EFFICIENT DATA STREAM ANOMALY DETECTION

INTRODUCTION

Anomalies, also referred to as outliers, are data points that significantly differ from the expected patterns or behaviours within a dataset. Identifying these outliers is essential as they can indicate potential issues, fraudulent activities, or lead to valuable insights. Anomaly detection is crucial across numerous domains such as data analysis, machine learning, and network security, where recognizing deviations helps in maintaining system integrity and uncovering hidden patterns.

TYPES OF ANOMALIES

There are three main types of anomalies: point anomalies, contextual anomalies, and collective anomalies.

Point Anomalies (Global Anomalies): These represent individual data points that deviate significantly from the rest of the dataset. For example, an unusually large credit card transaction may be flagged as a point anomaly.

Contextual Anomalies (Conditional Anomalies): These anomalies are dependent on the context or surrounding conditions. They are often found in time-series data, where certain patterns change over time. An example would be an unexpected spike in temperature during the winter season within weather data.

Collective Anomalies: These occur when a group of related data points collectively exhibits abnormal behaviour, even though individual points may appear normal. Detecting these anomalies typically requires more sophisticated pattern-based algorithms, and they are often found in dynamic environments, such as network traffic data.

Anomaly detection is a powerful tool used across numerous industries to prevent fraud, ensure system health, optimize processes, and protect against threats. Anomaly detection has a wide range of applications across various industries and fields due to its ability to identify unexpected or rare events that deviate from the norm. Here are some of the main applications:

- **Business:** Fraud detection, risk management, customer behaviour analysis
- **Security:** Cybersecurity, intrusion detection, physical security
- **Manufacturing:** Predictive maintenance, quality control, process optimization
- **Healthcare:** Patient monitoring, medical imaging, disease outbreak detection
- **Environmental Monitoring:** Climate change, natural disaster prediction, pollution detection
- **IoT:** Smart grids, smart homes, traffic management
- **Transportation:** Vehicle anomaly detection, supply chain optimization
- **Social Media:** Bot detection, sentiment analysis
- **Gaming:** Cheat detection, in-game behaviour analysis
- **Space and Astronomy:** Satellite data analysis, astronomical event detection
- **Call Centers:** Customer interaction monitoring

Anomaly detection helps improve efficiency, detect risks, and optimize operations in these and other industries.

Selecting the right anomaly detection algorithm requires careful consideration of various factors. These factors include the size, structure, and quality of the data, the nature of the anomalies, computational resources, and the specific industry or application. Ignoring these factors can lead to suboptimal performance, false positives, and missed anomalies. By carefully evaluating these aspects, you can choose an algorithm that is well-suited to your needs and delivers accurate, reliable results.

When selecting an anomaly detection algorithm, the primary factors to consider include:

- Type, size, and dimensionality of data
- Data distribution and nature of anomalies
- Real-time vs batch processing
- Availability of labelled data (supervised/unsupervised)
- Scalability and computational complexity
- Interpretability and noise handling
- Time-series characteristics (for temporal data)
- Evaluation metrics relevant to the specific problem
- Importance of domain knowledge

Outline for the Report: Efficient Data Stream Anomaly Detection

1. State of the Art

2. Project Objectives

- Algorithm Selection
- Data Stream Simulation
- Anomaly Detection
- Optimization
- Visualization

3. Methods

1. Statistical Methods

   Z-Score, KNN, IQR, Moving Average (MVG)

2. Advanced Methods

   Extended Moving Average (EMVG)

   Isolation Forest

3. Hybrid Model

   Isolation Forest with seasonal change handling and concept drift detection

4. Approach

- Algorithm Selection
- Data Stream Simulation
- Anomaly Detection
- Optimization
- Visualization

5. Metrics Used to Evaluate Models

- Precision
- Recall
- F1 Score
- Confusion Matrix

6. Conclusion

- Summary of achievements
- Limitations of the model
- Future exploration possibilities

STATE OF ART

The paper "Anomaly Detection by Using Streaming K-Means and Batch K-Means" by Zhuo Wang et al. aims to compare the effectiveness of Batch K-Means and Streaming K-Means algorithms in detecting anomalies within the MovieRate dataset. The methodology involved applying both algorithms to the dataset, assessing their performance based on cluster distribution, cost values, and anomaly detection consistency rates across various thresholds. Results indicated that while Streaming K-Means initially performed poorly in anomaly detection compared to Batch K-Means, its performance improved significantly with additional training iterations, leading to a more concentrated cluster distribution. However, Streaming K-Means exhibited a higher loss rate in detecting anomalies (nearly 30%), which could be reduced by extending the training window and allowing for more training time. The study highlights the potential of Streaming K-Means for real-time applications while emphasizing the need for further optimization to enhance its performance in big data contexts.

The paper "Review of Anomaly Detection Algorithms for Data Streams" aims to systematically classify and review existing algorithms for detecting anomalies in data streams, emphasizing their importance in real-time applications due to the continuous influx of data from sources like IoT and cloud computing. The authors categorize these algorithms into three types: offline learning based, semi-online learning based, and online learning based, and propose a new classification framework for systematic comparison. They plan to conduct qualitative and quantitative analyses using unified datasets to identify the strengths and limitations of each algorithm. The findings highlight a significant gap in research focused on real-time anomaly detection, leading to proposed future directions that include improving algorithm efficiency, enhancing data privacy and security, and developing robust methods for diverse data characteristics. Overall, the paper provides valuable insights for guiding future research and practical applications in the field.

