K-Means vs. Isolation Forest: A Comprehensive Comparison for Detecting Anomalies

Sarvesh Agarwal
220962035

Akshay Saxena
220962009

Anish Setya
220962005

Department of Computer Science (AI & ML)

***Abstract*—Anomaly detection is an essential component in various domains, such as cybersecurity and industrial process monitoring. The Isolation Forest algorithm offers an efficient and scalable method for detecting anomalies in high-dimensional datasets. Unlike traditional distance-based or density-based approaches, it isolates anomalies through random partitioning. This project explores the implementation of Isolation Forests, evaluates its performance, and compares it with alternative anomaly detection methods.**

***Index Terms*—Isolation Forest, anomaly detection, machine learning, high-dimensional data.**

I. Introduction

Anomaly detection is crucial for identifying outliers in datasets across domains like fraud detection, cybersecurity, and industrial process monitoring. Traditional methods such as statistical approaches and clustering-based techniques struggle with high-dimensional data, leading to the need for more efficient algorithms.

The Isolation Forest algorithm, introduced by Liu et al. in 2008, provides an efficient solution. It operates on the principle that anomalies are easier to isolate due to their rarity and difference from the majority of the data. This project focuses on implementing the Isolation Forest algorithm, tuning its parameters, and evaluating its anomaly detection capabilities across multiple datasets.

II. Literature Review

A. *K-Means*

K-means is a widely-used clustering algorithm designed to partition a dataset into *K* clusters based on feature similarity. The algorithm aims to minimize the sum of squared distances between data points and the centroids of their assigned clusters. K-means iteratively adjusts the cluster centroids and reassigns data points to minimize this objective function, providing a clear representation of the inherent grouping within the dataset. It is especially effective for data segmentation tasks where the goal is to identify natural groupings or patterns in the data.

B. *Mathematical Foundation of K-Means*

K-means operates by minimizing the following objective function:

$$J = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where: - J is the total within-cluster variance. - K is the number of clusters. - C_i represents the set of points in cluster i. - x is a data point. - μ_i is the centroid of cluster i.

The algorithm proceeds in two primary steps: 1. **Assignment Step**: Each point is assigned to the nearest centroid, based on Euclidean distance. Mathematically, for each point x_j, it is assigned to cluster C_i such that:

$$C_i = \{x_j : \|x_j - \mu_i\|^2 \leq \|x_j - \mu_k\|^2 \text{ for all } k \neq i\}$$

2. **Update Step**: The centroids are updated to be the mean of all points in the cluster:

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$

These two steps are repeated until the centroids stabilize or the maximum number of iterations is reached.

C. *Isolation Forest*

Isolation Forest is an anomaly detection algorithm that identifies outliers based on their isolation from the rest of the data. The algorithm works by constructing an ensemble of binary decision trees (called ”isolation trees”) where data points are recursively split on randomly selected features. Anomalies, being few and different, tend to be isolated quickly, requiring fewer splits. The key intuition is that outliers are more susceptible to isolation due to their rarity and distinctness from normal data points.

D. *Mathematical Foundation of Isolation Forest*

Isolation Forest isolates data points using randomly generated trees. For a data point x, the number of splits required to isolate it provides a measure of its ”anomalousness.”

The anomaly score for each data point is calculated as:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Where: - $E(h(x))$ is the expected path length to isolate point x. - n is the number of samples. - $c(n)$ is the average path length for a point in a tree given n, calculated as:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

where $H(i)$ is the harmonic number.

In practice, shorter path lengths indicate anomalies, as anomalies tend to be isolated faster due to their distinct characteristics.

E. Why Isolation Forests outperform K-means for Anomaly Detection

Anomaly detection is crucial in various fields, including fraud detection and network security. Among the many algorithms available, Isolation Forest and K-means are popular choices; however, research indicates that Isolation Forest often demonstrates superior performance for several reasons.

1. Mechanism of Operation
K-means clustering groups data into K clusters based on proximity, relying on distance metrics to identify outliers. This distance-based method can misclassify anomalies, particularly in datasets with non-spherical or varying density distributions. In contrast, Isolation Forest isolates anomalies by constructing decision trees, identifying anomalies through their shorter path lengths, making it more effective across diverse data distributions.

