting Algorithm (GBM). The weights are modified by either increasing the weights of unclassified data or reducing the weights of easily classifiable data after the initial tree assessment. The second tree is created by repeating the process using weighted data as a reference, improving the forecast of the first tree. The gradient boosting algorithms are classified into 4 models used in the proposed study as described below.

Extreme Gradient Boosting Machine (XGBM):

A Gradient Boosting Machine (GBM) combines predictions from many decision trees to provide final predictions. Remember that every weak learner in a gradient-boosting machine is a decision tree. Additionally, every new tree takes into account the shortcomings or errors of the previous trees. Consequently, the flaws of the previous decision trees are used to build the next one. The trees in a gradient boosting machine algorithm are made in this sequence. Extreme Gradient Boosting, or XGBoost, is a well-known boosting technique. In actuality, XGBoost is an altered form of the GBM algorithm! XGBoost and GBM function in the same way. Trees are generated sequentially in XGBoost, with each tree trying to correct the errors of the trees that came before it.

Extreme Gradient Boosting (XGBoost) is a sophisticated gradient-boosting algorithm implementation that prioritizes performance, scalability, and efficiency. By training decision trees one after the other, each of which fixes the mistakes of the one before it, it creates a powerful predictive model. XGBoost employs a weighted quantile sketch approach to efficiently handle sparse data with regularization techniques (L1 and L2) to avoid overfitting. It is also much quicker than conventional boosting techniques since it permits parallelization. To provide an optimized final model, the model uses gradient descent to minimize a loss function and adds new trees to eliminate residual errors. Equation 1 below represents the Extreme gradient boosting algorithm.

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where $l(yi, \hat{y}i)$ Is the loss function that calculates how much the real and projected values diverge, and $\Omega(f_k)$ is a regularization term that penalizes model complexity to avoid overfitting the model is optimized using second-order Taylor expansion, which allows for faster convergence and more accurate updates, each new tree added to the model indicates the residuals of the prior trees, as well as the final prediction, is obtained by adding the outputs of all the trees [30].

Light Gradient boosting Machine (LGBM):

The Light GBM boosting method is gaining popularity due to its efficiency and speed, making it well-suited for handling large datasets. However, it performs poorly when dealing with small datasets. Unlike traditional methods that grow trees level by level, Light GBM grows tree leaf by leaf, splitting only the leaf node with the highest delta loss after the initial split.

An effective and high-performance gradient-boosting solution made to manage big datasets with little memory use is called a Light Gradient Boosting Machine (Light GBM). Light GBM employs a leaf-wise growth approach, which means it extends the leaf node with the biggest loss reduction instead of expanding all leaves at the same level, in contrast to conventional gradient-boosting techniques that grow trees level-wise. This method preserves the computing economy while producing deeper trees and increased accuracy. Furthermore, Light GBM uses histogram-based learning, which reduces complexity and memory utilization by classifying continuous feature values into discrete bins, accelerating training.

Like other gradient boosting models, Light GBM's optimization goal is to minimize a loss function while adding a regularization term:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where $l(yi, \hat{y}i)$ symbolizes the loss function that measures the discrepancy between real and expected values, and $\Omega(f_k)$ is a regularization term that regulates the complexity of the model. Gradient-based one-side sampling (GOSS), which gives preference to instances with bigger gradients to concentrate on informative samples, & Exclusive Feature Bundling (EFB), which lowers dimensionality by merging mutually exclusive features, are two other ways that Light GBM improves efficiency. Because of these improvements, Light GBM is among the most scalable and quick gradient-boosting algorithms on the market [30].

Adaptive boosting (Ada Boost):

Several poor classifiers are combined in the ensemble learning technique known as "Adaptive Boosting" (Ada Boost) to produce a powerful prediction model. It works most well with decision trees; as poor learners, it frequently uses shallow trees or stumps. Ada Boost initially assigns equal weights to all data points and iteratively adjusts these weights based on the errors of previous models. Misclassified samples receive higher weights in each iteration, guiding the next weak learner to correct those errors. This process continues until a specified number of iterations is reached or the model attains a satisfactory level of accuracy. Ada Boost operates by training weak classifiers one after the other, highlighting the mistakes of the prior classifier. Based on accuracy, the algorithm gives each weak classifier a weight α_m, which is determined as follows:

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - e_m}{e_m} \right)$$

Where e_m is the weak classifier's weighted error rate. Occurrences that are incorrectly classified have higher weights, whereas occurrences that are correctly classified have lower weights. This modification guarantees that later classifiers focus more on challenging instances, thereby enhancing the model's overall performance. The weighted outputs of each weak classifier are combined to provide the final prediction.

