Network Anomaly Detection Using Various Machine Learning Algorithms - A Comparative Study

Yashashvini Rachamallu

Masters Student
Department of Computer Science
Michigan State University

rachama2@msu.edu

1. Introduction

As the world increasingly pivots towards a digital-first paradigm, network infrastructure has become a fundamental pillar of modern society's functioning. This critical infrastructure serves as the nexus of a vast array of activities including, but not limited to, finance, healthcare, education, and governance. However, the escalation in reliance on network systems has been paralleled by a commensurate rise in the sophistication and frequency of cyber threats. Network anomalies, symptomatic of underlying security breaches, present a persistent challenge that necessitates continuous surveillance and rapid response to maintain the sanctity of data and services. The KDD Cup 1999 Dataset, a seminal benchmark in the cybersecurity domain, provides a comprehensive landscape for developing and validating anomaly detection methodologies. Our research is driven by a multi-pronged objective i.e, to develop, benchmark, and refine machine learning models that are adept at detecting network anomalies. We begin by dissecting the dataset to identify and label the various forms of network anomalies. Subsequent steps involve the deployment of a range of machine learning algorithms, each scrutinized for its detection accuracy and response agility. A comparative analysis framework is employed to assess the relative performance of these models, focusing on their precision and reliability in identifying specific anomaly patterns. The methodology is structured to ensure a rigorous evaluation process. Initial data collection and exploration set the groundwork for understanding the dataset's intricacies. Feature engineering follows, where data is preprocessed and transformed to highlight the most salient features for anomaly detection. The selection of models spans both traditional machine learning algorithms and contemporary deep learning techniques. After training these models, a comprehensive evaluation is conducted, utilizing metrics that provide a holistic view of their performance. Finally, the research synthesizes the insights drawn from the comparative analysis to formulate recommendations. These recommendations are designed to guide practitioners in the selection of the most effective machine learning algorithms for real-time network anomaly detection, tailored to the specificities of the threat environment and the operational context of the network infrastructure.

2. Problem statement

In the increasingly interconnected digital landscape, network security plays a critical role in safeguarding sensitive data and ensuring the uninterrupted operation of organizations. The sophistication and scope of cyberattacks and intrusions are constantly increasing, posing a serious threat to both individuals and companies. Network anomalies, often indicative of cyberattacks, can take various forms, including denial-of-service (DoS) attacks, probes, and unauthorized access attempts. Detecting and mitigating these anomalies in real-time is crucial for maintaining the integrity and availability of network resources. Network Anomaly Detection aims to create a strong and proactive solution to strengthen cybersecurity defenses. The creation, deployment, and comparison of machine learning models for detecting network anomalies are the main objectives of this project. Our goal is to quickly detect malicious activity and unexpected network behavior by utilizing data-driven insights and predictive analytics.

3. Dataset

The KDD Cup 99 dataset represents a significant benchmark in the domain of network intrusion detection, widely recognized for its utility in academic research. The complexity of this dataset stems from its vast size, containing 4,898,431 instances in the training set and 311,029 instances in the test set. A noteworthy aspect of the training data is its high redundancy rate of 78%, resulting in 1,074,992 unique data points. Data was created at MIT's Lincoln Laboratory during a DARPA-sponsored event in 1999, the dataset captures a diverse array of simulated attack scenarios against a military network environment, alongside regular network traffic. This mix of normal and malicious traffic allows for the development of intrusion detection systems that can be trained to recognize various cybersecurity threats. The dataset features a combination of numeric and categorical variables, accounting for different types of network traffic and connection details. Numeric features provide continuous data such as bytes sent and received, while flat features describe symbolic attributes including protocol types and service types. The broad span of features encapsulates the intricate dynamics of network interactions, offering a rich ground for the application of machine learning models to detect anomalies. However, the use of this dataset presents unique challenges. The skewed distribution of the target variable, with a minority of instances representing attacks, can complicate the model training process and necessitates careful consideration to ensure balanced model performance. Additionally, the redundancy within the training data requires diligent preprocessing to eliminate overfitting and to enhance the model's predictive accuracy on unseen data. In leveraging the KDD Cup 99 dataset for this research, the aim is to address these challenges and to harness the detailed feature set provided to develop a nuanced approach to network anomaly detection. The dataset not only serves as a rigorous platform for testing machine learning models but also as a conduit for gaining deeper insights into effective cybersecurity measures.

4. Data Preprocessing and Data Analysis

In the process of preparing our dataset for analysis and machine learning tasks, we initially merged the following three files i.e, kddcup.names, kddcup.data_10_percent, and training_attack_types. The 'kddcup.names' file provided essential information about the column names of our DataFrame.'kddcup.data_10_percent' contained a large volume of network traffic data, approximately 500,000 rows, with 42 columns representing various features. Meanwhile, 'training_attack_types' listed different training attack types and their corresponding labels, which was a key component for our analysis. One challenge we encountered during data preparation was the issue of class imbalance. We observed that the dataset had 23 distinct attack type labels. However, the distribution of these labels was highly skewed, with the majority of them having less than 5% representation in the data. This class imbalance could lead to inaccurate model performance as models tend to favor the majority class. To mitigate this problem, we decided to reduce the number of labels by grouping similar attack types together. We mapped several attack types to a common label, effectively consolidating them. This process was carried out to normalize the data distribution, making it more balanced. As a result, the dataset exhibited a more even representation of attack types, which is crucial for the reliability and performance of machine learning models. To visualize this transformation, we created plots illustrating the data distribution of attack types and labels both before and after the label reduction process. The initial distribution [Fig 1] showed the significant class imbalance, with a handful of labels dominating the dataset. In contrast, the second plot [Fig 2] demonstrated a more balanced distribution, indicating that the label reduction successfully addressed the class imbalance issue, making the dataset more suitable for machine learning applications, particularly for tasks related to intrusion detection and network security. We took the log of the count, for easier understanding of the distribution.