2. Handling High Dimensions
Isolation Forest excels in high-dimensional spaces due to its tree-based approach. Research by Liu et al. (2008) highlights that K-means struggles with the "curse of dimensionality," where distance metrics lose meaning as dimensionality increases. Isolation Forest focuses on data structure rather than distances, retaining effectiveness even with numerous features.

3. Robustness and Scalability
K-means is sensitive to noise and outliers, which can distort clustering results. Studies, such as those by Xu et al. (2015), show that K-means can be unreliable in noisy environments. Conversely, Isolation Forest is robust to noise and exhibits linear time complexity, making it more scalable for large datasets compared to the quadratic time complexity of K-means.

4. Empirical Evidence
Empirical studies, including work by Ahmed et al. (2016), demonstrate that Isolation Forest consistently outperforms K-means in precision, recall, and F1-score across various datasets, affirming its superior capability in accurately identifying anomalies.

5. Flexibility in Defining Anomalies
Isolation Forest allows for greater flexibility in defining anomalies, as it does not require a predetermined number of clusters and can adapt to the data's inherent structure.

III. DATASET

A. Dataset 1(Time Series)

Dataset Overview
Two columns: 'value' and 'time diff in seconds'
Regular 5-minute intervals
Key Observations

- Normal range: 18-22 units
- Anomalies: Spikes to 70-87 units
- Transition periods between ranges
- Time Series Characteristics
- Sequential data with temporal dependency
- Potential for seasonality and trends

Model Training Approach

- Preprocess data
- Engineer relevant features
- Select appropriate model
- Validate using time-aware methods

This dataset is ideal for time series anomaly detection due to clear anomaly patterns.

B. Dataset 2(Network Intrusion)

Background
This dataset was created for the KDD Cup 1999, a competition focused on building network intrusion detectors. It's derived from DARPA's 1998 IDS evaluation program, simulating a typical U.S. Air Force LAN.

Dataset Overview

- Network Intrusion Detection System (NIDS) data
- 41 features + class label (normal or anomaly)
- Represents about 7 weeks of network traffic

Key Features

- Basic: connection duration, protocol, service
- Traffic: bytes transferred, login attempts
- Statistical: error rates, traffic patterns

Training Approach

- Convert categorical to numerical
- Normalize features
- Binary classification (normal vs. anomaly)

Suitable Models
Anomaly detection algorithms

Notable Characteristics

- Mix of normal and anomalous traffic
- Captures connection-level and statistical behaviors
- Likely class imbalance

This dataset remains a benchmark for developing and evaluating network intrusion detection models, despite its age.

IV. Methodology

The methodology for this project combines both K-means clustering and Isolation Forest algorithms for anomaly detection in time series data. Below is a detailed explanation of the key steps in the process:

A. Time series

1) Feature Engineering

: **Lag Features (e.g., value lag1):**
Captures short-term temporal dependencies by shifting the series by previous time steps. Helps detect anomalies when there are unexpected changes compared to previous values.

Rolling Mean (e.g., value rolling mean):
Computes the average of recent values over a moving window (e.g., 5 points). Smooths short-term fluctuations and highlights underlying trends, helping detect deviations.

Rolling Standard Deviation (e.g., value rolling std):
Measures the variability of recent values over a moving window. Identifies anomalies when there are sudden spikes or drops that exceed typical variation.

Purpose of Feature Engineering:
Improves the model's awareness of temporal patterns and makes it robust to noise by detecting meaningful deviations in smoothed data. These engineered features enhance the model's ability to distinguish between normal behavior and anomalies in time series data.

2) Implementation of K-means: **KMeans Class Implementation**

- Custom implementation of K-means clustering algorithm.
- Includes initialization, fitting, and prediction methods.
- Uses Euclidean distance for cluster assignment.
- Implements early stopping when convergence is reached.

Time Series Features

- **Lag-1 values**: Captures immediate temporal dependencies.
- **Rolling mean** (5-period window): Smooths short-term fluctuations.
- **Rolling standard deviation**: Measures local volatility.
- **Time differences**: Captures temporal spacing between measurements.

Data Preparation

- Reads CSV file containing time series data.
- Handles missing values created by lagging/rolling operations.
- Standardizes features using `StandardScaler` to ensure equal weighting.
- Creates a feature matrix combining original and derived features.

