Delaney Caceres
Cynthia Di

Time-series Anomaly Detection Methods for Unlabeled Data

Goal:
Our goal was to explore how different unsupervised methods can discover abnormal behavior in network time-series data in unlabeled datasets. Additionally, we observed how different forms of models (tree-based vs clustering) affect what is detected as "anomalous". The models we evaluated were Isolation Forest, K-means, and DBSCAN models.

Link to Dataset: (Instructions to get data on last page)
https://drive.google.com/drive/folders/1YFE8zgne7KqoHoNRiiSGd_ONOV5fbd4t?usp=sharing

Note:
There are two notebooks, project.ipynb and Anomaly_Detection.ipynb. Anomaly_Detection.ipynb is where all the code is, the main notebook. Project.ipynb was for running the data separately when google drive branches didn't work, however in project.ipynb, the optimal k for k-means was k=6, while in Anomaly_Detection.ipynb, it's k=2

Preprocessing the Data

Initial Cleanup

Our dataset contains 6,867,538 rows with 13 features. All features were numerical, consisting of 4 int64 integers and 9 floats. The first thing we did was an initial data cleanup, handling missing values and duplicates.

The code to handle missing values worked like this:

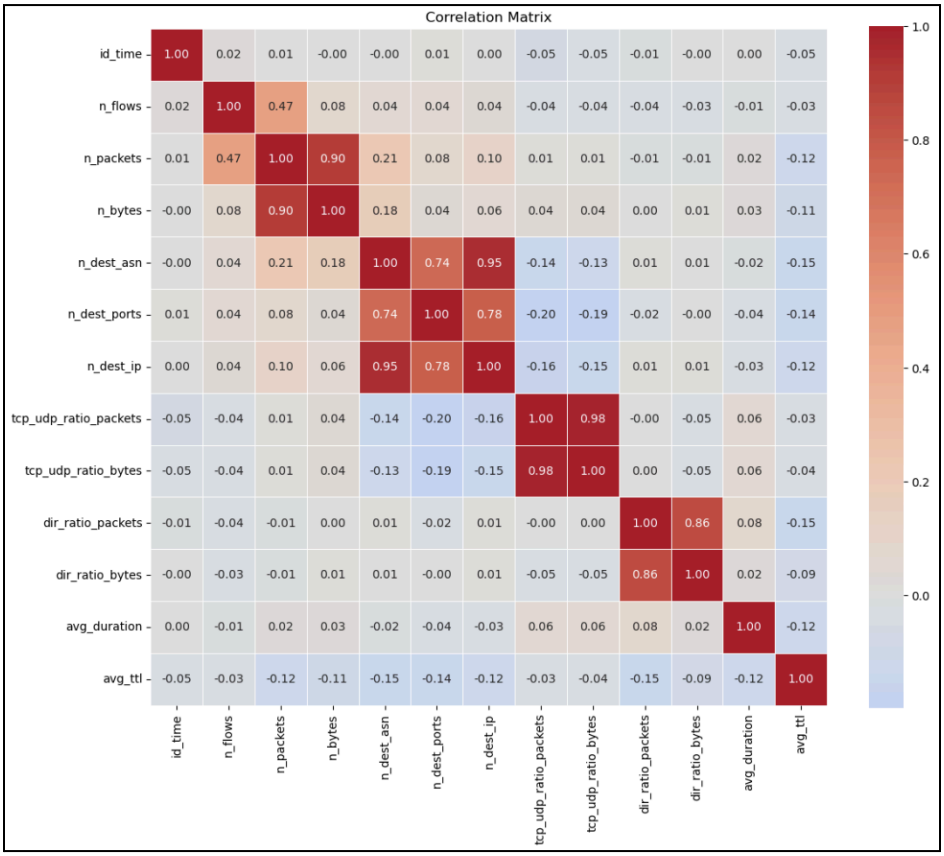
```python
missing_cols = df.columns[df.is_null().any()].tolist()

#for each column with missing values:
if numerical: fill with median
if categorical: fill with mode (or unknown if no mode)
```

The essential idea was to use the median to fill in numerical missing values since it would be robust to outliers and preserve data distribution. Mode was used for categorical data since the most frequent value maintains data patterns. For cases with no mode, "Unknown" was used. This step is key for preventing model failure from any NaN values, maintaining dataset integrity, and also implementing different strategies for different data types just in case one didn't work out for another.

For duplicates:

```python
duplicates = df.duplicated().sum()
df = df.drop_duplicates()
```

If any duplicates exist, they were immediately found and removed. Duplicate rows were also eliminated. This way, we could prevent any bias from forming in the model, ensuring each data point is unique, which is particularly important for time series to avoid overrepresentation.



Correlation Matrix

A correlation matrix analysis was conducted to examine the relationships between the 13 numerical features in the dataset. The analysis revealed significant multicollinearity, with four feature pairs demonstrating strong correlations ($|r| > 0.8$). Specifically, n_packets and n_bytes showed a correlation of 0.90, indicating that packet count and byte volume are inherently related measures of traffic volume. Similarly, n_dest_asn and n_dest_ip were highly correlated at 0.95, suggesting that the number of destination autonomous systems closely corresponds with the number of destination IP addresses in network traffic patterns.

