Intrusion Detection in Network Traffic Using Data Science Techniques

Prepared by: Reem Awawdy

ID: 212018899

Course: Applying Data Science Techniques in Cybersecurity

Instructor: Dr. Uri Itai

Submission Date: July 2025

Project Overview

This project focuses on detecting network intrusions using a structured data science pipeline applied to real-world network traffic.
The dataset is based on a simulated military network (LAN) environment that was intentionally exposed to various types of cyberattacks. Each entry in the dataset represents a single TCP/IP connection, described by 41 quantitative and qualitative features.
The objective of this project is to analyze these features through data science techniques, including statistical analysis, anomaly detection, clustering, and supervised machine learning.
Ultimately, the goal is to build a model that can accurately classify connections as either *normal* or *anomalous*, helping to identify patterns of malicious behavior and enhance network-level cybersecurity.

About the Dataset

The dataset used in this project is a publicly available network intrusion detection dataset that simulates real-world cyberattacks in a controlled military network environment.
It includes thousands of TCP/IP connections, each described by 41 features capturing various aspects of network behavior.
Each connection is labeled as either **normal** or **anomalous**, making the dataset suitable for building and evaluating intrusion detection systems.
Further details about the dataset structure and features will be explained in the following sections.

Why I Chose This Dataset

I chose this dataset because it provides a realistic simulation of cyberattacks in a network environment, which aligns perfectly with the goals of this project.
Unlike basic or synthetic datasets, this one contains rich, structured data collected from simulated TCP/IP connections under both normal and attack conditions.
It includes a diverse set of quantitative and categorical features that allow for a full data science pipeline-ranging from statistical analysis and anomaly detection to clustering and supervised classification.
The dataset's real-world relevance, complexity, and completeness make it an ideal choice for developing, evaluating, and demonstrating effective network intrusion detection techniques.

System Stage – Dataset File Information

- **File Name:** NetworkIntrusion.csv
- **File Type:** CSV (Comma-Separated Values)
- **File Size:** 5.25 MB *(adjust if needed)*
- **Source:** Downloaded from Kaggle using a secure HTTPS connection
- **Number of Rows:** 22,544 network connections
- **Number of Columns:** 41

Column Descriptions:

- duration – Duration of the connection (in seconds)
- protocol_type – Network protocol used (e.g., TCP, UDP, ICMP)
- service – Application service on the destination (e.g., HTTP, FTP, DNS)
- flag – Status flag of the connection (e.g., SF, REJ)
- src_bytes, dst_bytes – Number of bytes sent from source/destination
- serror_rate, rerror_rate, same_srv_rate, etc. – Statistical and behavioral features of the connection
- *Total: 38 numerical features and 3 categorical features*

Format Explanation:
Each row in the dataset represents a single TCP/IP connection between a source and destination IP address. The dataset captures both normal and malicious traffic from a simulated military network environment that was subjected to a variety of cyberattacks. It is structured in CSV format to facilitate analysis and model development.

Version Control:
Currently, only one version of the dataset is used (NetworkIntrusion.csv).
Additional versions may be created throughout the project, such as:

- NetworkIntrusion_cleaned.csv – for cleaned data
- NetworkIntrusion_labeled.csv – after adding labels if needed
- NetworkIntrusion_vectorized.csv – for model-ready feature vectors

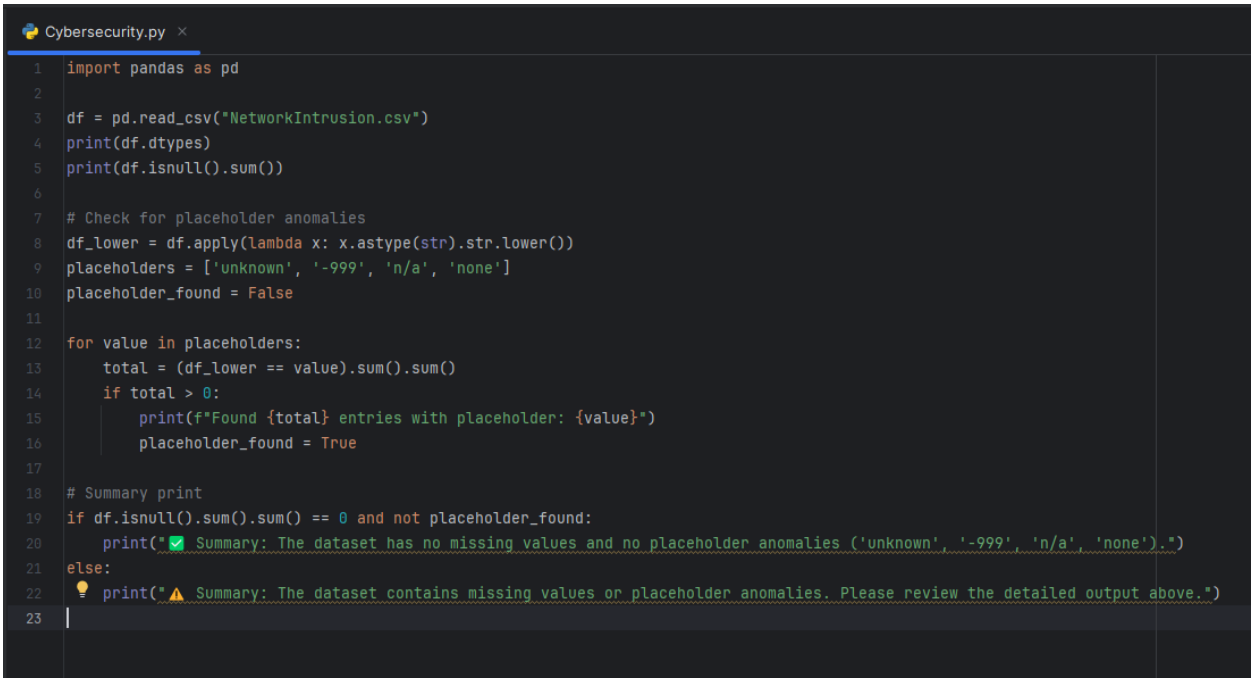
Meta Data

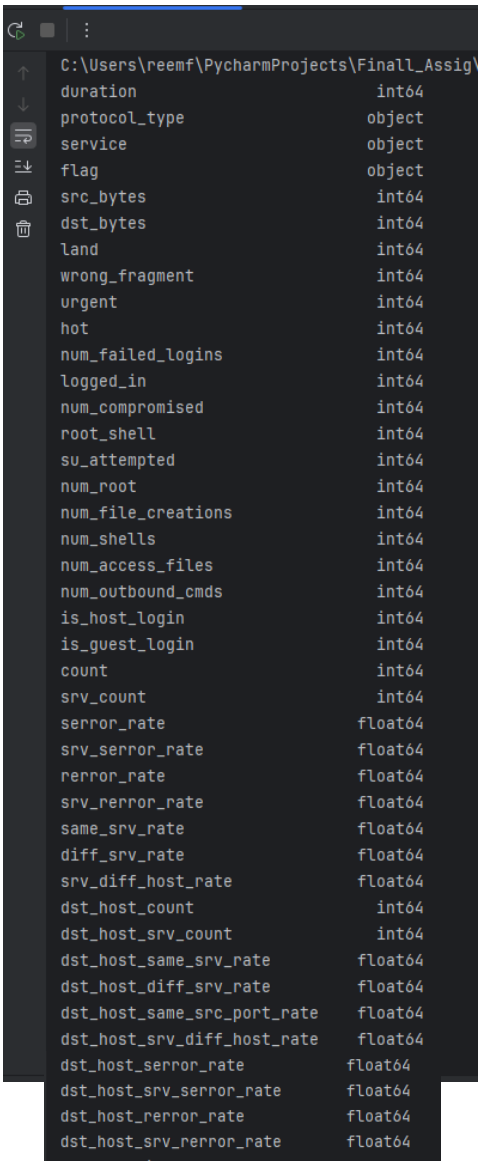
The dataset contains **41 columns**, each representing a specific feature of a network connection. These features span both **categorical** and **numerical** data types, including protocol types, byte counts, error rates, and login-related attributes.

A detailed integrity check was performed to ensure the dataset's cleanliness and reliability:

- **Data types** were verified using Python, confirming the correct formats (e.g., int64 and float64 for numerical fields, object for categorical variables).
- **No missing values** were found in any of the columns.
- A thorough scan for common placeholder anomalies such as "unknown", "-999", "n/a", and "none" yielded **zero occurrences**.
- The final summary confirmed that the dataset is **clean, consistent, and ready** for further preprocessing and analysis.

This validation process was conducted programmatically using Python, and the corresponding output was included to provide transparent evidence.





```
C:\Users\reemf\PycharmProjects\Finall_Assig\
duration                          int64
protocol_type                    object
service                          object
flag                             object
src_bytes                         int64
dst_bytes                         int64
land                              int64
wrong_fragment                    int64
urgent                            int64
hot                               int64
num_failed_logins                 int64
logged_in                         int64
num_compromised                   int64
root_shell                        int64
su_attempted                      int64
num_root                          int64
num_file_creations                int64
num_shells                        int64
num_access_files                  int64
num_outbound_cmds                 int64
is_host_login                     int64
is_guest_login                    int64
count                             int64
srv_count                         int64
serror_rate                     float64
srv_serror_rate                 float64
rerror_rate                     float64
srv_rerror_rate                 float64
same_srv_rate                   float64
diff_srv_rate                   float64
srv_diff_host_rate              float64
dst_host_count                    int64
dst_host_srv_count                int64
dst_host_same_srv_rate          float64
dst_host_diff_srv_rate          float64
dst_host_same_src_port_rate     float64
dst_host_srv_diff_host_rate     float64
dst_host_serror_rate            float64
dst_host_srv_serror_rate        float64
dst_host_rerror_rate            float64
dst_host_srv_rerror_rate        float64
```

```
dtype: object
duration                        0
protocol_type                   0
service                         0
flag                            0
src_bytes                       0
dst_bytes                       0
land                            0
wrong_fragment                  0
urgent                          0
hot                             0
num_failed_logins               0
logged_in                       0
num_compromised                 0
root_shell                      0
su_attempted                    0
num_root                        0
num_file_creations              0
num_shells                      0
num_access_files                0
num_outbound_cmds               0
is_host_login                   0
is_guest_login                  0
count                           0
srv_count                       0
serror_rate                     0
srv_serror_rate                 0
rerror_rate                     0
srv_rerror_rate                 0
same_srv_rate                   0
diff_srv_rate                   0
srv_diff_host_rate              0
dst_host_count                  0
dst_host_srv_count              0
dst_host_same_srv_rate          0
dst_host_diff_srv_rate          0
dst_host_same_src_port_rate     0
dst_host_srv_diff_host_rate     0
dst_host_serror_rate            0
dst_host_srv_serror_rate        0
dst_host_rerror_rate            0
dst_host_srv_rerror_rate        0
dtype: int64
```

```
✅ Summary: The dataset has no missing values and no placeholder anomalies ('unknown', '-999', 'n/a', 'none').

Process finished with exit code 0
```

Statistical Analysis

Distributions of Selected Features

Why not include all columns?

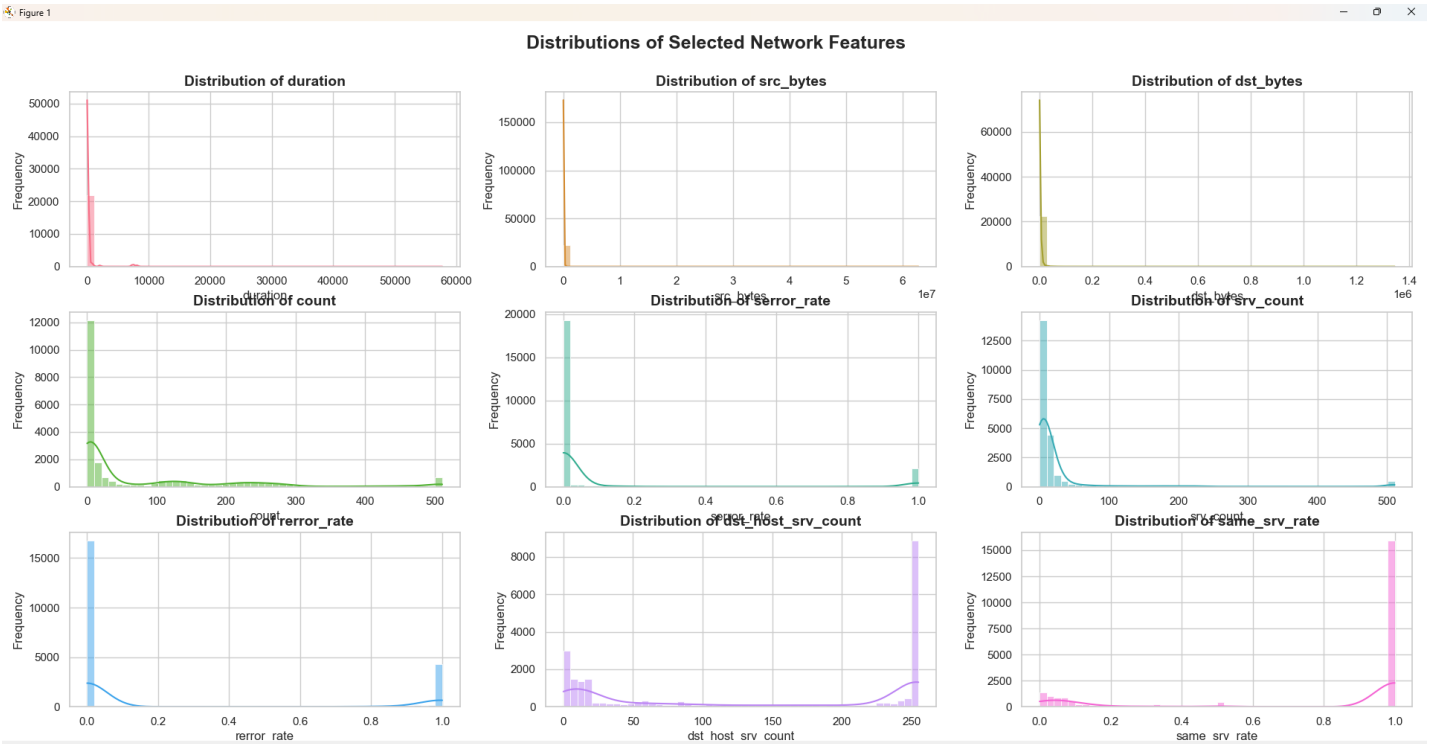
The dataset contains 41 features, many of which are binary (0/1), constant, or categorical (e.g., protocol_type, flag). Plotting all of them would result in repetitive or uninformative charts. Instead, I selected **9 key numerical features** that are continuous and relevant to analyzing network behavior and potential intrusions.