The paper presents Streaming Half-Space-Trees (HS-Trees), an innovative method for detecting anomalies in streaming data that operates efficiently with finite memory and adapts to evolving data distributions. By utilizing an ensemble of HS-Trees, the method processes data in a single pass and employs mass profiles to identify anomalies based on low-mass subspaces. Experimental evaluations across six large datasets, including SMTP and HTTP, demonstrate that the Streaming HS-Trees significantly outperform traditional methods like Hoeffding Trees, particularly in scenarios with distribution changes, achieving higher AUC scores and faster runtime performance. The findings highlight the effectiveness of the proposed approach in addressing the challenges of anomaly detection in dynamic data streams.

The paper "Deep Isolation Forest for Anomaly Detection" introduces the Deep Isolation Forest (DIF) algorithm, an advanced enhancement of the traditional Isolation Forest (iForest) model. DIF uses non-linear partitioning, improving its ability to detect anomalies in high-dimensional and non-linearly separable data. Key contributions include the Computation-Efficient Representation Ensemble (CERE), which enhances scalability, and the Deviation-Enhanced Anomaly Scoring (DEAS) function, improving accuracy in anomaly detection. Experimental results show that DIF outperforms iForest and other state-of-the-art methods in terms of robustness, efficiency, and adaptability to various datasets.

The paper "Anomaly detection in streaming data: A comparison and evaluation study" aims to evaluate various state-of-the-art algorithms for detecting anomalies in streaming data, focusing on their effectiveness in the presence of challenges such as concept drift and varying outlier ratios. The methodology involved selecting popular unsupervised algorithms, conducting experiments with simulated streaming datasets, and tuning hyperparameters, particularly the memory span of the algorithms. Results indicated that while most algorithms performed well under moderate concept drift, they struggled with strong drift, especially in overlapping data densities. Algorithms like SWKNN, RSHash, and SDOstream excelled in outlier detection, with memory span significantly impacting performance; for instance, SWLOF was limited by its short memory. The study emphasized the importance of using pre-knowledge for parameter tuning and highlighted the varying interpretability of models, providing valuable insights for researchers and practitioners in the field.

OBJECTIVES

1. Algorithm Selection:

Goal: Identify and implement a suitable anomaly detection algorithm.

Requirement: The algorithm must adapt to concept drift (changes in data distribution over time) and handle seasonal variations.

2. Data Stream Simulation:

Goal: Design a function to simulate a continuous data stream.

Requirement: The data stream should include regular patterns, seasonal element, and random noise, simulating real-world scenarios.

3. Anomaly Detection:

Goal: Develop a real-time mechanism for detecting anomalies.

Requirement: The system should accurately flag anomalies (such as exceptionally high values or deviations) as the data is being streamed.

4. Optimization:

Goal: Ensure the anomaly detection algorithm is efficient and optimized for real-time data streams.

Requirement: The system should be optimized for both speed and memory efficiency, making it suitable for continuous streaming.

5. Visualization:

Goal: Implement a real-time visualization tool for the data stream.

Requirement: The tool should display both the data stream and any detected anomalies in real time for easy monitoring and analysis.

Each of these objectives will guide the development and testing of your Efficient Data Stream Anomaly Detection system.

METHODS

When dealing with anomaly detection in data streams, it's crucial to account for **concept drift** (i.e., changing patterns in the data over time) and **seasonal variations** (i.e., recurring patterns or trends). Below are various algorithms and techniques designed to handle both:

1. Statistical Methods

Statistical approaches rely on measuring deviations from expected patterns based on historical data, often using methods like moving averages, Z-scores, or more complex techniques.

- **Z-Score with Adaptive Thresholds**: This method calculates the Z-score of each data point by comparing it to the mean and standard deviation. To handle concept drift, update the mean and standard deviation over time, and adjust thresholds based on the distribution of recent data points.
    - **Pros**: Simple, interpretable, and computationally inexpensive.
    - **Cons**: Not ideal for complex seasonal patterns.

- **Exponentially Weighted Moving Average (EWMA)**: This approach tracks the moving average of data points, giving more weight to recent observations. It can adapt to changes (concept drift) and works well when seasonal patterns are less pronounced.
    - **Pros**: Adapts to trends over time.
    - **Cons**: May not capture abrupt seasonal changes.
- **Seasonal Decomposition of Time Series (STL Decomposition)**: This technique decomposes the time series into trend, seasonal, and residual components. Anomalies can be detected in the residual component, and concept drift is handled by recalculating trends over time.
    - **Pros**: Explicitly separates seasonal and trend components, making anomalies easier to detect.
    - **Cons**: Requires periodic re-fitting, and is computationally more intensive.

2. Distance-Based Methods

These methods involve measuring the distance between data points in a feature space to identify anomalies. They can be adapted to detect patterns that change over time (concept drift) and seasonal effects.

- **K-Nearest Neighbours (KNN) with Adaptive Windowing**: KNN can be adapted for streaming data by calculating distances between new data points and their nearest neighbours in the feature space. To handle concept drift, use a sliding or adaptive window of recent data for comparison.
    - **Pros**: Flexible and non-parametric.
    - **Cons**: High computational complexity, especially for large datasets or streams.
- **Dynamic Time Warping (DTW)**: DTW is a distance-based technique for time-series comparison that aligns time series with different temporal dynamics. It can handle seasonal variations by adjusting the alignment, but needs regular updates to deal with concept drift.
    - **Pros**: Effective at comparing sequences with shifting patterns.
    - **Cons**: Computationally expensive for large datasets.

3. Ensemble Methods

Ensemble methods combine the outputs of multiple models or algorithms, providing greater adaptability to both concept drift and seasonal variations.