Two additional high-correlation pairs involved protocol and direction ratios. The TCP/UDP ratio measurements for packets and bytes were nearly identical with a correlation of 0.98, while directional ratios for packets and bytes showed substantial overlap with a correlation of 0.86. These strong correlations indicate redundancy in feature measurements, where multiple metrics capture essentially the same underlying network characteristics.

To address this multicollinearity and reduce feature dimensionality while preserving interpretability, we chose to remove one feature from each highly correlated pair based on interpretability and network

relevance. Packet-based ratios were prioritized over byte-based ratios for network focused analysis, byte volume was retained over packet count as a more informative traffic measure, and IP count was kept over ASN count for greater granularity in destination diversity. This process reduced the feature space from 13 to 9 dimensions, eliminating redundancy while maintaining critical network traffic characteristics essential for anomaly detection.

Feature Selection

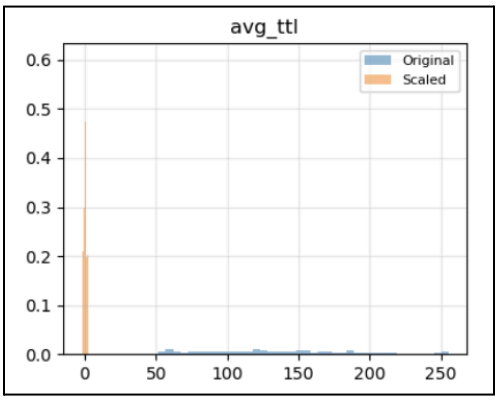
The resulting nine selected features each provide distinct network traffic characteristics essential for anomaly detection:

- **id_time:** serves as the temporary index for time-series analysis
- **n_flows:** represents connection count, indicating network activity level
- **n_bytes:** measures data volume transferred, crucial for detecting traffic floods or data exfiltration
- **n_dest_ports** and **n_dest_ip:** captures destination diversity, useful for identifying scanning activity or distributed attacks
- **tcp_udp_ratio_packets:** reveals packet protocol distribution patterns that may shift during certain attacks
- **dir_ratio_packets:** indicates traffic directionality, potentially showing inbound/outbound patterns
- **avg_duration:** represents connection longevity, which may shorten during scanning or lengthen during data exfiltration
- **avg_ttl:** average time to live provides insight into network path characteristics that may change

Together, these nine features provide a comprehensive yet non-redundant view of network behavior suitable for anomaly detection.

Feature Scaling and Normalization

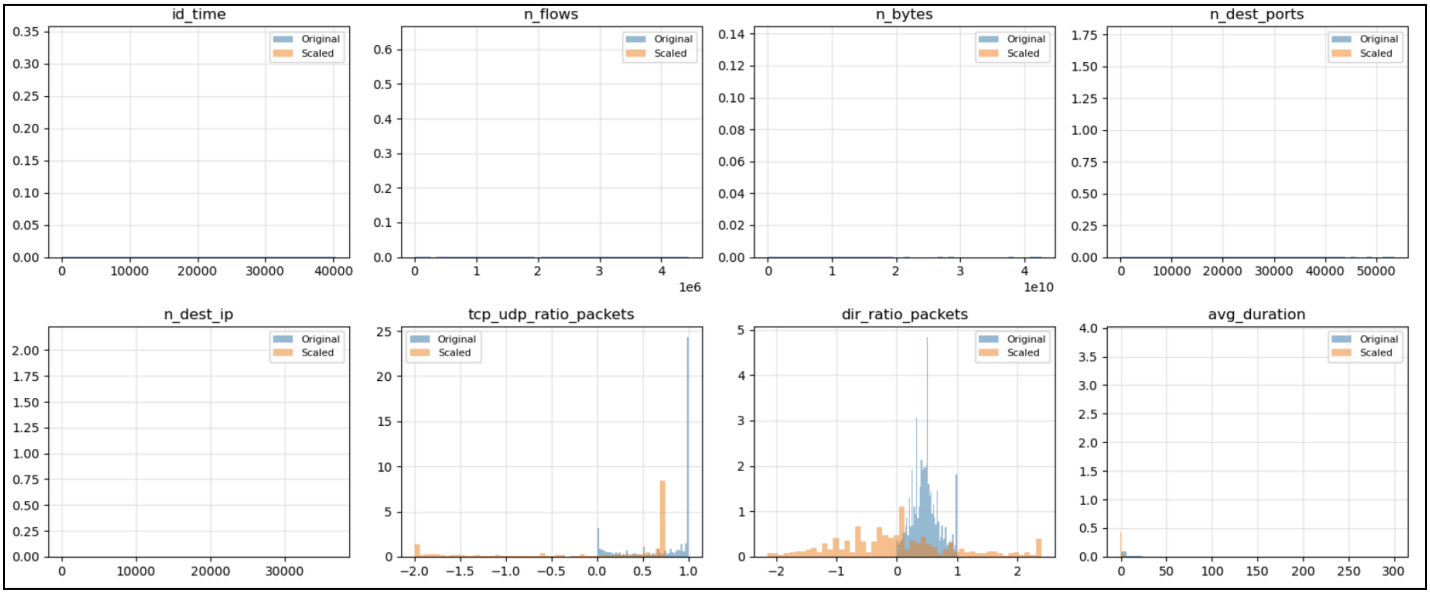
For the K-means and DBSCAN algorithms, StandardScaler was used to normalize features to zero mean and unit variance, providing optimal performance for these distance-based methods. However, for Isolation Forest - a tree based algorithm specifically designed for anomaly detection - RobustScaler was selected instead. This choice was deliberate because RobustScaler uses median and interquartile range scaling, making it resistant to the influence of outliers. Since Isolation Forest explicitly aims to isolate anomalies (outliers) from normal points, preserving their extremity without letting them distort the scaling of normal data is crucial. This dual scaling approach ensures each algorithm receives appropriately transformed data: StandardScaler for algorithms assuming normal distributions, and RobustScaler for algorithms where outlier preservation is essential to their detection mechanism. We initially used RobustScaler for all 3 algorithms, but only Isolation Forest kept using it after discovering that StandardScaler outputted better results for the other algorithms.