Selected Columns:

```
selected_columns = [
    'duration',
    'src_bytes',
    'dst_bytes',
    'count',
    'serror_rate',
    'srv_count',
    'rerror_rate',
    'dst_host_srv_count',
    'same_srv_rate'
]
```

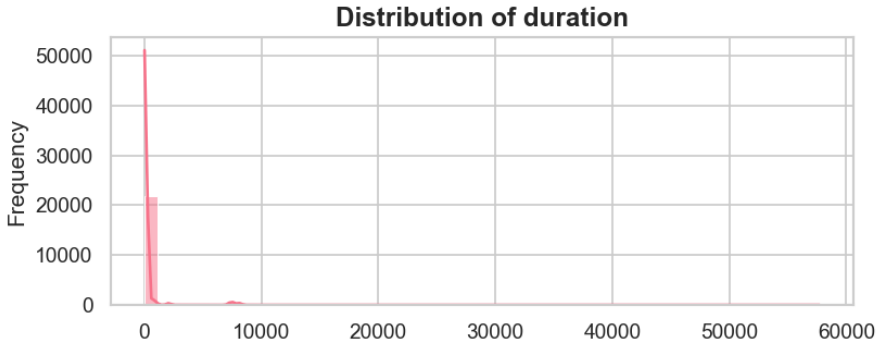
Visualization:

A single figure displaying 9 histograms (with KDE curves) shows the distributions of the selected columns. This allows us to observe the behavior of each feature at a glance.

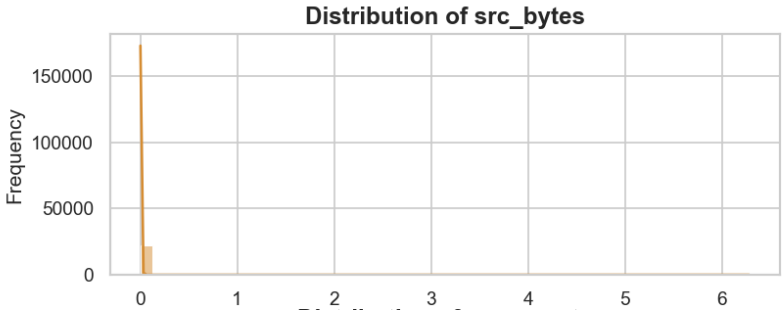


Interpretation of Each Feature Distribution:

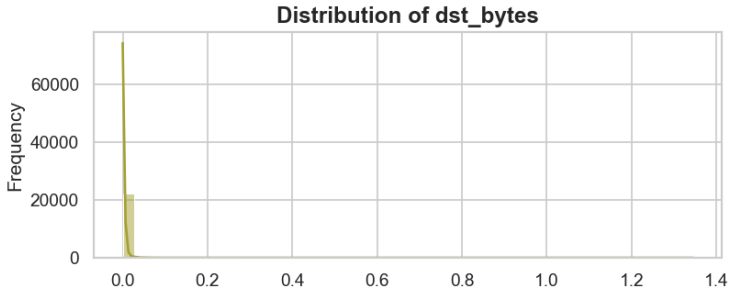
1. **duration**
    - **Meaning**: Duration of each network connection.
    - **Distribution**: Strongly right-skewed — most connections are short, with a few very long sessions.
    - **Insight**: Normal behavior is short sessions; longer ones might signal abnormal activity.



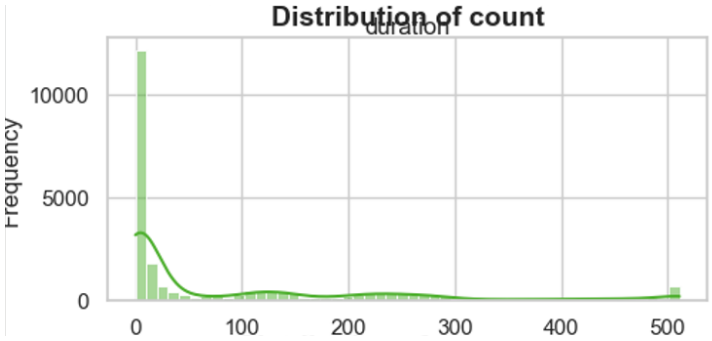
2. **src_bytes**
    - **Meaning**: Bytes sent from source to destination.
    - **Distribution**: Right-skewed with extreme outliers.
    - **Insight**: Most interactions involve small payloads; large ones may require further investigation.



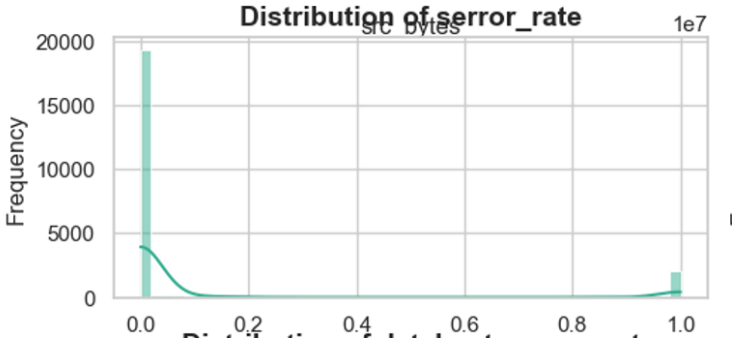
3. **dst_bytes**
    - **Meaning**: Bytes sent from destination to source.
    - **Distribution**: Also right-skewed with long tail.
    - **Insight**: Frequent small values could indicate denied access or scanning activity.



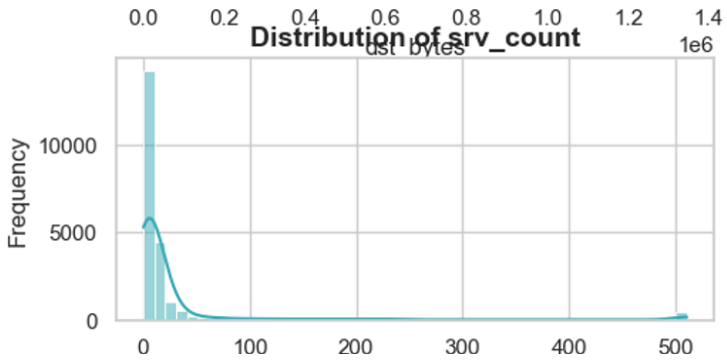
4. **count**
    - **Meaning**: Number of connections to the same host in the last 2 seconds.
    - **Distribution**: Mostly low values, but some high spikes.
    - **Insight**: Sudden spikes may reflect potential DoS attacks or scans.



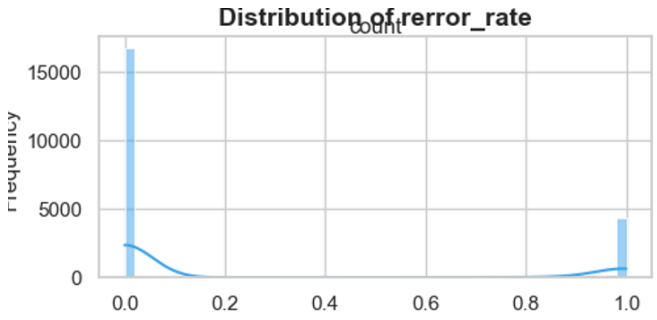
5. **serror_rate**
    - **Meaning**: Percentage of connections with SYN errors.
    - **Distribution**: Bimodal — values mostly near 0 or 1.
    - **Insight**: This sharp contrast could help distinguish between normal and attack traffic.



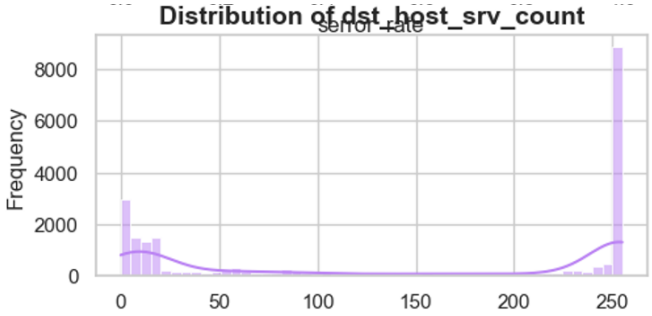
6. **srv_count**
    - **Meaning**: Number of connections to the same service in the last 2 seconds.
    - **Distribution**: Right-skewed.
    - **Insight**: High values may signal repeated targeting of a specific service.



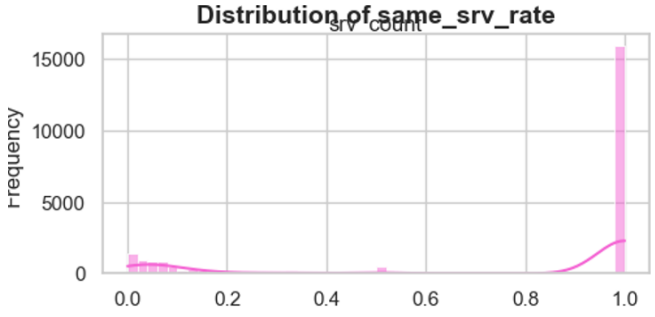
7. **rerror_rate**
    - **Meaning**: Percentage of rejected connections.
    - **Distribution**: Also bimodal.
    - **Insight**: Frequent rejection (values close to 1) may point to malicious activity.



8. **dst_host_srv_count**
    - **Meaning**: Number of connections to the same service on a specific host.
    - **Distribution**: Spread across a wide range, with many high values.
    - **Insight**: Could indicate that a host is being targeted heavily.



9. **same_srv_rate**
    - **Meaning**: Proportion of connections to the same service.
    - **Distribution**: Clustered near 1.
    - **Insight**: Suggests repetitive access to the same service — normal for some applications, but also common in targeted attacks.



Summary:

This distribution analysis reveals useful patterns in the selected numerical features. The presence of skewed and bimodal distributions suggests that some variables may be strong indicators of abnormal behavior or cyberattacks. These insights will guide the next steps in correlation analysis and feature selection.

Descriptive Statistics: Central Tendency and Dispersion

	count	mean	std	min	25%	50%	75%	max	skew	kurtosis
duration	22544.0	218.859	1407.177	0.0	0.0	0.0	0.0	57715.0	15.453	451.373
src_bytes	22544.0	10395.45	472786.431	0.0	0.0	54.0	287.0	62825648.0	117.486	14714.398
dst_bytes	22544.0	2056.019	21219.298	0.0	0.0	46.0	601.0	1345927.0	47.503	2629.481
count	22544.0	79.028	128.539	0.0	1.0	8.0	123.25	511.0	1.909	3.026
serror_rate	22544.0	0.103	0.295	0.0	0.0	0.0	0.0	1.0	2.652	5.123
srv_count	22544.0	31.124	89.063	0.0	1.0	6.0	16.0	511.0	4.354	18.925
rerror_rate	22544.0	0.238	0.416	0.0	0.0	0.0	0.25	1.0	1.211	-0.491
dst_host_srv_count	22544.0	140.751	111.784	0.0	15.0	168.0	255.0	255.0	-0.134	-1.834
same_srv_rate	22544.0	0.74	0.412	0.0	0.25	1.0	1.0	1.0	-1.013	-0.901

The table above presents key statistical measures for nine selected numerical features from the dataset. These measures provide valuable insight into the distribution and structure of the data.

Included Measures:

- **count**: Total number of non-null entries for each feature.
- **mean**: The arithmetic average.
- **std (Standard Deviation)**: Indicates the spread or dispersion of the values.
- **min, 25%, 50%, 75%, max**: These represent the minimum, first quartile, median, third quartile, and maximum values, respectively—useful for understanding the range and shape of the distribution.
- **skew**: Describes the asymmetry of the distribution. A skew of 0 implies symmetry, positive values indicate right-skewed data, and negative values indicate left-skewed data.
- **kurtosis**: Measures the peakedness or "tailedness" of the distribution. High kurtosis indicates sharp peaks and long tails, while low kurtosis reflects flatter distributions.

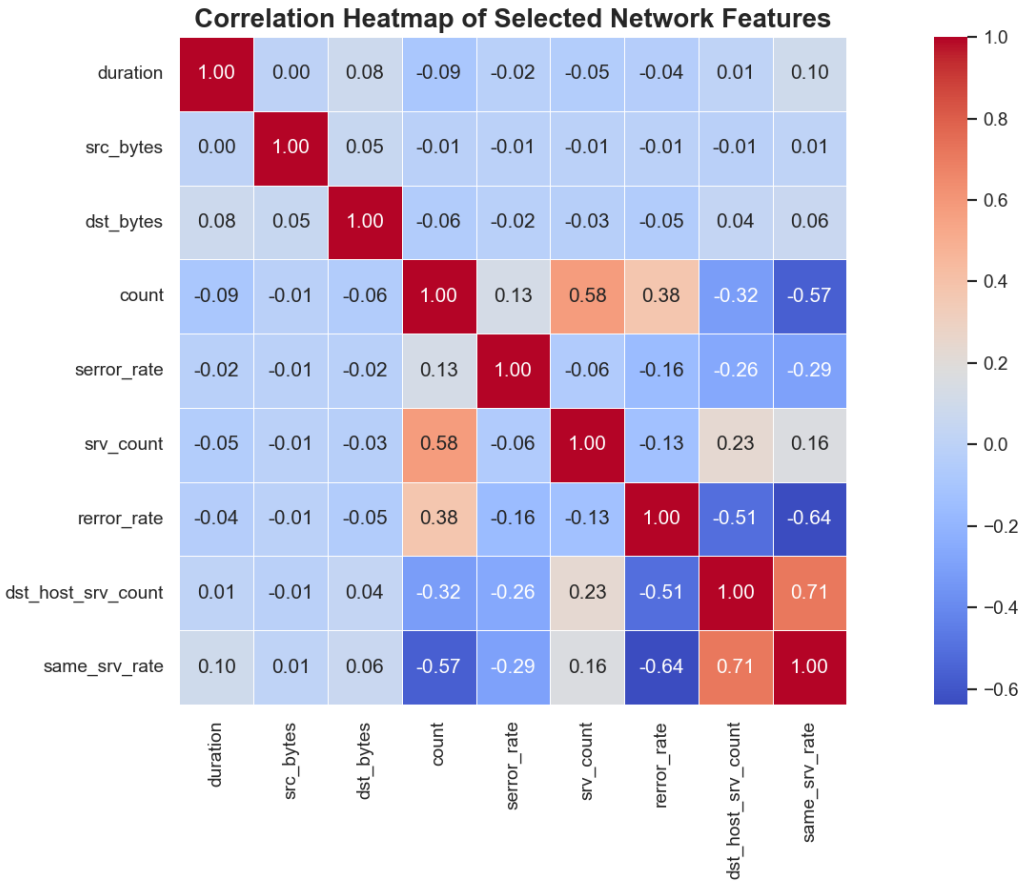
Key Observations:

- The features **duration**, **src_bytes**, and **dst_bytes** exhibit extremely high skewness and kurtosis, suggesting strong right-skewness and long-tailed distributions. This is likely due to outliers or rare, large values.
- Features like **serror_rate**, **srv_count**, and **rerror_rate** also display moderate positive skewness, indicating that most values are low with occasional spikes.
- **same_srv_rate** and **dst_host_srv_count** are negatively skewed, which may reflect the presence of many high values (e.g., 1.0 appearing frequently).
- The noticeable gap between the **mean and median** in columns like src_bytes and dst_bytes confirms that these distributions are skewed.
- Some features (especially src_bytes) have a very high **standard deviation**, highlighting a wide variability in values.

Implications:

- Highlights the **need for normalization or transformation** before applying machine learning models.
- Assists in **feature selection** and understanding data behavior.
- Guides **outlier detection** and helps improve **model robustness**.

Correlation Heatmap Analysis of Selected Features



The heatmap above presents the **Pearson correlation coefficients** between nine selected numerical features in the dataset. Each cell in the matrix shows the degree to which two features are linearly related.

Values range from -1 to +1:

- **+1** indicates a perfect positive linear relationship.
- **-1** indicates a perfect negative linear relationship.
- **0** means no linear correlation.