The benefit of Ada Boost is that it lowers bias and variance, which makes it resistant to overfitting, particularly when used with basic models. Outliers and noisy data, however, might have an excessive impact on the model since they are given greater weights. Despite this, Ada Boost is still a potent technique for classification problems and is frequently applied in text recognition and face detection applications. It is a well-liked option for boosting-based ensemble learning because of its ease of use and efficiency [30].

Cat Boost:

As the name implies, Cat Boost is a boosting technique that can handle categorical factors in data. When input comprises strings or categories, several machine learning methods cannot function. Therefore, converting categorical variables to numerical values is a crucial preprocessing step. Cat Boost is capable of handling categorical variables in data independently. These variables are transformed into ones using various statistics on feature sets.

A gradient boosting method called Cat Boost (Categorical Boosting) was created specially to deal with categorical data well. Cat Boost natively processes categorical features utilizing an ordered boosting approach as well as target-based encoding, in contrast to typical boosting approaches that need converting categorical variables into numerical representations (e.g., one-hot encoding). This increases model accuracy and helps avoid overfitting. Furthermore, Cat Boost utilizes symmetric decision trees, which enhance generalization and accelerate training by maintaining a consistent split structure across all trees. Due to its exceptional capability in handling categorical data, it is widely used in applications such as fraud detection, recommendation systems, and ranking problems.

Like other gradient boosting techniques, Cat Boost's optimization function involves regularization and loss function minimization:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where $l(yi, \hat{y}i)$ symbolizes the loss function that measures the discrepancy between real and expected values, and $\Omega(f_k)$ is a regularization word used to manage the complexity of the model. The use of ordered boosting, a fundamental component of Cat Boost, ensures that every data point is learned without "leakage" from subsequent observations, hence removing prediction bias. Because of this, Cat Boost works especially well with sequential data and high-cardinality categorical features. Furthermore, it performs better than other boosting models in terms of accuracy and speed due to its effective implementation [30].

Principal Component Analysis (PCA):

A popular dimensionality reduction method in data analysis and machine learning, principal component analysis (PCA) converts high-dimensional data into a lower-dimensional space while maintaining the most relevant variance. It is especially helpful when there is duplication in datasets due to strong feature correlations. To determine the directions (principal components) the fact that capture the most variation, PCA calculates the eigenvalues and eigenvectors of the dataset's covariance matrix. The original data is then projected onto a lower-dimensional space using the top-k main components, which successfully reduces complexity while preserving important information. This method is frequently used to reduce the curse of dimensionality, increase computing efficiency, and improve classification as well as clustering performance in domains including network traffic analysis, bioinformatics, and image processing.

The PCA working process consists of several sequential steps. First, the mean of each feature in the dataset is subtracted to ensure the data is centered at the origin. The covariance matrix is then calculated to determine how various characteristics relate to one another. The major components are then represented by the eigenvalues and accompanying eigenvectors obtained from the eigenvalues of this covariance matrix. The eigenvalues of the main components are used to rank them; the directions of maximum variance are indicated by the highest eigenvalues. To decrease the feature space, the dataset is finally processed by projecting it onto the chosen main components. PCA is a useful preprocessing step for a variety of machine learning models, such as classification, clustering, & anomaly detection, since it decreases dimensionality by removing less important components while simultaneously mitigating noise and redundancy [31].

Results Evaluation:

The final stage of our proposed methodology involves evaluating machine learning models, which is crucial for determining their effectiveness and overall performance. Metrics reveal details regarding each model's accuracy and performance [44]. In this step, test data that is, data not utilized during the training phase is used to assess the performance of the four models. To evaluate their performance and predictability, a variety of measures were used. Various assessment metrics are available based on the approach used. To offer a comprehensive view of each model's performance and data fit, the classification metrics used in this study include Accuracy (2), Precision (3), Recall (4), and F1-Score (5).

- **Confusion Matrix:** A confusion matrix is a performance measurement tool used in classification tasks that summarizes the prediction outcomes by displaying the amount of true positive (TP), true negative (TN), false positive (FP), as well as false negative (FN) occurrences. By offering a thorough analysis of classifier accuracy and showing the locations of misclassifications, it aids in determining the advantages and disadvantages of the model. This matrix makes it possible to thoroughly analyze the performance of the classification model, which is required to compute important evaluation metrics like accuracy, precision, recall, & F1-score [32].

- **Accuracy:** Accuracy is a performance statistic used in classification that quantifies the proportion of right predictions made by the model out of all predictions. It provides a simple measure of overall model accuracy and is computed by dividing the sum of true positive as well as true negative cases by the total number of occurrences. While accuracy is beneficial, it may be deceptive for imbalanced datasets as additional metrics, such as recall and precision, may be necessary to correctly assess the model's performance [32].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

- **Precision:** Precision is a classification performance parameter that quantifies the proportion of genuine positive predictions among all occurrences that the model predicts as positive. It illustrates how the model can detect positive instances while reducing false positives, which is crucial in situations when false positives might be harmful. Precision is crucial for determining the F1 score, particularly with unbalanced datasets, and is frequently paired with recall to offer a fair assessment of classifier performance [31].