Visualizations - Data Distribution

The distribution visualization revealed that several features exhibit heavy right skewness, a common characteristic of network traffic data. The y axis in the distribution plots represent density, and density

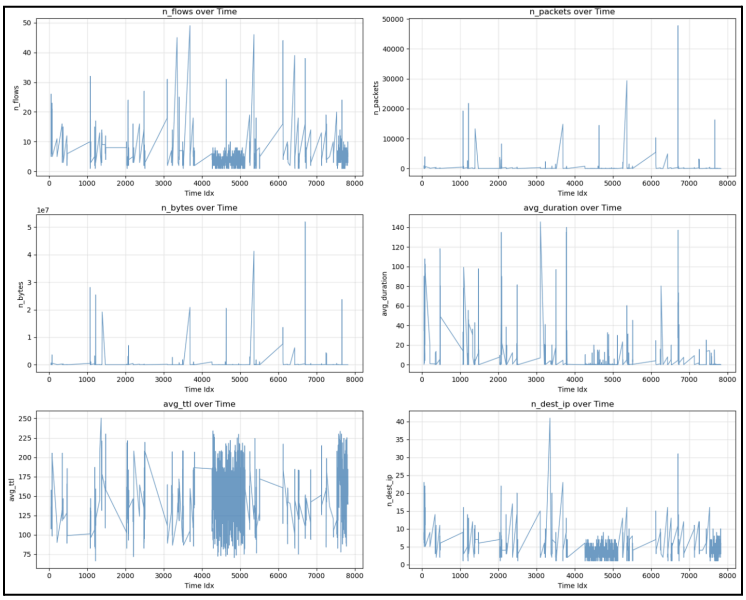
normalization scales each histogram so that the total area under the curve equals 1, similar to a probability distribution. This normalization causes features with extremely wide ranges to display very small y-axis values. For example, n_flows with values ranging from 0 to 4 million show a very tiny peak density around 0.14. Conversely, features with narrow ranges like tcp_udp_ratio_packets show much larger density values because the same total area is compressed into a smaller range.



For some features like id_time or n_flows, the densities are so low that it's hard to read, since they're so close to the x-axis. Applying logarithmic transformation to features like this would have been beneficial. Log transformation compresses the range of extreme outliers while expanding the scale for smaller values, resulting in more Gaussian-like distributions that better satisfy the assumptions of many machine learning algorithms. For distance based methods like K-means, log transformation prevents extreme outliers from disproportionately influencing cluster centroids and distance calculations. Additionally, in the context of network traffic analysis, logarithmic scales are often more meaningful because the difference between 10 and 100 flows may be as significant as the difference between 10,000 and 100,000 flows. In the future, we should consider implementing log transformation on any features that show the same issue as n_prints or are skewed prior to scaling to potentially improve clustering and interpretability.

Time Series Visualization

The time-series visualizations of key network metrics reveal highly volatile and non-stationary behaviors across the first 8,000 time indices. Multiple metrics exhibit extreme spikes that dominate their distributions: n_bytes shows massive spikes, while n_flows displays regular bursts. These extreme spikes likely represent



anomalous network events, bulk data transfers, or port scanning activity. The high temporal volatility is evident across all metrics, with avg_duration oscillating wildly between near-zero and over 140 seconds, and avg_ttl showing distinct behavioral regimes including a sustained high-activity period around time indices 4000-5500 where values cluster around 200 TTL. The sustained high activity period around 4000-5500 suggests a coordinated network event rather than random noise. Some metrics appear correlated, with n_flows and n_dest_ip showing similar patterns where increases in flows correspond to increases in destination IPs.