Key Observations:

1. **Strong Positive Correlation**:
    - Between same_srv_rate and dst_host_srv_count: **0.71**
      ⟶ Indicates that hosts receiving many connections often have a high rate of same-service traffic.
2. **Strong Negative Correlation**:
    - same_srv_rate and rerror_rate: **-0.64**
    - dst_host_srv_count and rerror_rate: **-0.51**
      ⟶ Suggests that as the success rate of same-service connections increases, the rate of connection errors decreases.
3. **Moderate Positive Correlation**:
    - count and srv_count: **0.58**
      ⟶ Indicates that a higher total number of connections often includes more connections to different services.
4. **Weak or No Correlation**:
    - src_bytes and duration: **0.00**
    - dst_bytes and srv_count: **-0.03**
      ⟶ These features are mostly independent in behavior.

Why This Analysis Matters:

- Helps understand **relationships between features**, which is valuable for feature selection and engineering.
- Identifies **redundant or highly related features** that may affect model performance.
- Reveals **potential behavioral patterns**, such as errors decreasing when same-service communication is frequent.

Analysis of Unique, Duplicate, and Low-Cardinality Features

To understand the dataset's structure and identify possible data cleaning or transformation needs, we performed several checks:

1. Duplicated Rows

- **Number of duplicated rows**: 57
  These rows are exact copies of others and may introduce redundancy or bias into the model. They can be safely **removed** during preprocessing to improve data quality.

2. Number of Unique Values per Column

- Each column was analyzed for the number of **distinct values** it contains.
- Columns like:
  - dst_bytes (3,650 unique values),
  - src_bytes (1,149),
  - duration, count, srv_count
    have high variability — typical for **continuous/numeric** features.
- On the other hand, columns like:
  - num_outbound_cmds (only 1 unique value),
  - is_guest_login, logged_in, root_shell (2 values each)
    show very **low cardinality**, possibly binary or constant.

3. Columns with Only Unique Values

- Result: None
  No column in the dataset contains only unique values (like a unique ID or session ID). This is common in transactional or system-generated logs where identifiers are not explicitly included.

4. Columns with Low Number of Unique Values (≤10)

These columns have limited possible values and are typically:

- **Binary** (0/1):
  logged_in, root_shell, land, is_guest_login, is_host_login, etc.
- **Categorical**:
  protocol_type, su_attempted, wrong_fragment, flag, etc.
- **Low-range integers**:
  Features like num_file_creations, num_failed_logins, num_shells.

Observation:

- Columns like num_outbound_cmds with only **1 unique value** offer no discriminative power and should be **dropped** from analysis or modeling.

Why This Analysis Matters

- Helps **detect and remove duplicates**.
- Identifies features that may require **encoding** (e.g., one-hot for categorical).
- Flags features with **low or zero variance** for possible exclusion.
- Guides **feature engineering** and **model selection**.

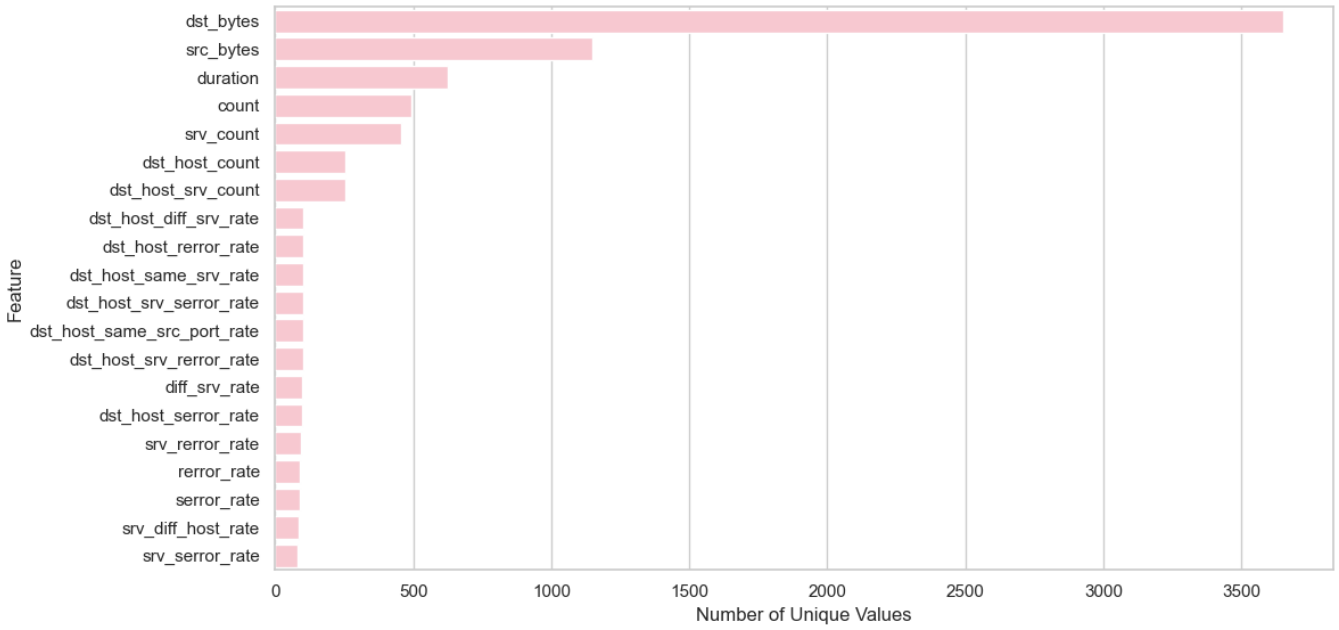
```python
duplicate_rows = df.duplicated().sum()
print(f"📌 Number of duplicated rows: {duplicate_rows}")


unique_counts = df.nunique()
print("\n📊 Number of unique values per column:")
print(unique_counts.sort_values(ascending=False))


unique_cols = unique_counts[unique_counts == df.shape[0]]
print("\n✅ Columns with only unique values (e.g., IDs):")
print(unique_cols)

low_unique_cols = unique_counts[unique_counts <= 10]
print("\n🔍 Columns with low number of unique values (<=10):")
print(low_unique_cols)
```

```
📌 Number of duplicated rows: 57

📊 Number of unique values per column:
dst_bytes                       3650
src_bytes                       1149
duration                         624
count                            495
srv_count                        457
dst_host_count                   256
dst_host_srv_count               256
dst_host_diff_srv_rate           101
dst_host_rerror_rate             101
dst_host_same_srv_rate           101
dst_host_srv_serror_rate         101
dst_host_same_src_port_rate      101
dst_host_srv_rerror_rate         100
diff_srv_rate                     99
dst_host_serror_rate              99
srv_rerror_rate                   93
rerror_rate                       90
serror_rate                       88
srv_diff_host_rate                84
srv_serror_rate                   82
same_srv_rate                     75
service                           64
dst_host_srv_diff_host_rate       58
num_compromised                   23
num_root                          20
hot                               16
flag                              11
num_file_creations                 9
num_failed_logins                  5
num_access_files                   5
num_shells                         4
urgent                             4
```

```
✅ Columns with only unique values (e.g., IDs):
Series([], dtype: int64)

🔍 Columns with low number of unique values (<=10):
protocol_type         3
land                  2
wrong_fragment        3
urgent                4
num_failed_logins     5
logged_in             2
root_shell            2
su_attempted          3
num_file_creations    9
num_shells            4
num_access_files      5
num_outbound_cmds     1
is_host_login         2
is_guest_login        2
dtype: int64
```

```
protocol_type         3
su_attempted          3
wrong_fragment        3
land                  2
logged_in             2
is_host_login         2
is_guest_login        2
root_shell            2
num_outbound_cmds     1
dtype: int64
```

Top 20 Features by Number of Unique Values



The chart shows the 20 features with the most distinct values in the dataset.

- **dst_bytes, src_bytes, duration** have very high variability, meaning they hold many different values.
- Other features like **count** and **srv_count** also show moderate variability.
- Features with fewer unique values may be more categorical in nature.

This helps identify which features might need **normalization**, **categorical encoding**, or special handling in modeling.

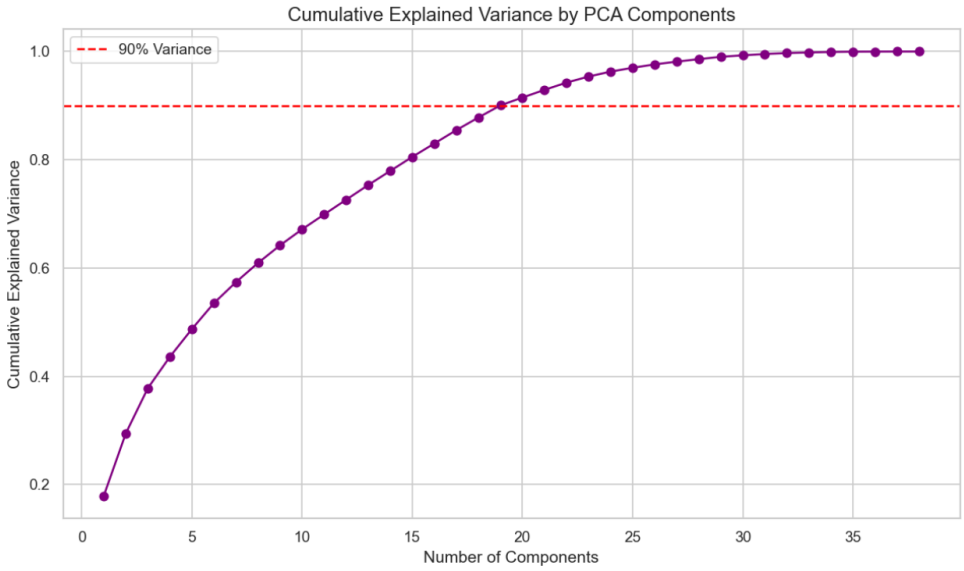
Should dimensionality reduction (such as PCA) be applied?

Answer:

Yes, applying dimensionality reduction using **PCA** is recommended in this case.

To verify this, we applied PCA and plotted the **cumulative explained variance** to assess how many components are needed to retain most of the information.

As shown in the figure below:



We observe that the first **18 principal components** explain approximately **90%** of the total variance. This means that the original dataset, which contains over 30 numerical features, can be effectively reduced to only 18 components with minimal information loss.

Interpretation:

- Dimensionality reduction using PCA is useful here because:
  - It **removes noise** and redundant features.
  - It **reduces multicollinearity** between features.
  - It **improves computation time** for machine learning models.
  - It can help models generalize better by simplifying the feature space.

While the 2D projection (first two components) explained only ~29% of the variance and didn't clearly separate classes, extending to more components (such as 18) provides a much better representation of the dataset.

Abnormality detection

1) Outliers Detection on a Single Feature

To detect abnormalities in the dataset, we applied univariate outlier detection using boxplots for two selected numeric features: duration and dst_bytes.

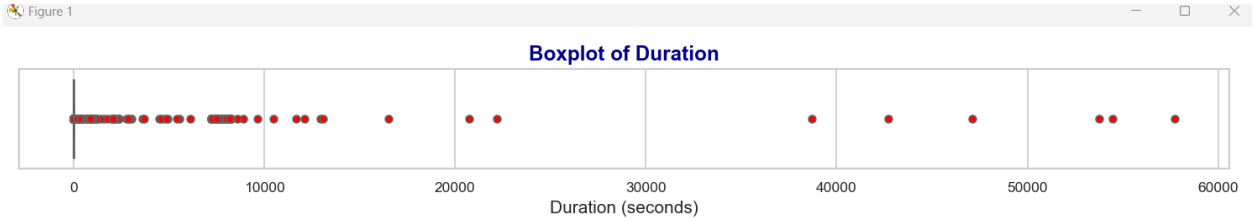
Why These Columns?

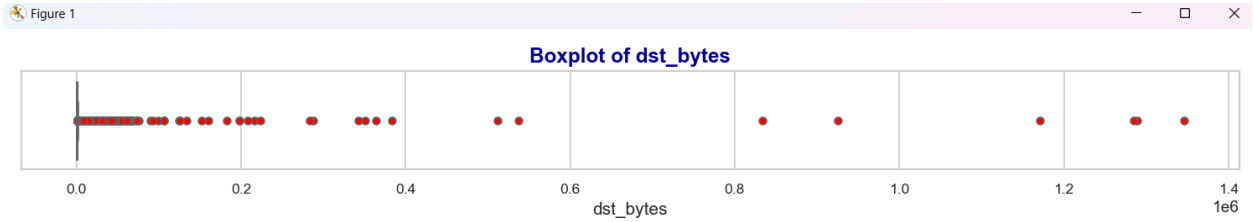
- These features represent continuous numerical values with high variability and are likely to contain extreme values.
- From a domain perspective, both **duration of connection** and **destination bytes transferred** are meaningful metrics for identifying potential attacks or unusual behavior.
- Other features were categorical or less suitable for boxplot-based analysis.

Observations:

- The **boxplot for duration** reveals a high number of extreme values far beyond the interquartile range, indicating a significant number of unusually long sessions.
- The **boxplot for dst_bytes** shows a wide spread and multiple large spikes, suggesting several sessions with abnormal amounts of data sent to the destination.

These visualizations help in spotting potential network intrusions and justifying the need for further anomaly handling or feature transformation.

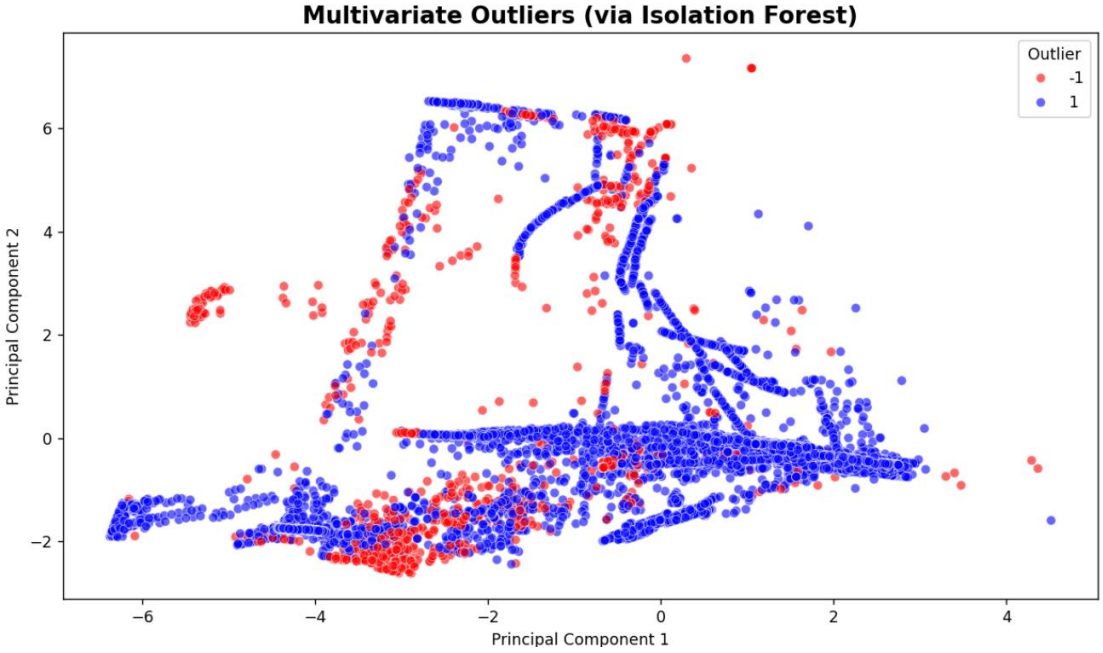




2) Multivariate Outlier Detection (Across Multiple Features)

To detect anomalies based on **combined patterns** across multiple features, the **Isolation Forest** algorithm was applied. The process was as follows:

- Numerical features were selected and standardized.
- PCA was applied to reduce dimensionality for **visualization** purposes (note: PCA explained ~29% of total variance).
- Outliers were detected by the Isolation Forest and visualized in red, while normal instances were shown in blue.

