Data Visualization and Preprocessing for Network Traffic Analysis in Cybersecurity

Mayank Kapadia, *Student, SJSU,* Andrew Dunton, *Student, SJSU,*
Chelsea Jaculina, *Student, SJSU,* Albert Ong, *Student, SJSU,* and David Thach, *Student, SJSU*

***Abstract*—In the age of cybersecurity, visualization and interpretation of network traffic data is very crucial for real-time intrusion detection. The proposed paper provides a data visualization-driven approach for analyzing network intrusions using the CICIDS 2017 dataset. The study also uses different preprocessing techniques, such as data cleaning, transformation, and feature selection, to make the data set ready for analysis. Principal Component Analysis, or PCA, is used for reducing dimensionality, helping to optimize memory, and clarifying visualizations. We will focus on the visualization of network attack patterns, model performance, and PCA results that provides actionable insights. Python libraries are used in conjunction with Power BI to create a data visualization platform that has interactive real-time visualizations for users to explore across attack types, feature importance, and model evaluation metrics. The goal is to show how effective data visualization can improve the understanding of complex network traffic data and help make better decisions in cybersecurity.**

***Index Terms*—anomaly detection, cybersecurity, data visualization, intrusion detection, network traffic analysis**

I. Introduction

CYBERSECURITY is one of the biggest challenges facing the digital world today. As the frequency and complexity of cyber-attacks continue to rise, organizations are more reliant on Intrusion Detection Systems to identify malicious network activities and secure their infrastructures. However, the high volume and complexity of network traffic data pose quite a few challenges to the effective detection and understanding of such attacks. Traditional network traffic analysis methods, based on raw statistics and numerical data sets, usually become insufficient when dealing with complexities inherent in high-dimensional data. In addition, although machine learning techniques can efficiently classify network intrusions, model interpretability and real-time decision-making capabilities are still limited. This research proposes one such methodology by combining strategies in data visualization with frameworks in machine learning. The CICIDS 2017 dataset was chosen because it is replete with network traffic information. This study, therefore, uses Principal Component Analysis for the reduction of dimensionality so that real-time visualization of network traffic can be achieved over platforms like Power BI for comprehension, model evaluation, and proactive security measures in detecting complex attack patterns.

A. Motivation

With increased volume and complexity of network traffic, coupled with modern complex cyberattacks, traditional methods of intrusion detection become less effective. Modern Network Intrusion Detection Systems can produce vast amounts of information, which may be complicated to analyze and interpret by themselves. Moreover, the very complexity of the models, with their powerfulness in forecasting attack types, may turn out to make cybersecurity professionals uncertain about what features are driving the prediction. What has driven this research, therefore, is a great interest in improving the result interpretation of machine learning and facilitating easier analysis of high-dimensional data in cybersecurity. Data visualization, in this line of thinking, provides a pathway for transforming complex network traffic data into more understandable visual forms. These visualizations are able to represent patterns of attack, feature importance, and relationships that may not be vividly apparent in the raw data alone. In addition, applying PCA [1] reduces the dimensionality of the dataset, enabling more efficient processing and clearer visualizations while saving computational costs without giving up important information. The goal is to arm cybersecurity professionals with hands-on and practical knowledge that can amplify their ability to detect, understand, and respond to network attacks. This becomes even more critical in real-time systems, where timely decision-making is critical to minimizing the impact of intrusions.

B. Contribution

This paper makes the following significant contributions to the field of network intrusion detection and data visualization:

1. Integration of Data Visualization and Machine Learning: We offer a novel approach to analyzing and presenting network traffic data that combines data visualization [2] and machine learning algorithms. The research focuses on converting sophisticated intrusion detection data into intuitive, interactive visual forms that are easier to understand for cybersecurity professionals.

2. Application of Principal Component Analysis (PCA): This study shows how PCA may be used to reduce the dimensionality of the CICIDS 2017 dataset [4], which has 85 features. Using PCA, we simplify the data, enhancing visualization clarity and model interpretability while keeping the majority of the variation.

3. Real-Time Visualization in Power BI: This study advances the usage of Power BI for real-time network traffic visualization. We demonstrate how Power BI can be used to dynamically simulate network traffic, resulting in real-time visualizations of attack types and model predictions to aid in proactive decision making.

4. Actionable Intelligence for Cybersecurity Decision Making: Using interactive dashboards, this study provides clear, actionable information into feature relevance, attack detection trends, and model performance. This enables cybersecurity professionals to make data-driven decisions faster and more confidently.

5. Real-Time Attack Simulation: Using Power BI, we simulate real-time attack detection, illustrating the capabilities of real-time intrusion detection systems powered by machine learning models and data visualization tools. This method can assist detect and respond to cyber threats dynamically.

C. Organization

The rest of the paper is organized as follows:

Section 2: Related Work

This section provides an overview of previous research, including studies on network intrusion detection, the application of machine learning for cybersecurity, and the role of dimensionality reduction techniques, like PCA, in improving model performance.

Section 3: Proposed Methodology

This section explains the step-by-step approach adopted in the study. It begins with the preprocessing of the CICIDS 2017 dataset, covering data cleaning and transformation. This is followed by the application of PCA for dimensionality reduction and its effect on feature selection. Machine learning models used for intrusion detection are also detailed, including discussions on accuracy, feature importance, and classification results.

Section 4: Data Visualization

This section highlights the visual outputs of the study, showcasing how data insights and model results were represented. Visualizations created using Python and Power BI are discussed, including real-time simulations and interactive dashboards.

Section 5: Future Work

This section details potential areas of further exploration, such as the use of advanced machine learning models, the application of other dimensionality reduction techniques, scalability to larger datasets, and the development of improved real-time data visualization frameworks.

Section 6: Paper Highlights

This section summarizes the key contributions and findings of the study, focusing on the integration of machine learning with data visualization to enhance the effectiveness of network intrusion detection systems.

Section 7: Conclusion

The paper concludes with a summary of the major insights gained from the research. It emphasizes the impact of combining dimensionality reduction, machine learning, and data visualization techniques to improve cybersecurity systems' interpretability and efficiency.

Figure 1 visually represents the major sections discussed in this work, offering an overview of the paper's structure and organization.

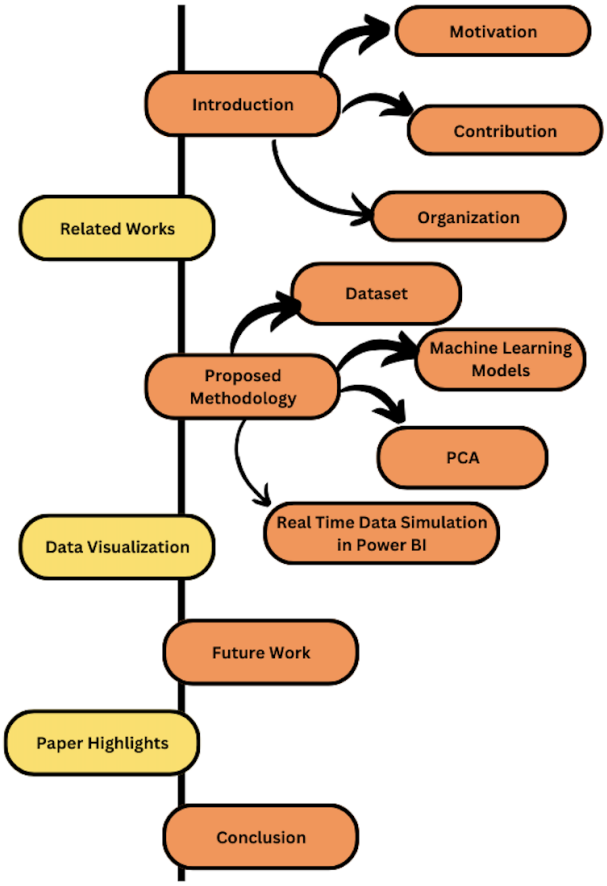

Fig. 1. Topics covered in this paper

II. Related Work

PCA is a powerful technique for network security analysis, particularly in the context of intrusion detection systems. Early work by Wang and Battiti [2] demonstrated PCA's effectivess in reducing high-dimensional network traffic data while maintaining detection accuracy, achieving rates above 99% on standard datasets. This research has been extended by recent studies, including Osho et al. [3] and Almaiah et al. [4], who explored combining PCA with various machine learning classifiers. These studies consistently show that PCA-based dimensionality reduction not only improves computational efficiency but also enhances the accuracy of anomaly detection when properly integrated with classification algorithms.

The challenges of visualizing complex network security data have been extensively studied, with researchers emphasizing the importance of creating intuitive and interactive visualization tools. Adams and Snider [5] conducted a comprehensive analysis of cybersecurity visualization challenges, identifying key issues such as tool rigidity, visualization complexity, and the critical need for user-centered design. Their research highlights that while data visualization can significantly improve the understanding of network traffic patterns, poor implementation can lead to tools that security analysts find difficult to

use or interpret. This aligns with our approach of combining dimensionality reduction with thoughtful visualization design to create more accessible and effective analysis tools.