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

- **Recall:** A classification metric called recall, sometimes referred to as sensitivity, shows the percentage of true positive events that the model correctly detected out of all true positive occurrences. It demonstrates the model's ability to find every pertinent instance in a dataset, which is essential when it becomes expensive to overlook excellent instances. Recall is a crucial component in determining the F1 score and is frequently assessed accurately to determine a model's overall performance [29].

$$Recall = \frac{TP}{TP+FN} (4)$$

- **F-Measures:** Recall and accuracy are combined into a single performance statistic called the F1 score by taking the harmonic mean of the two metrics. Because it balances false negatives and erroneous positives, it offers a more detailed picture of classifier performance than accuracy alone. This makes it especially helpful when dealing with imbalanced datasets. The F1 score provides a strong assessment tool by focusing on both memory and accuracy, particularly in situations where striking a balance between detecting positives and lowering false alarms is crucial [31].

$$F1 - Score = \frac{2*recall*precision}{Recall+Precision} \quad (5)$$

Result and Discussion:

The outcomes of using the proposed algorithms (discussed in methodology) as well as its integration with PCA, are shown and explained in this section. Before applying the algorithms, it is crucial to determine the length of the input sequence. The proposed research study utilized a publicly available dataset called Network Traffic Android Malware. The dataset was split into a 30:70 ratio for testing and training, respectively, using the K-fold cross-validation method with 10 folds. Boosting algorithms like XG Boost, light boost, ada boost, and cat boot along with PCA, are utilized for training.

Preliminaries:

The experiments were conducted on a system with an Intel Core i5 processor (2.0 GHz) and 8 GB of RAM, running Windows 10 as the operating system. The Keras Python module was used to test and train the model on all datasets. To investigate the performance of machine learning models, the algorithms Boosting algorithms like XG Boost, light boost, ada boost, and cat boot integrated with PCA, are contrasted concerning accuracy, recall, precision, and f-measure.

Results and Discussion:

In this study, four models underwent several tests, which were conducted using a variety of metrics, including f-measure, accuracy, recall, and precision. A list of models used in the simulation is shown below: 1. XG Boost, 2. Light GBM, 3. Ada Boost and 4. Cat Boost. 5. PCA-based XG Boost, 6. PCA-based Light GBM, 7. PCA-based Ada Boost and 8. PCA-based Cat Boost

Performance Evaluation of Boosting Algorithms:

The performance evaluation of various booting algorithms in terms of accuracy is shown in Figure 5 below. According to Figure 5, XGBoost outperformed all other boosting algorithms, achieving an accuracy of 90.42%. Following it, the light GBM achieved 89.02% of accuracy. Further cat boost achieved an accuracy rate of 86.87% while the accuracy rate of ada boost is too low as it achieved an accuracy rate of 83.07%. Consequently, in comparison to boosting algorithms for SDN traffic classification, XGBoost shows better performance as compared to others. Following the integration with PCA, XG Boost's accuracy rate was increased to 95.56%, while the accuracy of light GBM was increased to 93.42%. Furthermore, the accuracy of Cat Boost shows improvement, reaching 94.42%. Following that, the accuracy rate of the ada boost is increased to 94.56%. Figure 5 displays the accuracy plot of various models before and after the integration of PCA.