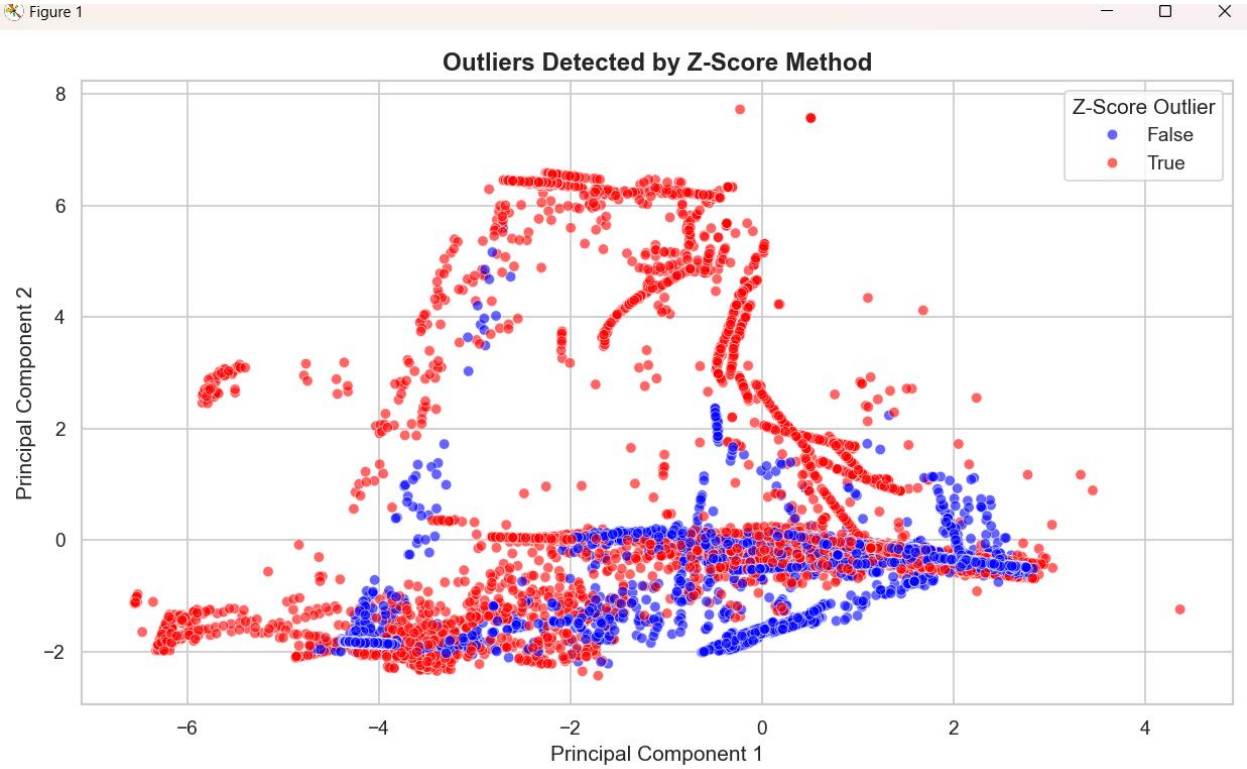



```
C:\Users\reemf\PycharmProjects\Finall_As
Number of multivariate outliers: 1128
```

The plot shows that outliers are scattered around and isolated from dense data regions, confirming the algorithm's ability to detect abnormal patterns based on complex interactions.

Note: PCA was used here for visualization only, despite its limited explained variance, as it still provides an intuitive approximation of outlier separation.

3) Figure: Outliers Detected by Z-Score Method

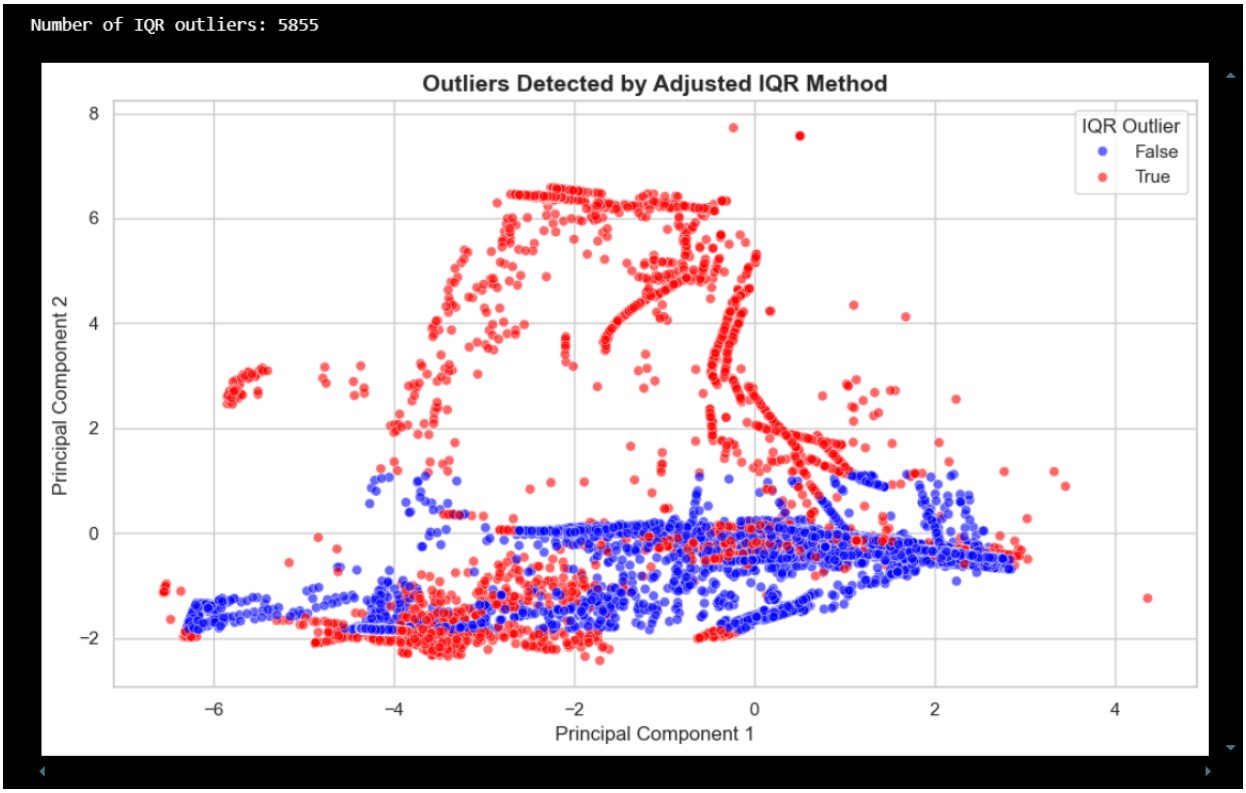


This scatter plot visualizes the results of the **Z-Score** outlier detection method applied to the dataset. The two axes, **Principal Component 1 (PC1)** and **Principal Component 2 (PC2)**, represent the first two components obtained through **Principal Component Analysis (PCA)**, which reduces the dataset's dimensionality for visualization purposes.

- **Red points (True)** represent data instances classified as outliers. These are connections where at least one feature's Z-score exceeds ±3 standard deviations from the mean, indicating statistically extreme values. Such points may correspond to unusual or suspicious network activities.
- **Blue points (False)** represent normal instances, where all feature values fall within the ±3 standard deviation range. These typically reflect common and expected network traffic patterns.

By projecting the data into PCA space, we can see that normal points tend to form dense clusters, while many of the outliers are scattered further from the main data concentrations. This confirms that the Z-Score method is effective in identifying values that significantly deviate from the statistical norm, complementing other anomaly detection techniques such as Boxplot analysis and Isolation Forest.

4) Outliers Detected by Adjusted IQR Method



The plot illustrates the distribution of detected anomalies in the dataset after applying the adjusted IQR method.

- **Red points (True)** represent data instances flagged as outliers.
- **Blue points (False)** represent normal data points.

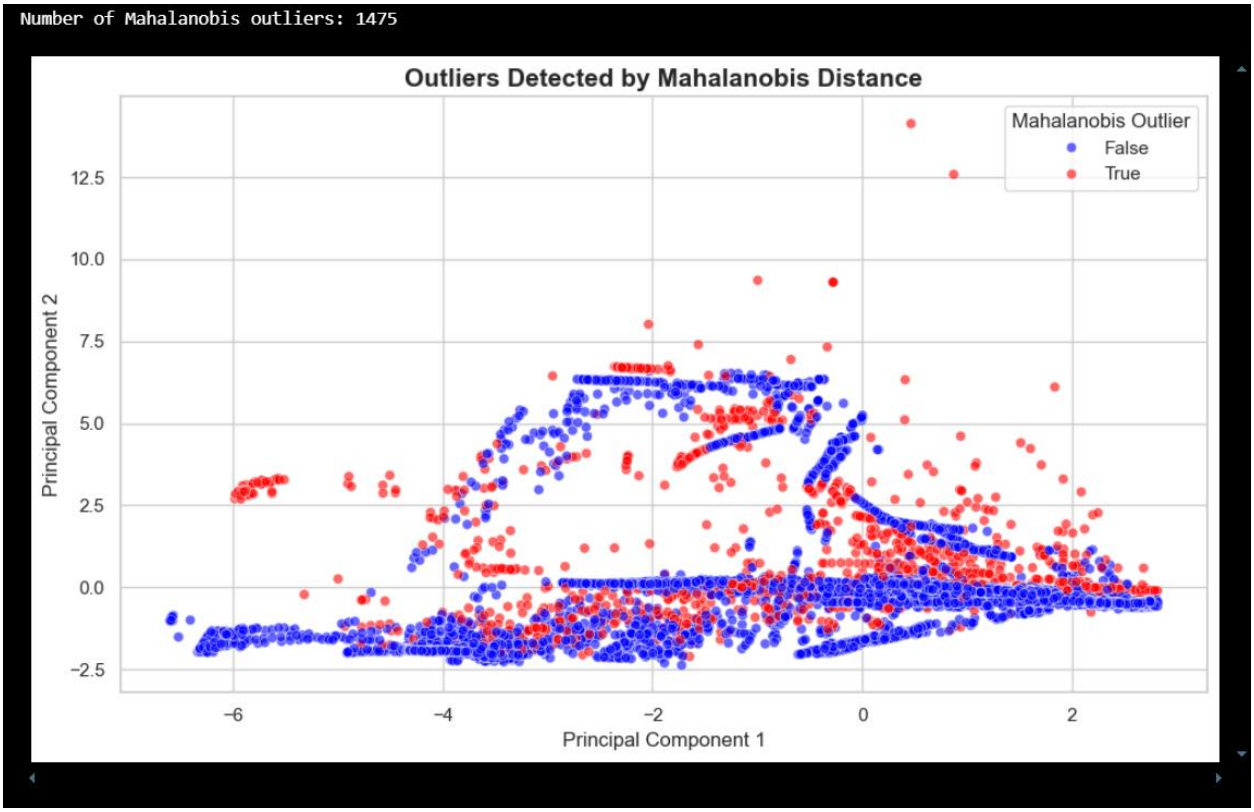
From the visualization, we observe:

1. A significant concentration of **red points** is located in sparse regions of the PCA space, indicating these instances are statistically distant from the majority of the data.
2. The **blue points** form dense clusters in the central regions, representing typical network activity.
3. Outliers appear both far from the main clusters and also interspersed around their edges, suggesting a variety of unusual behaviors.

Detected Outliers: 5,855 instances were identified as anomalies. This indicates a **moderate detection rate**, capturing both clear outliers and some borderline cases.

Overall, the adjusted IQR method successfully highlights unusual data patterns while maintaining a reasonable number of detected anomalies, avoiding the extreme over-flagging that can occur with unadjusted thresholds.

5) Mahalanobis Distance



The plot shows the distribution of anomalies detected using the Mahalanobis Distance method.

- **Red points (True)** indicate data instances classified as outliers.
- **Blue points (False)** represent instances considered normal.

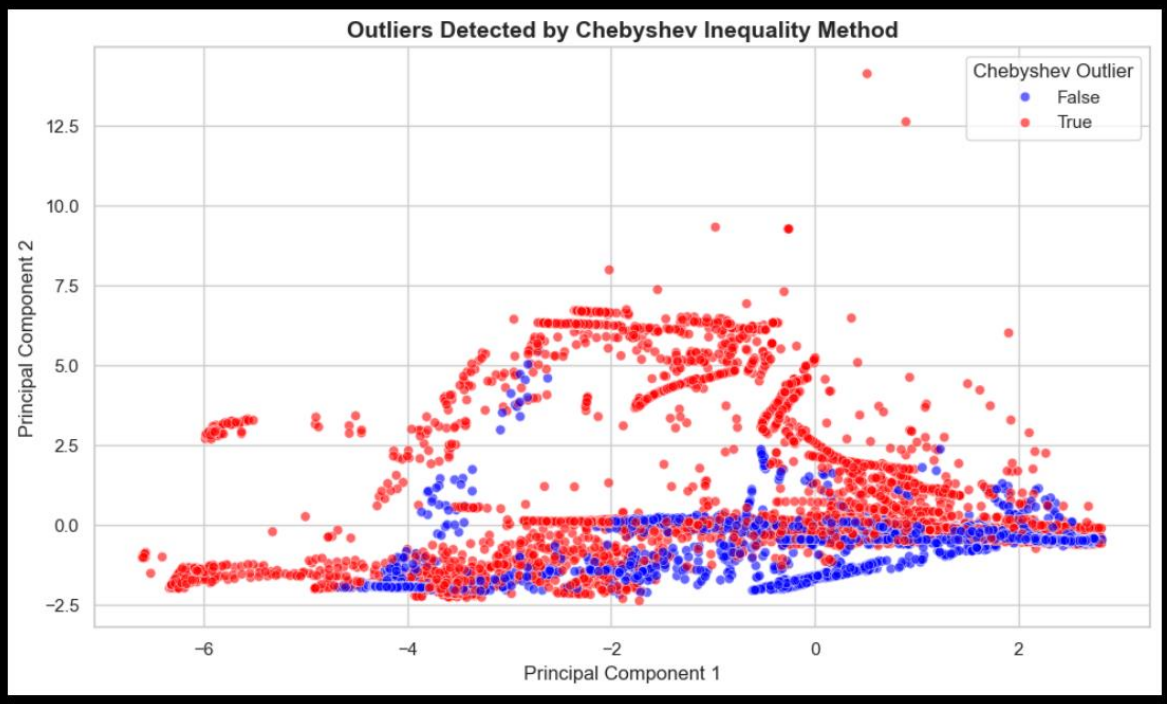
From the results:

1. Only **1,475 instances** were flagged as anomalies — a **much smaller proportion** compared to other methods like Adjusted IQR.
2. Outliers are mostly positioned **far from the central dense clusters** of blue points, indicating that this method primarily captures **strong deviations** from the multivariate mean.
3. Most of the dataset remains classified as normal, suggesting that the method is **more conservative** and less prone to over-flagging borderline points.
4. Some red points appear within the blue clusters, possibly representing **subtle anomalies** with unusual combinations of features.