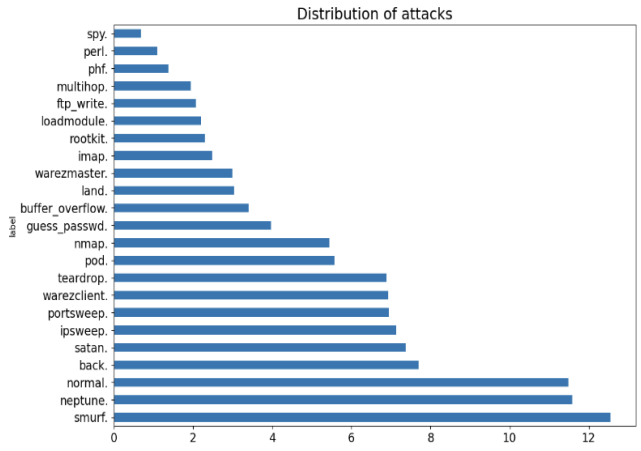

Fig 1

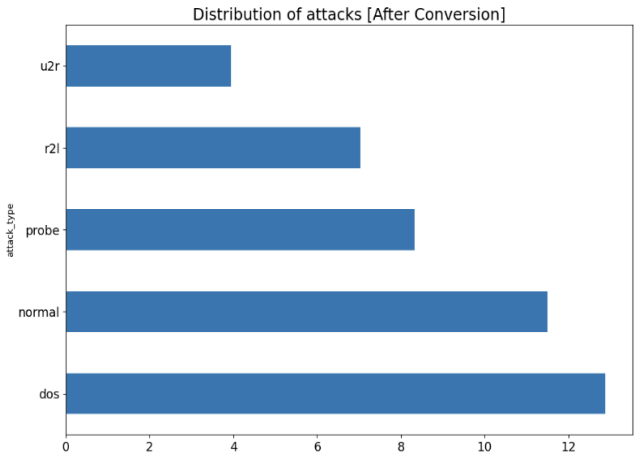

Fig 2

In our data exploration phase, we conducted an analysis of the dataset's attributes, specifically focusing on the count of unique values in each attribute. This examination revealed that two attributes, 'num_outbound_cmds' and 'is_host_login,' had a single unique value throughout the dataset, rendering them non-informative and non-contributory to our model training efforts. Consequently, we made a deliberate decision to eliminate these columns to streamline the dataset. Subsequently, we performed an array of data analyses supported by a variety of visualizations, including bar charts, pie charts, correlation matrices, heatmaps, histograms, density plots, and line charts. These analyses encompassed a thorough examination of attack type distributions, feature correlations, summary statistics, data distribution plots, class distributions. For a comprehensive understanding, please refer to the following visualizations. The following figure [Fig 3] talks about how the different attack_types exist in the corresponding protocol types like icmp, tcp and udp.

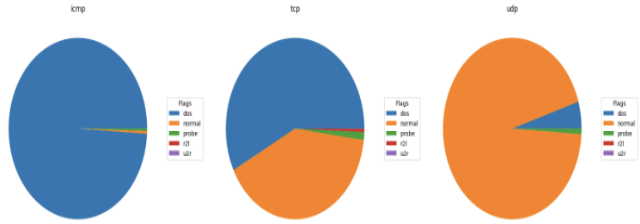

Fig 3

Based on the plots, it's evident that the 'tcp' and 'icmp' protocol types are experiencing a higher frequency of 'dos' attack types, whereas 'udp' protocol traffic appears to be predominantly normal. Next, we generated the following plot, which facilitated my comprehension of how each attack type is associated with distinct services.

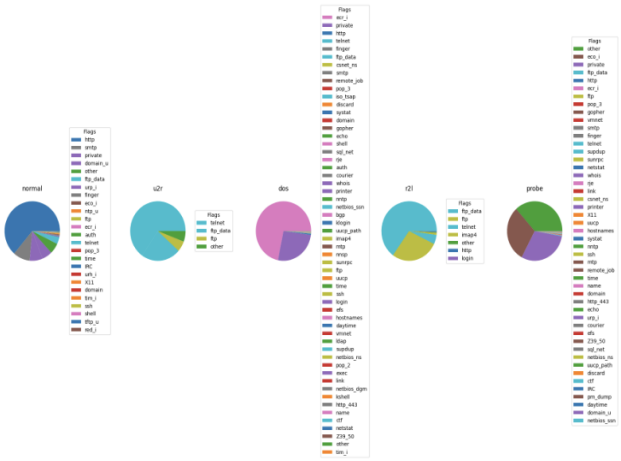

Fig 4

My conclusion was that the 'dos' attack type is linked to a greater variety of service types. Subsequently, we embarked on visualizing the distribution of each column, and we observed that the majority of columns exhibited skewness and had high kurtosis. The plot [Fig 5] presented below served as a valuable aid in enhancing our understanding of this phenomenon.