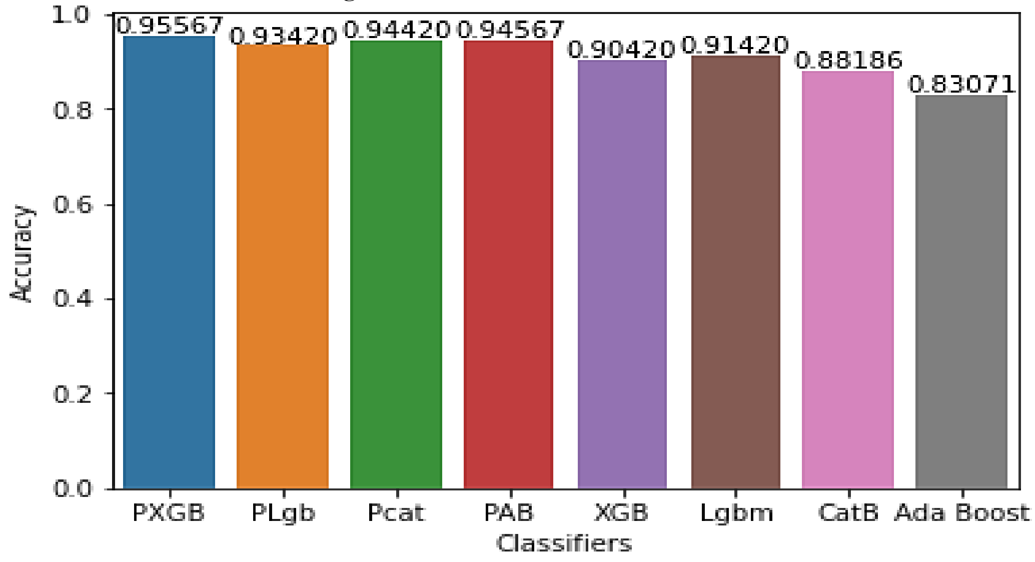
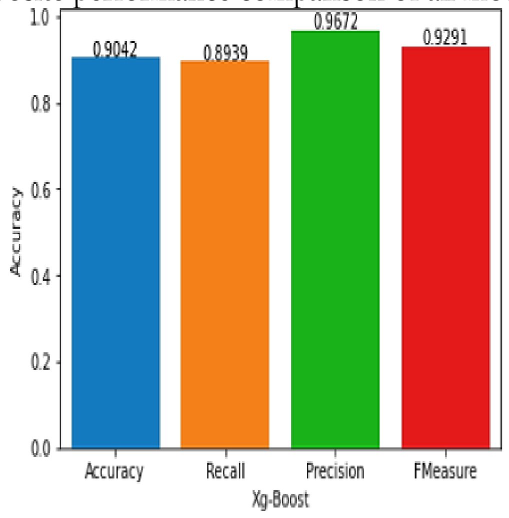
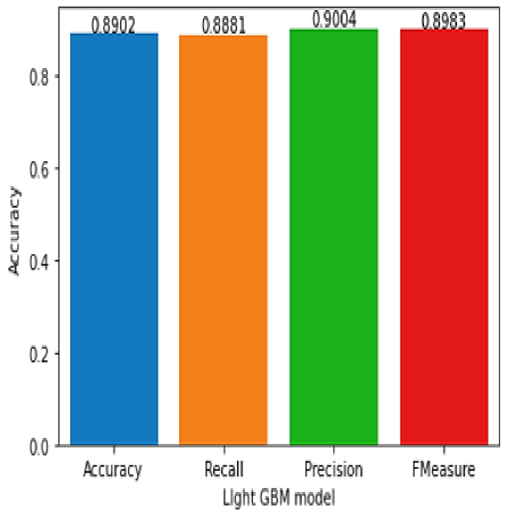


Figure 5. Accuracy report of various models

Figure 5 above displays the accuracy rate achieved by different boosting algorithms. It can be adopted from the above figure 8 that the XG boost outperforms all other boosting algorithms in terms of accuracy achieved. The complete performance evaluation of various boosting algorithms in terms of accuracy, precision, recall, and F-Measure has been displayed in Figure 7 and Table 2 below. According to Figure 7 and Table 2, it can be seen that the XGBM had 90.41% accuracy, 96.39% precision, 89.72% recall, and 92.91% f-measures, while the LGBM had 89.02% accuracy, 90.04% precision, 89.8% recall, and 89.83% f-measures. Nearly 86.87% accuracy, 83.97% recall, 89.43% precision, and 86.61% f-measure were attained with Cat Boost. Moreover, ada boost obtained 83.07% accuracy, 80% recall rate, 85.25 precision, and 82.58% f-measures. Following integration, XG Boost's accuracy rate increased to 95.56%, while its recall, precision, and F-Measure rates improved to 94.39%, 96.72%, and 93.91%, respectively. Similarly, the application of PCA enhances the performance of the Light GBM model, achieving 93.41% accuracy, 93.72% precision, 92.39% recall, and 92.91% F-measure. An improvement was also observed in the performance of PCA-integrated Cat Boost, achieving 94.41% accuracy, 93.72% precision, 92.39% recall, and 93.91% F-measure. Ada Boost's performance has also improved after its integration with PCA, as seen by its 94.56% accuracy rate, 94.72% precision rate, 93.39% recall, and 93.91% F-measure score. Following several simulations and performance assessments, it has been shown that combining PCA with boosting algorithms is an easy way to increase their effectiveness. In this case, each model's performance was increased by 10%. The performance evaluation of the proposed models is individually visualized in Figure 6, while Figure 7 and Table 2 present a composite performance comparison of all models.