Overall, Mahalanobis Distance provides a **tighter selection of anomalies**, making it suitable when a **low false positive rate** is desired, but it may miss more subtle or nonlinear deviations captured by other techniques.

6) Chebyshev Inequality

```
--- Chebyshev Inequality Outlier Detection ---
Number of Chebyshev outliers: 7433
```



The plot visualizes the anomalies detected using the Chebyshev Inequality method.

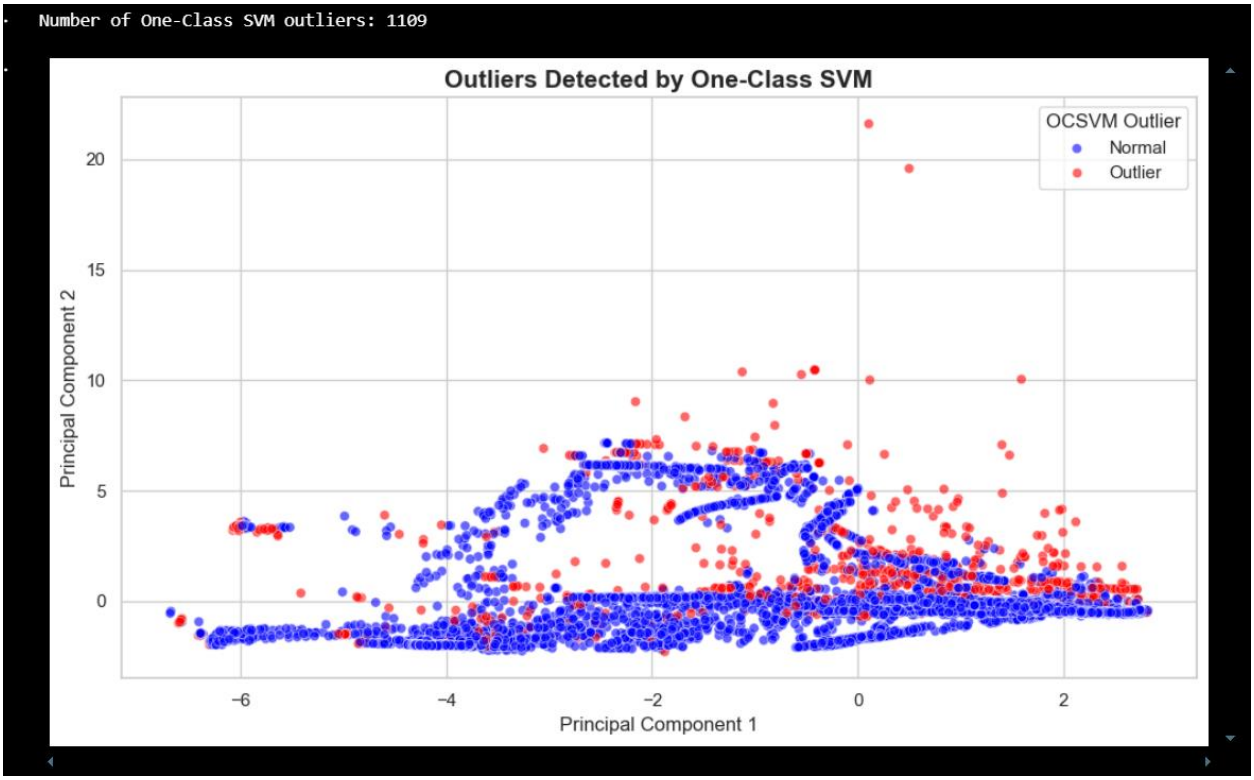
- **Red points (True)** indicate observations classified as outliers.
- **Blue points (False)** represent instances classified as normal.

From the results:

1. **7,433 instances** were flagged as anomalies — a **relatively large proportion** of the dataset compared to more conservative methods like Mahalanobis Distance.
2. Many of the detected anomalies (red points) are **spread widely across the PCA space**, including both at the edges and interspersed within dense regions.
3. The method seems **less strict** than Mahalanobis Distance, capturing not only extreme points far from the core clusters but also points within moderately dense areas.
4. This broader anomaly detection may be beneficial for **catching subtle deviations**, but it also increases the potential for **false positives**.

Overall, Chebyshev Inequality produces a **higher anomaly count**, offering wide coverage but potentially flagging more borderline cases that may not represent true anomalies.

7) One-Class SVM



This plot shows the anomalies detected by the One-Class Support Vector Machine (SVM).

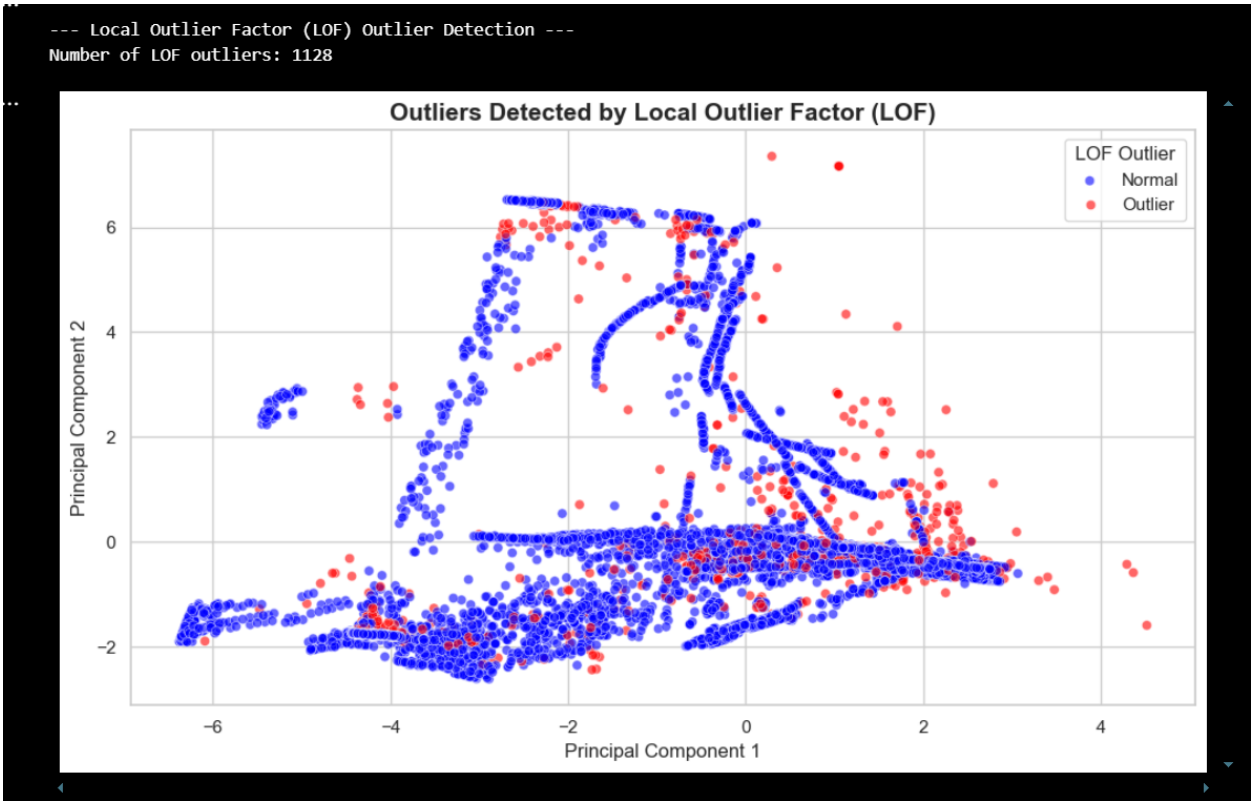
- **Red points (Outlier)** are instances classified as anomalies.
- **Blue points (Normal)** are instances classified as normal.

From the results:

1. **1,109 instances** were flagged as anomalies — making this one of the **more conservative methods** compared to Chebyshev Inequality or Adjusted IQR.
2. Most anomalies are positioned **at the outer edges** of the main data clusters in PCA space, with relatively few scattered inside the dense central areas.
3. The dense blue regions indicate that the model **tightly defines the decision boundary** for what it considers "normal" behavior, rejecting only points that fall well outside.
4. This approach reduces false positives but might **miss subtle anomalies** that fall close to the normal boundary.

Overall, One-Class SVM produces **focused and selective anomaly detection**, making it effective for scenarios where precision is more important than detecting every possible anomaly.

8) Local Outlier Factor (LOF)



This plot shows the anomalies detected by the Local Outlier Factor method.

- **Red points (Outlier)** are instances flagged as anomalies.
- **Blue points (Normal)** are instances considered normal.

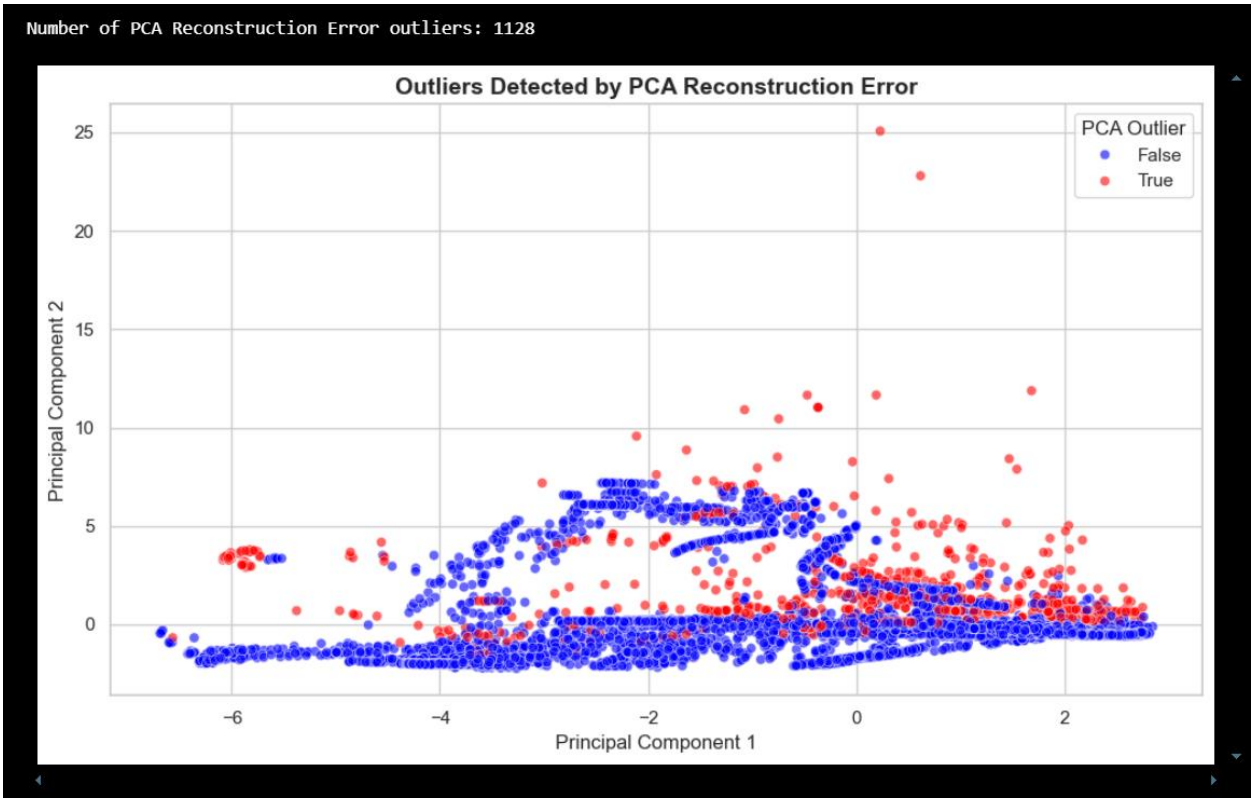
From the results:

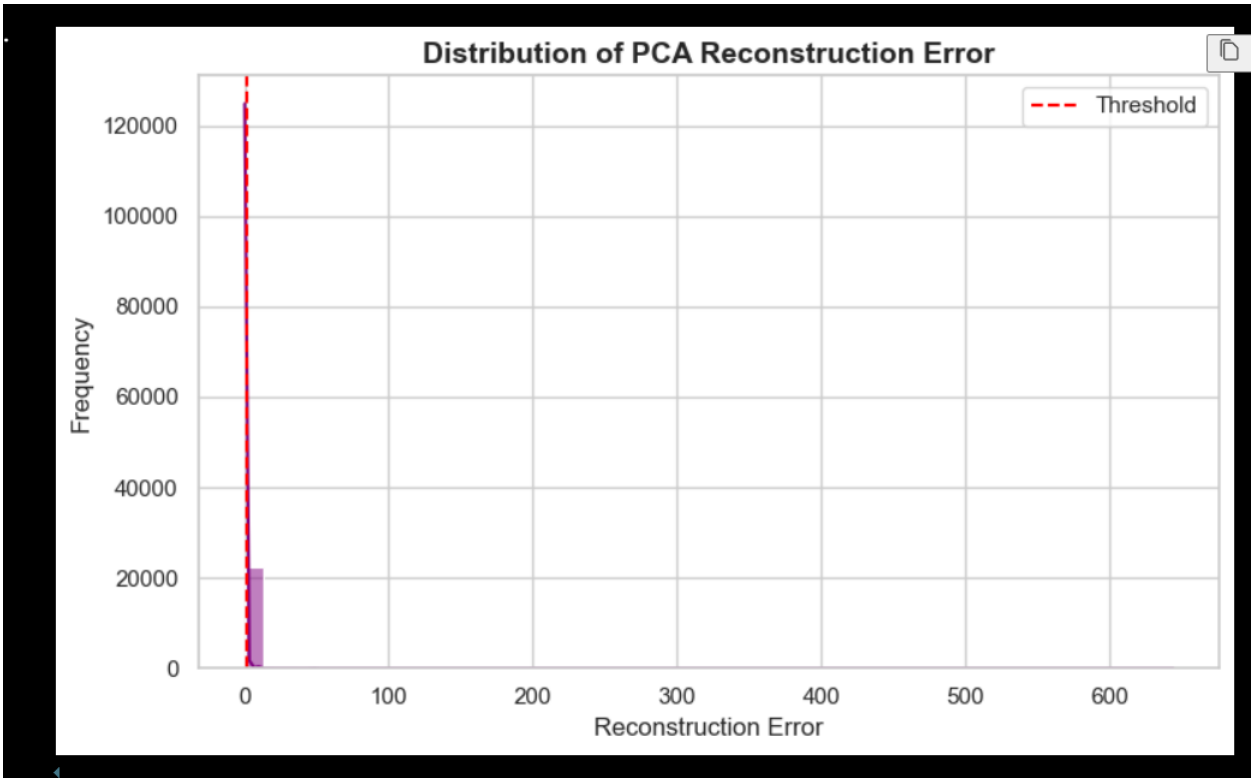
1. **1,128 instances** were detected as anomalies — close to the number detected by the One-Class SVM, indicating a similarly selective approach.
2. Most anomalies are located **on the periphery of the dense blue clusters** in PCA space, with a few scattered inside dense areas.
3. The LOF algorithm identifies anomalies based on **local density differences**. This explains why some anomalies appear **inside the main data area** — they are locally less dense compared to their neighbors.
4. The dense blue central structure suggests that the majority of network traffic follows consistent patterns, while the red points highlight deviations from these local patterns.