Recent work by Ullah et al. [6] has specifically focused on analyzing web browsing activity using PCA and demonstrates how focusing on the most relevant principal components can improve both analytical accuracy and computational efficiency. Their research showed that by identifying affected dimensions in network traffic and calculating appropriate anomaly scores, security analysts can more effectively detect and investigate potential threats. This work, along with similar studies in the field, suggests that successful network security visualization requires both effective dimensionality reduction and careful considering of how the reduced data is presented to analysts. Our research builds upon these findings by combining PCA-based dimensionality reduction with interactive visualization techniques, addressing the identified challenges while maintaining the computational efficiency necessary for real-time analysis.

III. Proposed Methodology

In this section, we describe the steps taken to preprocess the CICIDS 2017 dataset and the techniques used to analyze the network attack types. The goal of this methodology is to apply Principal Component Analysis (PCA) to reduce the dimensionality of the data while maintaining the accuracy of attack prediction. By leveraging PCA, we aim to achieve faster model training times and reduced memory usage without significantly affecting the model's accuracy. Additionally, real-time simulation and visualization are applied using Power BI to monitor and evaluate the model's performance in a dynamic environment. The following subsections detail the steps involved in preprocessing the data, applying PCA, and creating real-time visualizations for model evaluation.

A. Dataset

1) ***Dataset Overview****:* The CICIDS 2017 dataset, released by the Canadian Institute for Cybersecurity, includes tagged examples of various network attacks. It is a popular dataset for network intrusion detection research, containing 85 features such as flow statistics and network traffic patterns. These features collect critical network traffic information such as source and destination addresses, packet counts, and flow time, among others.

The dataset is separated into attack types, such as DoS (Denial of Service), DDoS (Distributed Denial of Service), Brute Force, and SQL Injection and many more as shown in Fig. 2 which allows for attack detection-based classification. For this study, we concentrate on preprocessing the dataset to eliminate extraneous features and convert the data into a format appropriate for machine learning analysis. In this research, we use the CICIDS 2017 dataset to predict the type of network assault, with a focus on improving model performance through dimensionality reduction using PCA and presenting the findings in real time with Power BI.

2) ***Pre-Processing****:* As shown in Fig. 3, to create a consistent and high-quality dataset, we performed the following pre-processing steps:

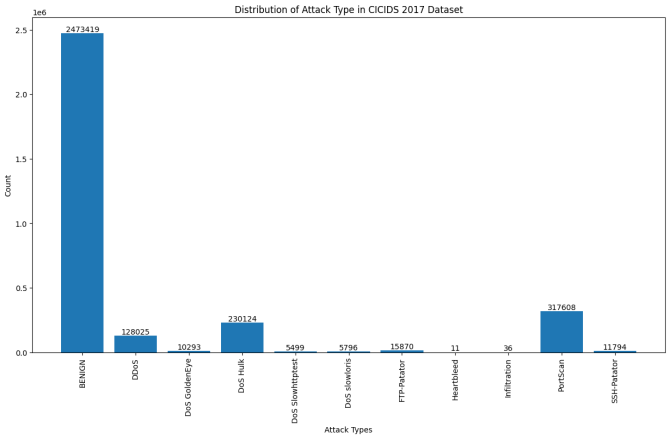

Fig. 2. Distribution of Attack Type in CICIDS 2017 Dataset

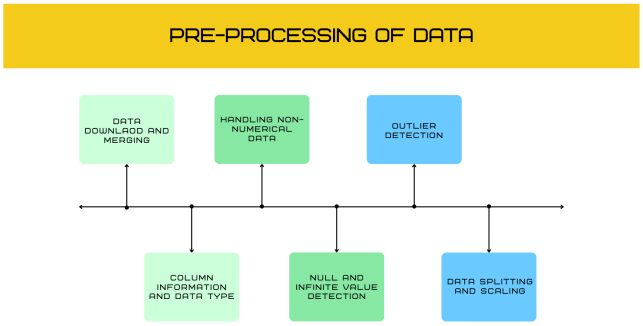

Fig. 3. Pre-Processing Steps

a) ***Data Download and Merging****:* The CICIDS 2017 dataset, containing daily logs from an experiment conducted over six days (Monday to Sunday), was downloaded from the official website as shown in Fig. 4. We then merged each CSV file into one unified dataset to simplify the analysis.

b) ***Column Information and Data Types****:* We retrieved and examined the column information from the merged dataset to identify the data types and values. Most of the columns were numerical, with exceptions being non-numerical columns such as Label, Flow ID, Timestamp, Source IP, and Destination IP (Shown in Fig. 5).

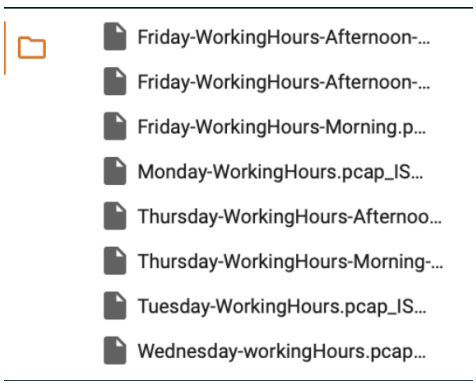

Fig. 4. Various CSV Files

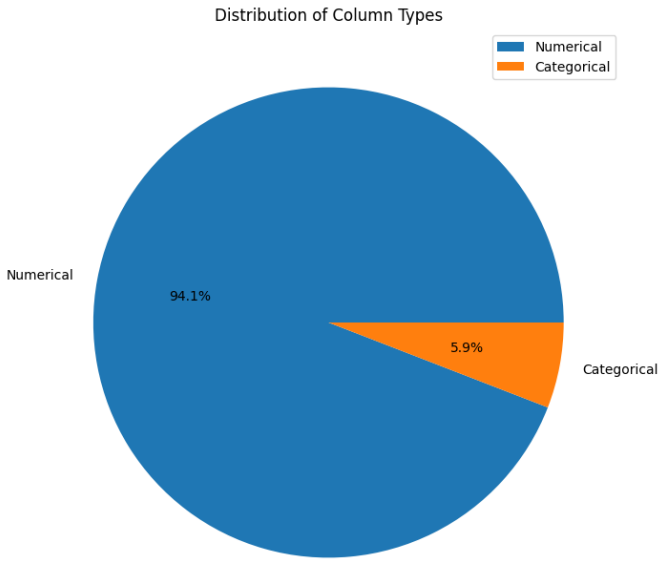

Fig. 5. Distribution of Column Types

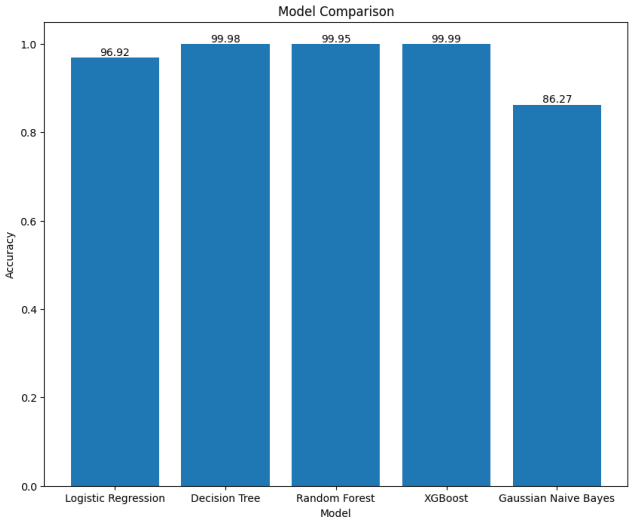

Fig. 6. Accuracy of Different Model

*c) **Handling Non-Numerical Data**:* The Label column, being categorical, was encoded to numerical values for machine learning purposes. Columns like Source IP, Destination IP, and Flow ID were identified as irrelevant for the analysis and were subsequently excluded from the dataset.

*d) **Null and Infinite Value Detection**:* The dataset was examined for null values, particularly in the Flow Bytes/s column, which contained missing data. We opted to impute the missing values with the median of each label group, as the median is more robust to outliers than the mean. Additionally, we checked for infinite values in columns like Flow Bytes/s and Packet, which were found and replaced with null values to ensure consistency.

*e) **Outlier Detection**:* After cleaning the data, we checked for outliers in the numerical columns but found none. Given the nature of this dataset and its relevance for real-time prediction, every piece of information was considered important.

*f) **Data Splitting and Scaling**:* We split the dataset into training and testing sets. Standard scaling was applied after the split to ensure that the model and visualizations work effectively with the scaled features. Scaling was not applied prior to the split to ensure that the data was properly represented for visualization in its original form.