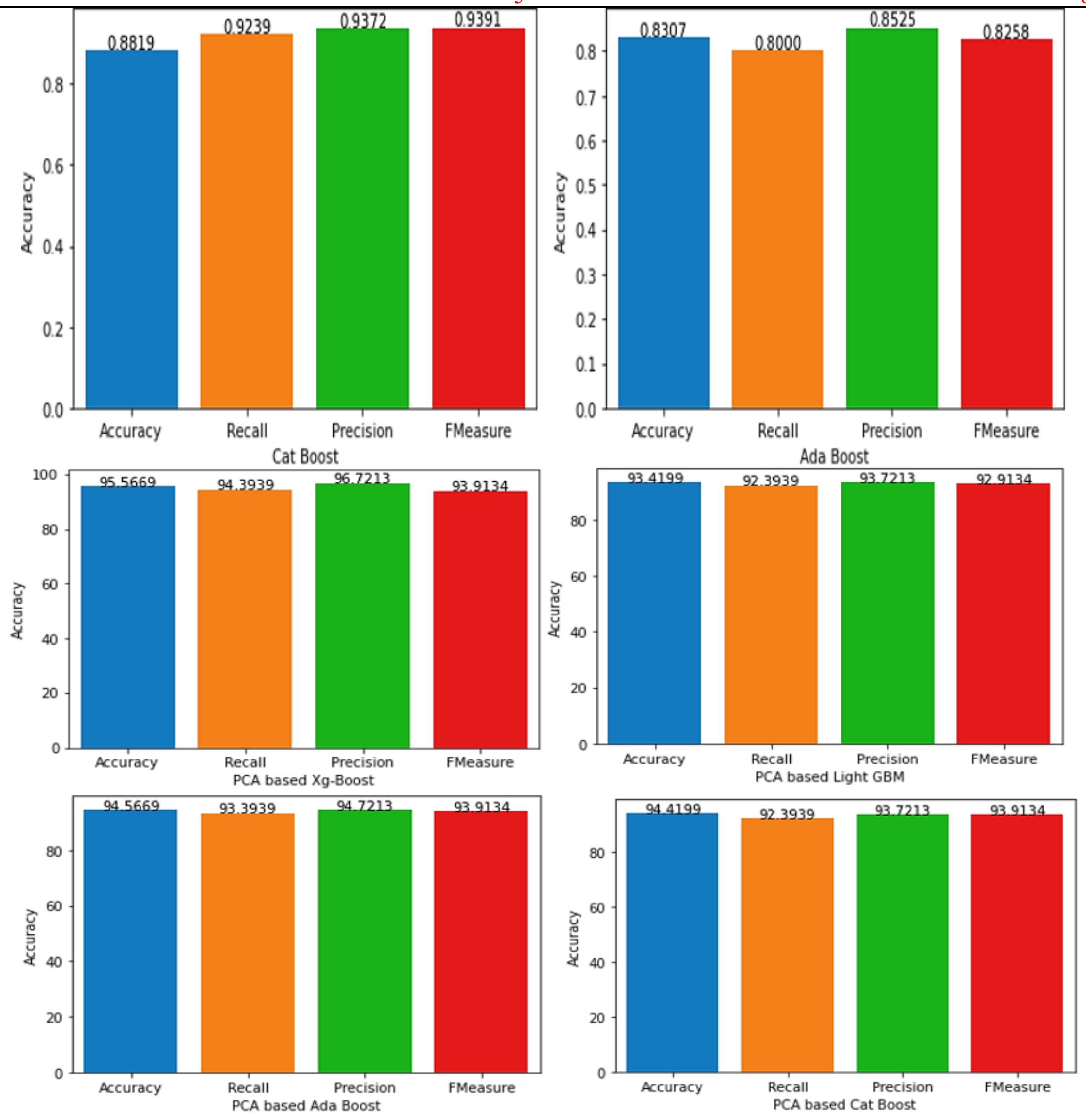


Figure 6. performance evaluation of each model

Figure 6 shows how each model performed in terms of accuracy, precision, recall, and f-measure for SDN traffic classification. Figure 6 is made up of bar graphs that show the performance outcomes of all boosting techniques before and after integration with PCA. Figure 6 depicts the performance of four boosting methods, as well as the results obtained by various boosting techniques after integration with principal component analysis. Figure 7 shows the performance of all proposed boosting techniques in solitary as well as after integration with PCA in a composite bar graph for easier comparison and comprehension.