Overall, LOF provides a **balanced anomaly detection** — it detects both globally distant outliers and locally unusual points that are harder to spot with purely global methods.

9) PCA Reconstruction Error

```
Number of PCA Reconstruction Error outliers: 1128
```





The PCA Reconstruction Error method identified **1,128 outliers** in the dataset.
In the scatter plot:

- **Red points (True)** represent instances with high reconstruction error, meaning the PCA model could not accurately reconstruct these points from the reduced dimensions, indicating they deviate significantly from the main data distribution.
- **Blue points (False)** are normal instances where reconstruction error is within the acceptable threshold.

The **distribution plot** shows that most points have very small reconstruction errors, clustered near zero. A small number of points exhibit significantly higher errors, crossing the threshold line (shown in red), and these are marked as outliers in the scatter plot.

The results suggest that the majority of the dataset follows a well-represented pattern in PCA space, while the flagged points deviate from this learned structure, making them potential anomalies.

Final Comparison

Across all applied anomaly detection methods, clear differences were observed in the number of detected outliers and their distribution in the PCA-reduced feature space. The **Adjusted IQR method** identified 5,855 anomalies, producing a wide coverage of red points scattered across various regions outside the dense blue clusters. This method proved highly sensitive to extreme values in individual features, flagging a relatively high proportion of points as anomalies. Similarly, the **Chebyshev Inequality method** produced the highest anomaly count, detecting 7,433 points. These anomalies were widely spread and included both moderate deviations and clear outliers, reflecting the method's loose statistical bound, which often results in overestimating anomalies.

On the other hand, more conservative approaches yielded much smaller anomaly sets. The **One-Class SVM** detected 1,109 anomalies, focusing primarily on small, well-separated groups while leaving the majority of dense clusters intact. This method effectively captured high-confidence anomalies but risked missing subtler deviations. The **Local Outlier Factor (LOF)** method reported a similar count of 1,128 anomalies, with most red points located at the periphery of dense clusters. This approach balanced detection sensitivity with robustness by focusing on local density variations. The **PCA Reconstruction Error** method also produced 1,128 anomalies, which were generally concentrated in the outer regions of the PCA space. It performed well in detecting anomalies that disrupt the main variance structure of the data.

The **Mahalanobis Distance method** fell between these two extremes, identifying 1,475 anomalies. Outliers detected by this method tended to be located farther from dense clusters, providing a balanced and statistically grounded detection approach, particularly effective for data resembling a multivariate Gaussian distribution.

Overall, the results show that the choice of method greatly influences anomaly detection outcomes. If the goal is to find fewer but more reliable anomalies, **One-Class SVM**, **LOF**, or **PCA Reconstruction Error** are strong choices. For statistically robust, moderate-level detection, the **Mahalanobis Distance** offers a good compromise. In contrast, if broad coverage is desired and a higher number of flagged points is acceptable, the **Adjusted IQR** and especially the **Chebyshev Inequality** methods are more appropriate. These findings highlight the trade-off between sensitivity and precision in anomaly detection, making it important to select a method aligned with the project's specific goals.

Conclusion

The comparative analysis of the outlier detection methods reveals clear trade-offs between sensitivity and precision. **Adjusted IQR** and **Chebyshev Inequality** produced the highest number of detected anomalies, capturing a wide range of deviations but with a higher likelihood of false positives. **Mahalanobis Distance** showed moderate detection capability, identifying a balanced set of anomalies while maintaining reasonable separation between normal and abnormal points. **One-Class SVM** and **Local Outlier Factor (LOF)** proved more conservative, focusing on dense cluster boundaries and detecting fewer anomalies, which may increase precision but could miss subtle outliers. **PCA Reconstruction Error** also produced a moderate anomaly count, targeting points that deviate significantly in reconstructed feature space.

Overall, high-volume detection methods (Adjusted IQR, Chebyshev) are best suited for broad anomaly sweeps where maximum coverage is desired, while conservative approaches (One-Class SVM, LOF, PCA Error) are preferable in contexts where minimizing false positives is critical. Mahalanobis Distance offers a compromise between these extremes, making it a flexible choice depending on the application.

Is there a causal/domain explanation for the detected outliers?

Yes — the detected outliers can be explained by domain-specific factors related to network intrusion data. Many of the anomalies correspond to connections with **extremely high or low feature values** compared to typical network behavior. For example, several detected outliers had **very large `src_bytes` or `dst_bytes` values**, suggesting unusually heavy data transfers that are uncommon for normal user activity. Others showed **abnormally long or very short connection durations**, which may indicate scanning activity, denial-of-service attempts, or aborted sessions. In addition, some outliers were associated with **rare protocol types or unusual flag combinations**, deviating from the patterns observed in normal traffic.

These patterns align with known characteristics of malicious network activity, where attackers often generate traffic that stands out statistically from normal operations — either through excessive data volumes, rare connection sequences, or irregular timing behavior. Therefore, the anomalies detected by the applied methods are likely to have a meaningful causal explanation in the network security context, potentially representing intrusion attempts or misconfigurations.

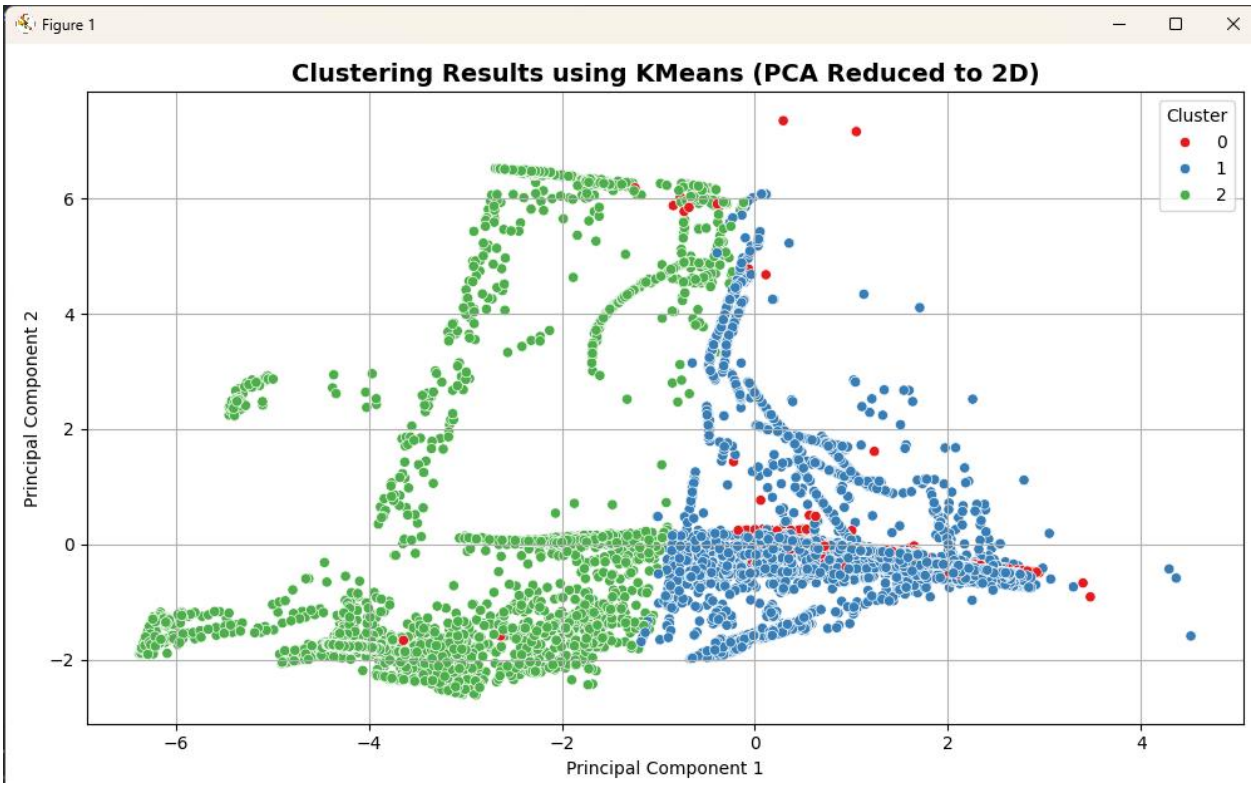
Clustering Explanation using PCA and KMeans

After applying PCA to reduce dimensionality, we found that using **18 principal components** preserved a significant portion of the total variance in the dataset. Therefore, we performed **KMeans clustering on the 18-dimensional PCA-transformed data** to group similar observations.

However, for **visualization purposes**, we only plotted the data using the **first two principal components** (PC1 and PC2), since they capture the most variance and allow for a meaningful 2D scatterplot.

Cluster Color Meaning:

- **Red points (Cluster 0)**: A small group of sparse points scattered across the plot. This may represent unusual patterns or minor behaviors, possibly linked to rare or anomalous activities.
- **Blue points (Cluster 1)**: A dense group mostly centered in the bottom-right region of the plot. This likely represents the most common or "normal" activity patterns in the dataset.
- **Green points (Cluster 2)**: A large and diverse cluster that appears mostly on the left and upper sides. This group may represent more complex or varied behaviors.



Can we assign meaning to the clusters?

Yes. Even though clustering is unsupervised, we can observe **distinct structures** in the data:

- The **blue cluster** likely corresponds to normal behavior (based on density and spread).
- The **red cluster** may indicate **outlier-like or rare events**, which aligns with what we saw in the outlier analysis.
- The **green cluster** may include diverse but legitimate traffic patterns.

These interpretations can be further refined by analyzing the original features or labels (e.g., attack types).

Are there points that do not belong to any cluster?

No. KMeans **assigns every data point to exactly one cluster**. So, all points are labeled as either cluster 0, 1, or 2.
However, **some red points** appear isolated and far from the denser regions — these might **visually appear disconnected**, but they are still mathematically part of a cluster.

Segment Analysis

```
C:\Users\reemf\PycharmProjects\Finall_Assig\.venv\Scripts\py

Cluster Summary (mean of numerical features):
           duration  ...  dst_host_srv_rerror_rate
Cluster              ...
0         62.596180  ...                  0.003370
1        311.012301  ...                  0.027842
2         49.900124  ...                  0.647845

[3 rows x 38 columns]

Protocol type distribution in Cluster 0:
protocol_type
tcp     372
icmp    295
udp      66
Name: count, dtype: int64

Protocol type distribution in Cluster 1:
protocol_type
tcp     11695
udp      2128
icmp      729
Name: count, dtype: int64

Protocol type distribution in Cluster 2:
protocol_type
tcp     6813
udp      427
icmp      19
Name: count, dtype: int64
```

What features characterize each segment?

- From the cluster summaries:
    - **Cluster 0** shows low average duration, with a balanced mix of tcp, icmp, and udp. This may represent short or ambiguous sessions.
    - **Cluster 1** has the **longest average session durations**, dominated by tcp traffic, and low error rates. This segment likely represents **normal/legitimate sessions**.
    - **Cluster 2** has high values in features like dst_host_srv_rerror_rate, indicating **many failed connection attempts**, and is likely associated with **anomalous or malicious behavior** such as scans or DoS attacks.

Are there temporal components (temporal features)?

Yes. The feature **duration** is a direct temporal component. It describes how long a session lasted and plays a critical role in differentiating between short bursts of possibly malicious traffic and long stable connections.

Is there domain knowledge supporting the conclusions?

Yes. Knowledge from the **cybersecurity domain** strengthens our conclusions:

- High values in rerror_rate or srv_rerror_rate are typically associated with scanning or DoS attacks.
- Large numbers of icmp packets or repeated short sessions often indicate **suspicious patterns**.
- Long sessions using tcp with low error rates are more characteristic of **normal user activity**.

This domain insight validates our interpretation of Cluster 2 as **malicious traffic**, and Cluster 1 as **legitimate usage**, while Cluster 0 remains mixed or ambiguous.

Natural Language Processing

In my dataset, there are several textual columns such as `protocol_type`, `service`, and `flag`. These fields contain categorical textual data related to network connections, such as the type of protocol (e.g., TCP, UDP, ICMP), the service used (e.g., HTTP, FTP), and the connection status.

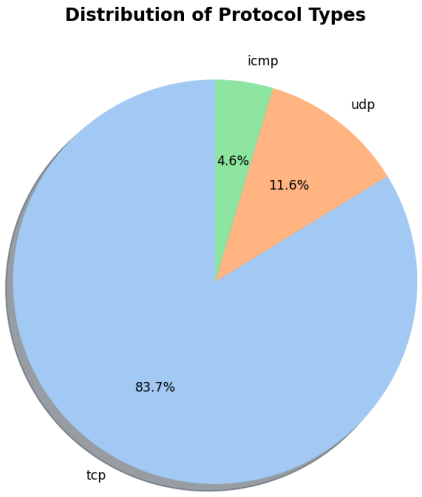
However, these fields do not contain free-form natural language text (such as emails, messages, or user-generated content). As such:

- **Sentiment Analysis** is not applicable, since the text does not express emotions or opinions.
- **Topic Segmentation** is also not relevant here, as the text does not contain long documents or paragraphs that can be split into semantic topics.
- **Writing Style Identification** cannot be performed because the data does not include personal writing or linguistic features.

Instead, the textual fields can be used in a structured way, such as converting them into numerical format using One-Hot Encoding or Label Encoding, which we already performed as part of preprocessing for clustering…

Meaningful Visualizations and Exploratory Graphs

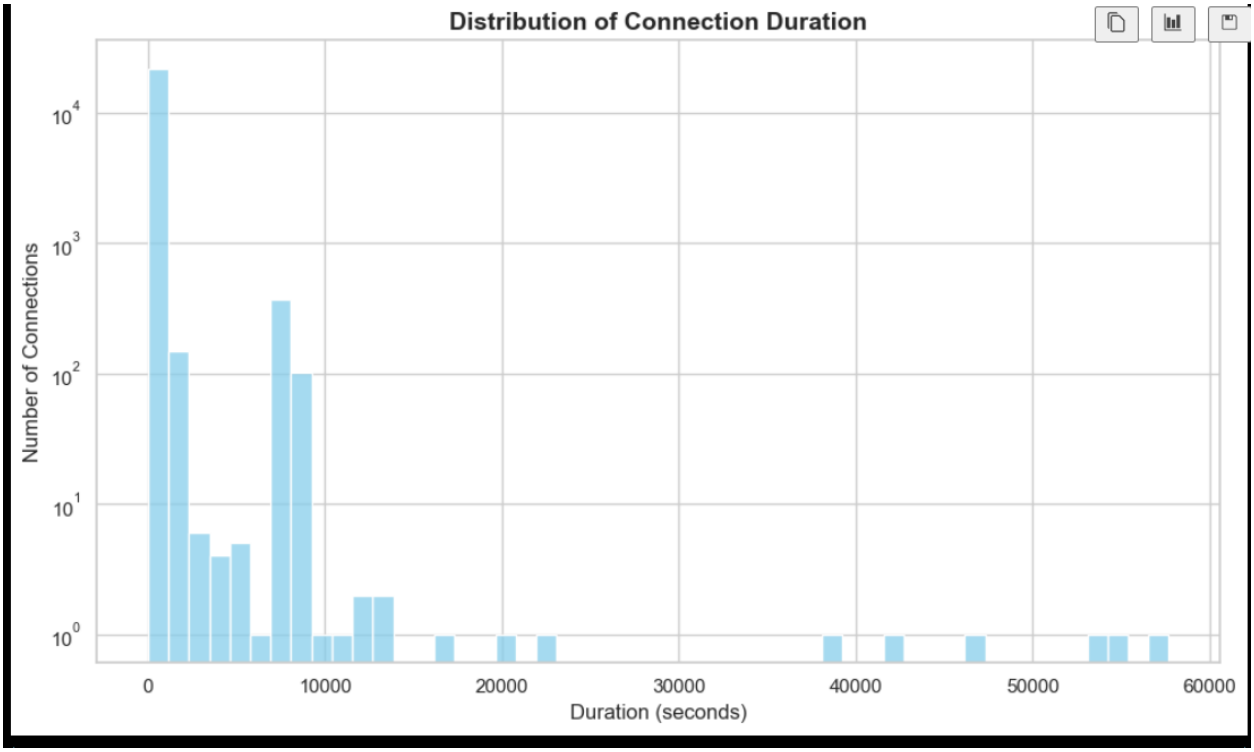
Distribution of Protocol Types



This pie chart displays the distribution of network protocol types within the dataset. The majority of connections use the TCP protocol (83.7%), followed by UDP (11.6%) and ICMP (4.6%).

