APPROACH DOCUMENTATION

Problem Statement: Real-Time Analysis and Detection of Security Incidents from Firewall Logs

Prepared By:
Nilesh Borana

Contact Details:
Nileshborana96@gmail.com

9783749891

Date: 27-Dec-2024

Contents

4. Machine Learning Implementation :

- Feature Selection :
- Approach 1 Binary Classification :
- Approach 2 Binary Classification with Class Imbalance Handling:
- Approach 3 Multi-Class Classification :

5. Reporting Dashboard:

Assignment Objective:

To utilize data analysis and machine learning techniques for the comprehensive examination of firewall logs, aiming to identify potential security incidents, including errors, threats, and suspicious activities in real time.

Key Components of the Solution:

1. **Exploratory Data Analysis (EDA):**
    - Perform detailed EDA to uncover insights from firewall logs.
    - Visualize significant findings with relevant explanations.
2. **Threat and Anomaly Detection System:**
    - Develop a machine learning model to detect threats, intrusion attempts, malware communications, and unusual activities.
    - Provide a system accuracy metric and detail the steps to improve its performance.
3. **Reporting Dashboard:**
    - Create a dashboard summarizing incidents across different timeframes (hourly, 12 hours, and 24 hours).
    - Showcase errors, threats, and unusual activities for better incident management and actionable insights.

Dataset:
Primary: Internet Firewall Data Set from Kaggle
Alternative: Relevant datasets sourced online or synthetic datasets aligned with the analysis objectives.

Output Expectation

In-Depth Exploratory Data Analysis (EDA)

To uncover significant patterns, anomalies, and insights in firewall logs, enabling a better understanding of the data and identifying potential areas for threat detection.

Tools and Environment:

- **Platform:** Google Colab **Command to Run Dashboard:** streamlit run dashboard.py
- **Libraries Used:** Pandas, NumPy, Matplotlib, Seaborn, Datetime, Scikit-learn (IsolationForest, train_test_split, RandomForestClassifier, classification_report, confusion_matrix, accuracy_score, roc_auc_score, roc_curve, compute_class_weight, LabelEncoder), Streamlit.

Data Loading:

Objective:

To load and inspect the dataset for preliminary understanding, including column details and overall structure.

Steps Taken:

1. The dataset was imported into Google Colab from Kaggle using direct download links and loaded into a Pandas DataFrame.
2. Inspected the data using functions like `.head()`, `.info()`, and `.describe()` to gain an initial overview.

Data Set Information:

- The dataset contains **12 features** in total.
- The **Action** feature serves as the target class, with **4 unique classes:**
  - **Allow**
  - **Action**
  - **Drop**
  - **Reset-both**

Attribute Information:

- **Source Port:** Source port of the network connection.

- **Destination Port:** Destination port of the network connection.
- **NAT Source Port:** Source port after network address translation (NAT).
- **NAT Destination Port:** Destination port after NAT.
- **Action:** Indicates the result of the connection (allow, drop, etc.).
- **Bytes:** Total number of bytes transferred during the connection.
- **Bytes Sent:** Number of bytes sent by the source.
- **Bytes Received:** Number of bytes received by the destination.
- **Packets:** Total number of packets transferred during the connection.
- **Elapsed Time (sec):** Total duration of the connection in seconds.
- **pkts_sent:** Number of packets sent by the source.
- **pkts_received:** Number of packets received by the destination.

Data Cleaning:

- Missing values were identified and handled appropriately (e.g., imputation or removal).
- Data types were checked and corrected for proper analysis.

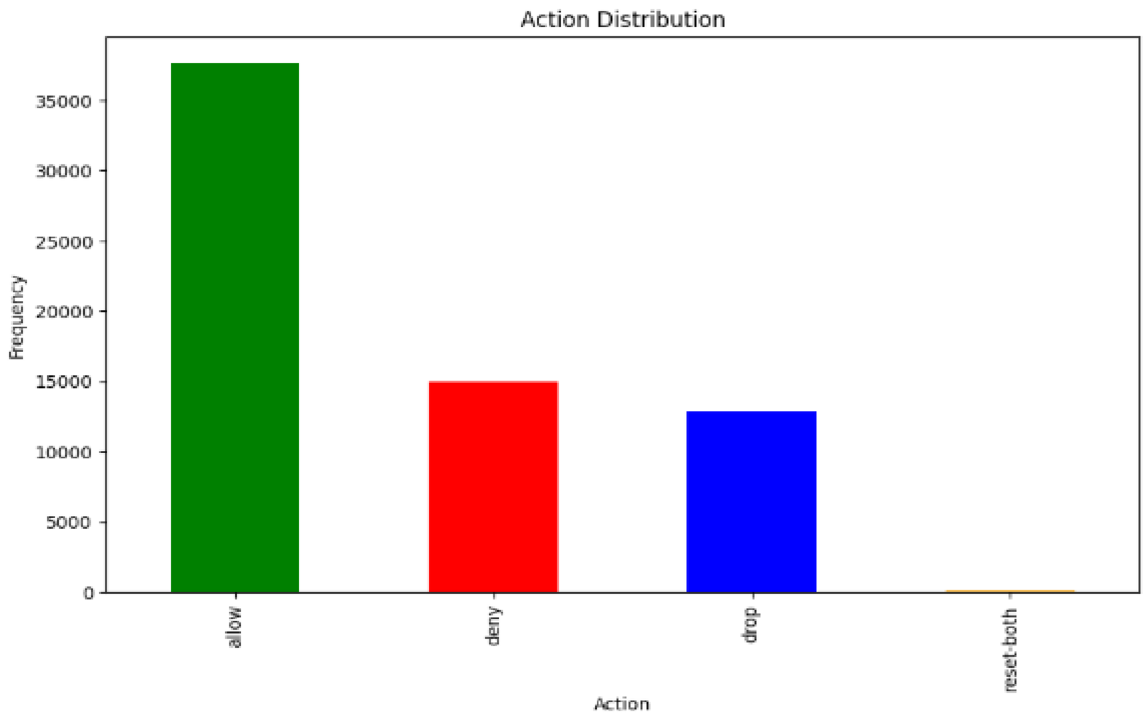
Preliminary Observations:

- The dataset appears clean, with no immediate irregularities in column naming or structure.

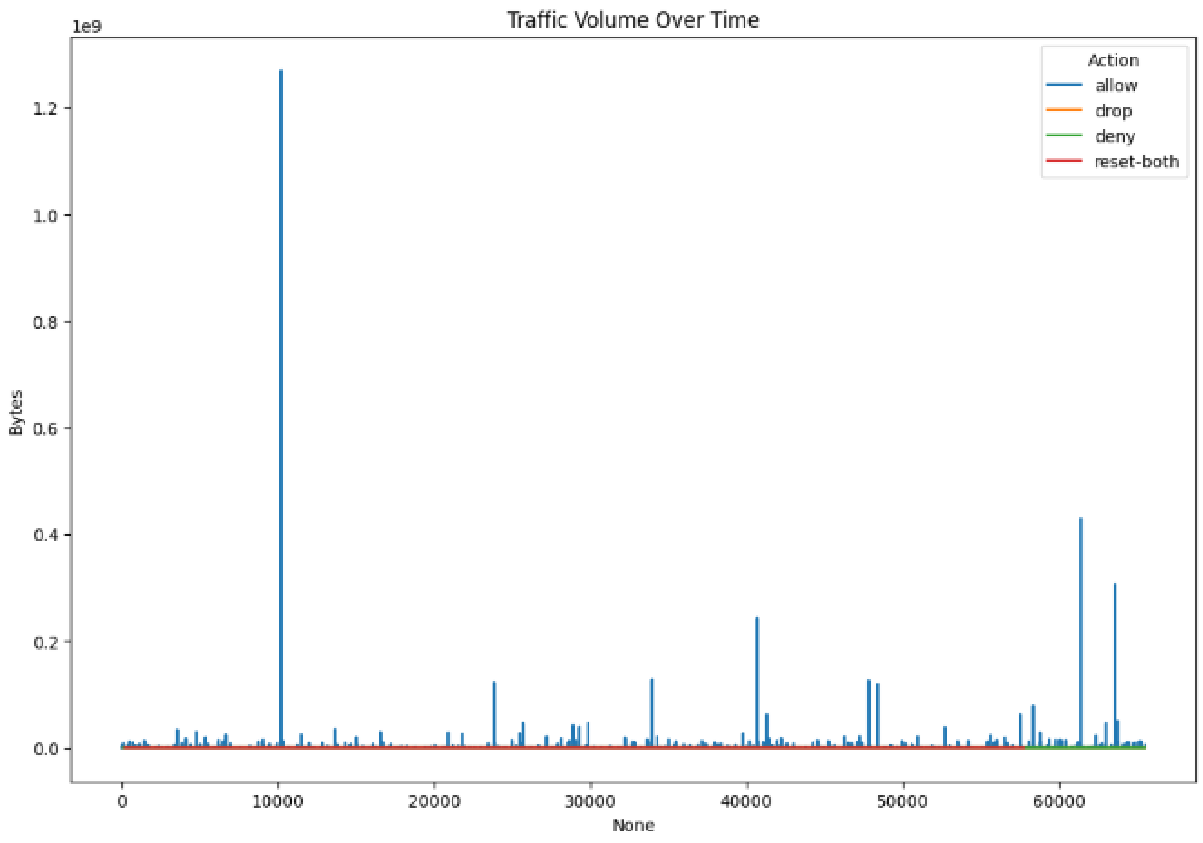
Visualization :
To visually explore the dataset and understand key patterns, trends, and distributions before applying machine learning models.