Data Preprocessing and Its Impact on Model Performance: By downloading and merging the CICIDS 2017 dataset, followed by a series of preprocessing steps including handling non-numerical columns, addressing missing and infinite values, and applying feature selection, we ensured that our dataset was clean and ready for analysis. Additionally, we standardized the data after splitting it into training and testing sets, which improved the model's ability to learn effectively. This rigorous preprocessing ensured fewer discrepancies during model training, leading to more efficient learning. By carefully cleaning and preparing the dataset, we enhanced the quality of the input data, resulting in a model that could learn more generalized features and perform well across a variety of attack types.

B. Machine Learning Models:

After preprocessing and scaling the dataset using Standard Scaling (Step f in Preprocessing), we proceeded to train several supervised machine learning models to classify the network traffic into benign and attack categories. The primary reason for applying standard scaling was to address the large range of numerical values in the dataset, which could lead to model instability or poor performance if not properly normalized. Scaling the data ensures that all features contribute equally to the model, with each having a mean of 0 and a standard deviation of 1, preventing certain features with larger values from disproportionately influencing the model.

Once the dataset was scaled, we used the scaled training and testing datasets to train and evaluate the following models:

- Decision Tree Classifier [11]
- Random Forest Classifier [3]
- XGBoost Classifier [5]
- Logistic Regression [7]
- Gaussian Naive Bayes [9]

Each of these models was selected based on its relevance and performance in classification tasks. We evaluated the models using accuracy as the primary metric.

Model Comparison: After training the models, we compared their performance and found that the XGBoost model produced the highest accuracy (as shown in Fig. 6), making it the best-performing model for this task. This was confirmed by calculating the accuracy for each model and evaluating it against the test dataset.

Confusion Matrix: To gain deeper insights into the performance of each model, we analyzed the confusion matrix for each. The confusion matrix is a valuable tool for understanding the types of errors each model made, especially in distinguishing between different attack types. For each model, we plotted the confusion matrix.

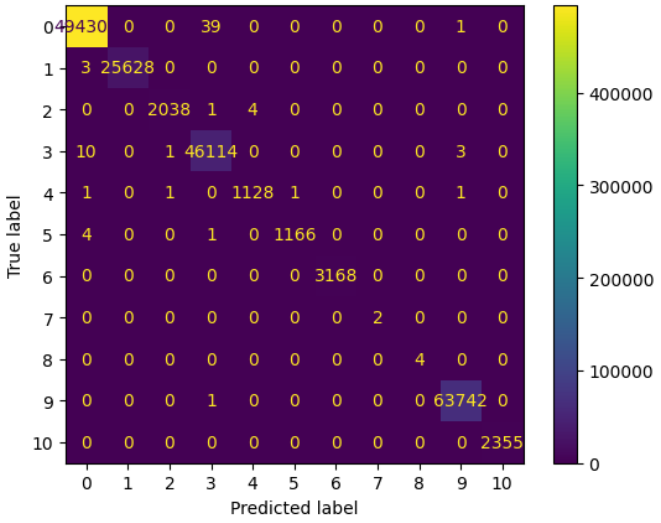

Fig. 7. Confusion Matrix of XGBoost Classifier

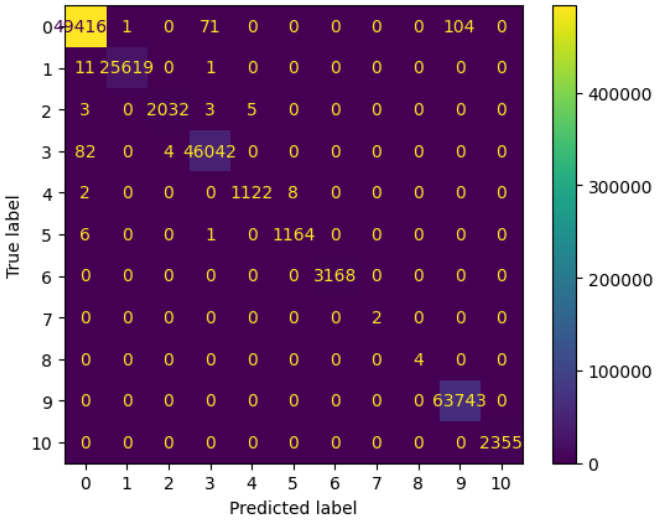

Fig. 8. Confusion Matrix of Random Forest Classifier

By carefully analyzing the confusion matrices presented in Fig. 7, 8, 9, 10, and 11, we observed distinct patterns in the classification performance of each model. Among all the models, the Gaussian Naive Bayes classifier demonstrated the weakest performance, as evidenced by the poor values along its diagonal, which correspond to correct predictions. This indicates a higher rate of misclassification compared to the other models, making it unsuitable for our analysis.

The Logistic Regression model also exhibited limitations, particularly with an elevated number of misclassified instances concentrated in the first row of its confusion matrix. This suggests a bias in its predictions, which compromises its overall reliability when compared to tree-based models. Focusing on the tree-based classifiers — Decision Tree, Random Forest [6] , and XGBoost — we evaluated their confusion matrices based on the number of off-diagonal elements, which represent misclassified samples. XGBoost emerged as the most robust model, with the fewest off-diagonal values, implying the highest precision in its predictions. Additionally, XGBoost's confusion matrix contained the largest proportion of zero entries outside the diagonal, further supporting its superior ability to accurately classify data across all categories. In summary, the confusion matrix analysis reinforces our conclusion that the XGBoost classifier significantly outperforms the other models, not only in terms of accuracy metrics but also in its ability to minimize misclassifications. This finding aligns with and substantiates the results derived from the accuracy metrics, solidifying XGBoost as the optimal choice for our dataset.

Feature Importance and Model Interpretability:

For the tree-based models (Decision Tree, Random Forest, and XGBoost), we plotted the feature importance, as shown in Fig. 12, 13, and 14, to analyze the relative contribution of each feature to the model's predictions. Feature importance quantifies [8] how much each feature influences the model's decision-making process, providing insights into the underlying factors driving predictions. This analysis is critical for understanding the model's behavior and identifying the key indicators of network traffic anomalies or attacks.