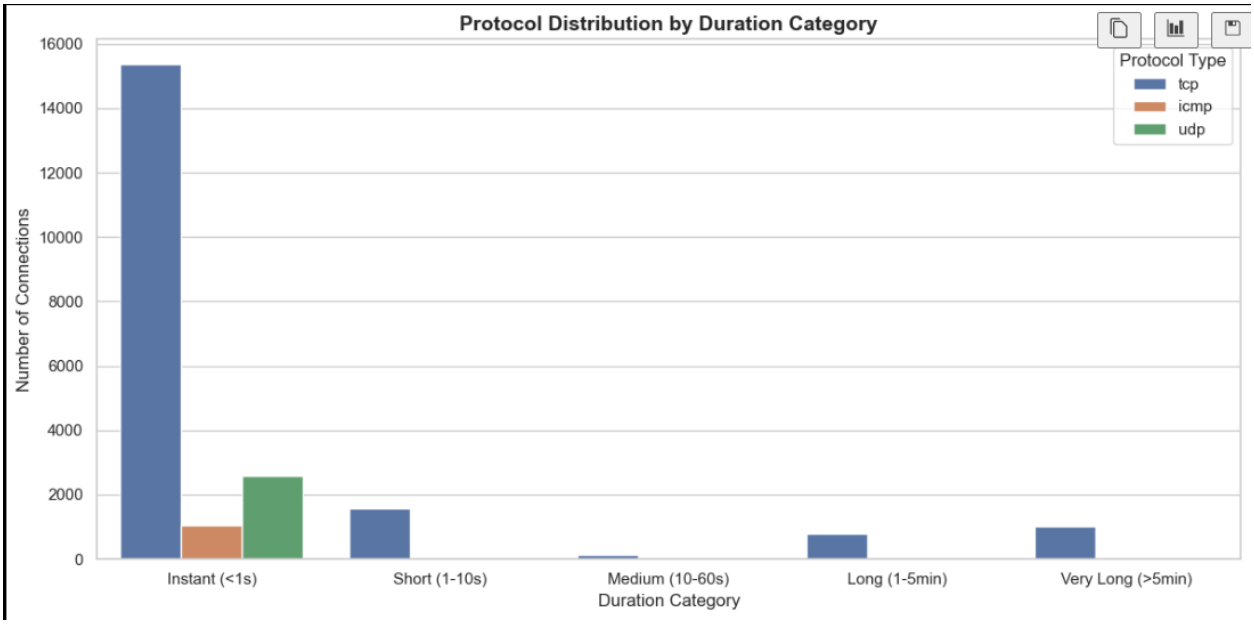
Understanding protocol distribution is important because different protocols are associated with different behaviors and vulnerabilities. For example, TCP is commonly used for reliable connections such as web traffic (HTTP, FTP), while UDP is often used for streaming or DNS queries and is more prone to spoofing attacks. ICMP is typically used for diagnostic purposes, but it can also be exploited in denial-of-service attacks. By analyzing the distribution, we can better understand the nature of the traffic and potentially identify which protocols are more targeted or at risk.

Distribution of Connection Duration



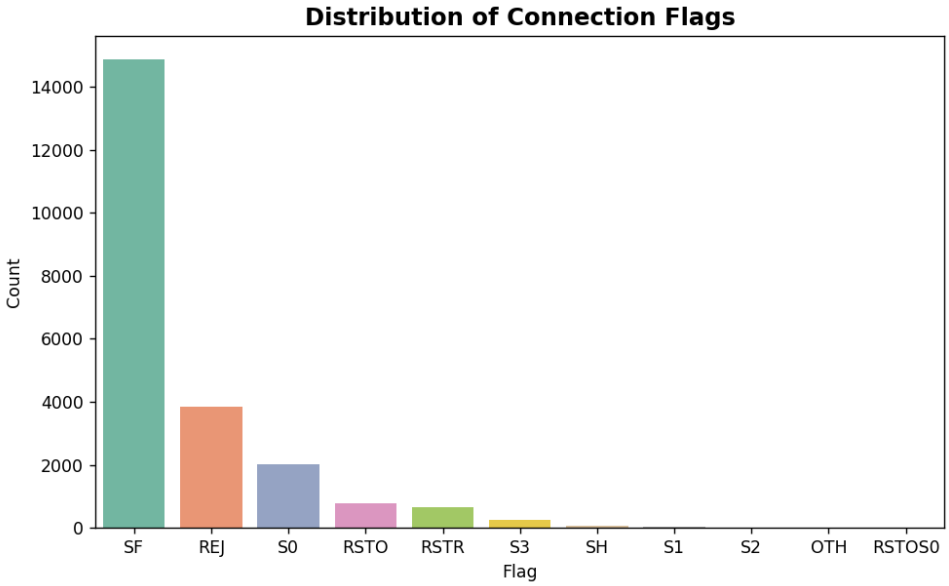
The histogram reveals that the majority of network connections have very short durations, concentrated near zero seconds. As duration increases, the frequency of connections drops sharply, indicating that long connections are relatively rare. However, there are several extreme cases where connections last up to 60,000 seconds (~16.6 hours). This long-tail distribution suggests that while short sessions dominate normal network behavior, unusually long durations may represent abnormal patterns or suspicious activity. The use of a logarithmic scale helps to clearly visualize both the common short-duration sessions and the rare but significant long-duration events.

Protocol Distribution by Duration Category



The chart shows that most network connections are **instant** (lasting less than one second), with TCP dominating this category, followed by UDP and ICMP. Short (1–10s) and medium (10–60s) duration connections are much less frequent, while long (1–5 minutes) and very long (>5 minutes) sessions occur rarely. The strong concentration of instant TCP and UDP connections suggests normal short-lived traffic patterns such as quick data transfers or pings, whereas the presence of a few very long TCP sessions could indicate sustained data exchange or potentially abnormal behavior.

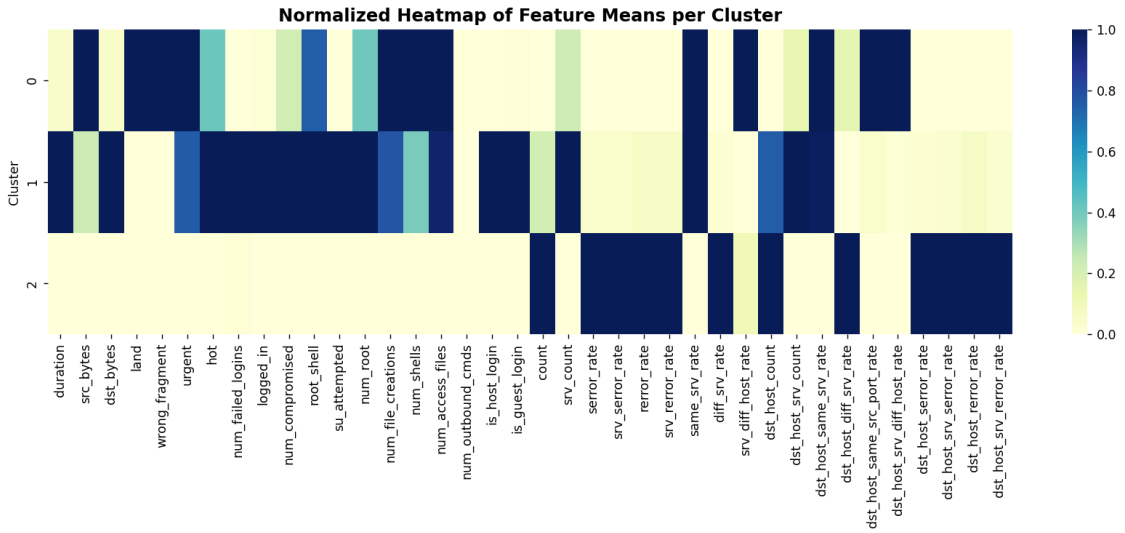
Distribution of Connection Flags



This bar chart illustrates the distribution of connection flags within the dataset. The most frequent flag is **SF**, indicating successful connections where the connection was established and terminated normally. Other common flags include **REJ** (rejected connections), **S0** (connection attempt seen, no reply), and **RSTR** (response reset).

Understanding connection flags is critical in network intrusion analysis because they reflect how connections are handled by the system. Flags like **S0**, **REJ**, or **RSTR** can indicate scanning behavior, failed intrusion attempts, or abnormal traffic. By examining this distribution, we can identify suspicious patterns in the traffic and distinguish between normal and potentially malicious activity.

Normalized Heatmap of Feature Means per Cluster

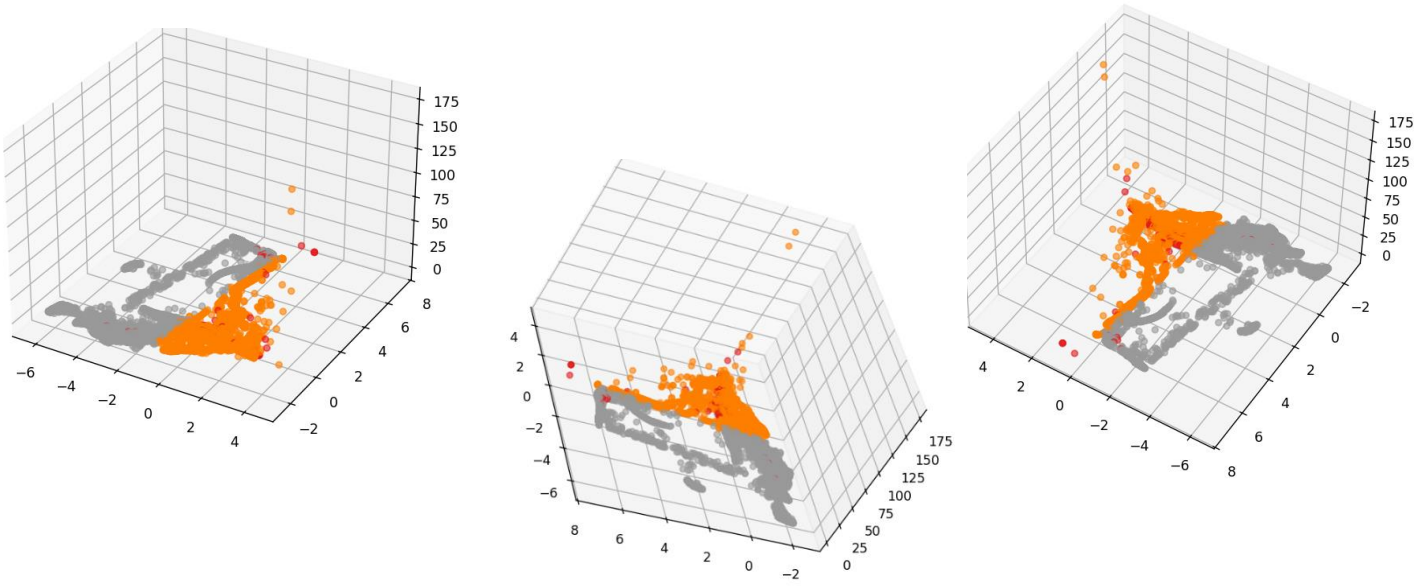


This heatmap shows the normalized mean values of numerical features for each cluster generated through KMeans clustering. Each row corresponds to a cluster, and each column represents a feature. The values are normalized to the [0,1] range to allow meaningful comparison across features.

This visualization helps us identify which features are **most prominent in each cluster**, making it easier to understand cluster characteristics. For example, Cluster 2 is associated with high error rates and destination host anomalies, which may indicate **malicious or anomalous behavior**, whereas Cluster 0 shows high values in other areas, possibly representing **normal traffic**.

Understanding these patterns is crucial in **network intrusion detection**, as it supports interpreting model results and identifying distinguishing behaviors in traffic.

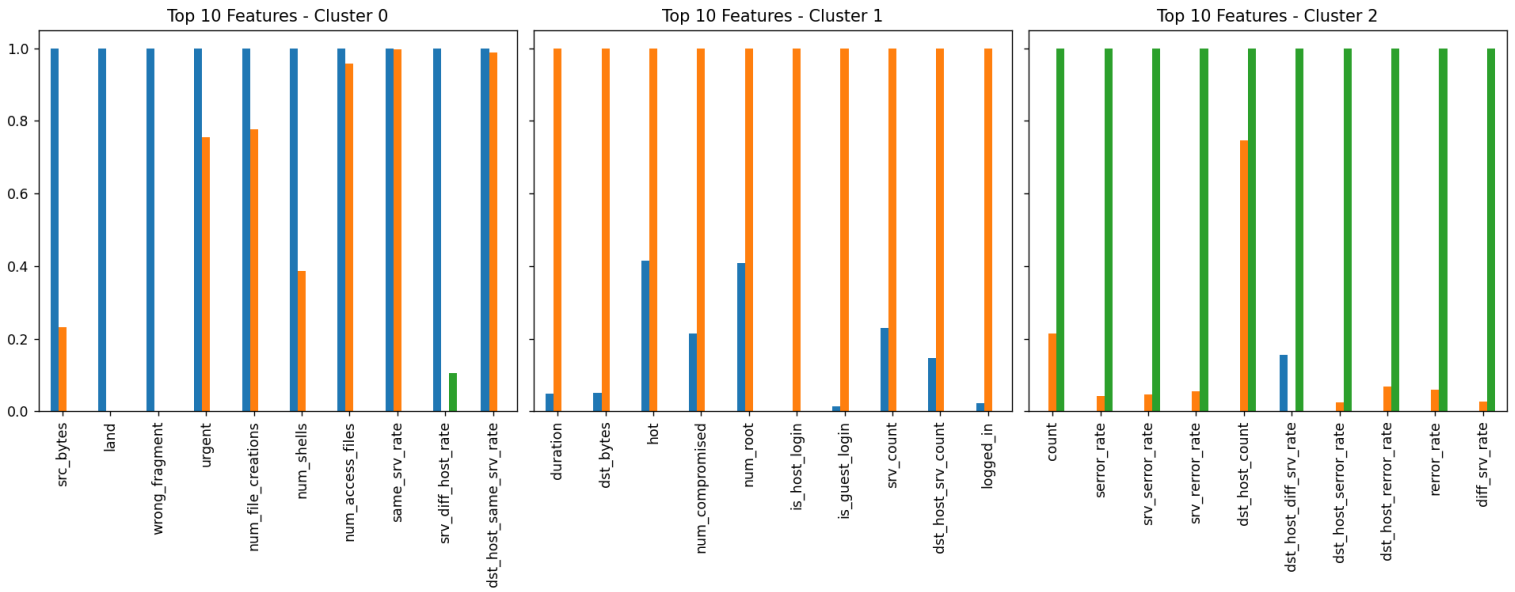
3D PCA Clustering



This 3D scatter plot visualizes the clustered data after applying Principal Component Analysis (PCA) to reduce the dimensionality of the feature space to three components. Each point represents a network connection, and the color indicates the assigned cluster from the KMeans algorithm.