Graph:

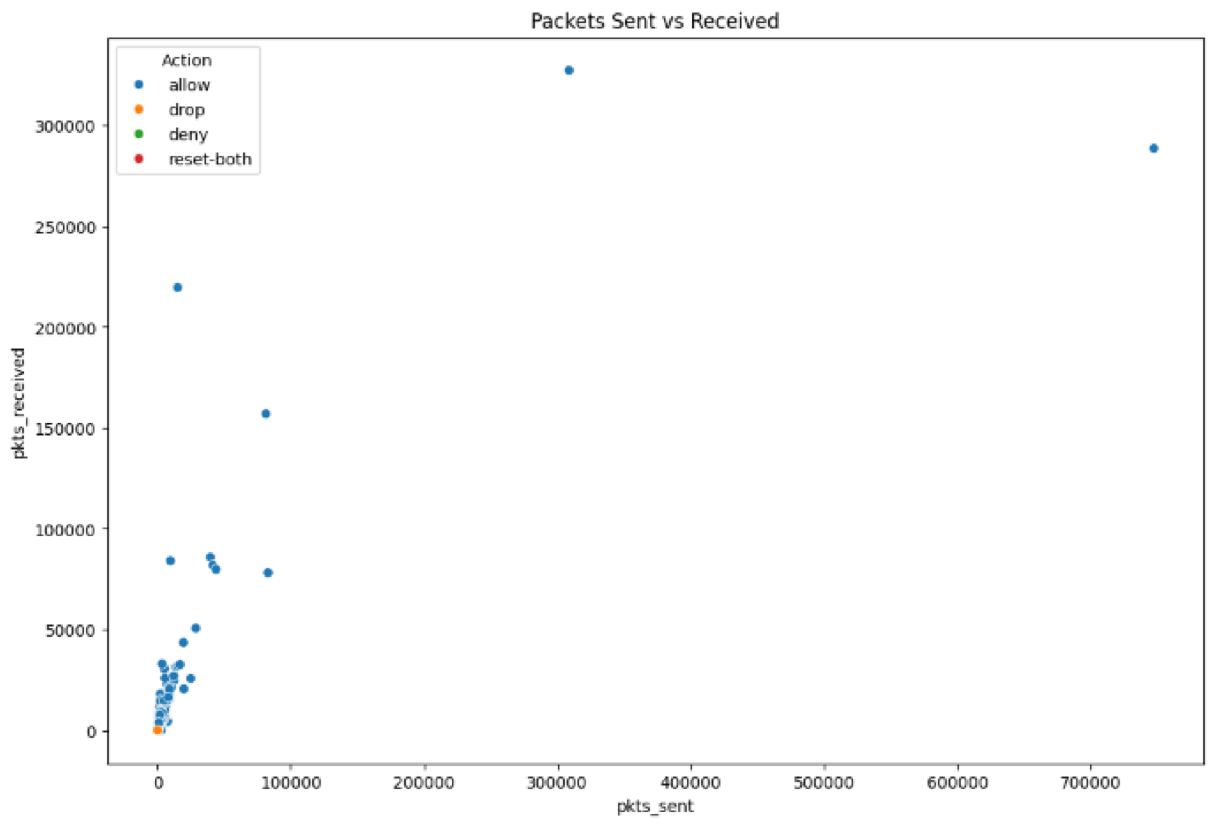
1. **Action Distribution:**
    - Created a graph to visualize the distribution of actions (e.g., "allow," "deny," "drop," or "reset-both") within the dataset.
    - This helped identify the frequency of each action, providing insights into the dataset's class balance.



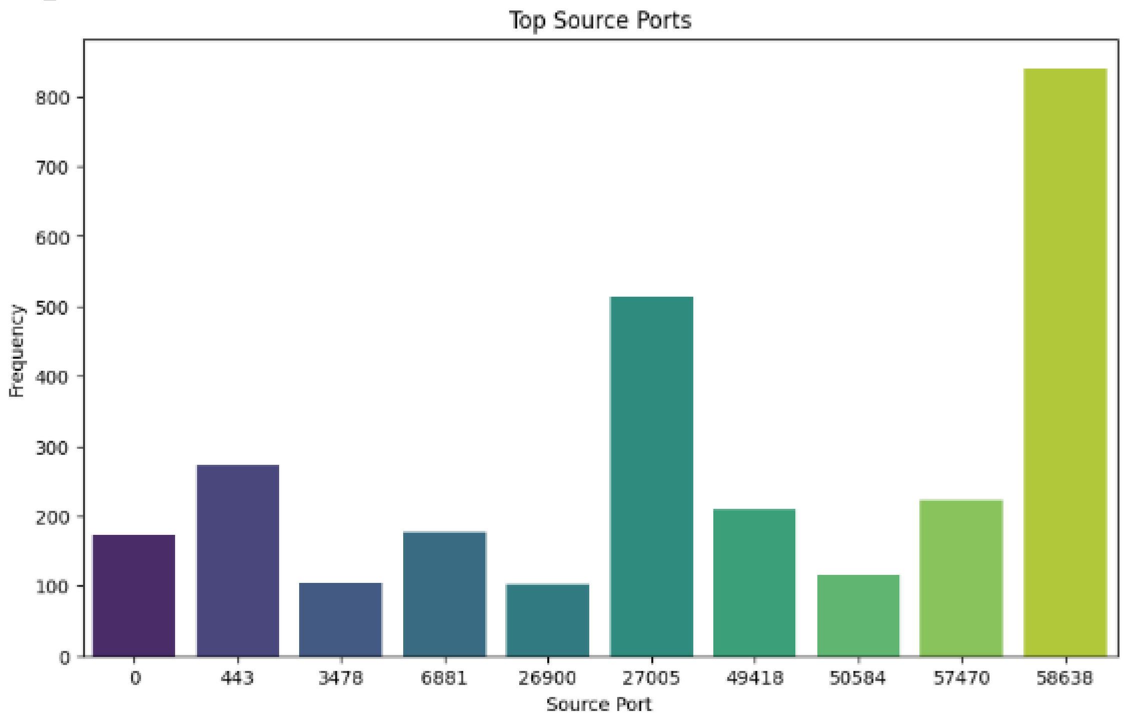
2. **Traffic Volume Over Time:**
   - Plotted traffic volume (in bytes) over time, categorized by action types.
   - This visualization revealed patterns in traffic trends, such as spikes or drops, and their association with specific actions.



3. **Packet Analysis:**
    - Generated a graph comparing packets sent and received, segmented by action type.
    - This visualization highlighted correlations and outliers in network activity, offering clues about anomalous behavior.



4. Top Source Ports :



Purpose:
These visualizations provided an initial understanding of the dataset, such as class distributions, traffic patterns, and potential anomalies. They served as a foundation for further analysis and model development.

Key Observations

1. **Action: "Allow"**
   - **Scenario 1:** Bytes sent and received are nearly equal, with `pkts_sent` and `pkts_received` also equal.
   - **Scenario 2:** Bytes sent and received differ, but `pkts_sent` and `pkts_received` remain close though not equal.
2. **Action: "Deny"**
   - **Scenario 1:** Bytes sent is non-zero, but bytes received is zero; `pkts_sent` is 1, and `pkts_received` is 0.
   - **Scenario 2:** Bytes sent and received differ, with `pkts_sent` and `pkts_received` being equal.
3. **Action: "Drop"**
   - Bytes sent is non-zero, but bytes received is zero; `pkts_sent` is 1, and `pkts_received` is 0.
4. **Action: "Reset-Both"**
   - **Scenario 1:** Bytes sent is non-zero, but bytes received is zero; `pkts_sent` is 1, and `pkts_received` is 0.
   - **Scenario 2:** Bytes sent and received differ, with `pkts_sent` greater than `pkts_received`.

Insights:

- These patterns provide a clear understanding of how network packets are transmitted under different actions.
- Scenarios like "Deny" and "Drop" with zero bytes received and minimal packets highlight potential anomalies or blocked connections.
- The differences in `pkts_sent` and `pkts_received` under "Reset-Both" suggest irregularities in connection resets, which could indicate suspicious activity.

These findings will guide feature engineering and model development for detecting anomalies and threats.

Feature Exploration

To create additional features and detect anomalies in the data to enhance the understanding of network behavior and improve model performance.

Traffic Over Time

This feature calculates and visualizes traffic data over time based on elapsed time in seconds since a predefined reference point. It converts the elapsed time into timestamps and then groups the traffic data by hour to plot the number of events over the course of a day.