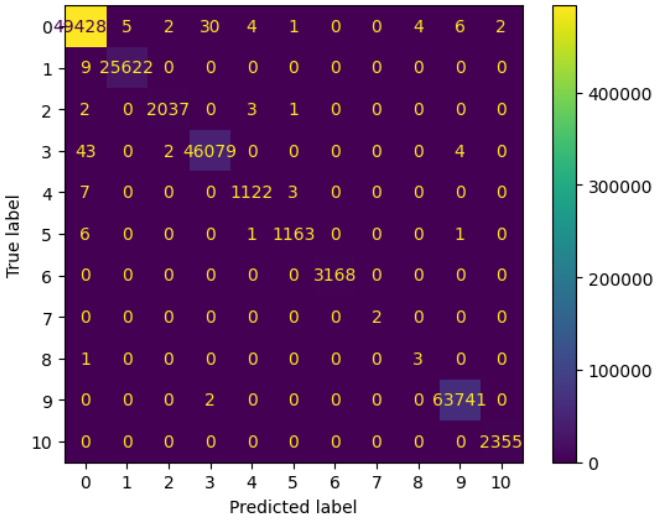

Fig. 9. Confusion Matrix of Decision Tree Classifier

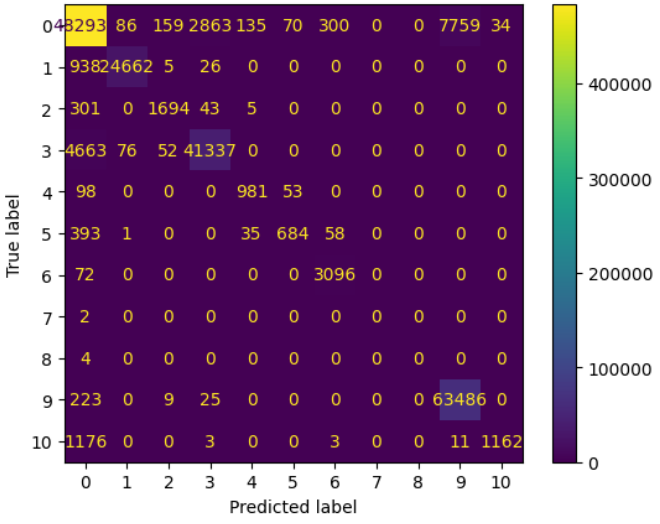

Fig. 10. Confusion Matrix of Logistic Regression

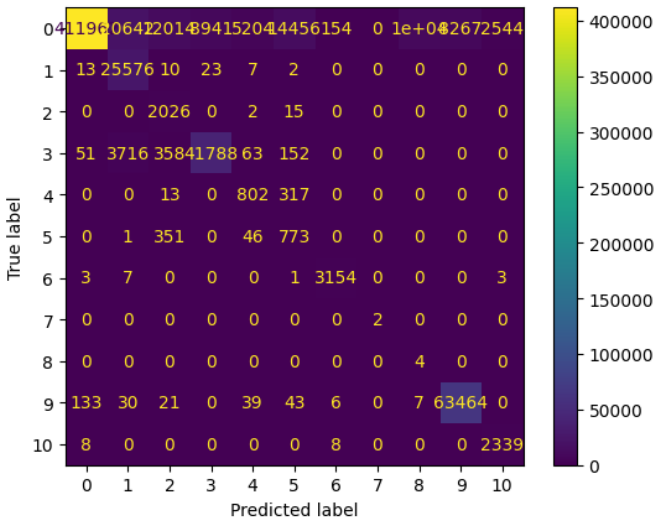

Fig. 11. Confusion Matrix of Gaussian Naive Bayes

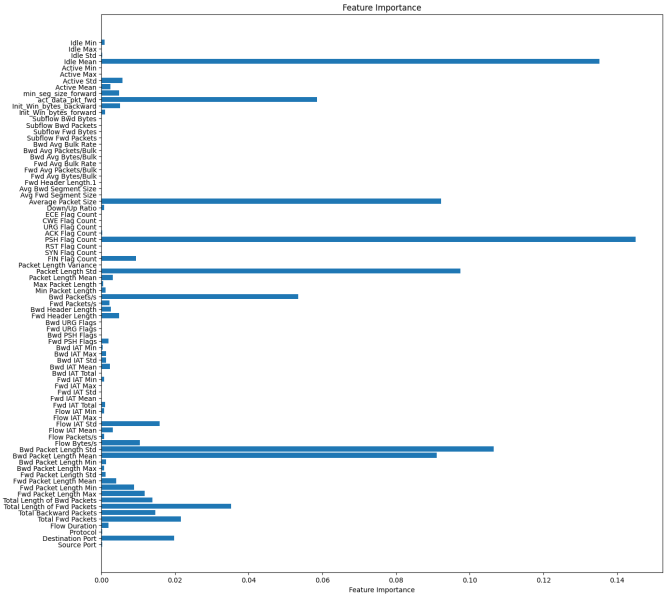

Fig. 12. Feature Importance from XGBoost

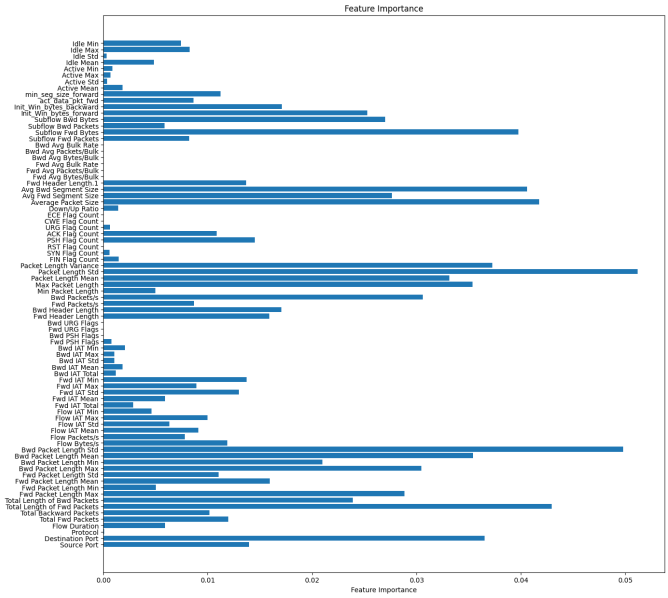

Fig. 13. Feature Importance from Random Forest

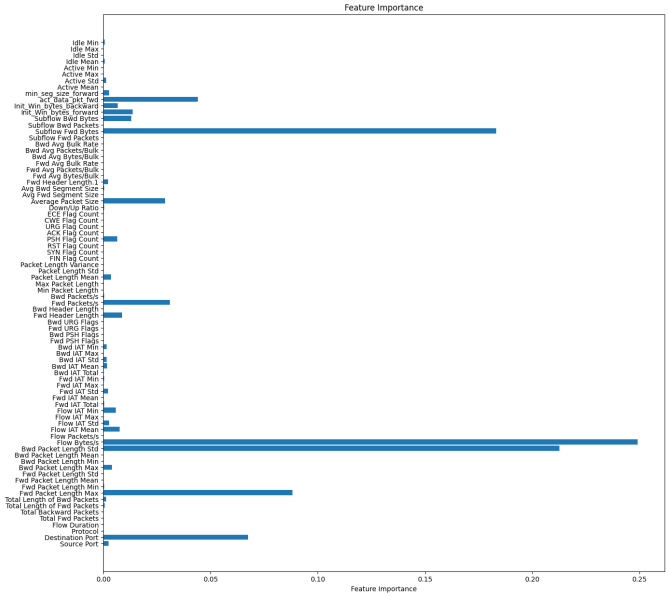

Fig. 14. Feature Importance from Decision Tree

Similarly, for Logistic Regression, we analyzed the coefficients of the model as shown in Fig. 15, which indicate the relationship between each feature and the model's predictions. Positive coefficients suggest that higher values of a feature increase the likelihood of an attack being classified, while negative coefficients suggest the opposite. We visualized these coefficients using bar plots for easier interpretation.

Bar plots were used to visualize the feature importance for each tree-based model, making it easier to compare their interpretations. From the feature importance plots, we observed that Random Forest emphasizes a broader range of features, assigning importance to many of them. In contrast, XGBoost and Decision Tree classifiers identify a more focused subset of significant features. This suggests that Random Forest may overfit the dataset by relying on too many features, which could reduce its generalization performance. On the other hand, XGBoost and Decision Tree models highlight fewer, more critical features, making them more efficient for our analysis.

Comparing XGBoost and Decision Tree, XGBoost provides a clearer and more distinct ranking of feature importance, supporting its overall superior performance in the accuracy metrics and confusion matrix analysis. This alignment across multiple evaluation criteria reinforces the conclusion that XGBoost is the best-performing model for our dataset. Consequently, we have chosen XGBoost for further visualization and analysis in this study.

While Logistic Regression offers interpretable coefficients that indicate the contribution of each feature, its feature importance plot shows a less clear delineation of significant

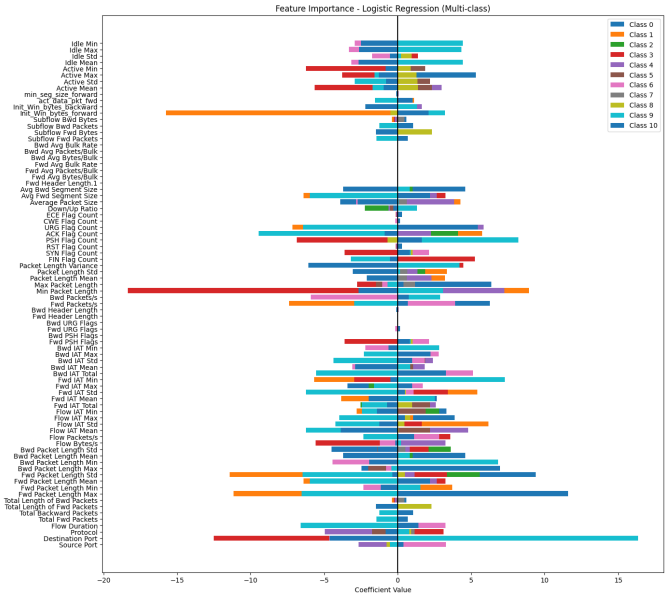

Fig. 15. Feature Importance from Logistic Regression

predictors compared to tree-based models. Logistic Regression assumes a linear relationship between features and the target variable, which may oversimplify the complex patterns inherent in our dataset. Additionally, its performance, as seen in the confusion matrix, is inferior to XGBoost, Random Forest, and Decision Tree, particularly in handling imbalanced classes. These limitations make Logistic Regression unsuitable for our objectives, and we prioritize tree-based models, with XGBoost being the most optimal choice.

C. PCA(Principal Component Analysis)

a) ***PCA Overview****:* Principal Component Analysis (PCA) [10] is a dimensionality reduction technique that transforms the original feature space into a smaller set of uncorrelated features, known as principal components. These components capture the maximum variance present in the data, reducing the number of features while retaining the most significant information. In the context of our project, PCA was applied after preprocessing to reduce the number of features, helping to mitigate the risk of overfitting and enhance computational efficiency. PCA allows the model to focus on the most important underlying patterns, enabling faster training times and lower memory consumption, while still maintaining a high level of accuracy.