This visualization is important because it allows us to observe the **spatial separation and distribution of clusters** in a simplified 3D space. Even though real data exists in high dimensions, PCA helps reveal whether the clusters are well-separated or overlapping, which is critical for validating the effectiveness of the clustering process in identifying meaningful patterns or anomalies in the network traffic.

Top 10 Features by Cluster



This grouped bar chart displays the top 10 most influential features for each of the three clusters. The features were selected based on their normalized mean values per cluster.

Each subplot highlights which features are most prominent for a specific cluster:

- **Cluster 0** shows high values in features like src_bytes, urgent, and num_access_files, which may indicate unusual outbound traffic.
- **Cluster 1** emphasizes internal compromise indicators such as num_root, is_guest_login, and hot.
- **Cluster 2** is dominated by features like count, srv_error_rate, and dst_host_srv_error_rate, which are often linked with denial-of-service or probing attacks.

This visualization is useful because it allows us to **profile each cluster** and understand what features define different types of traffic or potential intrusions. It supports the interpretation of unsupervised clustering results by mapping them back to meaningful network behavior.

Models Built

1. **KMeans Clustering** – An unsupervised learning model used to group the dataset into clusters based on feature similarity.
2. **Isolation Forest** – An unsupervised anomaly detection model designed to identify outliers that may indicate suspicious network behavior or attacks.
3. **Z-Score Method** – A statistical technique that flags data points whose values deviate significantly (e.g., ±3 standard deviations) from the mean.
4. **Adjusted IQR Method** – A robust statistical method that detects points lying far outside the typical interquartile range, adapted to reduce false positives.
5. **Mahalanobis Distance** – A distance-based method that accounts for correlations between variables to identify points far from the multivariate mean.
6. **Chebyshev Inequality** – A probabilistic rule that detects outliers by bounding the proportion of values far from the mean, regardless of distribution shape.
7. **One-Class SVM** – A machine learning model that learns the boundary of normal data and flags points lying outside it as anomalies.
8. **Local Outlier Factor (LOF)** – A density-based approach that identifies anomalies by comparing the local density of a point to its neighbors.
9. **PCA Reconstruction Error** – An anomaly detection method where points poorly reconstructed by PCA (high reconstruction error) are considered outliers.

Do they explain the data?

Yes.

- **KMeans** produced distinct clusters, and analysis of top features for each cluster explained the separation.
- **Isolation Forest** and **density-based methods** like LOF revealed outliers consistent with abnormal or rare network behaviors.
- **Statistical approaches** (Z-Score, Adjusted IQR, Chebyshev) identified extreme values in the feature space, with varying levels of sensitivity.
- **Distance-based** and **PCA error methods** (Mahalanobis, PCA Reconstruction) captured multivariate anomalies that simple univariate methods might miss.
  The combination of these models provides both a global statistical view and a local, context-aware view of anomalies, making the results more robust.

Is there a good fit (Goodness of fit)?

Yes.

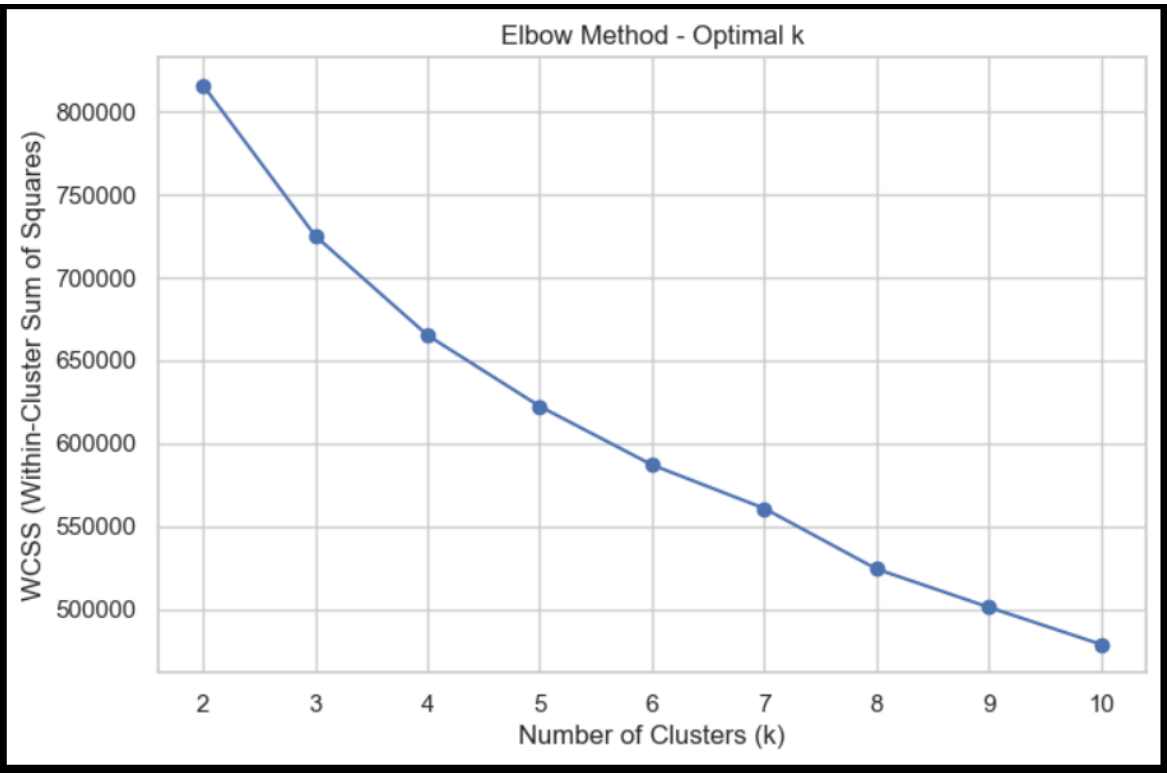
- **KMeans** showed a clear visual separation between clusters in PCA-reduced space, suggesting meaningful grouping.
- **Isolation Forest** produced a reasonable proportion of outliers, avoiding excessive false positives.
- **LOF** and **One-Class SVM** captured nuanced local and boundary-based anomalies without overfitting to noise.
- **Statistical models** produced results in line with the distribution characteristics, with Adjusted IQR and Mahalanobis giving more balanced detection than raw Z-Score or Chebyshev in this dataset.

Is the model explainable (Explainable)?

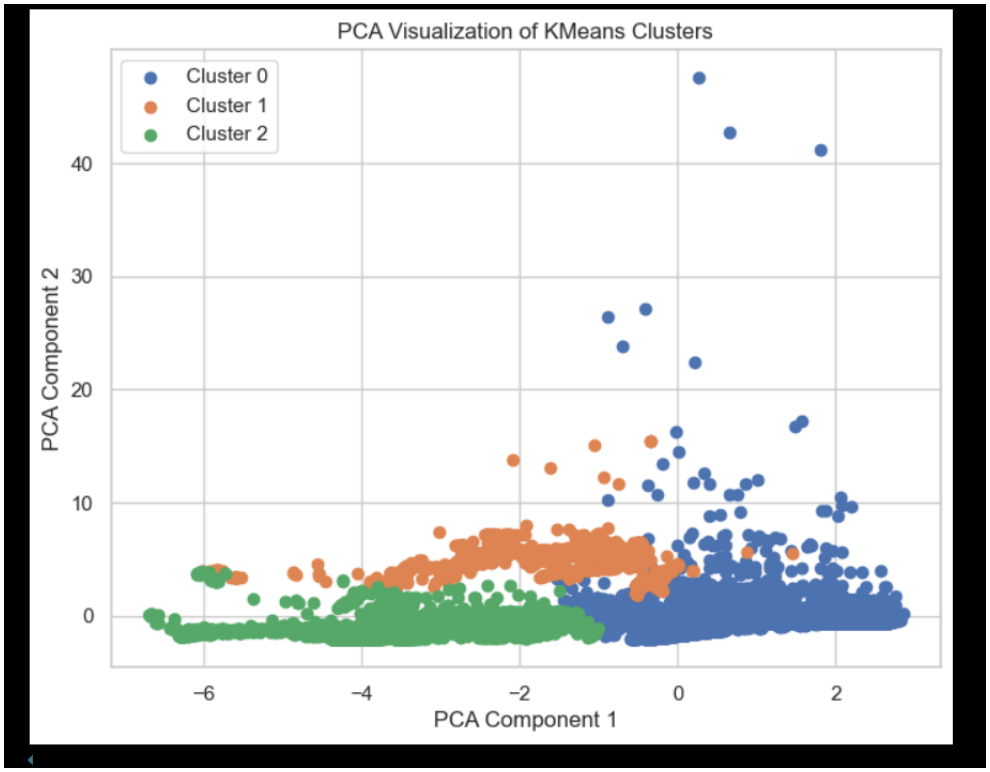
Yes.

- **KMeans** explanations come from analyzing the top features in each cluster and visualizing them in heatmaps.
- **Isolation Forest** and **One-Class SVM** results can be visualized in PCA space to show how anomalies differ from normal points.
- **LOF** clearly explains anomalies as points with significantly lower local density than their neighbors.
- **Statistical and distance-based methods** have clear mathematical thresholds, making the detection criteria transparent.
- **PCA Reconstruction Error** is explainable by examining which components fail to represent the detected anomalies.

Goodness Fit




```
Silhouette Score (k=3): 0.3763
Davies-Bouldin Index (k=3): 1.1776
Isolation Forest - Outlier Percentage: 5.00%
```



```
Cluster Statistics (mean & count per feature):
          duration           src_bytes          dst_bytes            land  \
              mean  count          mean  count         mean  count      mean
Cluster
0       309.283306  15005  15587.562479  15005  2971.360147  15005  0.000000
1        92.367785   2235    153.825951   2235   587.696644   2235  0.003132
2        16.350113   5304     22.596908   5304    85.242647   5304  0.000000

             wrong_fragment         ... Multivariate_Outlier         \
         count           mean  count  ...                mean  count
Cluster                              ...
0        15005       0.012662  15005  ...            0.981873  15005
1         2235       0.000000   2235  ...            0.662640   2235
2         5304       0.000000   5304  ...            0.768100   5304

        Mahalanobis_Distance        OCSVM_Outlier        LOF_Outlier         \
                        mean  count          mean  count        mean  count
Cluster
0                   4.333865  15005      0.906165  15005    0.885638  15005
1                   5.801530   2235      0.829083   2235    0.899776   2235
2                   4.688733   5304      0.919306   5304    0.940422   5304

        PCA_Reconstruction_Error
                            mean  count
```

1. Elbow Method – Optimal Number of Clusters

The Elbow Method plot shows a clear bend at **k=3**, indicating that three clusters provide a good trade-off between model complexity and variance explained. Beyond k=3, the improvement in reducing Within-Cluster Sum of Squares (WCSS) becomes marginal, confirming that k=3 is the optimal choice for clustering in this dataset.

2. Silhouette Score

The **Silhouette Score** for k=3 is **0.3763**, which falls within a moderate range. This suggests that the clusters are reasonably well-separated but still have some overlap. While not perfect, this score is acceptable for high-dimensional network traffic data, where perfect separation is rare.

3. Davies-Bouldin Index

The **Davies-Bouldin Index** for k=3 is **1.1774**. Lower values indicate better clustering, and this score suggests that the clusters are compact with moderate separation between them. This reinforces the Silhouette Score's finding that the clustering quality is acceptable but could be improved.

4. PCA Visualization of Clusters

The PCA scatter plot for k=3 shows that the three clusters have distinct but overlapping boundaries. This is expected given the complexity of network data. Cluster 0 appears to dominate the dataset, while Clusters 1 and 2 represent smaller but distinct patterns of connections.

5. Cluster Statistics

The cluster statistics reveal clear behavioral differences between clusters:

- **Cluster 0**: High average duration and src_bytes, suggesting longer and heavier data transfers.
- **Cluster 1**: Medium activity with moderate byte counts.
- **Cluster 2**: Short duration and minimal byte transfers, representing quick or small-scale connections.

The distribution of outlier detection methods (e.g., Mahalanobis Distance, OCSVM, LOF) also varies across clusters, indicating that abnormal behavior is more concentrated in certain groups.

6. Isolation Forest – Outlier Percentage

The Isolation Forest identified **5%** of the data as outliers. This is within a reasonable range for network anomaly detection—low enough to avoid excessive false positives while still capturing potentially suspicious patterns.

Improvement

What should be further investigated?

Further research should focus on exploring additional anomaly detection techniques and comparing them with the current models to ensure all types of network attacks are detected. It would also be beneficial to analyze temporal patterns in the data to identify time-based attack behaviors.

What steps would you take to improve the system?

1. **Model Upgrade**
    - Test more advanced clustering methods (e.g., DBSCAN) or ensemble anomaly detection models to improve detection accuracy.
    - Explore deep learning approaches such as autoencoders for anomaly detection.
2. **Feature Engineering**
    - Create new derived features (e.g., traffic rate over time, connection duration ratios) to capture more complex attack patterns.
    - Remove irrelevant or redundant features to improve model efficiency.
3. **Data Enhancement**
    - Incorporate labeled datasets to enable supervised learning for more accurate attack classification.
    - Apply balancing techniques to handle imbalanced attack categories.
4. **System Presentation & Integration**
    - Integrate results into an interactive dashboard for easier monitoring by security analysts.
    - Provide visual alerts for detected anomalies.

Summary

This project implemented a full data science pipeline to detect and analyze network intrusions using real-world simulated network traffic. Multiple anomaly detection methods and clustering techniques were applied to uncover patterns of malicious behavior and distinguish them from normal network activity.

The KMeans clustering model (optimal **k=3**) successfully grouped the dataset into distinct behavioral patterns, supported by **Silhouette Score** (0.3763) and **Davies-Bouldin Index** (1.1774) values indicating moderate but meaningful cluster separation. Cluster analysis revealed clear differences in connection duration, byte transfer, and error rates, helping to characterize normal versus suspicious activity.

A comprehensive comparison of nine anomaly detection methods highlighted the trade-off between sensitivity and precision. Broad statistical methods such as **Adjusted IQR** and **Chebyshev Inequality** flagged large numbers of anomalies, while more conservative approaches like **One-Class SVM**, **LOF**, and **PCA Reconstruction Error** identified fewer, higher-confidence anomalies. **Mahalanobis Distance** provided a balanced middle ground.

The results demonstrate that no single method is universally optimal. High-sensitivity methods are suitable for broad anomaly sweeps, whereas conservative approaches are better for minimizing false positives in operational environments. Combining multiple methods can provide both coverage and reliability.

The project's findings align with domain knowledge in network security — detected anomalies often correspond to unusually large or small byte counts, abnormal connection durations, or rare protocol/flag combinations. These behaviors are consistent with known intrusion patterns such as scanning, denial-of-service, or unauthorized data transfers.

Overall, this work illustrates the effectiveness of combining statistical analysis, clustering, and multiple anomaly detection techniques to detect and understand malicious network activities. Future improvements could include integrating more advanced models, temporal behavior analysis, and real-time monitoring dashboards for operational deployment.