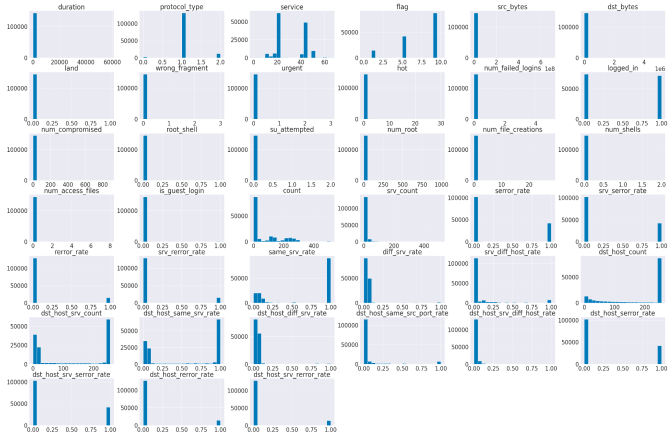

Fig 5

Moreover, our dataset contained categorical data, which needed to be transformed into a computer-readable format. To achieve this, I performed text vectorization and ended up having 41 columns and approximately 5 lakh data instances. A correlation plot [Fig 6] was then constructed to identify and eliminate features with high interdependencies, reducing multicollinearity that could adversely affect certain machine learning algorithms.

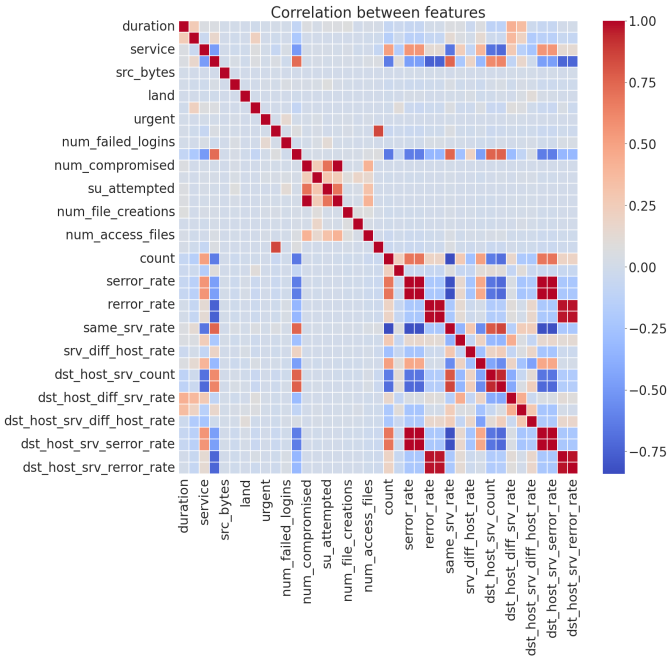

Fig 6

We observed a high degree of correlation among the majority of columns and decided to employ Principal Component Analysis (PCA) to reduce the dimensionality of the data. To determine the optimal number of components to retain, we conducted a variance explained plot [Fig. 7]. Upon examination, we noticed that the variance curve exhibited a sharp incline at around 20 components, indicating a substantial cumulative variance at this point. As a result, we selected 20 components as the appropriate parameter value for PCA dimension reduction.

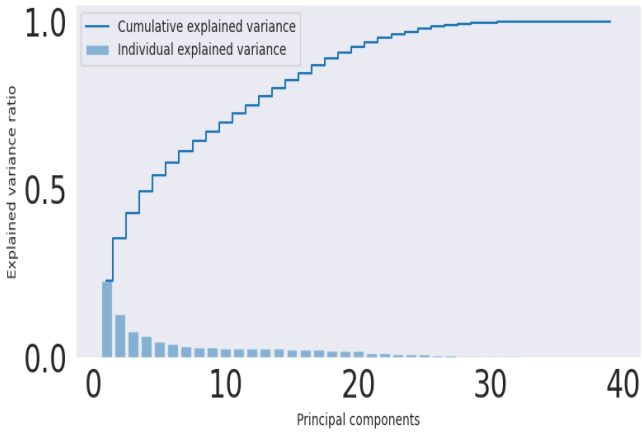

Fig 7

In our exploration of dimensionality reduction techniques alongside Principal Component Analysis (PCA), we specifically experimented with Incremental PCA. The objective was to identify the optimal number of components required to capture a substantial percentage of the dataset's variance. To achieve this, we generated a variance plot [Fig 8] that illustrates the cumulative explained variance against the number of components. This plot serves as a valuable tool for determining the point at which adding more components ceases to significantly contribute to capturing additional variance in the data. Upon careful analysis of the variance plot, we observed that achieving a 99% coverage of the data variance could be accomplished with just 20 components. This finding suggests that a dimensionality reduction to 20 components would be

both efficient and sufficient for retaining the critical information embedded in the dataset.