Based on the observations from the previous sections, where XGBoost was identified as the best-performing classifier, we extended our analysis to evaluate its performance from a different perspective. Rather than focusing solely on accuracy, we examined additional metrics such as time taken to train the model and memory usage. These factors are critical in practical applications, especially for real-time or resource-constrained environments. To further explore potential optimizations, we applied Principal Component Analysis (PCA) to investigate its impact on these performance metrics. PCA is a dimensionality reduction technique that transforms the data into a lower-dimensional space while retaining as much variance as possible [13]. This approach helps in reducing computational complexity and memory requirements without significantly affecting the model's predictive power.

b) ***Implementation of PCA****:* This section delves into the impact of applying PCA on model performance, focusing on the relationship between variance threshold, accuracy, memory usage, and training time [12]. We used the built-in PCA library in Python to reduce the dimensionality of the dataset, retaining varying proportions of the dataset's variance. A variance threshold specifies the amount of variance to retain, with higher thresholds preserving more information at the cost of computational complexity. To evaluate the trade-offs, we experimented with four different variance thresholds: 0.900, 0.950, 0.990, and 0.999, and analyzed their effects on three key metrics: Accuracy (%): The model's predictive performance. Memory Usage (GB): The memory consumed during training. Time Taken to Train Model (s): The computational efficiency of the model. The model used was the XGBClassifier.

Results and Observations:

The results, summarized in Table I, reveal that increasing the variance threshold slightly improves accuracy. However, this improvement comes at the cost of significantly higher training time, while memory usage remains relatively stable at higher thresholds (0.990 and 0.999). The model exhibits faster training times for thresholds 0.900 and 0.950.

To provide a clearer understanding, Fig. 16 visually illustrates the relationship between variance thresholds and the three performance metrics: Accuracy, Memory Usage, and Time Taken to Train. Below is an analysis of the trends observed in the graph:

1.Accuracy:

- Represented by the blue line, accuracy shows a gradual increase as the variance threshold rises.

- A noticeable improvement is observed from 0.90 to 0.99. However, beyond 0.99, the gains diminish, highlighting a point of diminishing returns.

2.Memory Usage:

- Depicted by the orange line, memory usage remains stable across higher thresholds (0.99 and 0.999).

- At lower thresholds (0.90 and 0.95), there is a minor reduction, but the effect on overall resource efficiency is minimal, suggesting that memory usage is not significantly impacted by variance thresholds beyond a certain point.

3.Time Taken to Train:

- The green line indicates a steep rise in training time as the threshold increases.

- Notably, the jump between thresholds 0.99 and 0.999 is substantial, demonstrating the trade-off between achieving marginally better accuracy and the increased computational cost.

The significant rise in training time between 0.99 and 0.999 emphasizes the inefficiency of extremely high variance thresholds. Since the differences in accuracy between 0.99 and 0.999 are minimal, and the training time for 0.99 is significantly lower (235 seconds compared to 272 seconds for 0.999), choosing 0.99 as the optimal variance threshold offers a practical balance. This threshold effectively retains sufficient

Variance Threshold	Accuracy (%)	Memory_Usage (GB)	Time Taken to Train Model(s)
0.900	99.745191	3.160116	160.895469
0.950	99.754102	3.026665	190.082128
0.990	99.806783	3.007600	235.115457
0.999	99.826480	3.007600	272.662835

TABLE I
PERFORMANCE METRICS FOR VARIANCE THRESHOLD

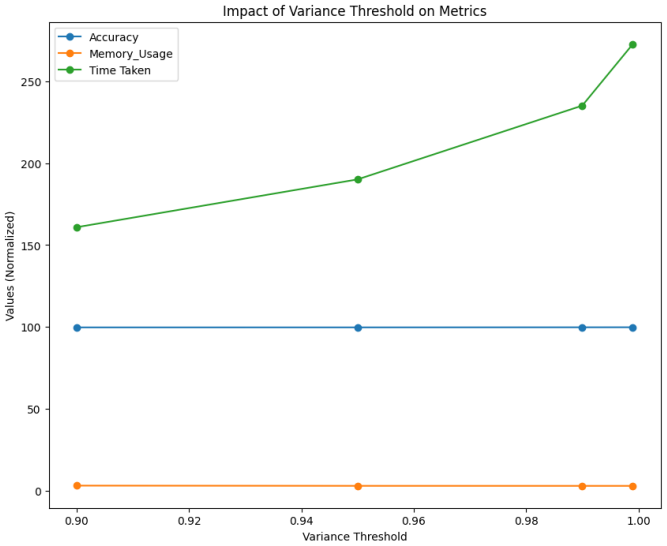

Fig. 16. Impact of Variance Threshold on Metrics

information while conserving computational resources, making it a more efficient choice for further analysis.

Based on the selected variance threshold (0.99) and the chosen model (XGBoost), PCA was applied to assess its impact on key performance metrics. The results, summarized in Table II and visually depicted in Fig. 17, compare the model's performance with and without PCA. The evaluated metrics include Accuracy, Memory Usage, and Time Taken for training.

To quantify the impact of PCA, a detailed percentage-based comparison is provided below, highlighting the trade-offs in terms of accuracy reduction, memory savings, and training time improvement. These insights help determine the practical benefits of applying PCA in the given context.

Observations and Calculations:

- **Accuracy:**

$$\text{Difference} = 99.989370 - 99.806158 = 0.183212\,(\%)$$

$$\text{Percentage Reduction} = \frac{0.183212}{99.989370} \times 100 \approx 0.18\,(\%)$$

- **Memory Usage:**

$$\text{Difference} = 24.395108 - 24.390344 = 0.004764\,(\text{MB})$$

Dataset	Accuracy(%)	Memory Usage (MB)	Time Taken(s)
Without PCA	99.989370	24.395108	290.131819
With PCA	99.806158	24.390344	210.937284

TABLE II
COMPARISON OF METRICS WITH PCA AND WITHOUT PCA

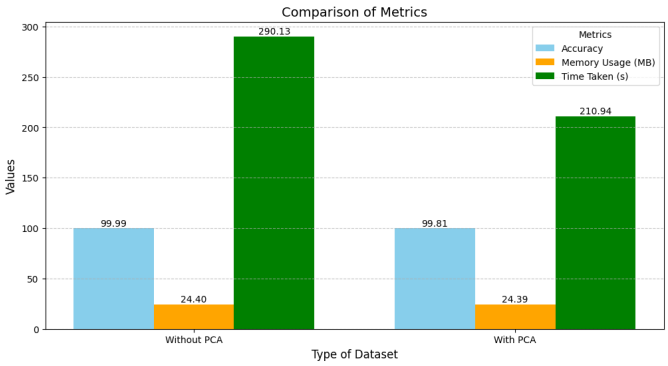

Fig. 17. Peformance Metrics Comparison with and without PCA

$$\text{Percentage Reduction} = \frac{0.004764}{24.395108} \times 100 \approx 0.02\,(\%)$$

- **Time Taken:**

$$\text{Difference} = 290.131819 - 210.937284 = 79.194535\,(\text{s})$$

$$\text{Percentage Reduction} = \frac{79.194535}{290.131819} \times 100 \approx 27.29\,(\%)$$

Analysis:

From the above comparison, it is evident that applying PCA with a variance threshold of 0.99 provides a significant reduction in training time while maintaining comparable accuracy and memory usage. The minor accuracy drop of 0.18% is an acceptable trade-off for the 27.29% reduction in training time, especially for scenarios where computational efficiency is prioritized.

This comparison and the Fig. 17 highlights the practicality of incorporating PCA in the preprocessing pipeline, particularly for large datasets where reducing training time can lead to faster iterations and model optimization.

D. Real Time Data Simulation in Power BI

a) Models and Tools Overview: Before diving into the implementation, the following models and tools were finalized for real-time data simulation and prediction:

- **PCA Instance:**Used for dimensionality reduction in the PCA-based approach.Saved and loaded using the joblib package.
- **XGB Model:**Two models were used: 1. XGB Model trained with PCA
  2. XGB Model trained without PCA.
  The built-in save and load functions of XGBoost were used for these models to preserve model weights and configurations.
- **Standard Scaler:**A separate scaler instance was trained for PCA-transformed data and the original dataset.Saved and loaded using the joblib package.
- **Label Encoder:**Common for both approaches to encode the predicted labels into human-readable or categorical formats. Saved and loaded using the joblib package.

This combination of tools and models ensures consistency, efficiency, and accuracy throughout the workflow as shown in Fig. 18.