Assumptions:

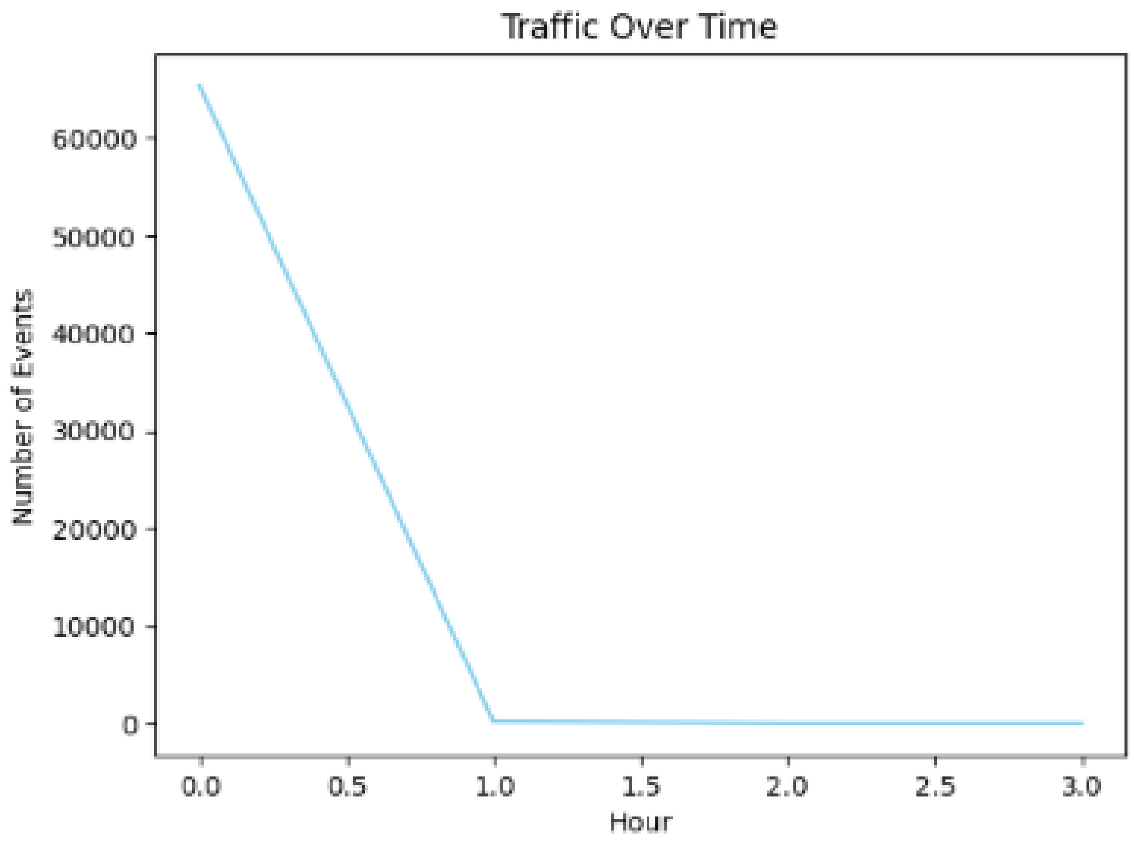
- The elapsed time is measured in seconds from a fixed reference point (e.g., Unix Epoch).
- The reference point is set to 2024-01-01 00:00:00 for this example, but should be adjusted according to the specific dataset context.

Steps:

1. **Define Reference Point:** Set a reference point for converting elapsed seconds into timestamps.

2. **Convert Elapsed Time:** Transform the elapsed time in seconds into actual timestamps using the predefined reference point.
3. **Analyze Hourly Traffic:**
   - Extract the hour from the timestamps.
   - Group the data by hour and count the number of events for each hour.
4. **Visualize Traffic Over Time:** Plot the hourly traffic data using a line graph to show the number of events occurring each hour.

Outcome: This feature helps in understanding traffic patterns by providing a clear visualization of events distributed over time.



Hours Summary

This feature provides a comprehensive summary of traffic data over specified time frames, along with insights into errors and traffic patterns. It includes calculations and visualizations to help understand the behavior and performance of the network over different periods.

Assumptions:

- The time_frame variable can take values like "Hourly", "12 Hours", or other defined intervals.
- Data fields such as Bytes, Packets, Elapsed Time (sec), and Action are available in the dataset.

Steps:

1. **Define Time Frame:** Set the time_frame variable to the desired interval (e.g., "Hourly" or "12 Hours") to group the traffic data accordingly.
2. **Calculate Summary Statistics:**
    - For hourly summaries, the data is grouped by each hour.
    - For 12-hour summaries, the data is grouped into two intervals: 0-12 hours and 12-24 hours.
    - For a full day (24 hours), total bytes and packets are summed up.
3. **Generate Error Insights:** Identify and count error events based on actions such as "drop" or "deny" to gain insights into network issues.

4. **Analyze Traffic Patterns:** Calculate the average packet size for sent and received packets to understand the data flow characteristics.
5. **Calculate Rates:**
    - **Traffic Rate (Bytes/sec):** Computes the data throughput by dividing the total bytes by the elapsed time.
    - **Packet Rate (Packets/sec):** Computes the packet throughput by dividing the total packets by the elapsed time.
6. **Encode Actions:** Map the action types to numerical values for easier analysis and visualization.

Unexpected Traffic Spikes

This feature detects and visualizes unexpected traffic spikes in the network data. It calculates the traffic rate and packet rate to identify outliers and anomalies, providing insights into unusual patterns that may indicate performance issues or security concerns.

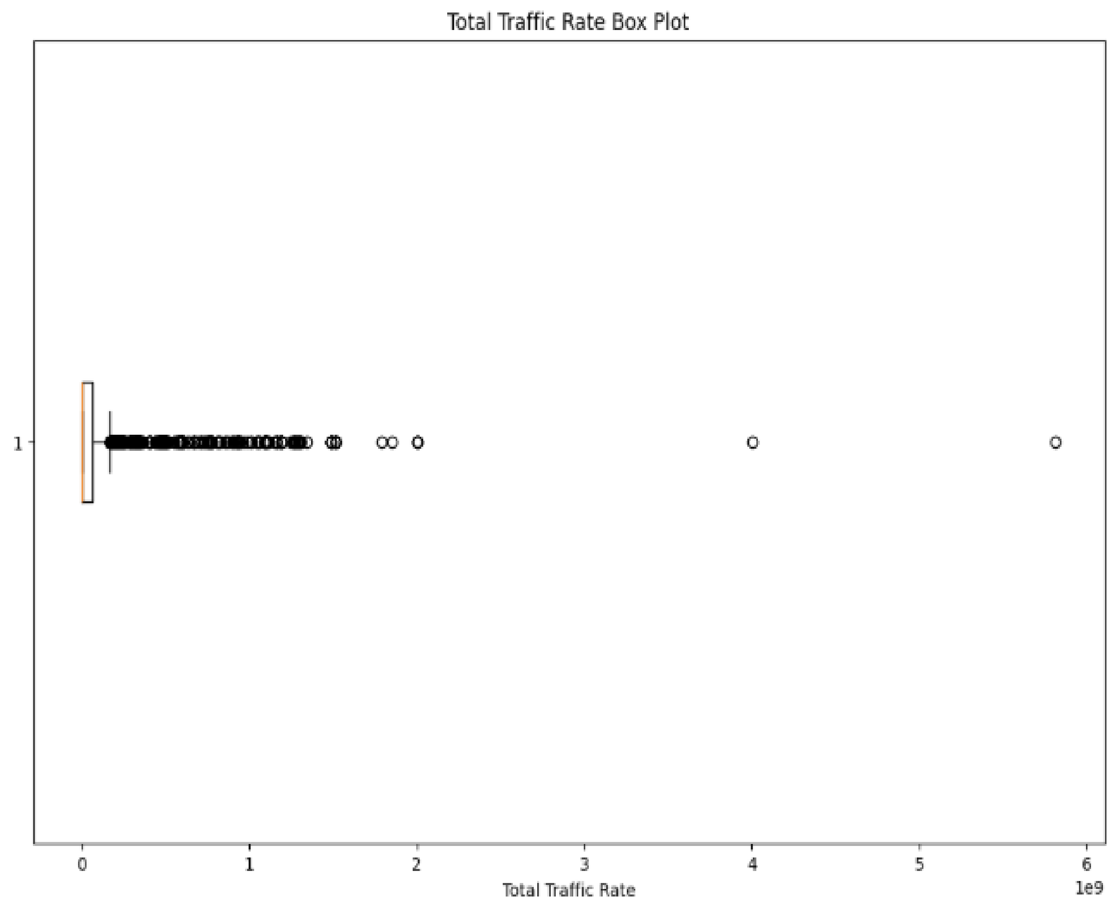
Assumptions:

- Data fields such as Traffic Rate and Packets are available in the dataset.