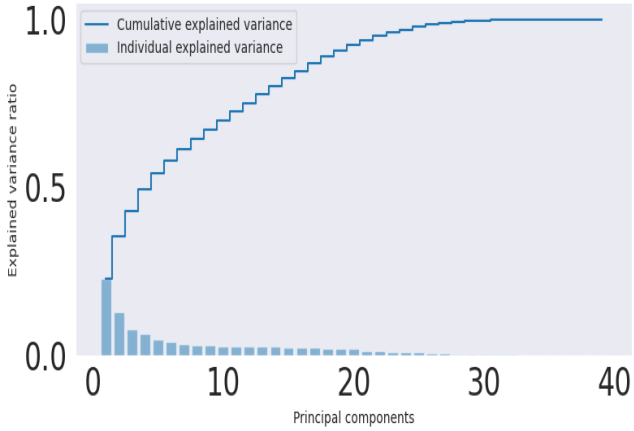

Fig 8

In our analysis, we expanded our exploration beyond Principal Component Analysis (PCA) to include Linear Discriminant Analysis (LDA). Through this process, we determined that incorporating four components from LDA yielded optimal results in terms of discriminating between classes and capturing relevant features in the data. Subsequently, we sought to further refine our dimensionality reduction methods by introducing a simple autoencoder. This autoencoder, composed of dense layers, allowed us to experiment with different numbers of components. Specifically, we systematically varied the number of components within the autoencoder, considering parameter values such as [2, 5, 10, 20, 23, 30, 35]. In order to provide a more comprehensive explanation, we calculated the Mean Squared Error (MSE) for all components, presenting the results in a tabulated format for a thorough examination of the reconstruction accuracy across various configurations [Table 1]. During this investigation, we focused on evaluating the Mean Squared Error (MSE) associated with each configuration. Notably, we observed that the MSE values remained consistently low for both 20 and 23 components. This observation indicated that the incremental increase in the number of components beyond 20 did not significantly contribute to improving the reconstruction accuracy or capturing additional information. Given the marginal difference in MSE between the configurations with 20 and 23 components, we made a strategic decision to opt for 20 components. This choice was driven by the desire to strike a balance between computational efficiency and data representation fidelity, ensuring that our subsequent analyses could be performed with a computationally feasible yet information-rich dataset. The selected configuration of 20 components in the autoencoder thus serves as an optimal compromise, aligning with our objectives of minimizing computational expense while retaining critical features for meaningful analysis

Num Components	Mean Square Error
2	0.7245
5	0.6782
10	0.6642
20	0.6613
23	0.6606
25	0.6632
30	0.6620

Table 1

The final step in our preprocessing was the application of Principal Component Analysis (PCA). This technique was employed to reduce the dimensionality of our dataset, condensing it into a set of principal components that encapsulate the most significant variance. PCA not only helped to mitigate the risk of overfitting but also enhanced the generalizability of the subsequent machine learning models. In addition to traditional PCA, we extended our preprocessing efforts to incorporate Incremental PCA, Linear Discriminant Analysis (LDA), and an autoencoder. These techniques collectively contributed to further refining our dataset. Notably,

we identified four distinct datasets resulting from each dimensionality reduction technique. The results from these dimensionality reduction methods are four different datasets namely, PCA Dataset, Incremental PCA Dataset, LDA Dataset, Autoencoder Dataset. This meticulous preprocessing, encompassing various dimensionality reduction methods, has culminated in a curated dataset that is specifically optimized for the task of detecting network anomalies. This refined dataset is now poised for the application of advanced machine learning techniques, setting the stage for robust and effective anomaly detection in our analysis.

5. Experimental Evaluation

After the completion of the preprocessing steps, including dimensionality reduction through techniques such as PCA, Incremental PCA, LDA, and an autoencoder, our next phase involved the application of machine learning models on the resulting five datasets. These datasets, each stemming from a distinct dimensionality reduction technique, were meticulously curated to enhance the performance of our models in the context of network anomaly detection. We employed a variety of machine learning models, leveraging their diverse strengths to capture the intricacies within the data. These models included KNN, Random Forest, Neural Networks, etc chosen for their suitability in handling the unique characteristics of each dataset.

The evaluation process focused on measuring the accuracy of each model in predicting network anomalies. Accuracy, a fundamental metric in classification tasks, reflects the proportion of correctly classified instances among the total instances. This assessment provided a comprehensive understanding of how well each model performed in capturing the underlying patterns within the data.The results of this evaluation not only shed light on the individual performance of each model but also allowed us to compare their effectiveness across the different datasets. Such a comparative analysis is crucial for identifying the model that excels in discerning anomalies within the specific context of each dimensionality-reduced dataset.

5.1 Machine Learning Models

As a crucial step in our machine learning workflow, we began by splitting the dataset into two subsets: the training set and the testing set. Approximately 70% of the data was allocated to the training set, while the remaining 30% was set aside for testing. This division ensures that we can train our models on one portion of the data and then evaluate their performance on unseen data.