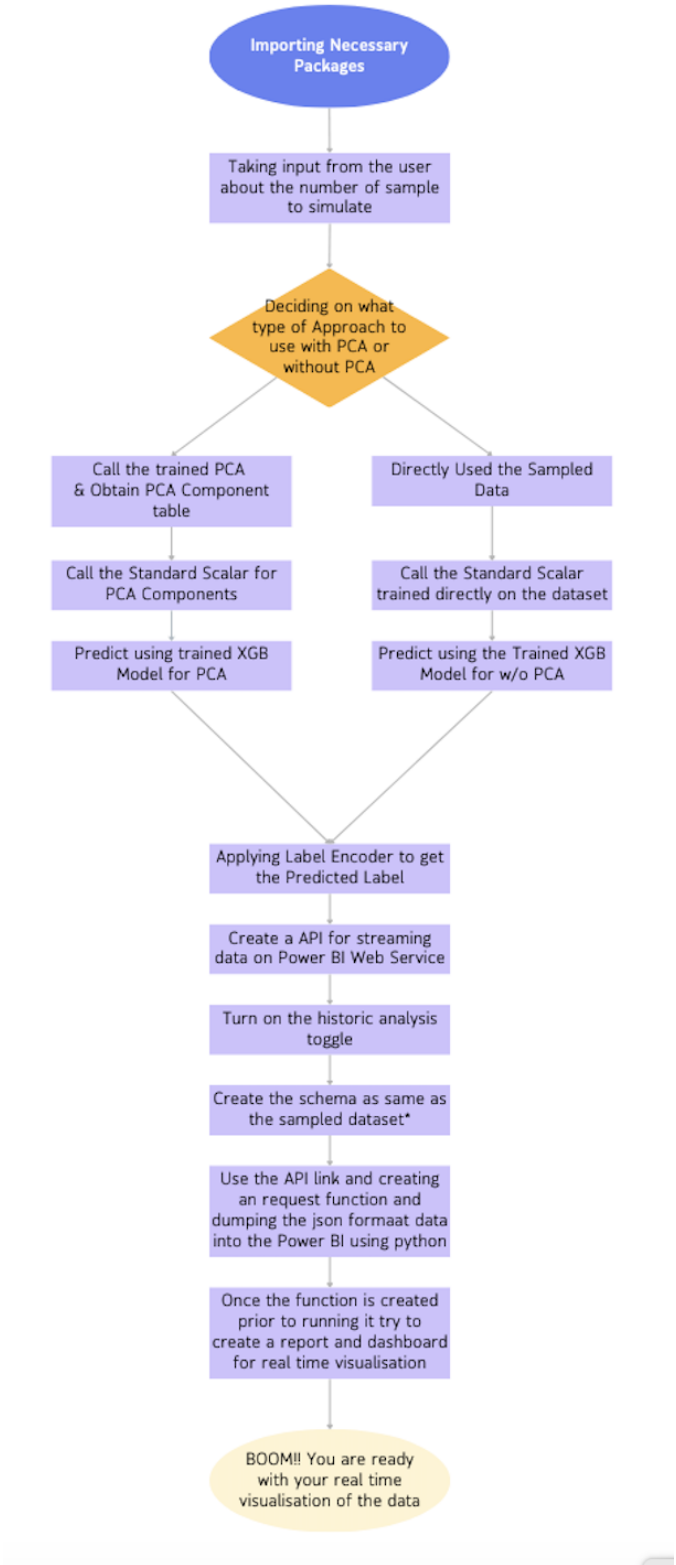

Fig. 18. Flowchart of the Implementation

b) Real-Time Data Simulation and Visualization Flowchart:

1. Importing Necessary Packages

Start by importing essential Python libraries required for preprocessing, scaling, encoding, modeling, and integration with Power BI.

2. Taking User Input

Collect input from the user to determine the number of samples to simulate for real-time prediction and visualization.

3. Deciding on the Approach

Based on the requirements, select one of the following approaches:

- With PCA: Use the pre-trained PCA model to transform the dataset into its principal components.Apply the Standard Scaler instance trained specifically for PCA-transformed data to ensure consistent scaling.Use the XGB Model trained with PCA to predict the outcome.
- Without PCA: Directly scale the sampled data using the Standard Scaler instance trained on the original dataset.Use the XGB Model trained without PCA to predict the outcome.

4. Applying Label Encoder

Use the Label Encoder instance to convert the predicted labels into human-readable or categorical outputs as required for Power BI visualization.

5. Streaming Data to Power BI

Step 1: Enable the Power BI service with the appropriate workspace and dataset for streaming.

Step 2: Turn on the ”Historic Data Analysis” toggle for the Power BI dataset to allow visualization of both past and real-time data.

Step 3: Create the schema for the streaming dataset, ensuring it matches the structure of the predicted data.

6. Building Reports and Dashboards

Create a report in Power BI that connects to the streaming dataset. Design visualizations to represent predictions, trends, and insights from the data. Publish a dashboard linked to the report for live updates.

7. Sending Data to Power BI Using Python

Use the Power BI API endpoint and Python’s requests library to send real-time simulated data (in JSON format) into Power BI. Ensure proper authentication and test the data pipeline to verify successful ingestion of data into Power BI.

IV. Data Visualization

Based on Fig. 19, we observe that the average flow duration was highest on 5th July 2017 and lowest on 6th July 2017. This could indicate that the experiments conducted on 5th July generated longer-lasting flows, potentially focusing on sustained traffic patterns. Conversely, the experiments on 6th July may have emphasized shorter-duration flows, or fewer experiments might have been performed that day.

The box plot in Fig. 20 visualizes the flow duration across various attack types. It shows that benign traffic, represented by the orange box, has the shortest flow duration, suggesting regular traffic has brief interactions. Attack types like FTP-Patator and SSH-Patator (yellow and green boxes) exhibit

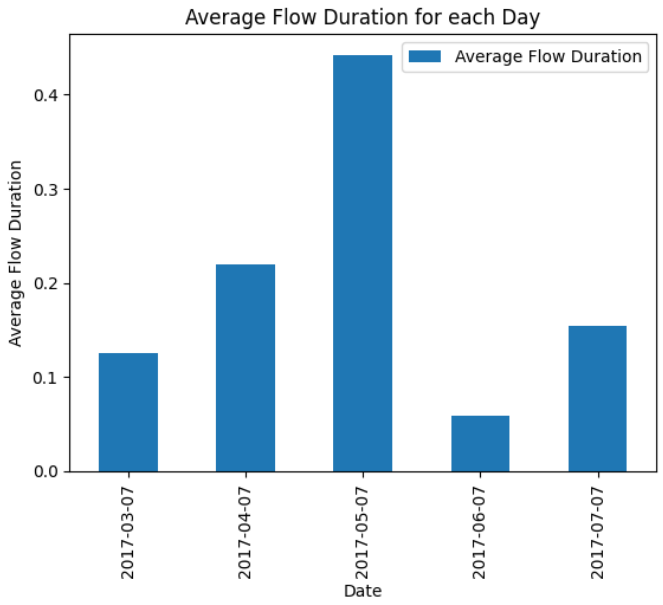

Fig. 19. Average Flow Duration for Each Day

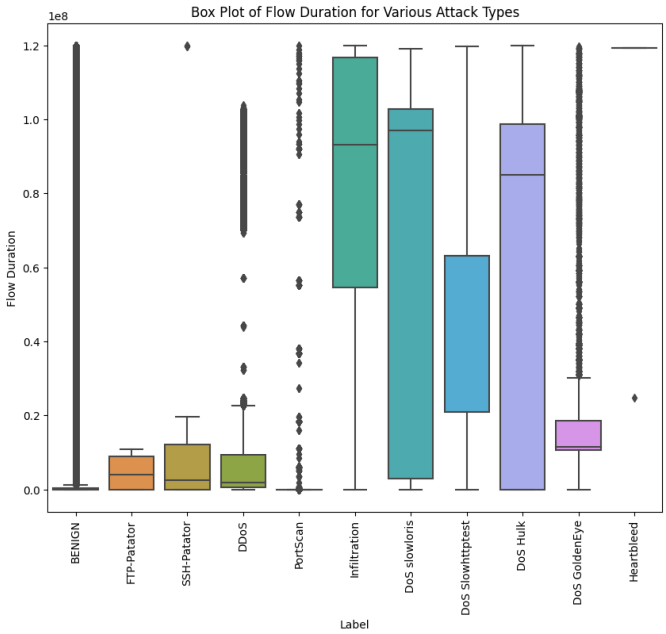

Fig. 20. Box Plot of Flow Duration for Various Attack Types

similarly short flow durations, reflecting the repetitive nature of these attacks, which involve quick, repeated login attempts. In contrast, attacks such as DDoS and PortScan (light blue and green boxes) demonstrate more variability in flow duration, indicating differing attack behaviors. More persistent attacks like Infiltration and DoS Slowloris (teal and blue boxes) show higher median flow durations, with wider interquartile ranges, suggesting longer attack durations. DoS Slowhttptest and DoS Hulk (light purple and purple boxes) exhibit particularly high median flow durations, indicating prolonged, intensive attacks. DoS GoldenEye and Heartbleed (purple and pink boxes) have the highest flow durations with significant outliers, reflecting highly persistent attacks. These outliers, representing extreme flow durations, are normalized during preprocessing to ensure that the model is trained on the full spectrum of values, preventing the outliers from disproportionately affecting the model's performance. This normalization helps in training the model with all possible values, allowing it to generalize better across different attack scenarios and network conditions.