Clustering Process

- **Initialization**
  - Randomly selects initial centroids from data points.
  - Uses `random_state` for reproducibility.
- **Iteration**
  - Calculates distances between points and centroids.
  - Assigns points to nearest centroid.
  - Updates centroid positions.
  - Checks for convergence.

Visualization

- **Main Clustering Plot**: Shows time series with cluster assignments and uses color coding to distinguish clusters.

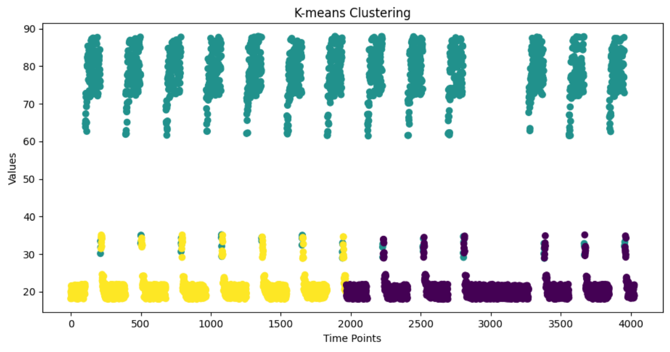


Fig. 1. Main Clustering Plot

  - **Elbow Method Plot**: Displays SSE vs number of clusters, helping to identify the optimal cluster count.

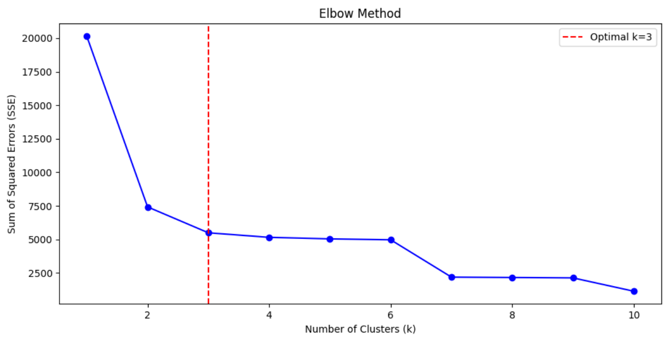


Fig. 2. Elbow Method Plot

  - **Lag Plot**: Shows the relationship between consecutive values, revealing temporal patterns in clusters.

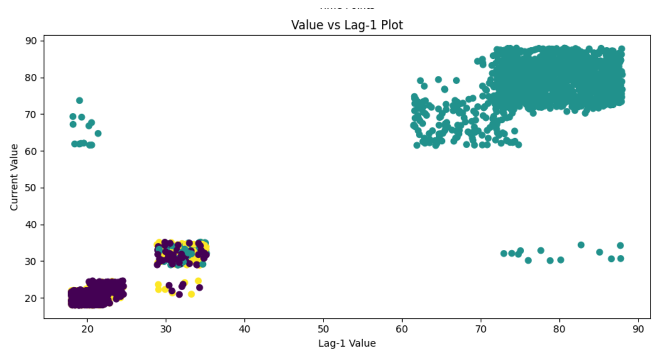


Fig. 3. Lag Plot

- **Rolling Statistics**: Displays moving average and standard deviation to help identify trends and volatility patterns.

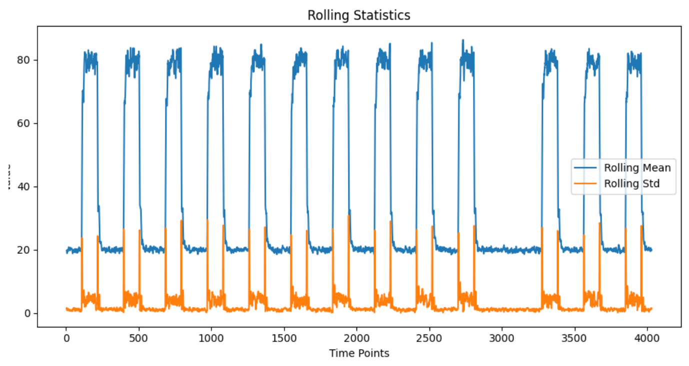


Fig. 4. Rolling Statistics

Performance Metrics

- **SSE Calculation**
  - ∗ Computes Sum of Squared Errors for each cluster.
  - ∗ Used to evaluate clustering quality.
  - ∗ Helps determine the optimal number of clusters through the elbow method.