- **Adaptive Random Forest (ARF)**: This ensemble-based method uses a set of decision trees to classify data and detect anomalies. ARF can adapt to concept drift by "growing" new trees and discarding outdated ones, effectively adjusting to changes over time.
    - **Pros**: Highly adaptable to changes in the data.
    - **Cons**: Requires careful tuning, and its effectiveness depends on the quality of the base models.
- **Bagging and Boosting Ensembles**: Methods like bagging and boosting can be used in combination with anomaly detection models. By training on different subsets of the data stream, they become more robust to concept drift and can handle recurring seasonal variations.
    - **Pros**: Increases the robustness of individual algorithms.
    - **Cons**: Increased computational complexity compared to single models.

4. Machine Learning Models

a. Supervised Learning

Supervised learning approaches can be applied if labelled data (normal vs. anomaly) is available. These models can be updated periodically to handle concept drift and trained on features that capture seasonal variations.

- **Random Forests or Gradient Boosting**: These are tree-based methods that can be retrained periodically to adapt to concept drift and use features derived from time-series decomposition to capture seasonality.
    - **Pros**: Effective for a variety of data types and can be extended to capture seasonal effects.
    - **Cons**: Requires labelled data and retraining, making them resource-intensive.

b. Unsupervised Learning

Unsupervised learning methods are widely used for anomaly detection because they don't require labelled data. These methods can adapt to concept drift by adjusting the learned models over time.

- **Isolation Forest**: A tree-based unsupervised learning algorithm that isolates anomalies by partitioning data points. It's highly efficient and can be updated to handle concept drift by using sliding windows.
    - **Pros**: Effective for both high-dimensional and seasonal data, works well with streams.
    - **Cons**: Requires tuning, and performance can degrade over time without regular updates.
- **One-Class SVM (Support Vector Machine)**: This method trains on "normal" data and detects anomalies by identifying points that don't fit the learned boundary. Concept drift can be handled by retraining periodically on new data.
    - **Pros**: Strong performance on non-linear data distributions.
    - **Cons**: Resource-intensive and requires careful tuning to handle streaming data effectively.

Summary Table

Algorithm	Concept Drift Handling	Seasonal Variations Handling	Pros	Cons
Z-Score with Adaptive Thresholds	Rolling window updates	Requires detrending or seasonal decomposition	Simple, computationally light	Struggles with complex seasonal patterns
EWMA (Moving Average)	Adapts with moving average	Handle via differencing	Easy to implement	Not good with abrupt seasonal changes
STL Decomposition	Recalculate trend over time	Explicitly separates seasonality	Powerful for time-series decomposition	Computationally intensive
Isolation Forest	Sliding window retraining	Seasonal preprocessing needed	Efficient for streaming data	Needs regular updates
One-Class SVM	Retrain periodically	Seasonal preprocessing needed	Strong performance on complex data	Computationally expensive
SARIMA (ARIMA for Seasonality)	Retrain periodically	Handles seasonality explicitly	Effective for seasonal data	Requires regular re-fitting
LSTM	Periodic retraining	Learns seasonal patterns automatically	Excellent for time-series	High computational cost
Autoencoder	Periodic retraining	Learns patterns, but seasonal preprocessing helps	Detects complex anomalies	Requires large training data

APPROACH

This report outlines the approach taken to achieve the objectives of the Efficient Data Stream Anomaly Detection project. Each objective, from the selection of anomaly detection algorithms to real-time visualization, was addressed using Python. The following sections provide detailed insights into how these objectives were met, including the use of statistical methods, moving averages, and a hybrid model based on Isolation Forest and seasonal adjustments.

1. Algorithm Selection

Objective:

The goal was to identify and implement a suitable anomaly detection algorithm that could handle concept drift and seasonal variations in streaming data.

Approach:

At the beginning of the project, I explored several statistical methods, including:

Z-Score: A method to detect anomalies based on how far data points deviate from the mean in terms of standard deviations.

K-Nearest Neighbours (KNN): An anomaly detection technique based on the distance between a data point and its neighbours.

Interquartile Range (IQR): A statistical method to detect outliers by measuring the spread of the middle 50% of the data.

Moving Average (MVG): A method for smoothing out data to capture the underlying trend and detect sudden deviations.

While all these statistical methods achieved high precision (1.0), they suffered from poor recall, meaning they failed to detect some anomalies, especially in cases of concept drift (changing data distributions) and seasonal variations. The inability of these methods to adapt to temporal dependencies and changing patterns in the data led me to explore more sophisticated techniques.

Evolution of the Approach:

After testing statistical methods, I moved on to Extended Moving Average (EMVG), which could observe and account for some previous data. While this improved performance slightly, it was not sufficient to handle seasonal changes and concept drift effectively.

Finally, I implemented the Isolation Forest algorithm, which is widely considered one of the best algorithms for streaming data. Isolation Forest works by isolating anomalies through random partitioning, making it efficient for high-dimensional data. While this approach improved anomaly detection, it still missed certain anomalies caused by seasonal patterns.

To address this, I developed a hybrid model that combined Isolation Forest with separate components to handle seasonal changes and concept drift. This hybrid model uses ensembling to merge the results of the individual models, allowing the system to capture both long-term seasonal trends and short-term deviations caused by concept drift.

2. Data Stream Simulation

Objective:

Design a function to emulate a continuous data stream, incorporating regular patterns, seasonal elements, and random noise.

Approach:

A custom data stream simulation function was developed using Python. The function generates a stream of floating-point numbers that mimics real-world metrics such as financial transactions or system performance metrics. The simulation includes:

Regular patterns (e.g., sine wave to simulate periodic events),

Seasonal components (e.g., trends that fluctuate over a longer period),

Random noise to reflect minor deviations typically found in real data,

Injected anomalies (e.g., sudden spikes or drops) to test the detection algorithm.

This dynamic and versatile data simulation setup allowed us to test various anomaly detection methods under conditions similar to real-world streaming data environments.