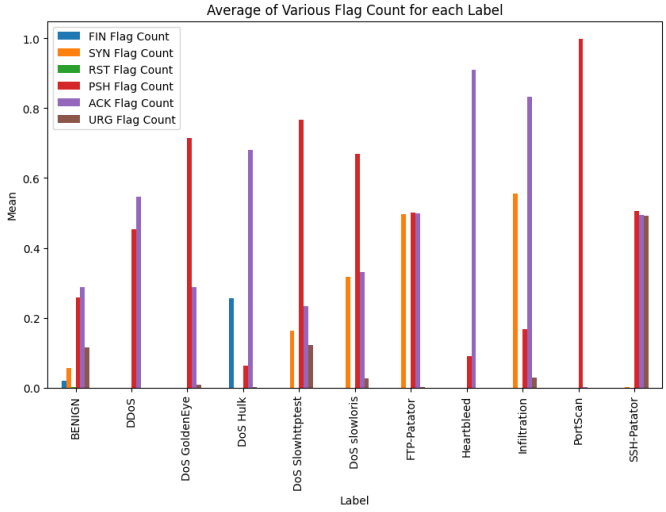

Fig. 21. Average of Various Flag Count for each Label

The bar chart in Fig. 21 displays the average count of various flags (FIN, SYN, RST, PSH, ACK, and URG) for each attack type. The BENIGN label shows relatively low values across all flags, indicating normal traffic with fewer network control signals. In contrast, attack types like DDoS, DoS GoldenEye, and DoS Hulk display significantly higher counts, especially for the URG and PSH flags, suggesting more intensive use of these flags in these attacks. Specifically, DoS Slowhttptest and DoS Slowloris exhibit high SYN and ACK flag counts, reflecting the slow and persistent nature of these attacks. FTP-Patator and SSH-Patator show relatively higher counts for the SYN and RST flags, which are typical for attempts to establish or reset connections. Heartbleed, Infiltration, PortScan, and SSH-Patator also have varying flag counts, but not as pronounced as the aforementioned attacks. The graph highlights how different attack types manipulate these flags differently to create malicious traffic, and this behavior helps in distinguishing between attack types and benign traffic.

From the correlation matrix in Fig. 22, we can observe several key relationships between variables in the dataset. The Total Forward Packets and Total Backward Packets have a perfect positive correlation of 1, indicating they increase or decrease together. There is a negative correlation between both Total Forward Packets and Flow Packets/s, as well as between Total Backward Packets and Flow Packets/s, suggesting that as the total number of forward or backward packets increases, the flow packets per second tends to decrease. The correlation between Flow Bytes/s and Flow Packets/s is modest, around 0.25, indicating a weak positive relationship, while the correlation between Total Forward Packets/Total Backward

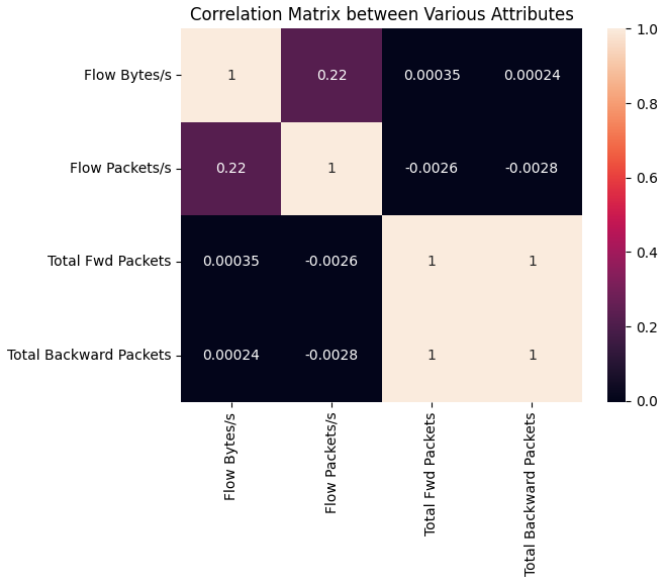

Fig. 22. Correlation Matrix between Attributes (such as Flow Bytes/s, Total Forward and Backward Packets,Flow Packets/s)

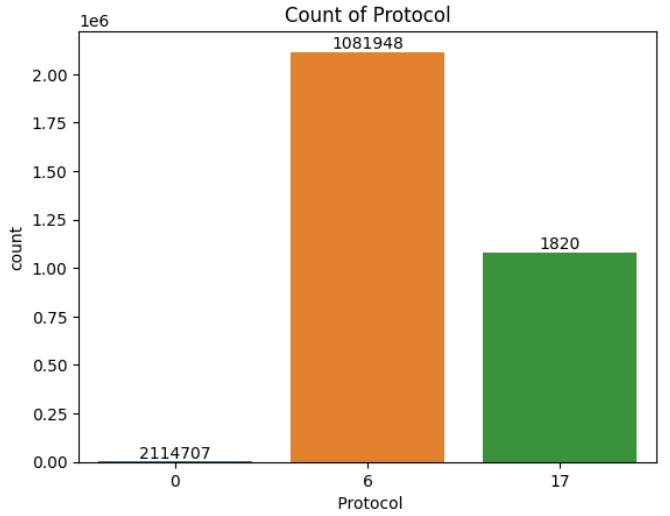

Fig. 23. Count of Protocol

Packets and Flow Bytes/s is close to zero, showing almost no correlation. This suggests that the total number of packets does not significantly affect the flow of bytes per second, likely due to other factors, such as packet size, influencing the flow of bytes more than the number of packets. These observations offer valuable insights into the relationships between different network traffic attributes.

From the Fig. 23 we can infer that the Protocol 6 (likely TCP) dominates the dataset, reflecting its widespread use in network traffic. This is consistent with TCP's role in ensuring reliable communication for many common applications such as web browsing, file transfers, and email. Protocol 17 (likely UDP) is the second most frequent protocol in the dataset. UDP is typically used in applications where low latency and speed are prioritized over reliability, such as video streaming, gaming, and VoIP.In contrast, Protocol 0 is rarely observed in

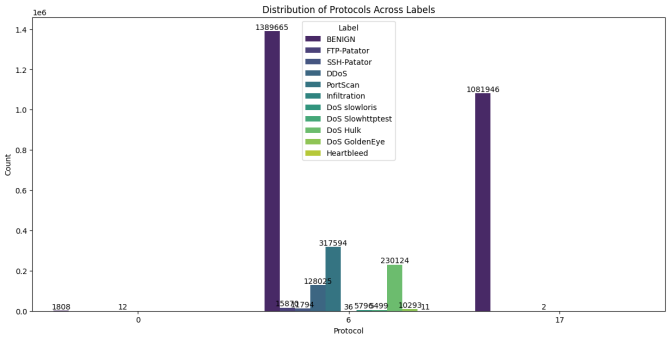

Fig. 24. Count of Label across various Protocol

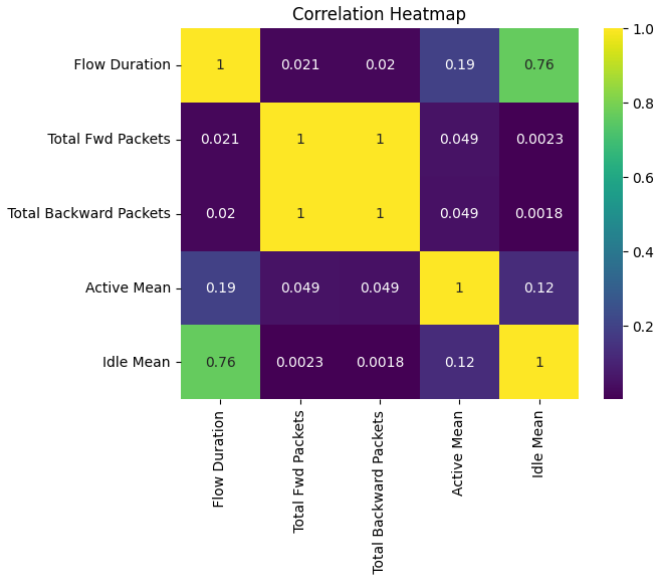

Fig. 25. Correlation Heatmap between attributes(such as Flow Duration, Total Fwd and Backward Packets, Active Mean, Idle Mean)

the dataset, suggesting it corresponds to specific or uncommon network events. Its infrequent use may indicate specialized traffic or non-standard network communications, which are not typically seen in regular network activity.