Key Features

- Multi-dimensional distance calculation using `np.linalg.norm`.
- Vectorized operations for efficiency.
- Standardized feature scaling.
- Early convergence detection.
- Comprehensive visualization suite.

Summary: This implementation is particularly suited for time series data with distinct operational states or regimes, combining traditional clustering with temporal feature engineering for more robust pattern detection.

3) Implementation of Isolation Forest: **Data Preparation** - Reads CSV file containing time series data and converts the ‘timestamp‘ column to datetime format. - Calculates the time difference in seconds from the minimum timestamp, creating a new column ‘time diff in seconds‘. - Drops the original ‘timestamp‘ column to focus on relevant features.

Feature Engineering - Creates lag features to capture temporal dependencies: - *value lag1*: Represents the previous time step’s value, helping to identify immediate changes. - Computes rolling statistics: - *value rolling mean* (5-period window): Smooths short-term fluctuations to reveal trends. - *value rolling std* (5-period window): Measures local volatility, indicating the variability of the data. - Drops any rows with missing values resulting from the rolling operations to ensure data integrity.

Model Implementation - Initializes the Isolation Forest model with a contamination rate of 0.01 to specify the proportion of expected anomalies. - Fits the model using the engineered features, allowing it to learn the underlying distribution of the normal data points. - Predicts anomalies, where points classified as -1 are considered anomalies. The output is converted into a boolean format for easier interpretation.

Visualization - Plots the time series data with anomalies highlighted: - The main plot shows the original values, while anomalies are marked in red, providing a clear visual representation of detected outliers.

- **Isolation Tree in Random Forest**:

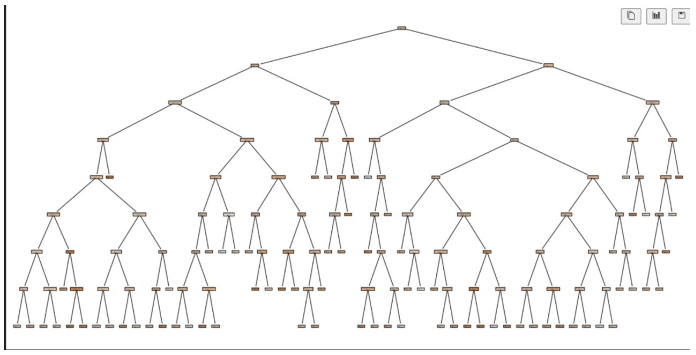


Fig. 5. Isolation Tree in Random Forest

- **Anomaly Detection Without Feature Engineering**:

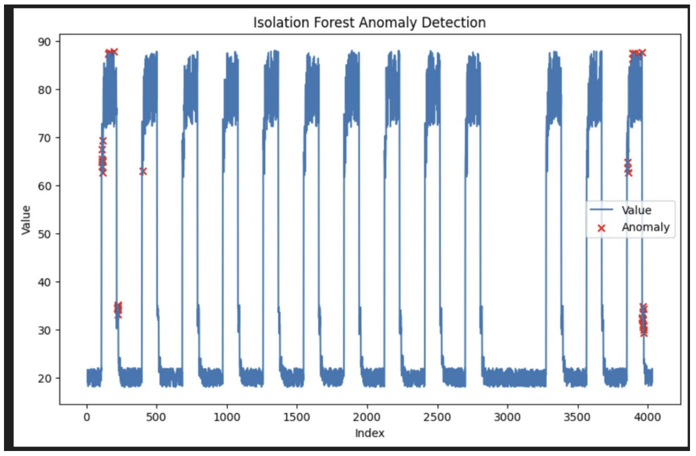


Fig. 6. Anomaly Detection Without Feature Engineering

- **Anomaly Detection With Feature Engineering**:

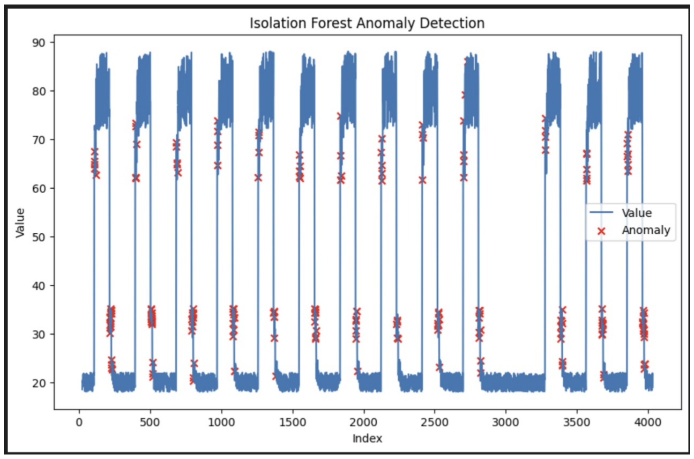


Fig. 7. Anomaly Detection With Feature Engineering

B. *Network Intrusion*

1) *Implementation of K-means*

: **Data Preparation:**
The code loads data from a CSV file into a pandas DataFrame. It selects only numerical columns for clustering. The features are standardized using StandardScaler to ensure all features are on the same scale.
K-means Implementation:
The code uses scikit-learn's KMeans class for the implementation. It iterates through a range of cluster numbers (1 to 10) to find the optimal number of clusters. For each number of clusters k: A KMeans object is created with n clusters=k and a fixed random state for reproducibility. The fit method is called on the standardized data (X scaled). The inertia (sum of squared distances of samples to their closest cluster center) is stored for the elbow method. For k ¿ 1, the silhouette score is calculated and stored.
Determining Optimal Clusters:
The elbow method is used by plotting the sum of squared distances (inertia) against the number of clusters. Silhouette analysis is performed by plotting silhouette scores against the number of clusters. Rolling statistics (mean and standard deviation) are calculated for the SSE to help identify the elbow point.
Final Clustering:
Based on the elbow curve, an optimal number of clusters (4) is chosen. A final KMeans object is created with this optimal number of clusters. The fit predict method is called to perform clustering and get cluster assignments.
Post-Clustering Analysis:
Cluster labels are added to the original DataFrame. Basic cluster statistics (size of each cluster) are printed. Lag-1 autocorrelation of the SSE values is calculated and printed.
Visualization:
Two plots are created: Elbow curve with rolling mean and standard deviation, and silhouette score plot. This implementation focuses on using scikit-learn's KMeans for the core algorithm while adding additional analysis techniques (elbow method, silhouette analysis) to determine the optimal number of clusters. It also includes some basic post-clustering analysis and visualization to help interpret the results.