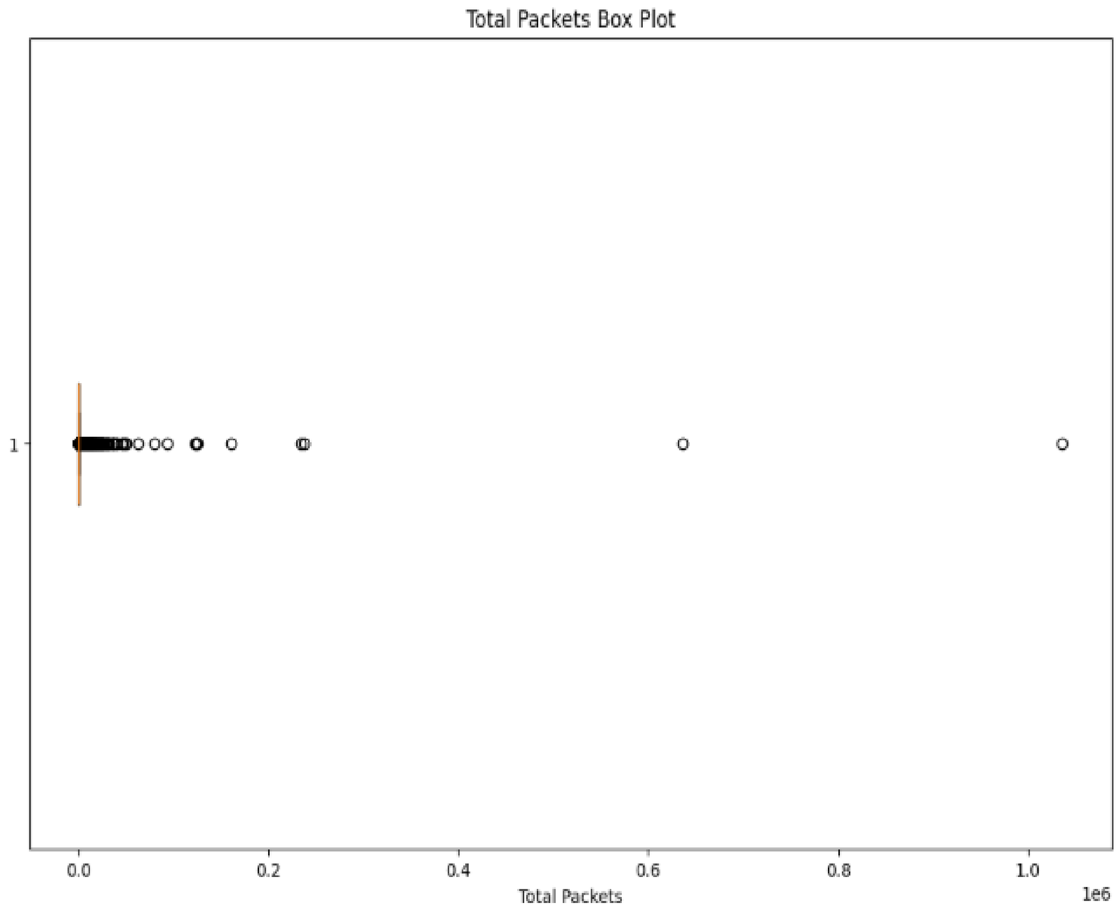
Steps:

1. **Calculate Traffic Statistics:**
   - Compute the mean and standard deviation of the traffic rate.
   - Identify traffic spikes by comparing the traffic rate against a threshold set at three standard deviations above the mean.
2. **Visualize Traffic Spikes:**
   - Generate a box plot of the traffic rate to visually identify outliers and spikes.
   - Provide a count of detected traffic spikes.
3. **Detect Packet Spikes:**
   - Define a threshold for packet spikes based on the mean and standard deviation of the total packets.
   - Identify rows in the dataset where the packet count exceeds this threshold.
4. **Visualize Packet Spikes:**

- Plot the total packets against the elapsed time (or any other time dimension) to visually identify spikes and anomalies.
- Create a box plot for the total packets to highlight outliers.



- Create a box plot for the total packets to highlight outliers.



Bytes Imbalance Analysis

- Bytes Imbalance: The difference between Bytes Sent and Bytes Received.
- Bytes Imbalance (%): The percentage difference between Bytes Sent and Bytes Received relative to the total bytes.

Thresholds for Anomalies:

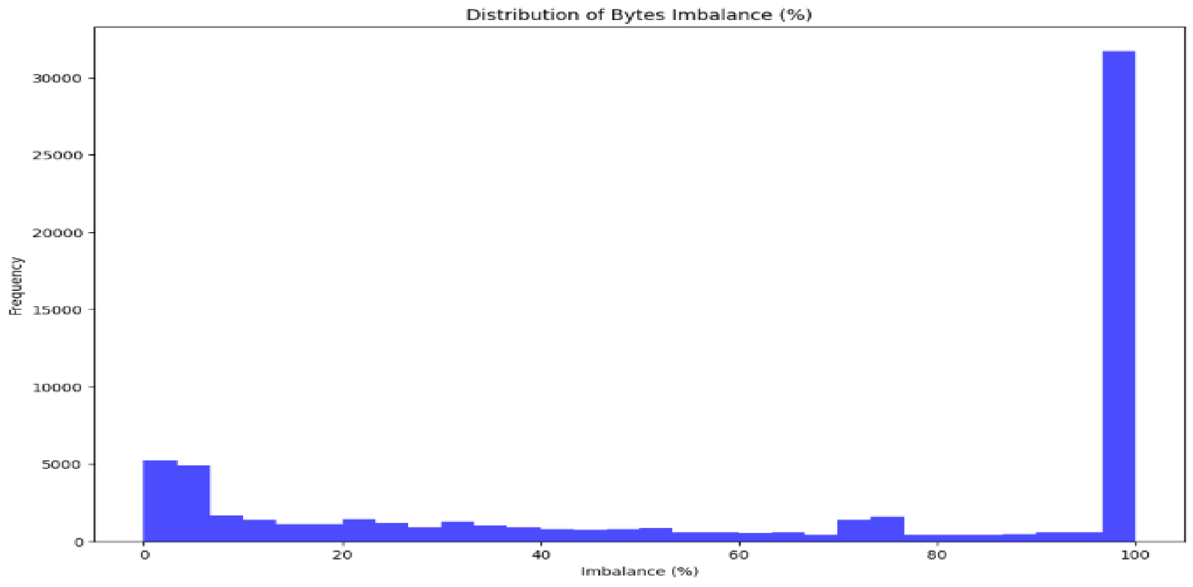
- Large discrepancies (>30%): Critical anomaly.
- Moderate discrepancies (10–30%): Warning.
- Small discrepancies (<10%): Acceptable.

Key Findings:

- Number of imbalanced rows: 53,719 (above a 10% threshold).

Visualization:

Distribution of Bytes Imbalance (%)



Packet Ratio Analysis

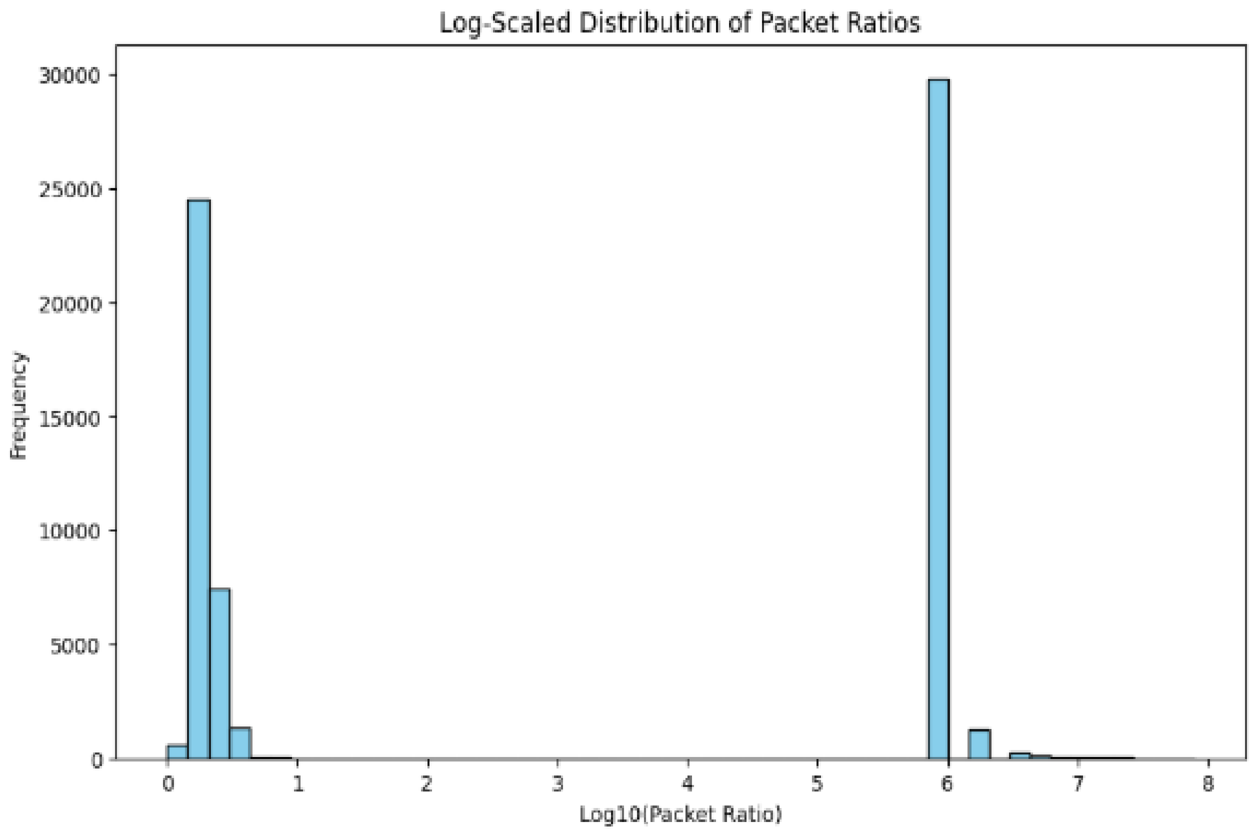
- Packet Ratio: Ratio of pkts_sent to pkts_received.
- Log Packet Ratio: Log-transformed Packet Ratio for better scaling and interpretation.