K-means Model Training and Evaluation

K-means serves as our baseline control precisely because it performs poorly at anomaly detection, which creates a meaningful comparison point for evaluating more sophisticated methods. K-means fundamentally assumes clusters are spherical and of equal size - an unrealistic assumption for network anomalies that typically represent rare, scattered deviations rather than cohesive groups. Furthermore, k-means forces every point into a cluster, including anomalies, thus diluting their distinctiveness. It's also highly sensitive to initialization and struggles with varying density regions where anomalies often reside.

```
dataset size: (4796604, 9)
Sample shape: (50000, 9)
on this k: 2
k=2: silhouette = 0.1444 (total time: 72.9s)
on this k: 3
k=3: silhouette = 0.1354 (total time: 109.0s)
on this k: 4
k=4: silhouette = 0.1334 (total time: 34.0s)
on this k: 5
k=5: silhouette = 0.1531 (total time: 32.0s)
on this k: 6
k=6: silhouette = 0.1909 (total time: 31.8s)
on this k: 7
k=7: silhouette = 0.1812 (total time: 29.7s)

 total time: 309.4190s
optimal k: 6
```

```
dataset size: (4796604, 9)
Sample shape: (50000, 9)
on this k: 2
k=2: silhouette = 0.2313 (total time: 33.9s)
on this k: 3
k=3: silhouette = 0.1709 (total time: 29.6s)
on this k: 4
k=4: silhouette = 0.1563 (total time: 32.4s)
on this k: 5
k=5: silhouette = 0.1437 (total time: 30.5s)
on this k: 6
k=6: silhouette = 0.1740 (total time: 28.6s)
on this k: 7
k=7: silhouette = 0.1955 (total time: 32.6s)

 total time: 187.5313s
optimal k: 2
```

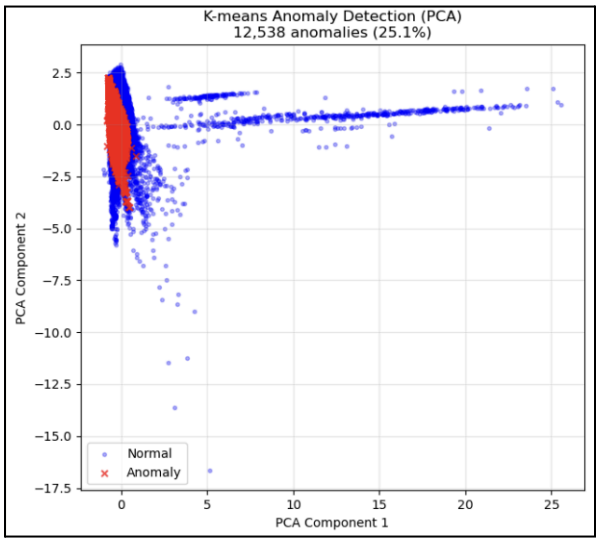
Due to the substantial size of the dataset - approximately 6.8 million entries - using the full dataset for hyperparameter tuning would have been computationally intensive. The silhouette score calculation has a complexity of $O(n^2)$, making it infeasible to evaluate multiple k values on millions of data points. Therefore, a representative sample of 50,000 observations was randomly selected for hyperparameter optimization. This does pose the risk of random sampling possibly mixing distinct traffic regimes such as diurnal patterns. However, the aggregation granularity mitigates the severity of this effect compared to raw or fine-grained time series.

The K-means hyperparameter tuning process evaluated cluster counts from k=2 to k=7 using the silhouette score as the optimization metric. The silhouette score measures how similar each point is to its own cluster compared to other clusters. For our sample, the highest silhouette score of 0.1909 was achieved at k=6, suggesting six clusters provide the most distinct separation in the data. However, all silhouette scores were relatively low (0.13-0.19), indicating weak cluster structure overall. We also had a

discrepancy in results, with another output of optimal k=2 with a silhouette of 0.2313 can be attributed to several factors inherent to K-means initialization and random sampling. First, different random samples from the same population will naturally yield slightly different optimal k values, especially when cluster separation is weak. Second, k-means uses random initialization for cluster centers, and different random seeds produce different starting points that can converge to different local optima. Third, MiniBatchKMeans introduces additional stochasticity through its batch sampling approach. These variations are expected in large, complex datasets where cluster boundaries are not sharply defined.