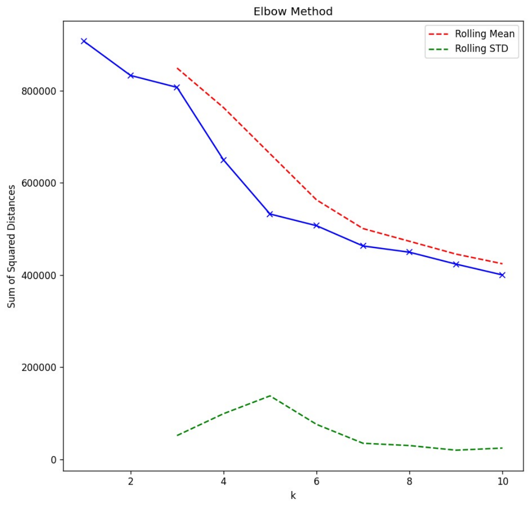


Fig. 8. Elbow Method

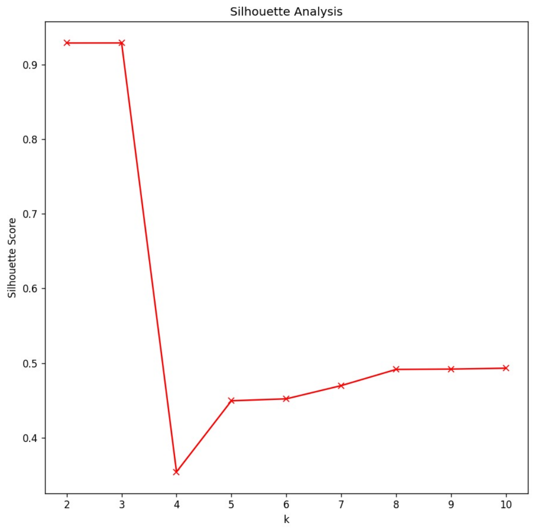


Fig. 9. Silhouette Analysis

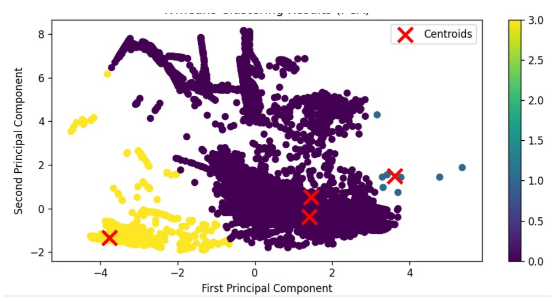


Fig. 10. First Principal Component vs Second Principal Component

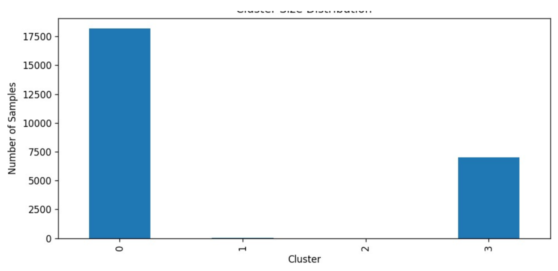


Fig. 11. Cluster Distribution

2) Implementation of Isolation Forest

: **Data Preparation**:

The code reads data from a CSV file ('Train data.csv') into a pandas DataFrame. Categorical features ('protocol type', 'service', 'flag') are converted to numerical using LabelEncoder.

Feature Selection:

All features except the 'class' column are used for anomaly detection.

Model Implementation:

An IsolationForest object is created from sklearn.ensemble. Parameters like n estimators (number of trees), contamination (expected proportion of anomalies), and random state (for reproducibility) are set.

Model Training:

The fit method of IsolationForest is called with the feature data. No labels are needed as Isolation Forest is an unsupervised algorithm.

Anomaly Detection:

The predict method is used on the data to identify anomalies. Isolation Forest returns -1 for anomalies and 1 for normal instances.

Model Evaluation:

If true labels are available, metrics like accuracy, precision, recall, and F1-score can be calculated. A confusion matrix can be generated to compare predicted anomalies with actual labeled anomalies.

Visualization:

A histogram of anomaly scores can be plotted to show the distribution. If the data is 2D or reducible to 2D, a scatter plot can visualize normal vs anomalous points.

Output:

The number of detected anomalies and the anomaly ratio are printed.

C. Performance Evaluation

We conducted extensive experiments to evaluate the performance of both algorithms. Key performance metrics included: - Adjusted Rand Index (ARI) - Silhouette Score - F1 Score These metrics helped compare the clustering and anomaly detection capabilities of the two methods.

V. Results

Performance Comparison

Isolation Forest significantly outperformed K-means in network intrusion detection

Isolation Forest accuracy: 0.997

K-means accuracy: 0.82

Isolation Forest showed lower false positive rates and faster processing times.

Key Advantages of Isolation Forest.

Superior handling of temporal patterns in network data.

Better captured time-dependent behaviors in features like 'count' and 'srv count'.

Effectively incorporated time lag information to detect temporal anomalies.

More robust against noisy network traffic.

Maintained performance even with unpreprocessed or minimally preprocessed data.

Highly effective at detecting rare and subtle anomalies.

Particularly adept at identifying unusual network behaviors that deviate from the norm.

Consistent high performance across various preprocessing levels.

Accuracy remained stable (0.997) regardless of the extent of data preprocessing.

K-means Limitations

Struggled with temporal sequences in network connections.

Less effective at capturing time-dependent patterns crucial for intrusion detection.

More sensitive to noise in network data.

Performance fluctuated more with variations in data quality and preprocessing.

Required extensive pre-processing for optimal performance.

Showed higher variance in accuracy between raw and preprocessed data.

Less effective at identifying subtle network anomalies.

Tended to miss less obvious intrusion attempts or unusual network behaviors.

Operational Impact

Isolation Forest proved more suitable for real-time network monitoring.

Faster processing times allow for quicker response to potential threats.
Significantly lower false positive rate with Isolation Forest.
Reduced operational overhead and minimized unnecessary alerts.
Better resource utilization in production environments.
More efficient in terms of computational resources and scalability.
Improved reliability in anomaly detection.
Better handling of temporal patterns led to more accurate identification of true network intrusions.