Insights:

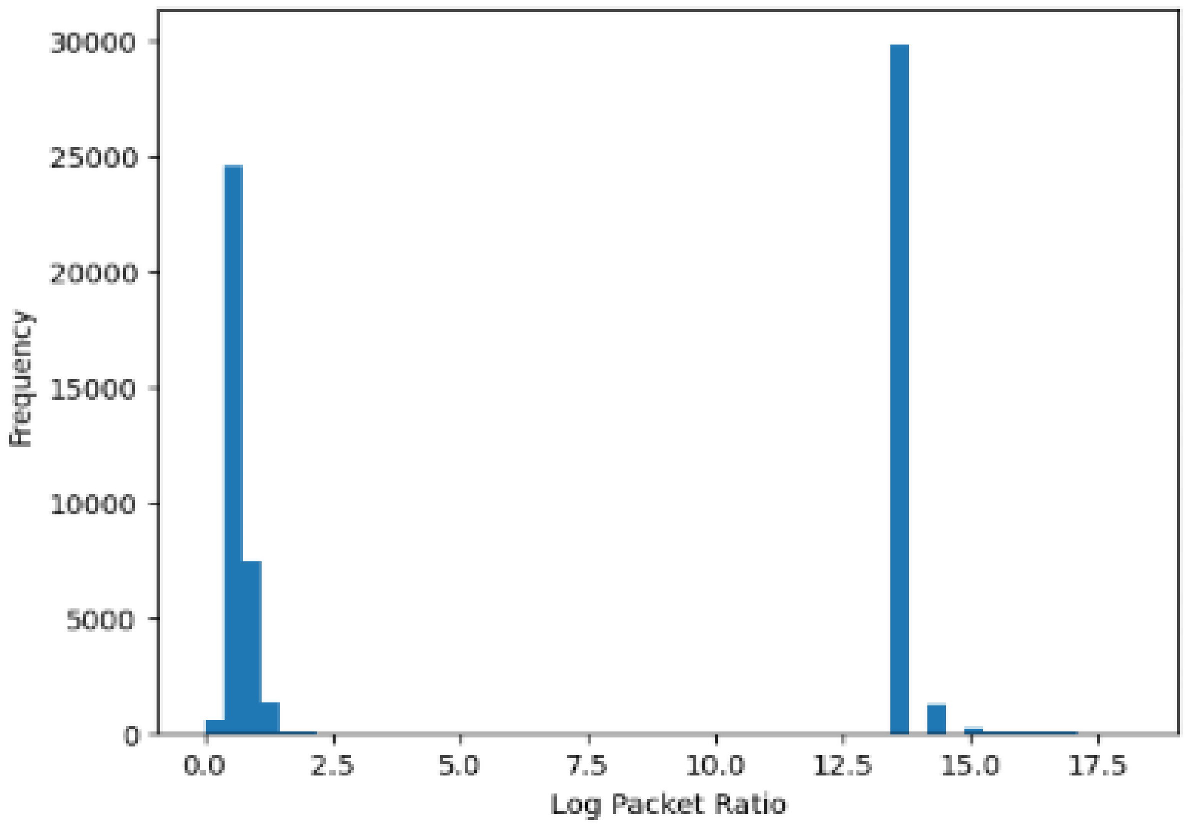
- Maximum Packet Ratio: 79,000,000.
- Minimum Packet Ratio: 0.0057.
- Maximum Packets Sent: 747,520.
- Maximum Packets Received: 327,208.

Visualization:

Log-Scaled Distribution of Packet Ratios



Histogram of Log Packet Ratio



Log Packet Ratio Classification

Categories Defined:

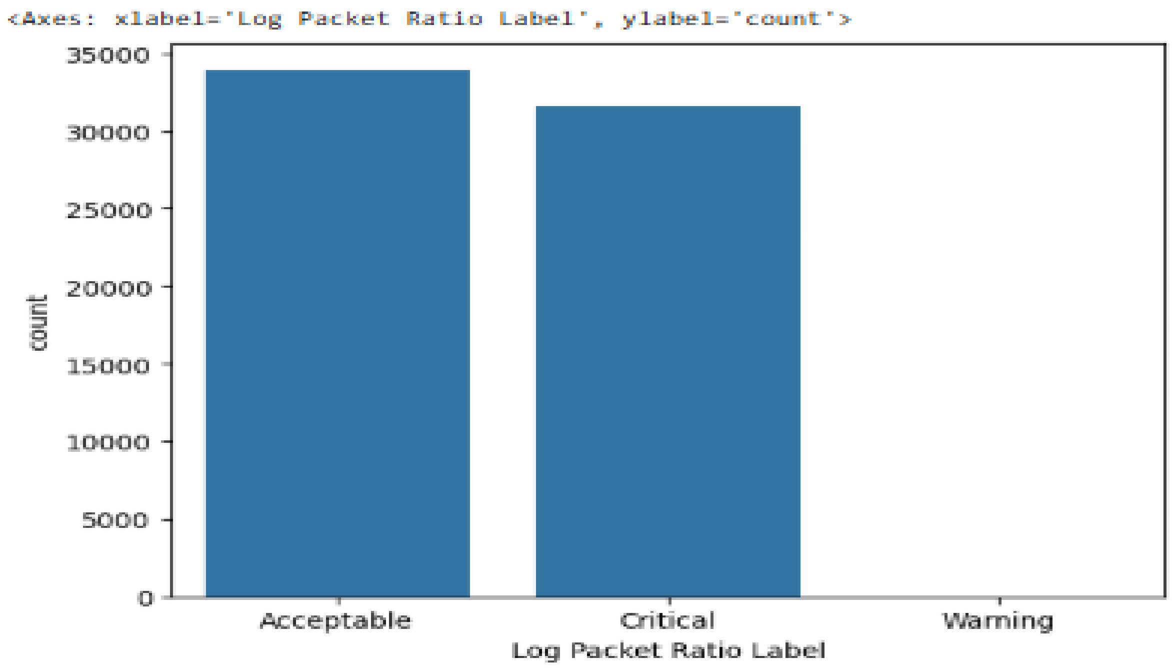
Acceptable (Log Ratio ≈ 0 to 2): Normal and balanced traffic.

Warning (Log Ratio ≈ 2 to 12): Noticeable discrepancies.

Critical (Log Ratio > 12): Extreme discrepancies requiring immediate attention.

Visualization:

Log Packet Ratio Classification Counts



```
<Axes: xlabel='Log Packet Ratio Label', ylabel='count'>
```

Machine Learning Implementation

Feature Selection :

For the machine learning models, the following features were selected for training:

- Selected Features: Bytes, Bytes Sent, Bytes Received, Packets, pkts_sent, pkts_received, Anomaly
- Dropped Features: Source Port, Destination Port, NAT Source Port, NAT Destination Port, Action, Bytes Imbalance, Bytes Imbalance (%), Packet Ratio, Log Packet Ratio, Log Packet Ratio Label, Timestamp, Elapsed Time (sec), hour, Avg Packet Size Sent, Avg Packet Size Received, Traffic Rate, Packet Rate, Action.

Approach 1

Binary Classification

In this approach, the goal is to identify normal behavior and anomalies (critical/warning) in network traffic data. By combining the categories, the model is simplified, making it easier to interpret the results and alert when anomalies are detected.

Objective:

- Simplify the problem by combining categories into two classes:
  - 0 = Normal Behavior
  - 1 = Anomaly (Critical/Warning)

Algorithm Used:

- Random Forest Classifier

Steps:

1. **Prepare Data:** Select relevant features such as Bytes, Bytes Sent, Bytes Received, Packets, Elapsed Time (sec), pkts_sent, pkts_received, and action_encoded from the main training dataset. Create a new target variable

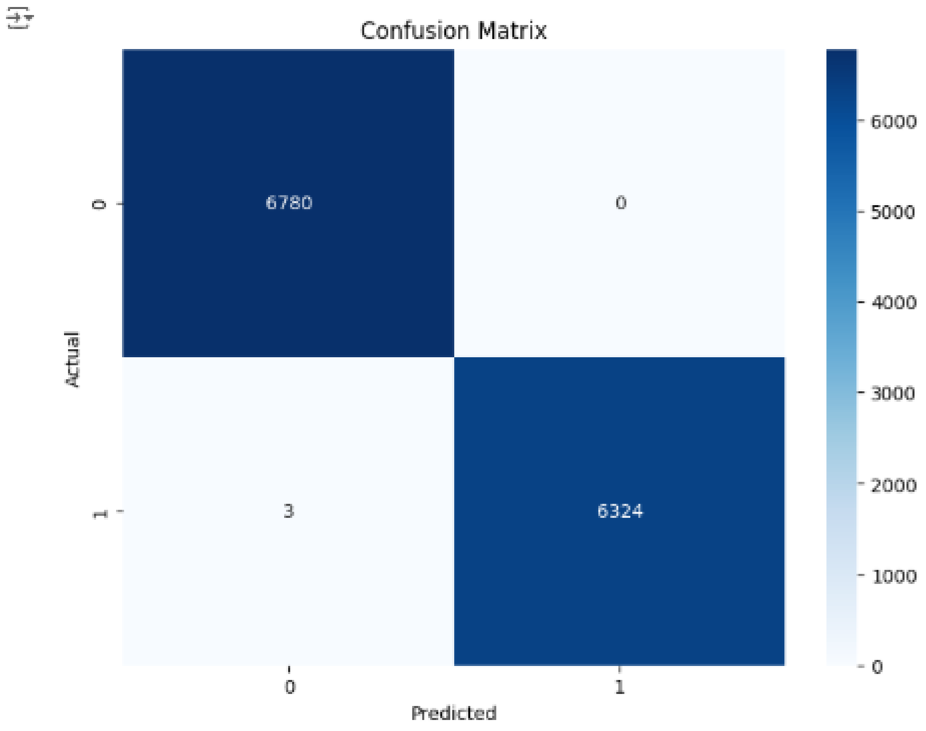
Anomaly_Label where critical and warning logs are labeled as 1 (anomalies) and the rest as 0 (normal behavior).