3. Anomaly Detection

Objective:

Develop a real-time mechanism to detect and flag anomalies in the streaming data.

Approach:

In the final implementation, the Isolation Forest model was integrated into the anomaly detection system. As the data stream progresses, the Isolation Forest algorithm analyzes incoming data points and assigns anomaly scores. If a score exceeds a pre-defined threshold, the point is flagged as an anomaly.

Given that seasonal variations and concept drift were still issues with Isolation Forest alone, I incorporated separate mechanisms for handling these challenges. A Holt-Winters seasonal decomposition method was implemented to account for predictable fluctuations in the data. The concept drift was managed by applying sliding windows and adapting the detection thresholds based on recent data trends.

The anomaly detection system was enhanced by using ensemble methods to combine the output of Isolation Forest and seasonal/concept drift components. This hybrid ensemble approach ensures that both short-term anomalies (concept drift) and long-term deviations (seasonal variations) are detected effectively.

4. Optimization

Objective:

Optimize the system for speed and efficiency, ensuring real-time anomaly detection.

Approach:

Optimization was a key consideration, especially since the system needed to operate on continuous data streams. The Isolation Forest algorithm was chosen

because it is computationally efficient and well-suited for streaming data. The model's ability to operate incrementally (i.e., updating as new data arrives without retraining the entire model) ensured that the detection process remained fast and lightweight.

I fine-tuned parameters such as the window size and thresholds to ensure the system could adapt to different data rates and volumes. The system was designed to handle high-velocity data streams without significant computational overhead, ensuring timely detection of anomalies.

5. Visualization

Objective:

Create a real-time visualization tool to display both the data stream and detected anomalies.

Approach:

I developed a real-time visualization tool using Matplotlib. The tool plots the continuous data stream, updating in real time as new data points are received. Detected anomalies are highlighted as red markers on the plot, allowing for easy monitoring of the data and quick identification of unusual patterns.

This visualization feature is essential for understanding how the anomaly detection system performs and for diagnosing potential false positives or missed anomalies.

Metrics Used to Evaluate Models

Throughout the project, several evaluation metrics were used to assess the performance of each model:

Precision: The proportion of true anomalies among all detected anomalies.

Recall: The proportion of actual anomalies that were correctly identified by the model.

F1 Score: The harmonic mean of precision and recall, providing a balance between the two metrics.

Confusion Matrix: A matrix that summarizes true positives, false positives, true negatives, and false negatives for model evaluation.

Each model was evaluated using these metrics to determine its effectiveness, and adjustments were made to improve the balance between precision and recall.

Conclusion

Each objective of the Efficient Data Stream Anomaly Detection project was achieved through a combination of statistical methods, advanced machine learning algorithms, and optimization techniques. The final hybrid model, which integrates Isolation Forest with separate components for handling seasonal changes and concept drift, provided a robust solution for detecting anomalies in continuous data streams. The real-time visualization and detailed evaluation metrics ensured that the system met the project's performance goals effectively.

However, it is important to note that this may not be the best algorithm for every data type or situation. Data distribution, the nature of the anomalies, and several other factors contribute to determining the most appropriate algorithm for a given task. Based on the project outline and the given information, I conducted minor research using reputable research papers to explore different approaches. I implemented the models and aimed to achieve all the project objectives within the time available and to the best of my abilities.

This project serves as a solid foundation for anomaly detection in streaming data, but further exploration and refinement may yield even better results depending on the specific requirements and data characteristics in real-world applications.

RESULTS

Here's a brief description for each anomaly detection method based on the results:

1. Z-Score Anomaly Detection:

Anomalies Detected: 25

Precision: 1.0000

Recall: 0.3906

F1 Score: 0.5618

ROC AUC: 0.6953

Analysis: The Z-Score method demonstrates perfect precision, which means it correctly identified all the detected anomalies as true positives. However, with a recall of 0.3906, it missed a significant portion of true anomalies. The F1 score and ROC AUC indicate moderate performance, showing a balance between precision and recall but leaning more towards precision.