5.1.1 Random Forest Classifier

We implemented the Random Forest Classifier, where training was conducted on 70% of the dataset. Remarkably, the model demonstrated high test accuracy across various datasets: 99.8% for the normal dataset, 99.8% for PCA, 99.8% for Incremental PCA, 99.5% for LDA, and 99.8% for the autoencoder. These accuracy scores underscore the classifier's effectiveness in accurately predicting anomalies across diverse scenarios. Following the training phase, the Random Forest Classifier was applied to the test data for evaluation. The achieved test accuracy rates serve as a crucial performance metric, highlighting the model's ability to generalize well to new, unseen data.

To gain deeper insights into the model's performance, we provide an accuracy report for each test dataset—normal, PCA, Incremental PCA, LDA, and autoencoder [Table 2,3,4,5,6]. These matrices offer a comprehensive breakdown of true positives, true negatives, false positives, and false negatives, providing valuable information on precision and recall. Precision measures the accuracy of positive predictions, while recall assesses the model's ability to capture all positive instances.Examining the confusion matrices for each test dataset allows for a detailed evaluation of the Random Forest Classifier's performance under various dimensionality reduction techniques. This nuanced analysis is essential for understanding not only overall accuracy but also the model's capacity to correctly identify anomalies,

minimize false positives or negatives, and ensure robust predictive capabilities

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	1.00	0.98	0.99	653
3	0.99	0.95	0.97	303
4	0.82	0.53	0.64	17
accuracy			1.00	43676
macro avg	0.96	0.89	0.92	43676
weighted avg	1.00	1.00	1.00	43676

Table 2

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.95	0.97	653
3	0.98	0.94	0.96	303
4	0.90	0.53	0.67	17
accuracy			1.00	43676
macro avg	0.97	0.88	0.92	43676
weighted avg	1.00	1.00	1.00	43676

Table 3

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.96	0.97	653
3	0.98	0.94	0.96	303
4	0.89	0.47	0.62	17
accuracy			1.00	43676
macro avg	0.97	0.87	0.91	43676
weighted avg	1.00	1.00	1.00	43676

Table 4

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.97	0.91	0.94	653
3	0.96	0.91	0.93	303
4	0.82	0.53	0.64	17
accuracy			1.00	43676
macro avg	0.95	0.87	0.90	43676
weighted avg	1.00	1.00	1.00	43676

Table 5

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.98	0.96	0.97	653
3	0.98	0.93	0.95	303
4	0.67	0.35	0.46	17
accuracy			1.00	43676
macro avg	0.93	0.85	0.88	43676
weighted avg	1.00	1.00	1.00	43676

Table 6

5.1.2 K- Nearest Neighbors

We employed the K Nearest Neighbors (KNN) method, training it on 70% of the dataset. The model demonstrated impressive test accuracy across diverse datasets: 99.7% for the normal dataset, 99.7% for PCA, 99.7% for Incremental PCA, 99.4% for LDA, and 99.7% for the autoencoder. These scores highlight KNN's effectiveness in accurately predicting anomalies across various scenarios. After training, the KNN method was applied to the test data, achieving noteworthy accuracy rates. An accuracy report for each test dataset normal, PCA, Incremental PCA, LDA, and autoencoder [Table 7,8,9,10,11 provides insights into true positives, true negatives, false positives, and false negatives, indicating precision and recall. Precision assesses positive prediction accuracy, while recall measures the model's ability to capture all positive instances.

Examining confusion matrices for each test dataset allows a detailed evaluation of the KNN method's performance under various dimensionality reduction techniques. This analysis is crucial for understanding overall accuracy, the model's proficiency in identifying anomalies, and its capability to minimize false positives or negatives.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.96	0.97	653
3	0.95	0.93	0.94	303
4	0.86	0.35	0.50	17
accuracy			1.00	43676
macro avg	0.96	0.85	0.88	43676
weighted avg	1.00	1.00	1.00	43676

Table 7

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.96	0.97	653
3	0.94	0.94	0.94	303
4	0.86	0.35	0.50	17
accuracy			1.00	43676
macro avg	0.96	0.85	0.88	43676
weighted avg	1.00	1.00	1.00	43676

Table 8

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.96	0.97	653
3	0.94	0.94	0.94	303
4	0.83	0.29	0.43	17
accuracy			1.00	43676
macro avg	0.95	0.84	0.87	43676
weighted avg	1.00	1.00	1.00	43676

Table 9

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.96	0.91	0.93	653
3	0.93	0.90	0.92	303
4	0.56	0.29	0.38	17
accuracy			0.99	43676
macro avg	0.89	0.82	0.85	43676
weighted avg	0.99	0.99	0.99	43676

Table 10

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.98	0.96	0.97	653
3	0.91	0.92	0.91	303
4	0.75	0.35	0.48	17
accuracy			1.00	43676
macro avg	0.93	0.85	0.87	43676
weighted avg	1.00	1.00	1.00	43676

Table 11

5.1.3 Naive Bayes

We utilized the Naive Bayes method, training on 70% of the dataset, and achieved impressive test accuracy: 73.7% for normal,74% PCA, 81.9 %Incremental PCA, and 79% autoencoder, and 96.7% for LDA. Post-training, Naive Bayes exhibited noteworthy accuracy on test data. An accuracy report for each dataset—normal, PCA, Incremental PCA, LDA, and autoencoder [Table 12,13,14,15,16] details true positives, true negatives, false positives, and false negatives, providing insights into precision and recall. Precision gauges positive prediction accuracy, while recall measures the model's ability to capture all positive instances. Examining confusion matrices for each test dataset allows a detailed evaluation of Naive Bayes' performance under various dimensionality reduction techniques, crucial for understanding overall accuracy and the model's anomaly detection capabilities.