2. **Split Data:** Divide the dataset into training and testing sets, with an 80-20 split, to ensure that the model is trained on one portion of the data and evaluated on another.
3. **Train Random Forest Classifier:** Use the Random Forest algorithm to train the model on the training data. This step involves fitting the model to the training data with class weights.
4. **Predictions:** Generate predictions on the test data using the trained Random Forest model. Calculate probabilities for the positive class (anomalies).
5. **Evaluate Model:**
    - **Accuracy:** Measure the accuracy of the model by comparing the predicted labels with the actual labels.
    - **Precision, Recall, and F1-Score:** Evaluate the model's performance using these metrics for both normal and anomaly classes.
    - **Confusion Matrix:** Display the confusion matrix to visualize the performance of the model in terms of true positive, false positive, true negative, and false negative predictions.

Performance Metrics:

- **Accuracy:** 1.00
- **Classification Report:**
    - Precision, Recall, and F1-Score for both classes were 1.00, demonstrating excellent performance.

Confusion Matrix: A visual representation of predictions confirmed zero misclassifications.



Approach 2

Binary Classification with Class Imbalance Handling:

In this approach, the goal is to identify normal behavior and anomalies (critical/warning) in network traffic data. By combining the categories, the model is simplified, making it easier to interpret the results and alert when anomalies are detected, but addressed the potential issue of class imbalance.

Objective:

- Simplify the problem by combining categories into two classes:
  - 0 = Normal Behavior
  - 1 = Anomaly (Critical/Warning)

Algorithm Used:

- Random Forest Classifier

Steps:

6. **Prepare Data:** Select relevant features such as Bytes, Bytes Sent, Bytes Received, Packets, Elapsed Time (sec), pkts_sent, pkts_received, and action_encoded from the main training dataset. Create a new target variable

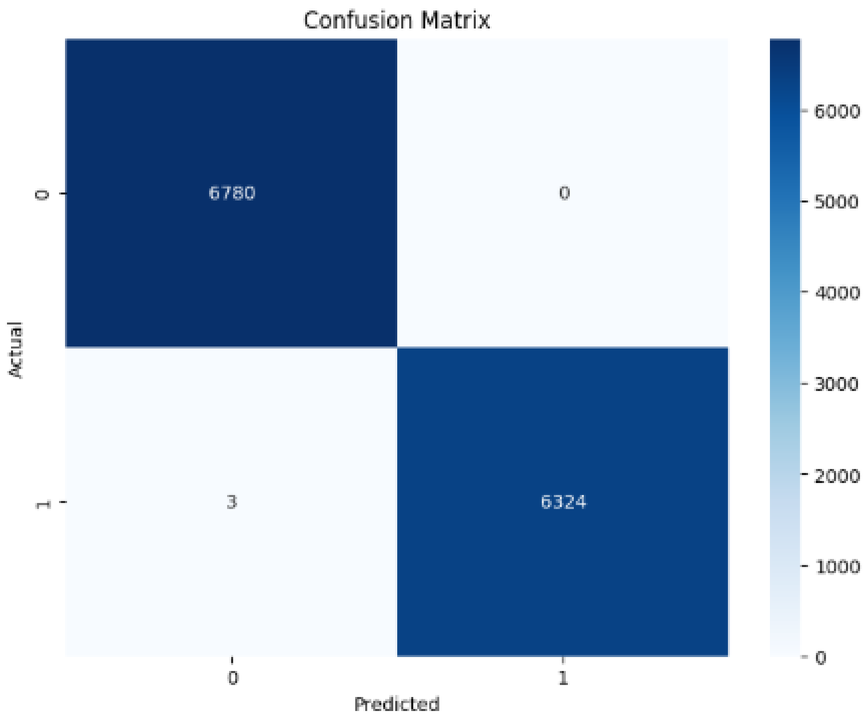
Anomaly_Label where critical and warning logs are labeled as 1 (anomalies) and the rest as 0 (normal behavior).

7. **Split Data:** Divide the dataset into training and testing sets, with an 80-20 split, to ensure that the model is trained on one portion of the data and evaluated on another.
8. **Train Random Forest Classifier:** Use the Random Forest algorithm to train the model on the training data. This step involves fitting the model to the training data with class weights.
9. **Predictions:** Generate predictions on the test data using the trained Random Forest model. Calculate probabilities for the positive class (anomalies).
10. **Evaluate Model:**
    - **Accuracy:** Measure the accuracy of the model by comparing the predicted labels with the actual labels.
    - **Precision, Recall, and F1-Score:** Evaluate the model's performance using these metrics for both normal and anomaly classes.
    - **Confusion Matrix:** Display the confusion matrix to visualize the performance of the model in terms of true positive, false positive, true negative, and false negative predictions.

Performance Metrics:

- **Accuracy:** 1.00
- **Classification Report:**
    - Precision, Recall, and F1-Score for both classes were 1.00, demonstrating excellent performance.

Confusion Matrix: A visual representation of predictions confirmed zero misclassifications.



Approach 3

Multi-Class Classification :

This approach aims to classify network traffic logs into three categories: Acceptable, Critical, and Warning. By encoding these categories and computing class weights, the model can be trained to detect and alert on anomalies effectively.

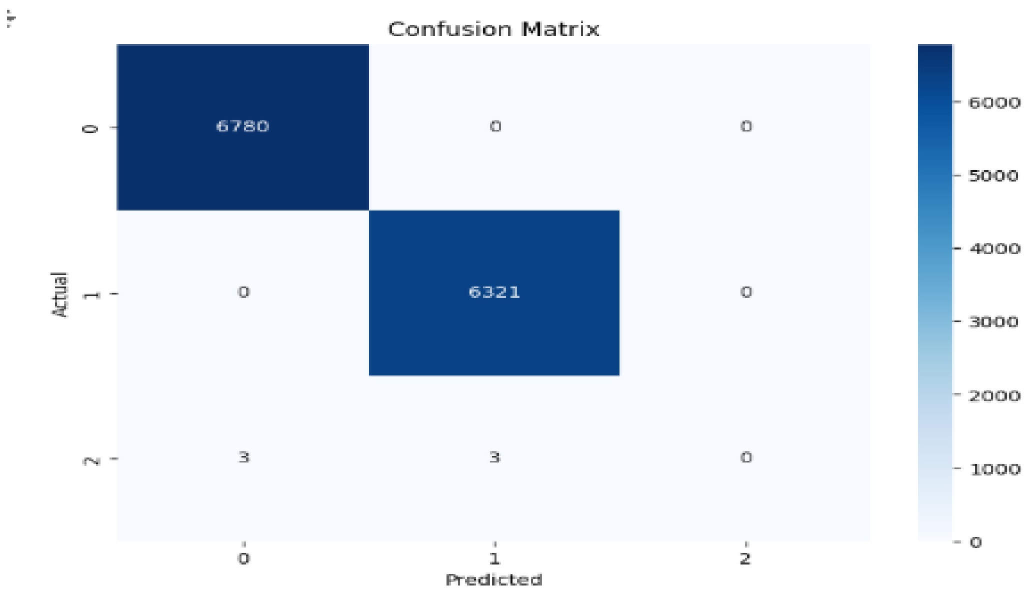
Steps:

1. **Encode Categorical Labels:**
   - Use LabelEncoder to transform categorical labels across the entire dataset into numerical labels.
   - Categories include:
     - Acceptable
     - Critical
     - Warning
2. **Prepare Data:** Select relevant features such as `Bytes`, `Bytes Sent`, `Bytes Received`, `Packets`, `Elapsed Time (sec)`, `pkts_sent`, `pkts_received`, and `action_encoded` from the main training dataset. The target variable is the encoded label of `Log Packet Ratio Label`.
3. **Split Data:** Divide the dataset into training and testing sets, with an 80-20 split, ensuring the model is trained on one portion of the data and evaluated on another.
4. **Compute Class Weights:** Calculate class weights to handle the imbalanced dataset and ensure the model treats each class appropriately.

5. **Train Random Forest Classifier:** Use the Random Forest algorithm with the computed class weights to train the model on the training data.
6. **Predictions and Decoding:** Generate predictions on the test data and decode the predicted labels back to the original categories for evaluation.
7. **Evaluate Model:**
    - **Accuracy:** Measure the accuracy of the model by comparing the predicted labels with the actual labels.
    - **Precision, Recall, and F1-Score:** Evaluate the model's performance using these metrics for each class.
    - **Confusion Matrix:** Display the confusion matrix to visualize the model's performance in terms of true positive, false positive, true negative, and false negative predictions.

Results:

- **Accuracy:** 1.00
- **Classification Report:**
    - Precision, Recall, and F1-Score for both classes were 1.00, demonstrating excellent performance.