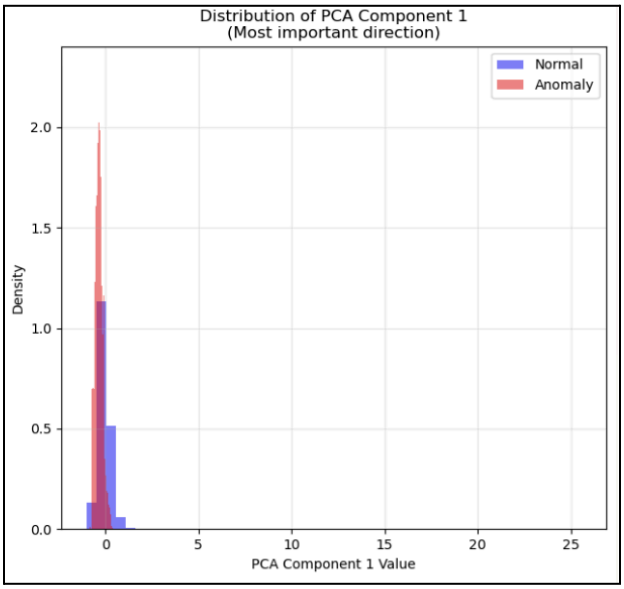
The final K-means model was trained using six clusters as determined by silhouette score optimization, employing MiniBatchKMeans with enhanced parameters for improved convergence. The resulting cluster distribution revealed an uneven partition of the data, with clusters ranging from 21.6% to 25.1% of samples, indicating that network traffic naturally organizes into multiple behavioral patterns rather than a simple normal/anomaly dichotomy. Two distinct anomaly detection strategies were evaluated: a distance based approach identifying points far from their cluster centroids and a cluster based approach designating the entire smallest cluster as anomalous. The distance based method proved overly conservative, identifying only 1.60% of samples as anomalies - a rate potentially too low for realistic network monitoring where 5-10% anomaly rates are common. Conversely, the cluster based approach classified 21.64% of samples as anomalous, representing an entire behavioral pattern within the network. This higher rate, while substantial, may more accurately reflect the diversity of atypical network behaviors presented in traffic. The cluster based strategy was ultimately selected for further analysis as it better aligns with the multi-modal nature of network traffic and provides clearer interpretability—each cluster represents a distinct traffic pattern, with the smallest cluster representing the most deviant behavior worthy of investigation.

K-means Clustering Scatter Plot (PCA Space)

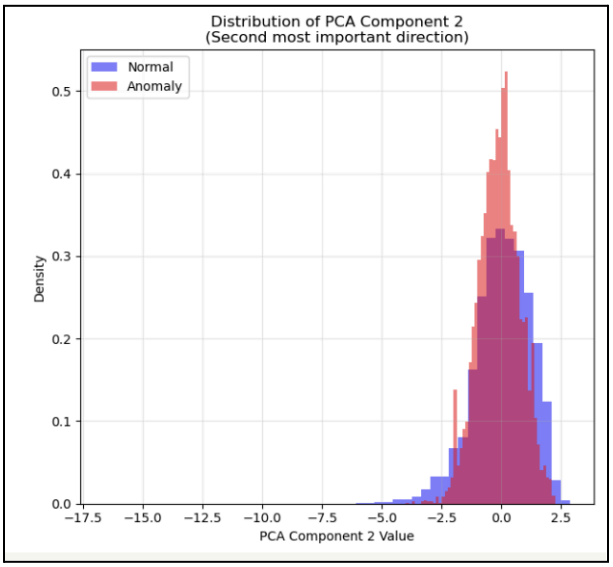


The scatter plot displays the k-means anomaly detection results in the two-dimensional PCA space, revealing 12,538 anomalies (25.1% of the 50,000 sample dataset). The visualization shows clear separation between the two clusters: anomalies (red) form a dense, concentrated cluster near the origin with PCA Component 1 values close to 0 and Component 2 values ranging primarily from -5 to 2.5. In contrast, normal traffic (blue) exhibits much greater spread across the feature space, extending along Component 1 to values beyond 25 and showing substantial vertical dispersion in Component 2 from approximately -15 to 2.5. This separation indicates that the k-means algorithm successfully identified a distinct subset of traffic with anomalous characteristics. The small number of scattered anomaly points below -7.5 may represent particularly unusual outliers within the cluster itself.

Distribution of PCA Component 1



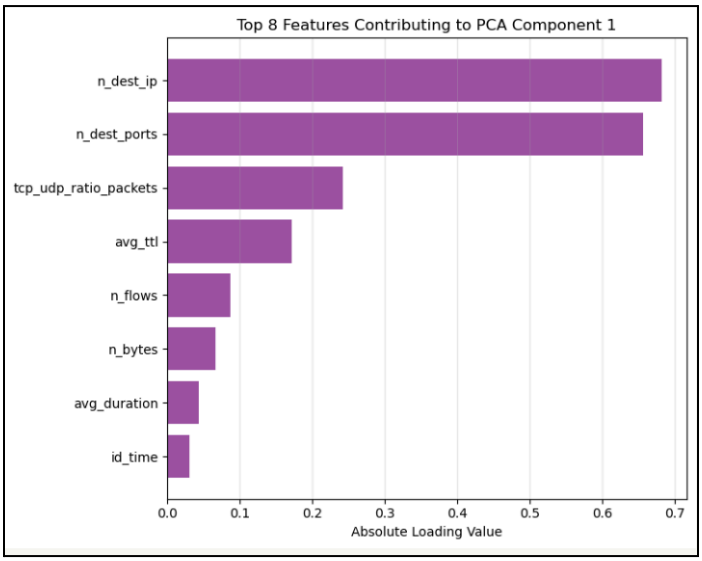
The histogram shows dramatically different distributions between normal and anomaly classes. Both distributions peak sharply near zero, with the anomaly distribution (red) forming an extremely narrow, tall spike with density reaching approximately 2.0, indicating that nearly all anomalies have Component 1 values very close to 0. The normal traffic distribution (blue) also peaks near zero but with a much broader spread and lower peak density around 0.5, extending noticeably into positive values up to approximately 5. This suggests that while both classes center near the origin in this dimension, anomalies are far more homogeneous in their Component 1 characteristics. Given that Component 1 explains 22.48% of the variance and is dominated by n_dest_ip and n_dest_ports (as shown in the feature contribution plot), this distribution indicates that anomalous traffic contacts fewer or more consistent numbers of destinations compared to the connectivity patterns seen in normal traffic.