	precision	recall	f1-score	support
0	0.92	0.79	0.85	16297
1	0.98	0.70	0.82	26406
2	0.08	0.99	0.15	653
3	0.41	0.49	0.44	303
4	0.01	0.71	0.01	17
accuracy			0.74	43676
macro avg	0.48	0.74	0.45	43676
weighted avg	0.94	0.74	0.82	43676

Table 12

	precision	recall	f1-score	support
0	0.75	0.95	0.84	16297
1	0.97	0.74	0.84	26406
2	0.26	0.88	0.40	653
3	0.52	0.39	0.44	303
4	0.02	0.47	0.03	17
accuracy			0.82	43676
macro avg	0.50	0.69	0.51	43676
weighted avg	0.88	0.82	0.83	43676

Table 13

	precision	recall	f1-score	support
0	0.65	0.95	0.77	16297
1	0.97	0.62	0.75	26406
2	0.24	0.88	0.38	653
3	0.45	0.39	0.42	303
4	0.03	0.47	0.05	17
accuracy			0.74	43676
macro avg	0.47	0.66	0.47	43676
weighted avg	0.83	0.74	0.75	43676

Table 14

	precision	recall	f1-score	support
0	1.00	0.97	0.98	16297
1	0.99	0.97	0.98	26406
2	0.40	0.92	0.55	653
3	0.67	0.86	0.75	303
4	0.35	0.41	0.38	17
accuracy			0.97	43676
macro avg	0.68	0.83	0.73	43676
weighted avg	0.98	0.97	0.97	43676

Table 15

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.98	0.96	0.97	653
3	0.91	0.92	0.91	303
4	0.75	0.35	0.48	17
accuracy			1.00	43676
macro avg	0.93	0.85	0.87	43676
weighted avg	1.00	1.00	1.00	43676

Table 16

5.1.4 AdaBoost Classifier

We implemented AdaBoost technique, training on 70% of the dataset, and achieved strong test accuracy: 96.2% for normal, 89.4% for PCA, 75.8% for Incremental PCA, and 97% for LDA and 95.6% for encoders. Post-training, AdaBoost demonstrated notable accuracy on the test data. An accuracy report for each dataset—normal, PCA, Incremental PCA, LDA, and autoencoder [Table 17, 18, 19, 20, 21]—details true positives, true negatives, false positives, and false negatives, providing insights into precision and recall. Examining confusion matrices allows a detailed evaluation of AdaBoost's performance under different dimensionality reduction techniques, essential for understanding overall accuracy and anomaly detection capabilities.

	precision	recall	f1-score	support
0	1.00	0.95	0.97	16297
1	0.95	0.99	0.97	26406
2	0.79	0.59	0.68	653
3	0.00	0.00	0.00	303
4	0.11	0.41	0.18	17
accuracy			0.96	43676
macro avg	0.57	0.59	0.56	43676
weighted avg	0.96	0.96	0.96	43676

Table 17

	precision	recall	f1-score	support
0	0.98	0.80	0.88	16297
1	0.98	0.96	0.97	26406
2	0.12	0.74	0.20	653
3	0.66	0.65	0.66	303
4	0.04	0.12	0.06	17
accuracy			0.89	43676
macro avg	0.55	0.65	0.55	43676
weighted avg	0.96	0.89	0.92	43676

Table 18

	precision	recall	f1-score	support
0	0.95	0.40	0.56	16297
1	0.99	0.98	0.98	26406
2	0.05	0.81	0.10	653
3	0.57	0.79	0.66	303
4	0.25	0.41	0.31	17
accuracy			0.76	43676
macro avg	0.56	0.68	0.52	43676
weighted avg	0.96	0.76	0.81	43676

Table 19

	precision	recall	f1-score	support
0	0.98	0.98	0.98	16297
1	0.97	0.99	0.98	26406
2	0.67	0.55	0.61	653
3	0.12	0.01	0.02	303
4	0.32	0.53	0.40	17
accuracy			0.97	43676
macro avg	0.61	0.61	0.60	43676
weighted avg	0.97	0.97	0.97	43676

Table 20

	precision	recall	f1-score	support
0	1.00	0.93	0.96	16297
1	0.98	0.98	0.98	26406
2	0.35	0.81	0.49	653
3	0.45	0.53	0.49	303
4	0.37	0.41	0.39	17
accuracy			0.96	43676
macro avg	0.63	0.73	0.66	43676
weighted avg	0.97	0.96	0.96	43676