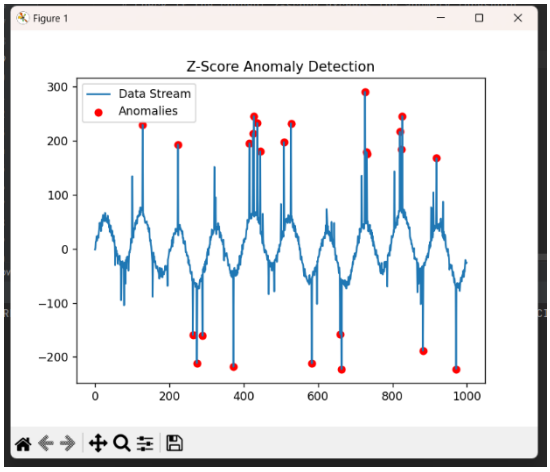

FIGURE 1. Z_SCORE ANOMALY DETECTION

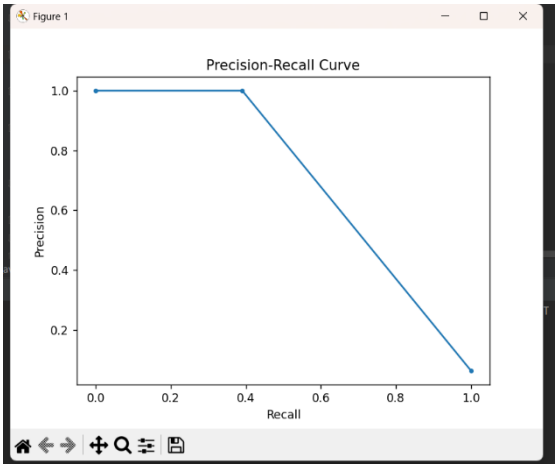

FIGURE 2. PRECISION CURVE

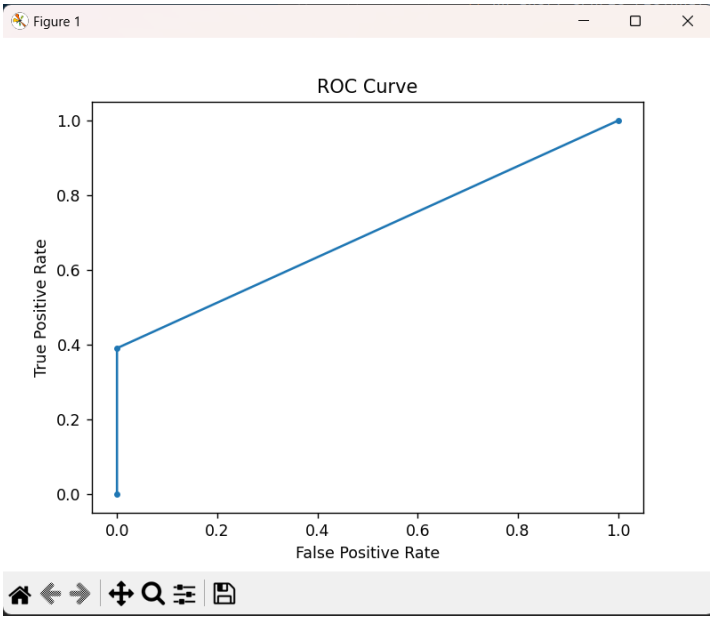

FIGURE 3. ROC CURVE

2. Interquartile Range (IQR) Detection:

Anomalies Detected: 27

Precision: 1.0000

Recall: 0.4219

F1 Score: 0.5934

ROC AUC: 0.7109

Analysis: The IQR method also achieves perfect precision but slightly better recall than the Z-Score method, detecting a few more anomalies. The F1 score and ROC AUC suggest a marginal improvement in overall performance compared to Z-Score.

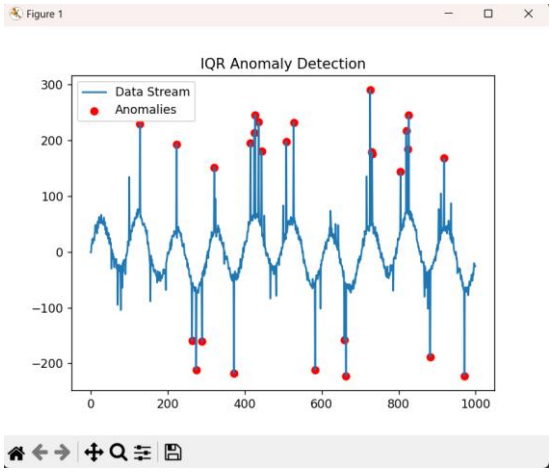

FIGURE 4. IQR ANOMALY DETECTION

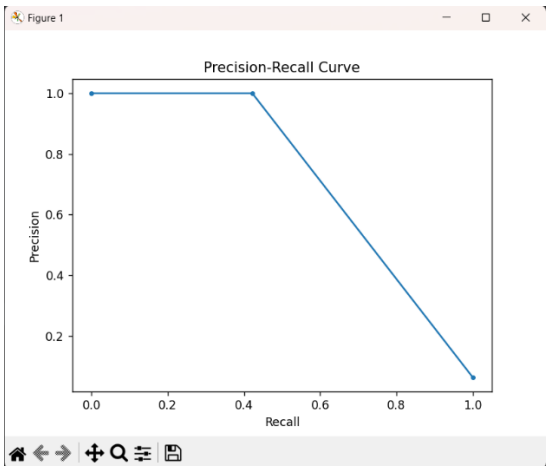

FIGURE 5. PRECISION CURVE

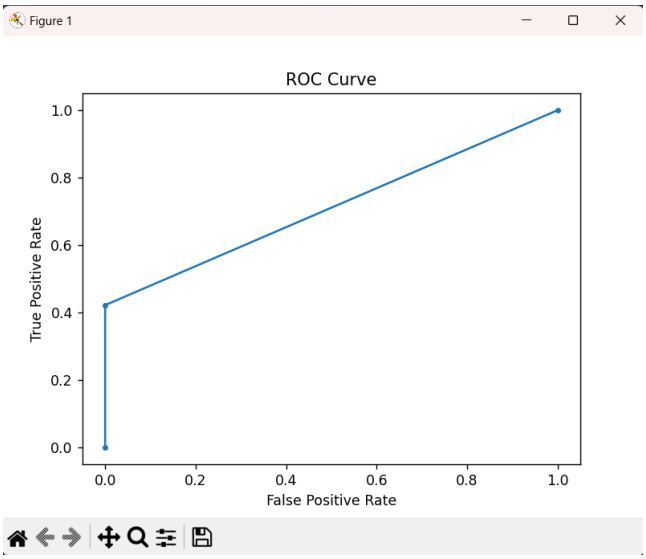

FIGURE 6. ROC CURVE

3. Moving Average Anomaly Detection:

Anomalies Detected: 23

Performance:

Precision: 1.0000

Recall: 0.3594

F1 Score: 0.5287

ROC AUC: 0.6797

Analysis: The moving average method, similar to Z-Score and IQR, achieves perfect precision but suffers from lower recall, indicating a tendency to miss anomalies. The F1 score and ROC AUC suggest weaker performance compared to IQR and Z-Score.