Distribution of PCA Component 2

This histogram reveals an even more striking separation between classes than Component 1. The anomaly distribution forms a bell-shaped curve, while the normal traffic distribution is also bell shaped, but doesn't peak as high as the anomaly distribution. The substantial overlap between the two distributions in this component, combined with their different shapes, suggests that Component 2 captures behavioral patterns where anomalies exhibit more consistent, predictable values while normal traffic shows greater variability.

Top 8 Contributing Features



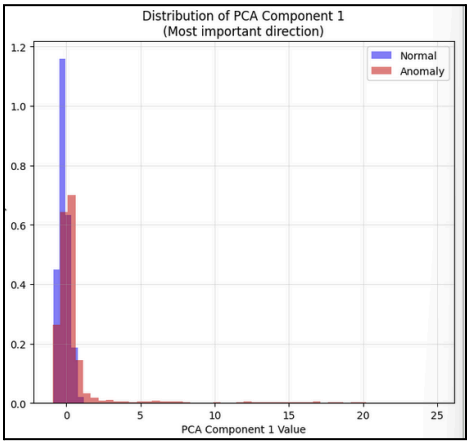
The feature contribution bar chart reveals that Component 1 is primarily driven by network connectivity diversity metrics rather than volume-based features. The top two contributors are n_dest_ip and n_dest_ports, indicating that how broadly traffic is distributed across destinations is the strongest differentiator between

normal and anomalous behavior. The third significant contributor is tcp_udp_ratio_packets suggesting protocol composition also plays a role in distinguishing traffic types. Mid level contributors include avg_ttl, n_flows, and n_bytes showing that after proper scaling, the algorithm considers routing characteristics, flow counts, and data volume but gives them less weight than connectivity patterns. The minimal contributions from avg_duration and id_time indicate these temporal features are less important for the primary axis of variation.

DBSCAN Model Training and Evaluation

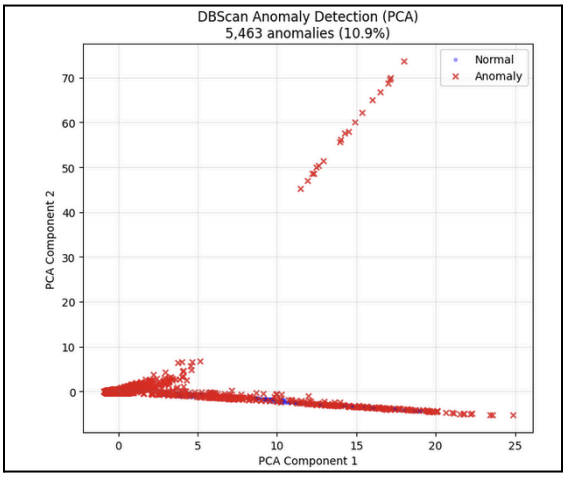
Because DBSCAN is non-parametric and computationally expensive on large datasets, a sample size of 50,000 observations was also selected for the DBSCAN model using the same sampling process described in our K-means model. We then fit the DBSCAN model on the selected features using the parameters eps (neighborhood radius) and min_samples (minimum points per cluster).

After fitting, we applied a DBSCAN to a test dataset to identify anomalies. To better visualize and interpret the results, we projected the classified data into its first two principal components and plotted clustered points (normal) in blue and anomalous points in red.



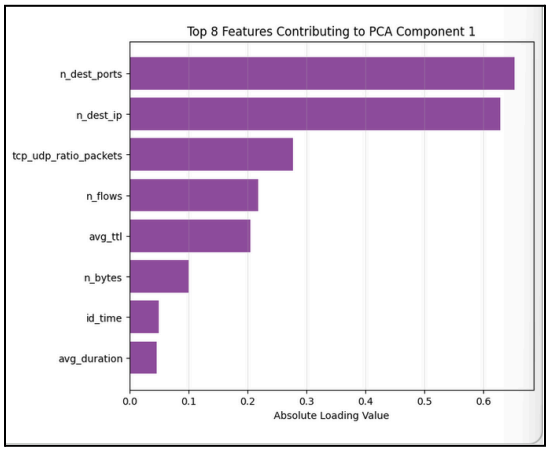
Distribution of PCA Component 1

Both anomalous and normal data peak near a value of zero along PCA Component 1. However, the normal traffic exhibits a sharper and more concentrated peak, while the anomalous traffic displays a broader and flatter distribution. This wider spread suggests that anomalous flows deviate from the expected value along this principal direction. In other words, PCA Component 1 appears to represent a "typical" traffic pattern that anomalies fail to conform to.



DBSCAN Clustering Scatter Plot (PCA Space)

Compared to K-means, DBSCAN categorizes anomalous data across a much wider region in both PCA Component 1 and PCA Component 2. This behavior is expected as DBSCAN identifies anomalies based on local density rather than distance to a centroid. This results in points lying in sparse regions being flagged as anomalies, regardless of their global position.