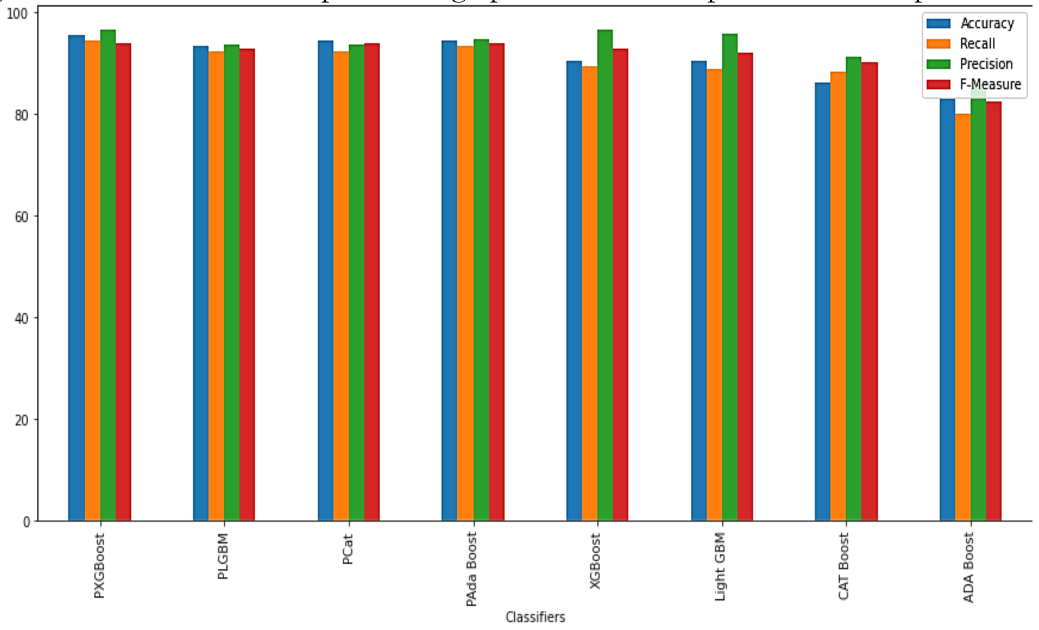


Figure 7. Performance evaluation of all models

Table 2. Tabular representation of performance evaluation of various boosting models

Model	Accuracy %	Precision %	Recall %	F-Measure %
XGBM	90.42	96.72	89.72	92.91

Light GBM	89.02	90.04	88.81	89.83
Cat Boost	86.87	89.43	83.97	86.61
Ada Boost	83.07	85.25	80.00	82.58
PCA based XGBM	95.56	96.72	94.39	93.91
PCA based LGBM	93.41	93.72	92.39	92.91
PCA-based cat Boost	94.41	93.72	92.39	93.91
PCA-based ada Boost	94.56	94.72	93.39	93.91

Discussion:

The performance evaluation of machine learning models, including XGBoost, Light GBM, Ada Boost, and Cat Boost, is presented in Figure 5 and the table above. For the same task, different machine learning models can yield varying performance levels (poor, good, or excellent) due to differences in their design and learning characteristics. Using a network traffic Android malware dataset, the performance evaluation of boosting algorithms for SDN traffic categorization demonstrates the greater effectiveness of XGBoost, It achieved the highest accuracy (90.41%), precision (96.39%), and F1-score (92.91%). Light GBM finished second, with balanced precision-recall values as well as an accuracy of 89.02%, indicating it is a viable replacement. Ada Boost had the lowest accuracy of 83.07%, indicating a weaker capacity to handle complex traffic patterns, whereas Cat Boost performed on average with 86.87% accuracy. These findings suggest that XGBoost has high accuracy and overall classification efficacy. After integrating with PCA, XG Boost's accuracy rate grew to 95.56%, while recall, precision, as well as F-Measure rates, increased to 94.39%, 96.72%, & 93.91%, respectively. Similarly, the integration of PCA improves the performance of the light GBM model. It achieved 93.41% accuracy, 93.72% precision, 92.39% recall, and 92.91% F-measures. Following this, PCA integrated cat boost's performance improved, reaching 94.41% accuracy, 93.72% precision, 92.39% recall, and 93.91% F-Measures. Ada Boost's performance has also increased, as seen by its 94.56% accuracy rate, 94.72% precision rate, 93.39% recall, & 93.91% F-measure score. After multiple simulations and performance assessments, it was discovered that combining PCA with boosting algorithms is a simple method to improve their efficacy. In this scenario, each model's performance has improved by around 10%, making it the best option for SDN traffic classification.

Conclusion:

Using a network traffic Android malware dataset, four boosting algorithms XGBoost, Light GBM, Cat Boost, along Ada Boost were assessed for SDN traffic categorization in this study. The findings show that XGBoost performed better than the other models, obtaining the greatest F1-score (92.91%) and accuracy (90.41%), making it the best option for this classification assignment. Light GBM performed competitively next, while Ada Boost and Cat Boost were less successful, with Ada Boost showing the worst outcomes. According to these results, boosting-based models, in particular, XGBoost offer great accuracy and recall, making them ideal for classifying SDN traffic. After that Following integration, XG Boost's accuracy rate increased to 95.56%, while its recall, precision, and F-Measure rates improved to 94.39%, 96.72%, and 93.91%, respectively. Likewise, the use of PCA enhances the performance of the light gbm model. It was able to get 93.41% accuracy, 93.72% precision, 92.39% recall, and 92.91% f-measures. Following this, it was possible to see an improvement in the performance of PCA integrated cat boost, as it reached 94.41% accuracy, 93.72% precision, 92.39% recall, and 93.91% F-measures. Ada Boost's performance has also improved, as seen by its 94.56% accuracy rate, 94.72% precision rate, 93.39% recall, and 93.91% F-measure score. Following several simulations and performance assessments, it has been shown that combining PCA with boosting algorithms is an easy way to increase their effectiveness. In this case, each model's performance has increased by about 10%. Future studies might investigate deep learning techniques or hybrid models to improve classification performance even further.

Future Research:

In the proposed study, network traffic is classified using boosting methods such as the Extreme Gradient Boosting Model (XGBM), Light Gradient Boosting Model (LGBM), Cat Boost, & Ada Boost with the integration of Principle component analysis (PCA). Accuracy, precision, recall, F-Measure, and confusion matrix are used to compare the outcomes of various models. This study trains each boosting algorithm for SDN traffic classification solitary and also with the integration of PCA. After the simulation, it was achieved that in solitary the XG boost achieved the best results as compared to other boosting algorithms. But still, the accuracy rate needs further improvement. After the integration with PCA, we observed that the performance of each model has been improved by up to 10 %. So, it has been achieved that the integration of PCA with machine learning models can improve overall performance. The research concept should further be extended to some tasks as below.

- To integrate the PCA with other machine learning models like decision tree, random forest, and K-nearest neighbor to achieve best results.
- The concept should be used in various fields rather than network traffic classification.
- The integration of machine learning models with similarity matrixes can also enhance overall performance.

Acknowledgment: Acknowledgments are considered necessary.

Author's Contribution: Muhammad Muntazir Khan (M. Muntazir Khan) conceptualization, implementation, and primary writing. Dr. Muhammad Ishaq's funding sources and final checking. Zubair Ahmad Shams literature preparation. Haseeb Ullah Jan methodology, Muhammad Ghayoor Jan (M. ghayoor Jan) results analysis and review. Hussan Fatima validation. All authors have read and agreed to the published version of the manuscript.

Conflict of Interest: All authors have read and agreed to the published version of the manuscript in IJIST.

References:

[1] Nuñez-Agurto, D., Fuertes, W., Marrone, L., Benavides-Astudillo, E., Coronel-Guerrero, C., & Perez, F. (2024). A novel traffic classification approach by employing deep learning on software-defined networking. *Future Internet*, *16*(5), 153.

[2] Belkadi, O., Vulpe, A., Laaziz, Y., & Halunga, S. (2023). Ml-based traffic classification in an sdn-enabled cloud environment. *Electronics*, *12*(2), 269.

[3] Ayoubi, S., Limam, N., Salahuddin, M. A., Shahriar, N., Boutaba, R., Estrada-Solano, F., & Caicedo, O. M. (2018). Machine learning for cognitive network management. *IEEE Communications Magazine*, *56*(1), 158-165.

[4] Clark, D. D., Partridge, C., Ramming, J. C., & Wroclawski, J. T. (2003, August). A knowledge plane for the internet. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications* (pp. 3-10).

[5] Trois, C., Del Fabro, M. D., de Bona, L. C., & Martinello, M. (2016). A survey on SDN programming languages: Toward a taxonomy. *IEEE Communications Surveys & Tutorials*, *18*(4), 2687-2712.

[6] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14-76.

[7] Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., & Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. *IEEE Communications Surveys & Tutorials*, *21*(1), 393-430.

[8] Moore, A. W., & Papagiannaki, K. (2005, March). Toward the accurate identification of network applications. In *International workshop on passive and active network measurement* (pp. 41-54). Berlin, Heidelberg: Springer Berlin Heidelberg.

[9] Nguyen, T. T., & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, *10*(4), 56-76.

[10] Finsterbusch, M., Richter, C., Rocha, E., Muller, J. A., & Hanssgen, K. (2013). A survey of payload-based traffic classification approaches. *IEEE Communications Surveys & Tutorials*, *16*(2), 1135-1156.

[11] Li, G., Dong, M., Ota, K., Wu, J., Li, J., & Ye, T. (2016, December). Deep packet inspection based application-aware traffic control for software defined networks. In *2016 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.

[12] Nunez-Agurto, D., Fuertes, W., Marrone, L., & Macas, M. (2022). Machine Learning-Based Traffic Classification in Software-Defined Networking: A Systematic Literature Review, Challenges, and Future Research Directions. *IAENG International Journal of Computer Science*, *49*(4).