Table 21

5.1.5 XGBoost Classifier

We applied XGBoost training on 70% of the dataset, and achieved strong test accuracy: 99.9% for normal, 74% for PCA, 81.9% for Incremental PCA, and 74% for LDA. Post-training, XGBoost demonstrated notable accuracy on the test data. An accuracy report for each dataset normal, PCA, Incremental PCA, LDA, and autoencoder [Table -22,23,24,25,26] details true positives, true negatives, false positives, and false negatives, providing insights into precision and recall. Examining confusion matrices allows a detailed evaluation of XGBoost's performance under different dimensionality reduction techniques, essential for understanding overall accuracy and anomaly detection capabilities.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	1.00	0.99	0.99	653
3	1.00	0.98	0.99	303
4	0.83	0.59	0.69	17
accuracy			1.00	43676
macro avg	0.97	0.91	0.93	43676
weighted avg	1.00	1.00	1.00	43676

Table 22

	precision	recall	f1-score	support
0	1.00	0.93	0.96	16297
1	0.98	0.98	0.98	26406
2	0.35	0.81	0.49	653
3	0.45	0.53	0.49	303
4	0.37	0.41	0.39	17
accuracy			0.96	43676
macro avg	0.63	0.73	0.66	43676
weighted avg	0.97	0.96	0.96	43676

Table 23

	precision	recall	f1-score	support
0	1.00	0.93	0.96	16297
1	0.98	0.98	0.98	26406
2	0.35	0.81	0.49	653
3	0.45	0.53	0.49	303
4	0.37	0.41	0.39	17
accuracy			0.96	43676
macro avg	0.63	0.73	0.66	43676
weighted avg	0.97	0.96	0.96	43676

Table 24

	precision	recall	f1-score	support
0	1.00	0.93	0.96	16297
1	0.98	0.98	0.98	26406
2	0.35	0.81	0.49	653
3	0.45	0.53	0.49	303
4	0.37	0.41	0.39	17
accuracy			0.96	43676
macro avg	0.63	0.73	0.66	43676
weighted avg	0.97	0.96	0.96	43676

Table 25

	precision	recall	f1-score	support
0	1.00	0.93	0.96	16297
1	0.98	0.98	0.98	26406
2	0.35	0.81	0.49	653
3	0.45	0.53	0.49	303
4	0.37	0.41	0.39	17
accuracy			0.96	43676
macro avg	0.63	0.73	0.66	43676
weighted avg	0.97	0.96	0.96	43676

Table 26

5.1.6 Ensemble Methods

We implemented an ensemble method using a Voting Classifier with AdaBoost, SVC, GaussianNB, KNeighbors, and RandomForest. Trained on 70% of the dataset, the ensemble demonstrated strong test accuracy: 99.7% for normal, 99.6% for PCA, 99.6% for Incremental PCA, and 99.8% for LDA and 99.5% for encoders. Post-training, the ensemble showcased robust performance in predicting anomalies.An accuracy report for each dataset—normal, PCA, Incremental PCA, LDA, and autoencoder [Table 12, 13, 14, 15, 16]—provides insights into true positives, true negatives, false positives, and false negatives, detailing precision and recall. Examining confusion matrices allows a thorough evaluation of the ensemble's performance under different dimensionality reduction techniques, crucial for overall accuracy and anomaly detection capabilities. The ensemble's combination of classifiers through soft voting enhances its ability to capture complex patterns within the data, contributing to its overall effectiveness in anomaly detection

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.96	0.97	653
3	0.95	0.89	0.92	303
4	0.58	0.41	0.48	17
accuracy			1.00	43676
macro avg	0.90	0.85	0.87	43676
weighted avg	1.00	1.00	1.00	43676

Table 23

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.94	0.96	653
3	0.94	0.83	0.88	303
4	1.00	0.35	0.52	17
accuracy			1.00	43676
macro avg	0.98	0.82	0.87	43676
weighted avg	1.00	1.00	1.00	43676

Table 23

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.99	0.94	0.96	653
3	0.94	0.83	0.88	303
4	1.00	0.35	0.52	17
accuracy			1.00	43676
macro avg	0.98	0.82	0.87	43676
weighted avg	1.00	1.00	1.00	43676

Table 23

	precision	recall	f1-score	support
0	1.00	0.98	0.99	16297
1	0.99	1.00	0.99	26406
2	0.91	0.85	0.88	653
3	0.95	0.83	0.89	303
4	0.75	0.35	0.48	17
accuracy			0.99	43676
macro avg	0.92	0.80	0.85	43676
weighted avg	0.99	0.99	0.99	43676

Table 23

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16297
1	1.00	1.00	1.00	26406
2	0.97	0.94	0.96	653
3	0.88	0.80	0.83	303
4	0.60	0.35	0.44	17
accuracy			1.00	43676
macro avg	0.89	0.82	0.85	43676
weighted avg	1.00	1.00	1.00	43676

Table 23

5.1.7 Neural Networks

We implemented a simple neural network with dense layers, trained for 10 epochs using a batch size of 32. We utilized sparse categorical entropy as the loss function and the Adam optimizer. The network yielded noteworthy accuracies, complementing results obtained from previous models, including the ensemble trained on 70% of the dataset. Notably, the ensemble exhibited robust performance across diverse datasets, achieving strong test accuracy: 99.7% for normal, 99.6% for PCA, 99.6% for Incremental PCA, and 99.6% for LDA and 99.5% for encoders. The ensemble's post-training accuracy emphasizes its effectiveness in predicting anomalies. This collective analysis provides a comprehensive understanding of anomaly detection capabilities across different models and techniques.