Top 8 Contributing Features

Analysis of PCA loadings shows that n_dest_ports and n_dest_ip are the top two contributors to PCA Component 1. This suggests that destination diversity plays a significant role in defining typical versus atypical traffic behavior. Flows that communicate with unusually many ports or IP addresses are more likely to fall outside dense regions and be flagged as anomalies by DBSCAN.

Compared to our results in K-means the clusters are far more spread out which is to be expected. We did not use silhouette score as a way to evaluate our DBSCAN performance as silhouette assumes convex, centroid-based clusters and tends to produce low values for density-based methods even when clustering is meaningful. Instead, we used a Davies-Bouldin Index to measure how "good" the clustering was in the dataset.

The DBSCAN model achieved a DBI score of 0.73, indicating relatively low intra-cluster distance and good inter-cluster separation. This suggests that the identified clusters are well formed and the noise points are genuinely outside of the clusters. Overall, the DBSCAN model detected 10.93% of the data as anomalous. While a DBSCAN model enables the discovery of diverse anomalous patterns, it may also include traffic that is mildly unusual but not truly malicious. Thus, there is a strong possibility of the model overcategorizing the data as anomalous.

Isolation Forest Model Training and Evaluation

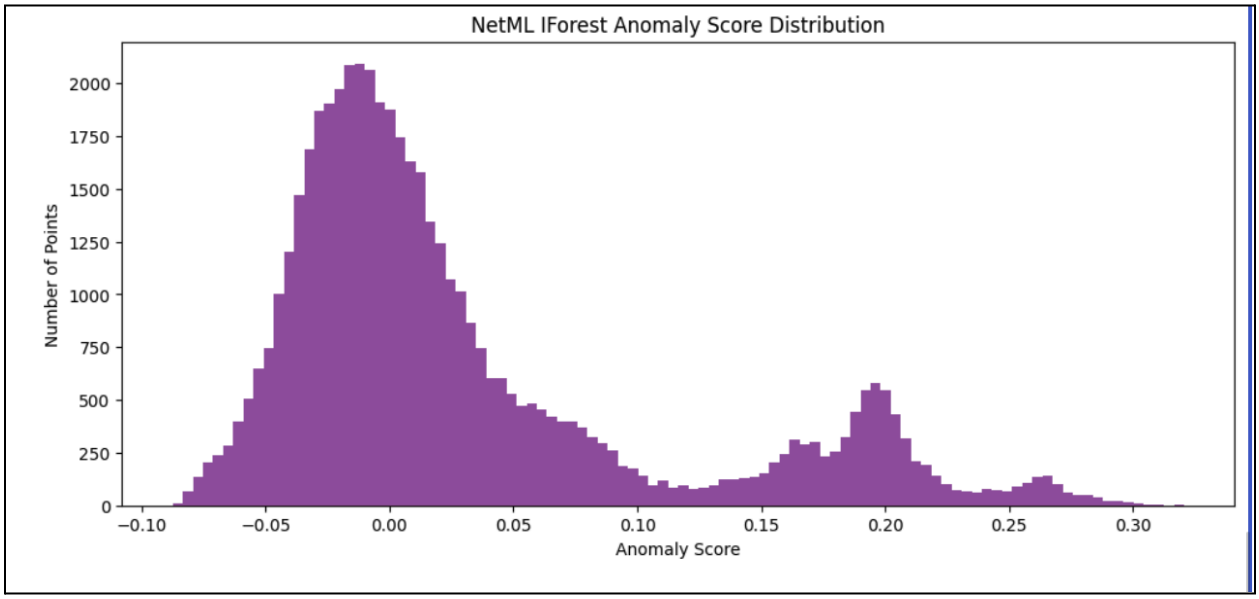
For our Isolation Forest model, we used NetML's IForest model to learn an internal score distribution by isolating data points through randomized tree partitions. The underlying assumption is that anomalous points are easier to isolate and therefore receive higher anomaly scores.

We fit an unsupervised NetML Isolation Forest on the first 70% of the CESNET traffic, letting it learn normal traffic patterns. Then we applied it to the final 50,000 entries of the data to identify points that deviate strongly from this learned baseline. Because NetML's IForest doesn't enforce a fixed anomaly proportion at prediction time, we observed a very low anomaly rate at test time despite setting contamination to 0.3. This behavior reflects IForest's conservative nature: it flags only the most extreme outliers in feature space.

We used iforest_model.decision_function(features_test) to obtain continuous anomaly scores which allowed us to analyze the distribution of isolation strengths rather than solely relying on binary labels.

Anomaly Score Distribution

The Isolation Forest score distribution is strongly right-skewed, with over 98% of observations receiving scores below 0.1 and fewer than 1% exceeding 0.2. This indicates that the majority of traffic closely follows learned normal behavior while only a small fraction exhibits anomalous behavior. Overall, the IForest model did have a very low anomaly detection rate with a rate of 0.71% even when contamination was set to the maximum value of 0.5. Because such a small fraction of the data exceeds high anomaly score thresholds, the model acts conservatively and flags only the most extreme deviations from normal traffic patterns. This behavior suggests that IForest is well-suited for identifying strong outliers but likely under-detects subtler or locally sparse anomalies that density-based methods like DBSCAN are more likely to capture.