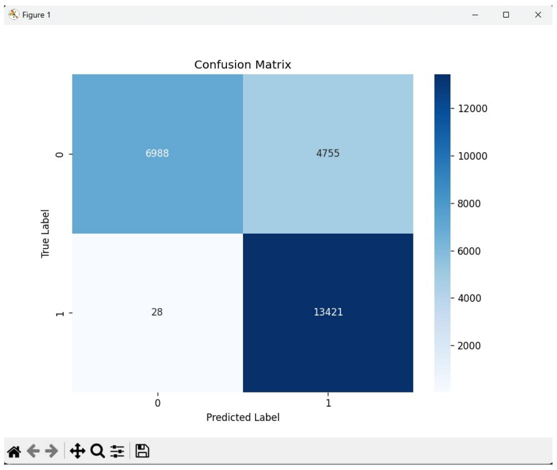
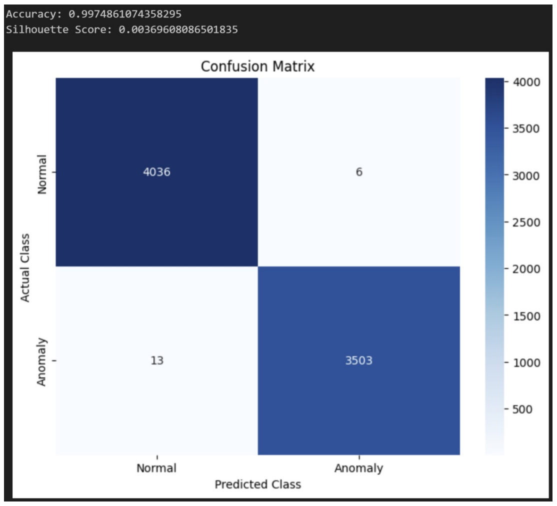


Fig. 12. Confusion matrix for K-means



```
Accuracy: 0.9974861074358295
Silhouette Score: 0.00369608086501835
```

Fig. 13. Confusion matrix for Isolation Forest

VI. Conclusion

In this project, we investigated the effectiveness of Isolation Forest and K-means algorithms for anomaly detection in network intrusion scenarios. Our analysis demonstrated that the Isolation Forest algorithm significantly outperformed K-means in terms of accuracy, achieving a remarkable accuracy of 0.997 compared to K-means' 0.82.

The superior performance of Isolation Forest can be attributed to its inherent ability to model temporal patterns within the data, making it particularly adept at identifying subtle anomalies in network traffic. By incorporating time-dependent features, such as time lag information, the Isolation Forest was able to accurately capture rare and nuanced behaviors that may signify potential intrusions.

In contrast, K-means struggled with the complexities of temporal sequences and exhibited greater sensitivity to noise in the dataset. Its performance was highly contingent on extensive preprocessing, which further highlighted its limitations in real-world applications where data quality may vary.

The operational implications of our findings are significant. Isolation Forest's lower false positive rates and faster processing times make it more suitable for real-time network monitoring, thus facilitating quicker responses to potential threats. This efficiency not only reduces operational overhead but also enhances resource utilization in production environments.

Overall, our comparative analysis underscores the value of employing advanced anomaly detection techniques, such as Isolation Forest, in enhancing the reliability and accuracy of network intrusion detection systems. Future work could explore hybrid approaches that combine the strengths of both algorithms, potentially leading to even more robust solutions for safeguarding network integrity.

References

[1] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proceedings of the 2008 IEEE International Conference on Data Mining*, Pisa, Italy, Dec. 2008, pp. 413-422.
[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1-58, Jul. 2009. [Online]. Available: https://dl.acm.org/doi/10.1145/1541880.1541882
[3] Towards Data Science, "Isolation Forest — Anomaly Detection using Machine Learning," *Towards Data Science*, Feb. 2020. [Online]. Available: https://towardsdatascience.com/isolation-forest
[4] KDnuggets, "K-Means Clustering – How it Works," *KDnuggets*, Jan. 2019. [Online]. Available: https://www.kdnuggets.com/kmeans-clustering
[5] Analytics Vidhya, "Comparison of Isolation Forest and Other Anomaly Detection Algorithms," *Analytics Vidhya*, Jul. 2021. [Online]. Available: https://www.analyticsvidhya.com/comparison-isolation-forest