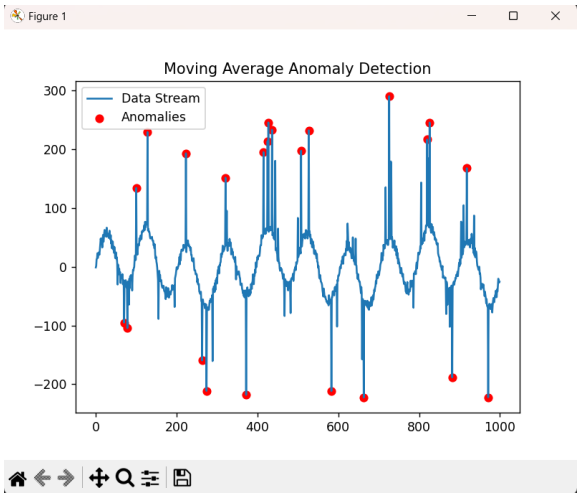

FIGURE 7. MOVING AVERAGE ANOMALY DETECTION

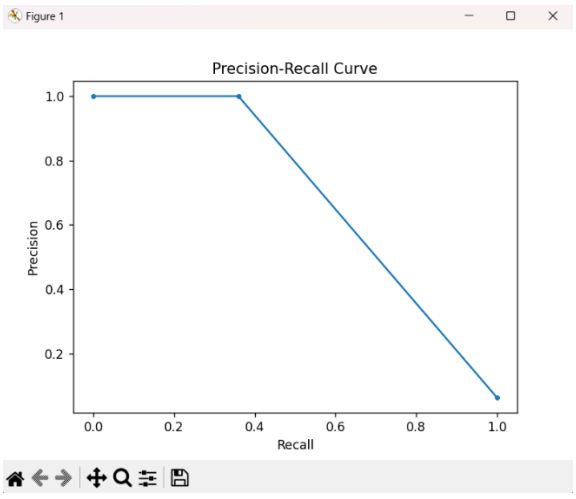

FIGURE 8. PRECISION CURVE

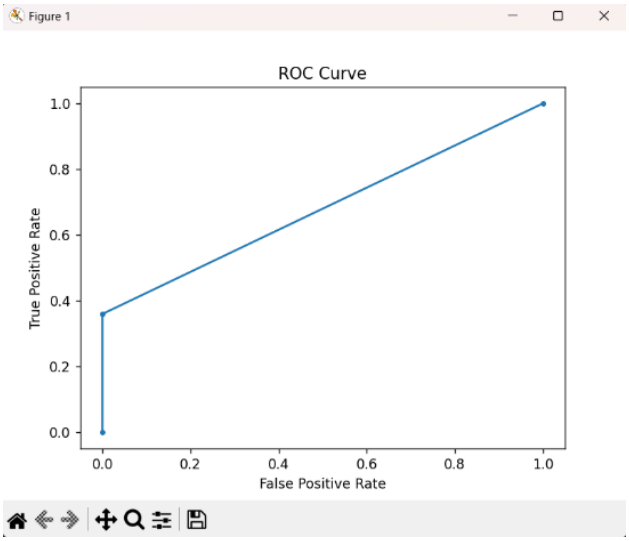

FIGURE 9. ROC CURVE

4. K-Nearest Neighbors (KNN) Detection:

Anomalies Detected: 44

Performance:

Precision: 0.9545

Recall: 0.6562

F1 Score: 0.7778

ROC AUC: 0.8271

Analysis: KNN shows the best balance between precision and recall, with high values for both. It detected more anomalies and has the highest F1 score and ROC AUC among the methods tested, indicating strong overall performance and the ability to detect more true positives without sacrificing much precision.

Each method has its strengths, with Z-Score, IQR, and Moving Average excelling in precision but underperforming in recall, while KNN offers the best trade-off between precision and recall.