Model Comparison and Evaluation

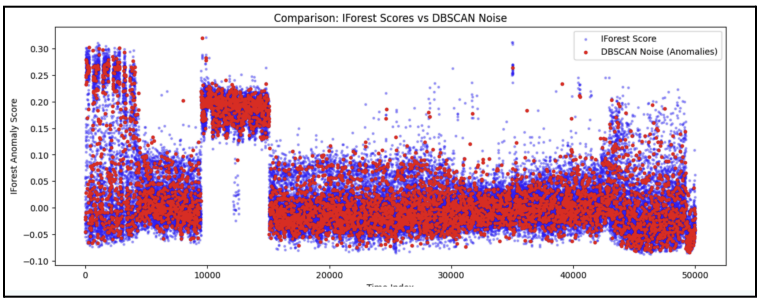
The feature contribution patterns for both DBSCAN and K-means reveal both similarities and notable differences between the two clustering algorithms' perspectives on what defines anomalous network behavior. Both algorithms identify n_dest_ports and n_dest_ip as the top two most important features. This consistency across both methods strongly validates that network connectivity diversity - specifically how many different destinations and ports traffic connects to - is the fundamental characteristic distinguishing anomalous traffic from normal behavior in this dataset. Both K-means and DBSCAN also have tcp_udp_ratio_packets ranked third

However, the algorithms diverge in their ranking of secondary features. While all features are there, both algorithms ranked the features differently. This difference suggests that DBSCAN's density based approach focuses more narrowly on connectivity metrics and flow characteristics, while K-means' centroid based method considers a broader range of features including volume and temporal aspects. The

overall similarity in the top features, however, indicates that both algorithms have successfully identified the core behavioral patterns that distinguish anomalous network traffic from normal activity in this dataset.

Additionally, K-means clusters were less well-defined compared to DBSCAN's clusters. This is likely due to K-means' assumption of spherical clusters and uniform density which does not align well with the heterogeneous nature of aggregated network flows. In contrast, DBSCAN's density-based approach enables it to identify arbitrarily shaped clusters and isolate low-density regions, making it better suited for modeling network traffic behavior.

When mapping DBSCAN-flagged anomalous points onto the Isolation Forest anomaly score distribution, we observe that these points span the full range of anomaly scores. This suggests that DBSCAN captures both extreme outliers and locally sparse behaviors that Isolation Forest does not necessarily consider anomalous. Only 0.46% of DBSCAN anomalies were also flagged by Isolation Forest, highlighting the complementary nature of the two methods. While Isolation Forest acts conservatively and identifies only the most extreme deviations, DBSCAN flags a broader set of unusual traffic patterns based on local density.



Overall, we observe minimal overlap between the anomaly sets identified by K-means, DBSCAN, and Isolation Forest. Specifically, fewer than 1% of K-means anomalies are also flagged by Isolation Forest, while 5.13% of DBSCAN anomalies overlap with K-means. This low agreement reflects the fundamentally different anomaly definitions employed by each method rather than poor model performance. Distance-based (K-means), density-based (DBSCAN), and isolation-based (IForest) approaches emphasize complementary aspects of abnormal network behavior. Consequently, the limited overlap suggests that each model identifies distinct classes of anomalous traffic rather than redundant signals.

Conclusion

In this study, we evaluated three unsupervised anomaly detection methods: K-means, DBSCAN, and Isolation forest on CESNET traffic data. Our results demonstrate that no single method provides a comprehensive characterization of anomalous behavior. Instead, each model captures a distinct notion of abnormality based on its underlying assumptions.

DBSCAN identifies traffic occurring in low-density regions of features space and identifies suspicious traffic occurring in low-density regions of feature space, allowing the discovery of rare patterns at the cost of a moderately high anomaly rate. Isolation Forest, by contrast, produces a highly conservative set of anomalies, flagging only traffic that is strongly isolated from the majority of observations. K-means occupies an intermediate position, detecting broad deviations from cluster centroids but struggles to

model complex, non-spherical traffic distributions. Notably, we observe minimal overlap between the anomaly sets produced by these methods, with less than 1% agreement between Isolation Forest and either clustering approach.

These findings suggest that unsupervised anomaly detection works best when combining complementary methods or selecting models based on the desired balance between sensitivity and precision. This solution is likely to yield more robust and interpretable results in real-world network monitoring. Future work should explore ensemble approaches that integrate clustering and isolation based models to better capture the full spectrum of anomalous network behavior.

Obtaining our data and Running the Notebook

To run our notebook, please go to this link to obtain our data which is labeled under "CESNET_agg_10_minutes":
https://drive.google.com/drive/folders/1YFE8zgne7KqoHoNRiiSGd_ONOV5fbd4t?usp=sharing

Then, pull the code we've pushed to the Github repository. The data should be placed in the "data" folder under "CESNET_agg_10_minutes/". Adjust the DIRECTORY variable appropriately, and the notebook will be ready to run.