[13] Fan, Z., & Liu, R. (2017, August). Investigation of machine learning based network traffic classification. In *2017 International Symposium on Wireless Communication Systems (ISWCS)* (pp. 1-6). IEEE.

[14] Tahaei, H., Afifi, F., Asemi, A., Zaki, F., & Anuar, N. B. (2020). The rise of traffic classification in IoT networks: A survey. *Journal of Network and Computer Applications*, *154*, 102538.

[15] Hayes, M., Ng, B., Pekar, A., & Seah, W. K. (2017). Scalable architecture for SDN traffic classification. *IEEE Systems Journal*, *12*(4), 3203-3214.

[16] Sun, W., Zhang, Y., Li, J., Sun, C., & Zhang, S. (2022). A deep learning-based encrypted VPN traffic classification method using packet block image. *Electronics*, *12*(1), 115.

[17] Zaki, F. A. M., & Chin, T. S. (2019). FWFS: Selecting robust features towards reliable and stable traffic classifier in SDN. *IEEE Access*, *7*, 166011-166020.

[18] Wang, P., Ye, F., Chen, X., & Qian, Y. (2018). Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access*, *6*, 55380-55391.

[19] Zhang, C., Wang, X., Li, F., He, Q., & Huang, M. (2018). Deep learning–based network application classification for SDN. *Transactions on Emerging Telecommunications Technologies*, *29*(5), e3302.

[20] Lim, H. K., Kim, J. B., Kim, K., Hong, Y. G., & Han, Y. H. (2019). Payload-based traffic classification using multi-layer lstm in software defined networks. *Applied Sciences*, *9*(12), 2550.

[21] Lin-Huang, C., Tsung-Han, L., Hung-Chi, C., & Cheng-Wei, S. (2020). Application-based online traffic classification with deep learning models on SDN networks. *Advances in Technology Innovation*, *5*(4), 216.

[22] Al-Fayoumi, M., Al-Fawa'reh, M., & Nashwan, S. (2022). VPN and Non-VPN Network Traffic Classification Using Time-Related Features. *Computers, Materials & Continua*, *72*(2).

[23] Chiu, K. C., Liu, C. C., & Chou, L. D. (2020). CAPC: Packet-based network service classifier with convolutional autoencoder. *Ieee Access*, *8*, 218081-218094.

[24] Wang, P., Wang, Z., Ye, F., & Chen, X. (2021). Bytesgan: A semi-supervised generative adversarial network for encrypted traffic classification of sdn edge gateway in green communication network. *arXiv preprint arXiv:2103.05250*.

[25] Mohamed, S. A. A., & Kurnaz, S. (2023). Classified VPN Network Traffic Flow Using Time Related to Artificial Neural Network.

[26] Setiawan, R., Ganga, R. R., Velayutham, P., Thangavel, K., Sharma, D. K., Rajan, R., ... & Sengan, S. (2022). Encrypted network traffic classification and resource allocation with deep learning in software defined network. *Wireless Personal Communications*, 1-17.

[27] Ahn, S., Kim, J., Park, S. Y., & Cho, S. (2020). Explaining deep learning-based traffic classification using a genetic algorithm. *IEEE Access*, *9*, 4738-4751.

[28] Jang, Y., Kim, N., & Lee, B. D. (2023). Traffic classification using distributions of latent space in software-defined networks: An experimental evaluation. *Engineering Applications of Artificial Intelligence*, *119*, 105736.

[30] Singh, A. (4). boosting algorithms you should know–GBM, XGBoost, LightGBM, & Catboost. *Consulted on*, *13*, 08-20.

[32] Khan, A., Khan, A., Khan, M. M., Farid, K., Alam, M. M., & Su'ud, M. B. M. (2022). Cardiovascular and diabetes diseases classification using ensemble stacking classifiers with SVM as a meta classifier. *Diagnostics*, *12*(11), 2595.

[31] Lashari, S. A., Khan, M. M., Khan, A., Salahuddin, S., & Ata, M. N. (2024). Comparative Evaluation of Machine Learning Models for Mobile Phone Price Prediction: Assessing Accuracy, Robustness, and Generalization Performance. *Journal of Informatics and Web Engineering*, *3*(3), 147-163.

[29] Alhwaiti, Y., Khan, M., Asim, M., Siddiqi, M. H., Ishaq, M., & Alruwaili, M. (2025). Leveraging YOLO deep learning models to enhance plant disease identification. *Scientific Reports*, *15*(1), 7969.

[30] M. Ishaq, S. Zahir, L. Iftikhar, M. F. Bulbul, S. Rho and M. Y. Lee, "Machine Learning Based Missing Data Imputation in Categorical Datasets," in *IEEE Access*, vol. 12, pp. 88332-88344, 2024, doi: 10.1109/ACCESS.2024.3411817.