From the Fig. 24 we can infer that the "BENIGN" label predominantly utilizes Protocol 6, with significantly fewer instances of other protocols such as Protocol 17 or Protocol 0. This indicates that benign network traffic primarily consists of communication using TCP, while other protocols are less frequently observed in normal traffic.On the other hand, attack types like "PortScan" and "DoS Hulk" exhibit a more diverse distribution across protocols. This suggests that certain attack types are more closely associated with specific protocol usage patterns. For instance, PortScan may involve multiple protocols to probe various network services, while DoS Hulk could show a stronger correlation with a particular protocol, reflecting the nature of the attack and its impact on the network.

From the Fig. 25 we can infer that Flow Duration is positively correlated with Active Mean and Idle Mean, indicating longer flows have higher active or idle times. There's minimal correlation between Total Backward Packets and

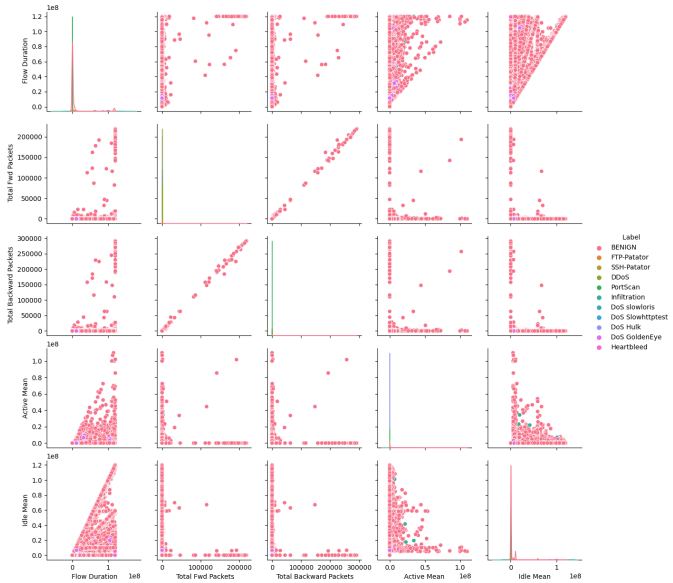

Fig. 26. Pair Plot on Attributes such as Flow Duration, Total Fwd Packets, Total Bwd Packets, Active and Idle Mean

other features, suggesting independent behavior.

Based on the Fig. 26 we can infer that the relationships between features like Flow Duration, Total Fwd Packets, and Total Backward Packets reveal distinct clusters, potentially aiding in label classification. Attack labels such as "DoS Hulk" or "Port Scan" tend to show dispersed values, distinguishing them from BENIGN traffic, which is more clustered. Feature Relationships: The diagonal histograms reveal that features like Flow Duration and Active Mean are heavily skewed, showing a concentration of values near the lower end .Label Separation: Certain attack types, such as "DoS Hulk," are distinguishable in specific feature pairings (e.g., Flow Duration vs. Total Fwd Packets), showing potential for effective classification.

V. Future Work

1. Advanced Dimensionality Reduction Techniques

Explore other dimensionality reduction methods like t-SNE, UMAP, or LDA to compare their effectiveness with PCA for both visualization and classification performance. Investigate nonlinear PCA or kernel-based techniques for datasets with complex feature interactions.

2. Model Optimization

Conduct hyperparameter tuning using techniques like grid search or Bayesian optimization to further improve the performance of the XGBoost model. Experiment with ensemble techniques like Stacking or Boosting by combining PCA and non-PCA models for better accuracy.

3. Scalability Analysis

Test the model's performance on larger datasets to evaluate the scalability of XGBoost with and without PCA in terms of computation time and memory usage. Integrate distributed frameworks such as Apache Spark or Dask to handle datasets beyond current hardware limitations.

4. Real-Time Analysis

Implement the model in a real-time streaming environment (e.g., using Apache Kafka or Flink) to analyze live network traffic data for intrusion detection. Develop a real-time dashboard to monitor predictions and visualize critical metrics dynamically.

5. Comparison with Other Models

Evaluate the performance of deep learning models (e.g., Autoencoders, LSTMs) for dimensionality reduction and classification. Compare tree-based models with other classifiers, such as SVM, Neural Networks, or k-NN, to determine the best approach for specific scenarios.

6. Improved Visualization Techniques

Develop more interactive visualizations (e.g., in Power BI or Tableau) to better demonstrate the impact of PCA on different metrics. Utilize 3D or multidimensional plots to visualize relationships between PCA components and classification outcomes.

VI. Paper Highlights

- **Superior Performance of XGBoost:** Among the five machine learning models evaluated, XGBoost emerged as the most accurate and effective in detecting cyber attacks, demonstrating its robustness in handling complex patterns and feature interactions.
- **Feature Importance Analysis:** Random Forest tended to assign importance to a wide range of features, potentially leading to overfitting. In contrast, XGBoost and Decision Tree focused on a smaller, more critical subset of features, enhancing their interpretability and efficiency. Logistic Regression, while offering interpretable coefficients, struggled to clearly identify distinct features due to its linear assumptions, making it less suitable for the dataset's complexities.
- **Impact of PCA on Model Efficiency:** Implementing Principal Component Analysis (PCA) led to a negligible reduction in accuracy but significantly improved computational efficiency and reduced training time. This trade-off highlights the practicality of dimensionality reduction techniques in handling large datasets without compromising performance.
- **Insights into Attack Detection:** The analysis revealed that different types of cyber attacks manipulate network flags in distinct ways. This behavior provides a key indicator for distinguishing between malicious and benign traffic, underscoring the importance of feature engineering and domain-specific knowledge in cyber attack detection.

VII. Conclusion

This paper highlights the importance of data visualization and preprocessing in enhancing network traffic analysis for cybersecurity. Our team used the CICIDS 2017 dataset to process, analyze, and visualize network traffic in order to visual patterns in network traffic and gain better insights into cybersecurity threats. Additionally, data cleaning, transformation, and feature selection were used to improve model performance. In particular, Principal Component Analysis (PCA) was used

to reduce the dimensionality of the data and increase computational efficiency without compromising accuracy. Among the five models, XGBoost, Random Forest, Decision Tree, Logistic Regression, and Gaussian Naive Bayes, that were evaluated in this report, XGBoost was found to have the highest accuracy at detecting intrusions. Using Python and PowerBI, we also created an interactive platform for real-time data simulation that analyzes attack types, feature importance, and performance metrics. Our visualizations highlighted how different types of cyberattacks manipulated network flags and that this distinction can be used to detect malicious traffic. Our findings emphasize the importance of data analytics and visualization when improving cybersecurity infrastructure and intrusion detection.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to San José State University (SJSU) for providing the opportunity and resources to work on this project. We would like to thank Dr. Shih Yu Chang for his guidance on the project, as well as Sourab Saklecha for providing resources in deepening our understanding of data visualizations. We would also like to acknowledge Canadian Institute of Cybersecurity for their open-access database that helped make this project possible.

REFERENCES

[1] M. A. Almaiah et al., ”Performance Investigation of Principal Component Analysis for Intrusion Detection System Using Different Support Vector Machine Kernels,” Electronics, vol. 11, no. 21, p. 3571, 2022.

[2] C. N. Adams and D. H. Snider, ”Effective Data Visualization in Cybersecurity,” in 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018, pp. 1-8.

[3] L. Breiman, ”Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.

[4] Canadian Institute for Cybersecurity, ”CICIDS 2017 Dataset,” Available: https://www.unb.ca/cic/datasets/ids-2017.htmlhttps://www.unb.ca/cic/datasets/ids-2017.html

[5] T. Chen and C. Guestrin, ”XGBoost: A Scalable Tree Boosting System,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794.

[6] R. Genuer, J. M. Poggi, and C. Tuleau-Malot, ”Variable selection using random forests,” Pattern Recognition Letters, vol. 31, no. 14, pp. 2225-2236, 2010.

[7] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, Applied Logistic Regression, 3rd ed. New York: John Wiley & Sons, 2013.

[8] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, ”Understanding variable importances in forests of randomized trees,” Advances in Neural Information Processing Systems, vol. 26, 2013.

[9] K. P. Murphy, ”Naive Bayes classifiers,” University of British Columbia, vol. 18, 2006.

[10] O. Osho and S. Hong, ”Network Intrusion Detection System Using Principal Component Analysis Algorithm and Decision Tree Classifier,” in 2021 International Conference on Computational Science and Computational Intelligence (CSCI), 2021, pp. 273-279.

[11] L. Rokach and O. Maimon, ”Decision Trees,” in Data Mining and Knowledge Discovery Handbook, Springer, 2010, pp. 165-192.

[12] I. Ullah, K. Mengersen, R. J. Hyndman, and J. McGree, ”Detection of cybersecurity attacks through analysis of web browsing activities using principal component analysis,” arXiv preprint arXiv:2107.12592, 2021.

[13] W. Wang and R. Battiti, ”Identifying Intrusions in Computer Networks with Principal Component Analysis,” in First International Conference on Availability, Reliability and Security (ARES’06), 2006, pp. 270-279.