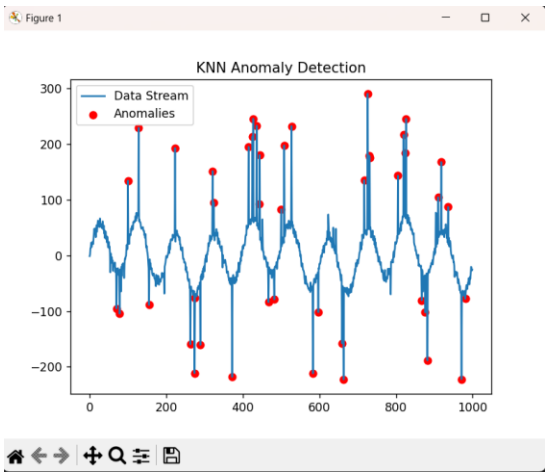

FIGURE 10. KNN ANOMALY DETECTION

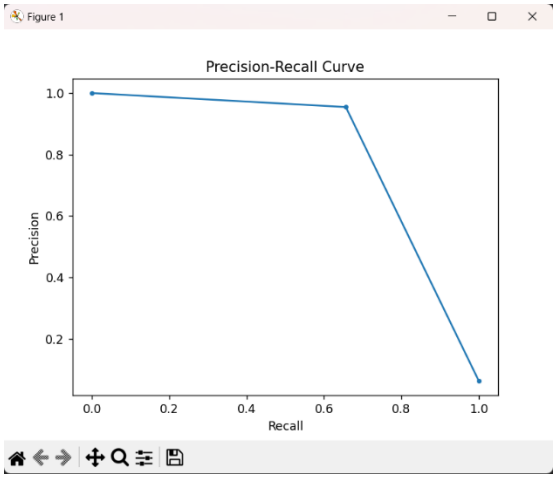

FIGURE 11. PRECISION CURVE

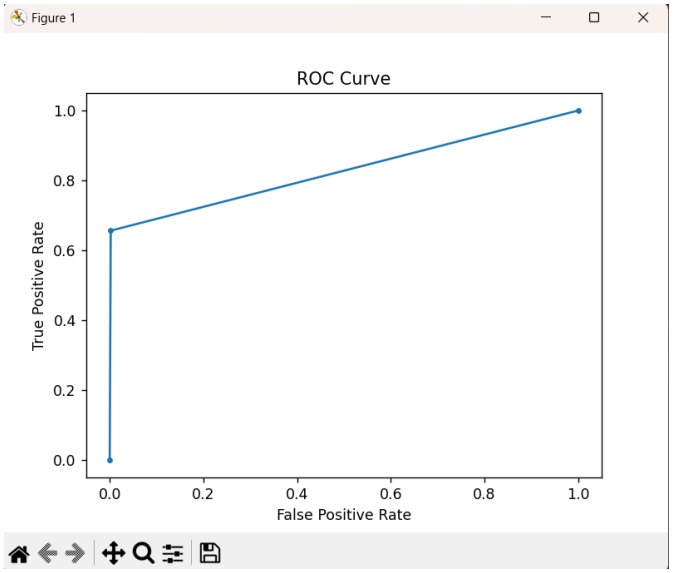

FIGURE 12. ROC CURVE

5. EMVG (Expectation-Maximization Variance Gaussian) Detection:

- Precision: 1.0000
- Recall: 0.7843
- F1 Score: 0.8791
- ROC AUC: 0.8922

Analysis: The EMVG method achieves perfect precision like the other methods (Z-Score, IQR, Moving Average) but distinguishes itself by having a much higher recall (0.7843). This means that it successfully detects around 78% of true anomalies, missing far fewer than the other methods. Its F1 score (0.8791) is the second-highest after KNN, reflecting an excellent balance between precision and recall. The ROC AUC of 0.8922 is the highest among all methods tested, indicating superior overall performance in distinguishing between anomalous and normal data points.