Graphical Analysis

For all approaches, confusion matrix graphs were generated to visually assess the model’s predictions. These graphs confirmed the model's high accuracy and ability to handle both binary and multi-class classifications effectively.

Reporting Dashboard

This dashboard is designed to provide a comprehensive overview of network incidents by leveraging Exploratory Data Analysis (EDA) and the outputs from various anomaly detection systems. It offers summaries of incidents over different time frames, identifies errors, detects threats, and highlights unusual activities.

Key Features:

1. **Incident Summaries:**
    - Summarizes traffic data over different time frames (hourly, 12 hours, and 24 hours).

    - Allows users to select the desired timeframe for detailed insights.
2. **Error Insights:**
    - Displays the total number of errors such as network failures and packet loss.
    - Provides detailed information on each error event.
3. **Threat Detection:**
    - Identifies and lists threats including intrusion attempts and malware communication.
    - Counts the total threats detected for easy monitoring.
4. **Traffic Volume Analysis:**
    - Visualizes the traffic volume using a line chart.
    - Helps in understanding the data flow and identifying peak traffic times.
5. **Rare IP Activity:**
    - Highlights rare source and destination ports to detect unusual IP activities.
    - Lists the least common ports to aid in anomaly detection.
6. **Unexpected Traffic Spikes:**
    - Detects and reports unexpected traffic spikes by analyzing traffic rates.
    - Provides a detailed view of each detected spike.

How to Use:

1. **Upload Data:**
    - Upload your firewall log file “output.csv” in CSV format through the provided file uploader.
2. **Feature Engineering:**

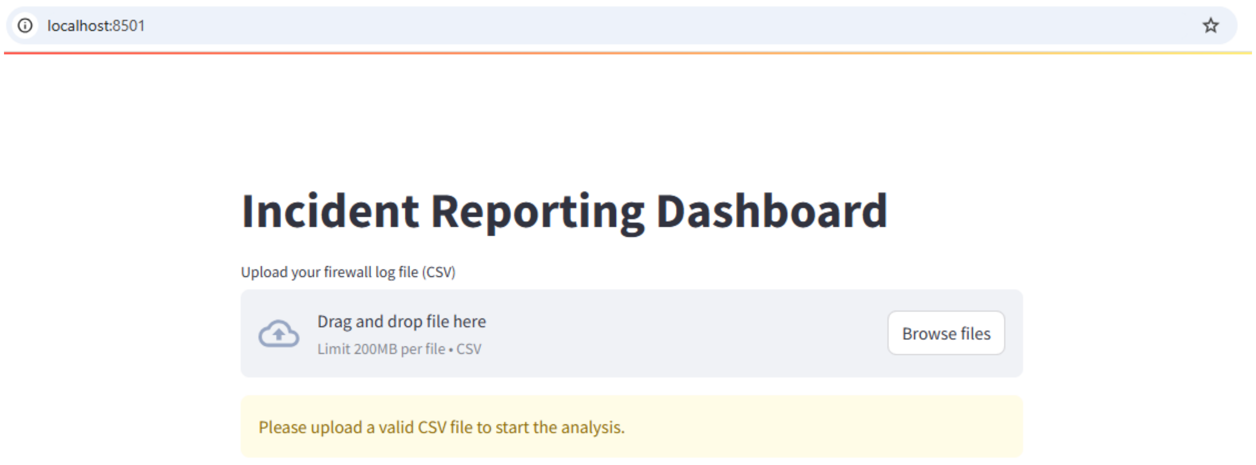
    - The dashboard preprocesses the data to derive various features such as timestamps, average packet sizes, traffic rates, and packet rates.
3. **Anomaly Detection:**
    - Utilizes Isolation Forest to detect anomalies based on selected features like source port, destination port, traffic rate, and packet rate.
4. **Select Time Frame:**
    - Choose the desired timeframe (hourly, 12 hours, or 24 hours) to view incident summaries.
5. **Review Insights:**
    - Examine the summarized data, error insights, threat detections, traffic volume analysis, rare IP activities, and unexpected traffic spikes.

Command to Run:

```
streamlit run dashboard.py
```

Screenshot:

- Include your dashboard screenshot here for a visual representation.


localhost:8501

Incident Reporting Dashboard

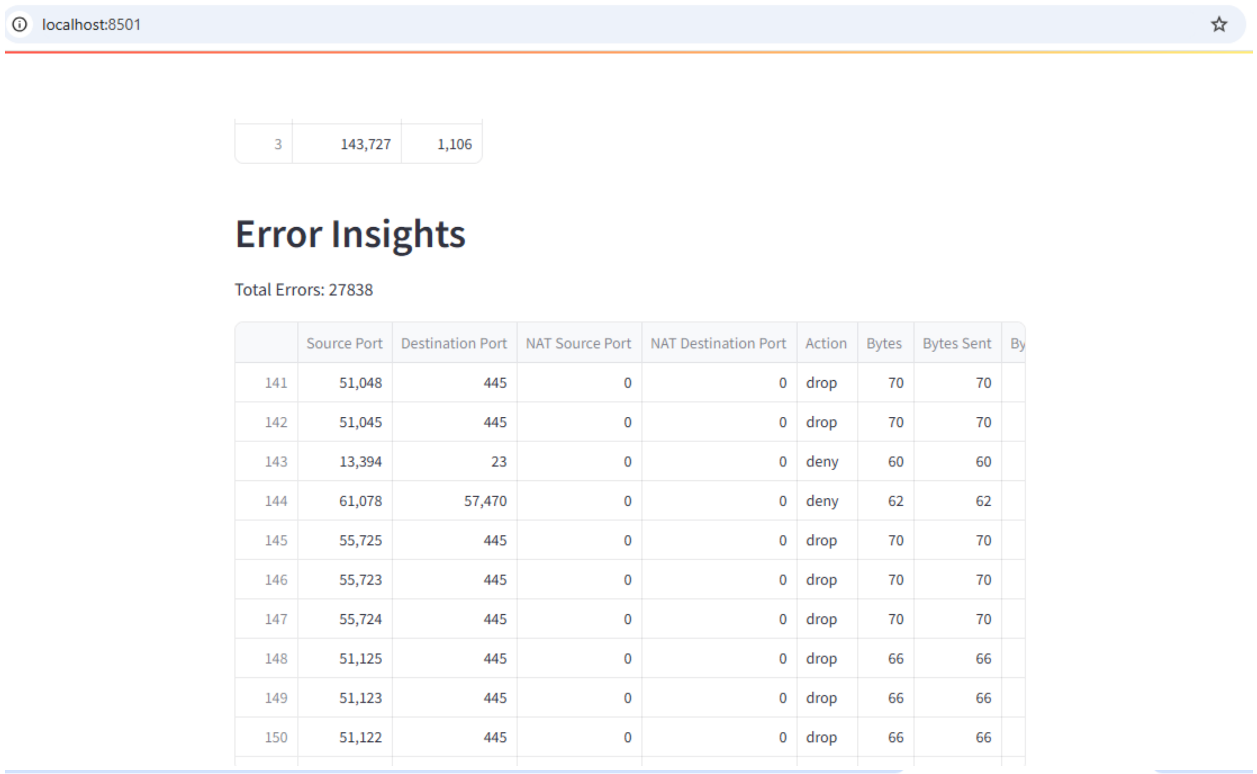
Upload your firewall log file (CSV)

Drag and drop file here

Limit 200MB per file • CSV

Browse files

Please upload a valid CSV file to start the analysis.