6. Generative Adversarial Networks

In response to the dataset imbalance, where classes 0 and 1 had a higher number of instances than classes 3, 4, and 5, I employed Generative Adversarial Networks (GANs). Three distinct GAN models were crafted, each specialized in generating synthetic data for the underrepresented classes (3, 4, and 5). This strategic approach aimed to rectify the class distribution imbalance. The GAN models were trained to discern patterns and features inherent in the original data for the respective minority classes. Their purpose was to generate synthetic instances closely resembling the characteristics of the underrepresented classes. The result was the creation of a new dataset with an equal number of instances for each class, effectively alleviating the imbalance. This balanced dataset, generated through GANs, enhances model training by ensuring equitable representation across all classes. This information is pertinent to the comprehensive report, showcasing the implementation of GANs as a valuable tool for addressing class imbalance challenges in our machine learning approach.

6.1 GAN Architecture

The GAN consists of three main components: a generator, a discriminator, and the GAN model itself. The generator is responsible for creating synthetic data points that mimic the underlying patterns of the original dataset. It comprises three dense layers, each introducing non-linearity through Rectified Linear Unit (ReLU) activation functions. Batch normalization is applied after the first two layers to stabilize and expedite the training process. The final layer uses the hyperbolic tangent (tanh) activation function to ensure the generated data falls within the range of -1 to 1, aligning with the distribution of the original data. The discriminator evaluates whether a given data point is real (from the original dataset) or synthetic (generated by the generator). It consists of three dense layers with ReLU activations in the initial layers and a sigmoid activation function in the output layer, providing a probability score between 0 and 1. Dropout layers are inserted after the first two dense layers to mitigate overfitting and enhance the model's generalization capabilities. The GAN model serves as the overarching framework, combining the generator and discriminator. During training, the discriminator's weights are frozen to prevent updates, ensuring that the generator is challenged to produce synthetic data capable of fooling the discriminator. The GAN model is compiled using the Adam optimizer and binary cross-entropy loss. The training loop iterates over a specified number of epochs, with each epoch involving the training of both the discriminator and generator. Real and synthetic data batches are sampled, and the discriminator is trained to distinguish between them. The generator, in turn, is

trained to produce synthetic data that is convincing enough to deceive the discriminator. This adversarial training process continues to refine both the generator and discriminator over epochs.

6.2 Classification Model

Upon obtaining a more balanced dataset, a two-pronged approach was undertaken. First, Principal Component Analysis (PCA) was applied to the data, serving the dual purpose of reducing dimensionality and capturing essential features that contribute to the variance in the dataset. This preprocessing step aimed to enhance the efficiency of subsequent machine learning models.Following PCA, we subjected the transformed datasets to the scrutiny of three distinct models: Ensemble, XGBoost, and Neural Network (NN). Each of these models brings a unique set of strengths to the table. The Ensemble model amalgamates the predictive capabilities of multiple classifiers, harnessing their collective power for improved accuracy and robustness. XGBoost, a renowned gradient boosting algorithm, excels in handling complex datasets and capturing intricate patterns. Meanwhile, the Neural Network, a deep learning model, showcases its capacity to discern nonlinear relationships within the data.

The resulting accuracies, meticulously recorded in the presented table [Table 24], serve as a comprehensive evaluation of model performance on the balanced datasets. This scrutiny goes beyond mere accuracy metrics, aiming to uncover nuanced insights into how each model interprets and responds to the rebalanced data. By employing diverse modeling techniques, this analysis seeks to offer a thorough understanding of the strengths and limitations of Ensemble, XGBoost, and Neural Network in handling datasets with improved class balance.

	Accuracy	
Technique	**Original Data**	**PCA**
XGBoost	0.925	0.918
Ensemble	0.920	0.913
Neural Nets	0.908	0.905

Table 24

7. Conclusions

The comprehensive exploration and modeling of the KDD 99 dataset for network anomaly detection unfolded through a multifaceted approach. In the initial phase, a meticulous analysis of the dataset was conducted, revealing areas characterized by class imbalance, notably among instances with limited occurrences. This identification of imbalance prompted the implementation of an inventive solution – a data resampling technique based on Generative Adversarial Networks (GANs). This pioneering method involved generating synthetic data instances through GANs, effectively mitigating the class imbalance and ensuring a more balanced representation of underrepresented classes in the dataset.

A critical aspect of the study involved a deep dive into the impact of dimensionality reduction techniques during the model training phase, with a specific emphasis on methodologies such as Principal Component Analysis (PCA). Dimensionality reduction emerged as a pivotal strategy for refining the dataset, reducing its complexity, and optimizing it for subsequent machine learning models. By employing techniques like PCA, LDA, encoders, Incremental PCA the dataset was transformed into a condensed set of essential features, thereby enhancing the interpretability and efficiency of the subsequent models.

This holistic analysis and modeling strategy not only demonstrated a nuanced understanding of the intricacies within the KDD 99 dataset but also underscored the profound significance of innovative methodologies, including GANs and dimensionality reduction. These techniques emerged as powerful tools in the arsenal of network anomaly detection systems, significantly contributing to improved overall performance and the ability to effectively address challenges posed by class imbalance and high dimensionality in real-world datasets.