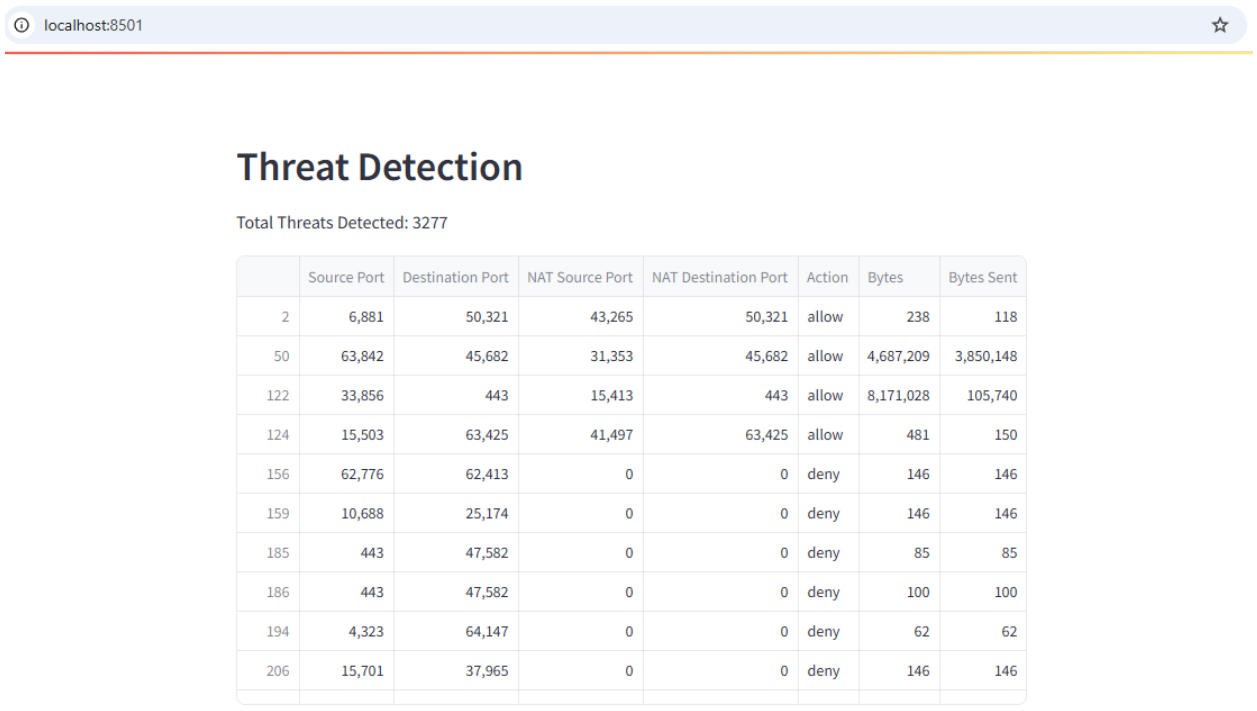
localhost:8501

3	143,727	1,106

Error Insights

Total Errors: 27838

	Source Port	Destination Port	NAT Source Port	NAT Destination Port	Action	Bytes	Bytes Sent	By
141	51,048	445	0	0	drop	70	70	
142	51,045	445	0	0	drop	70	70	
143	13,394	23	0	0	deny	60	60	
144	61,078	57,470	0	0	deny	62	62	
145	55,725	445	0	0	drop	70	70	
146	55,723	445	0	0	drop	70	70	
147	55,724	445	0	0	drop	70	70	
148	51,125	445	0	0	drop	66	66	
149	51,123	445	0	0	drop	66	66	
150	51,122	445	0	0	drop	66	66	

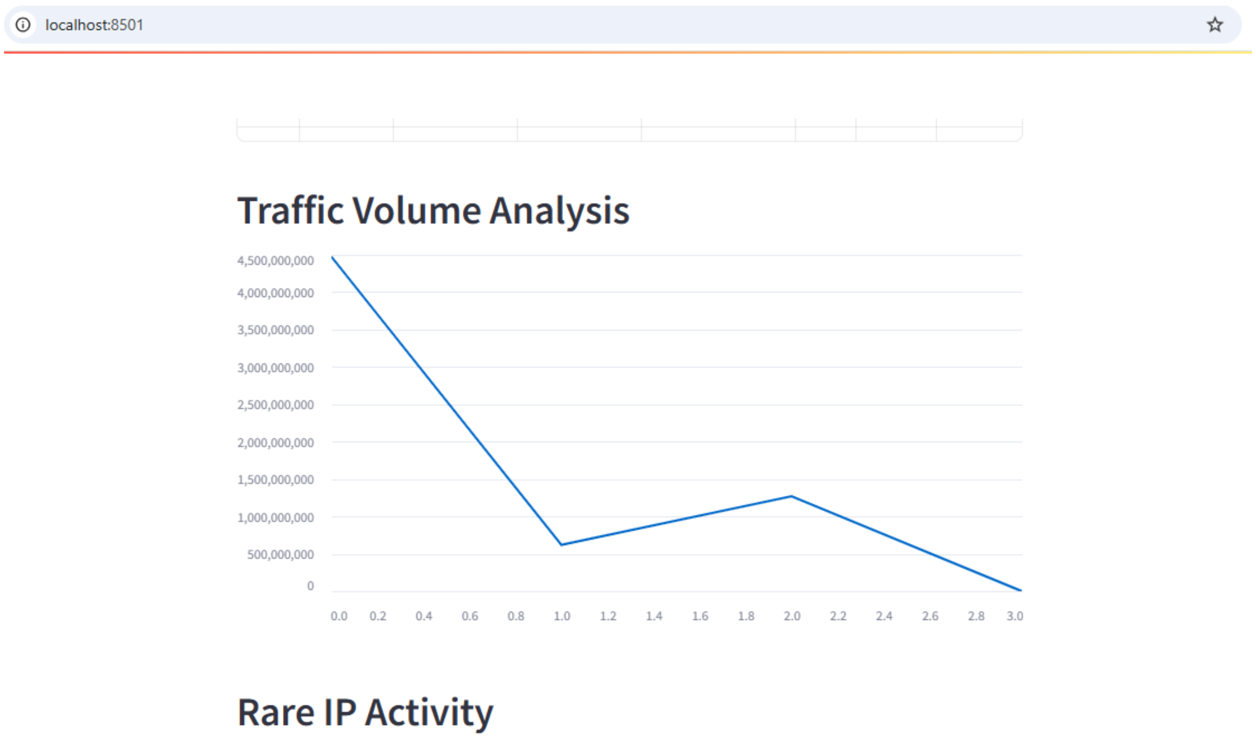


localhost:8501

Threat Detection

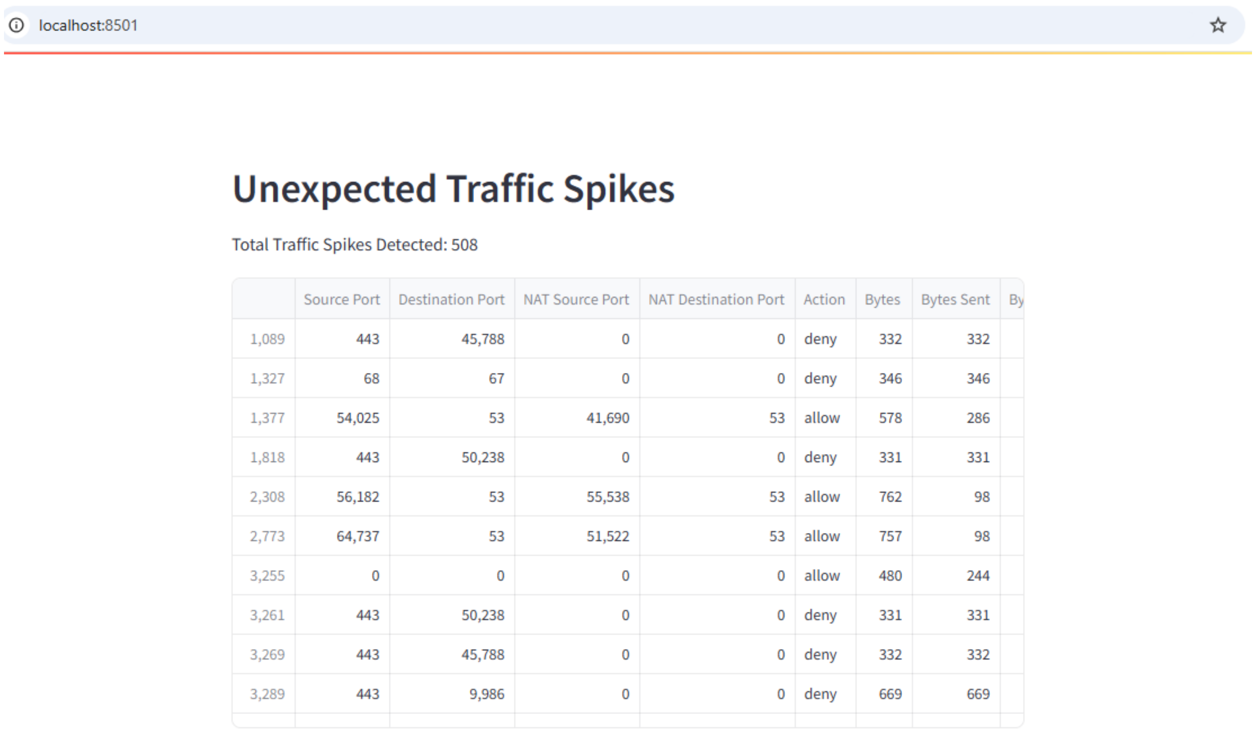
Total Threats Detected: 3277

	Source Port	Destination Port	NAT Source Port	NAT Destination Port	Action	Bytes	Bytes Sent
2	6,881	50,321	43,265	50,321	allow	238	118
50	63,842	45,682	31,353	45,682	allow	4,687,209	3,850,148
122	33,856	443	15,413	443	allow	8,171,028	105,740
124	15,503	63,425	41,497	63,425	allow	481	150
156	62,776	62,413	0	0	deny	146	146
159	10,688	25,174	0	0	deny	146	146
185	443	47,582	0	0	deny	85	85
186	443	47,582	0	0	deny	100	100
194	4,323	64,147	0	0	deny	62	62
206	15,701	37,965	0	0	deny	146	146



Traffic Volume Analysis

Rare IP Activity



Unexpected Traffic Spikes

Total Traffic Spikes Detected: 508

	Source Port	Destination Port	NAT Source Port	NAT Destination Port	Action	Bytes	Bytes Sent	By
1,089	443	45,788	0	0	deny	332	332	
1,327	68	67	0	0	deny	346	346	
1,377	54,025	53	41,690	53	allow	578	286	
1,818	443	50,238	0	0	deny	331	331	
2,308	56,182	53	55,538	53	allow	762	98	
2,773	64,737	53	51,522	53	allow	757	98	
3,255	0	0	0	0	allow	480	244	
3,261	443	50,238	0	0	deny	331	331	
3,269	443	45,788	0	0	deny	332	332	
3,289	443	9,986	0	0	deny	669	669	

Prepared For:

Data Scientist Role Assignment