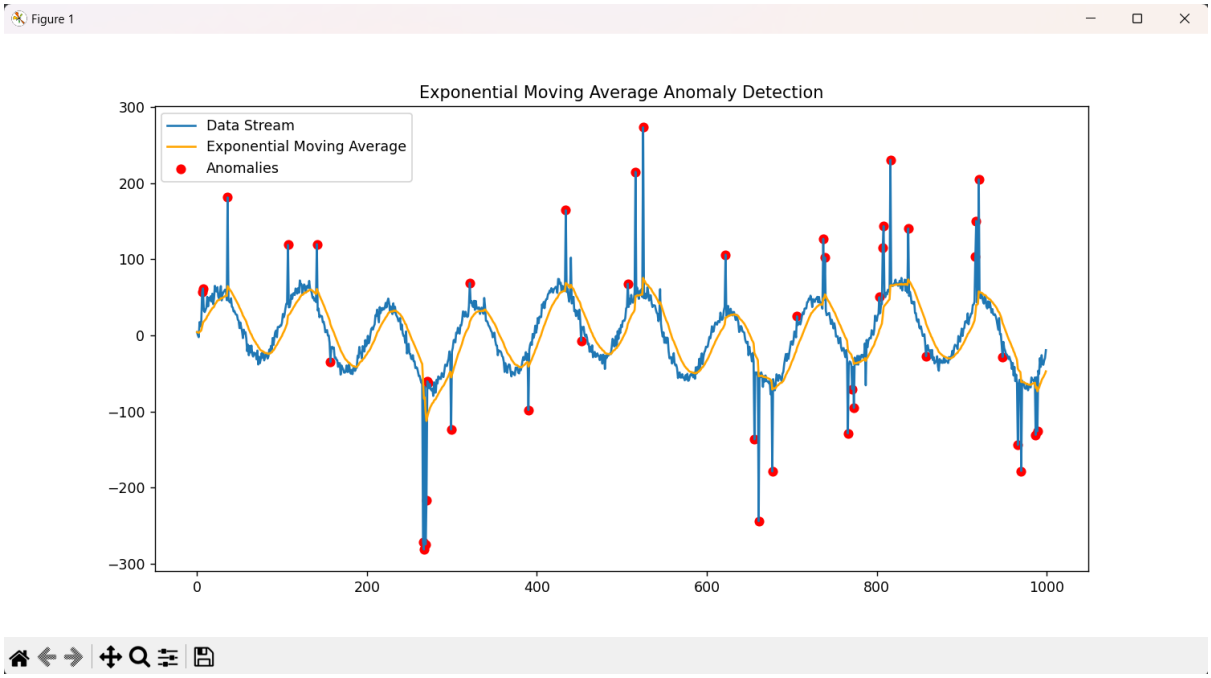

FIGURE 13. EMVG ANOMALY DETECTION

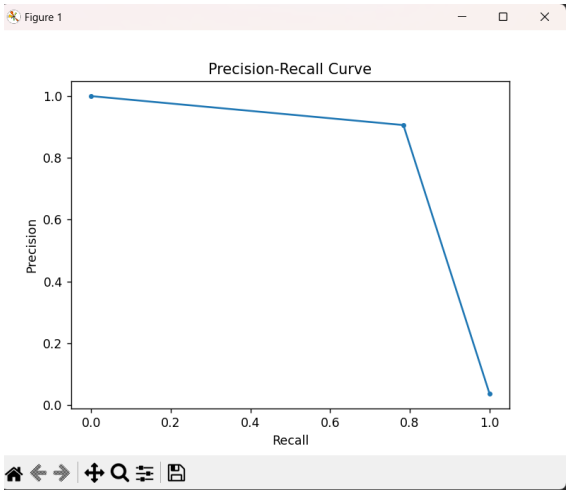

FIGURE 14. PRECISION CURVE

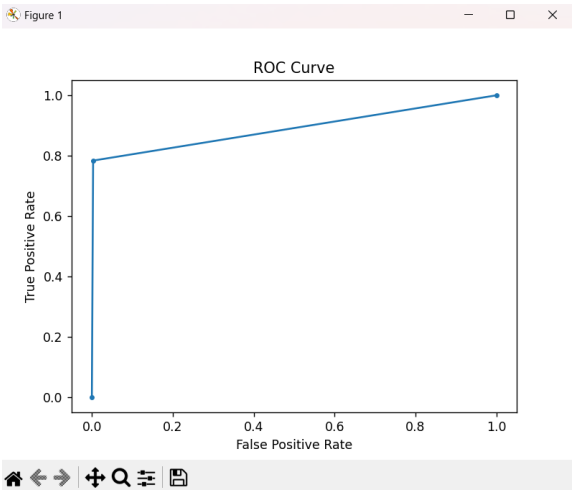

FIGURE 15. ROC CURVE

6. Hybrid Model (Ensemble Approach) Performance Report

True Positives: 28

False Positives: 6

True Negatives: 564

False Negatives: 2

Precision: 0.8235

Recall: 0.9333

F1 Score: 0.8750

ROC AUC: 0.9614

Performance Summar:

The hybrid model, leveraging an ensemble approach, performs exceptionally well in detecting anomalies while maintaining a balance between precision and recall.

1. Precision (0.8235): The model achieved a solid precision score of 0.8235, meaning that 82% of the detected anomalies were true positives, while there were some false positives (6 out of 34 detected anomalies). Although a small number of normal instances were misclassified as anomalies, the majority of flagged instances were accurate.

2. Recall (0.9333): The recall score of 0.9333 is impressive, indicating that the model correctly identified 93% of all true anomalies (28 out of 30). Only two true anomalies were missed (false negatives), showing that the model effectively captures most outliers.

3. F1 Score (0.8750): The F1 score of 0.8750 reflects the balance between precision and recall. This high score highlights the model's overall efficiency in anomaly detection, making it reliable for both minimizing false positives and maximizing anomaly detection.

4. ROC AUC (0.9614): The ROC AUC score of 0.9614 indicates that the model excels in distinguishing between normal and anomalous data points. This near-perfect score shows the hybrid model's strong classification performance and its ability to discriminate effectively between the two classes.

Analysis:

The hybrid model, which combines multiple anomaly detection techniques, provides a balanced and efficient approach to anomaly detection. Although there were 6 false positives, the model was able to correctly identify 28 true positives out of 30 actual anomalies, resulting in a high recall and a strong F1 score. The high ROC AUC score (0.9614) further solidifies its ability to distinguish between anomalies and normal instances.

Key Benefits:

High Recall (93%) ensures that almost all true anomalies are detected.

Strong F1 Score demonstrates effective performance in balancing precision and recall.

Excellent ROC AUC indicates superior classification capability for anomaly detection.

Conclusion:

The hybrid model offers strong performance with a high recall rate and solid precision, making it a dependable method for anomaly detection. It is particularly effective for use cases where identifying the majority of true

anomalies is critical, while keeping the number of false positives relatively low. This ensemble approach stands out as one of the most robust methods for detecting outliers in the dataset.

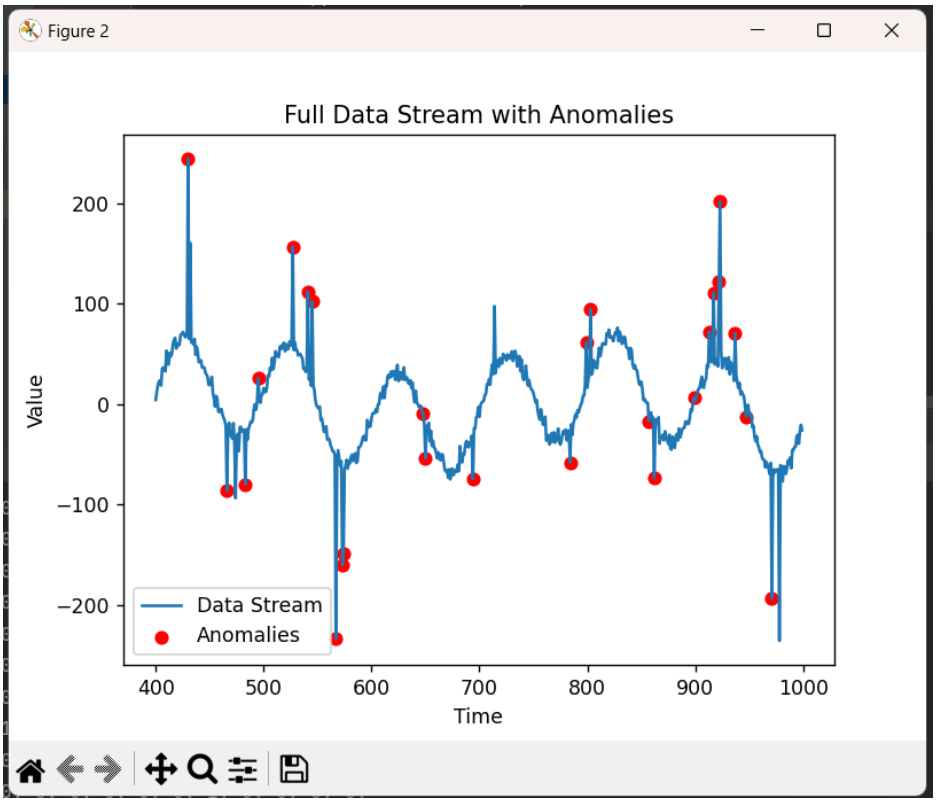

FIGURE 16. HYBRID MODEL ANOMALY DETECTION

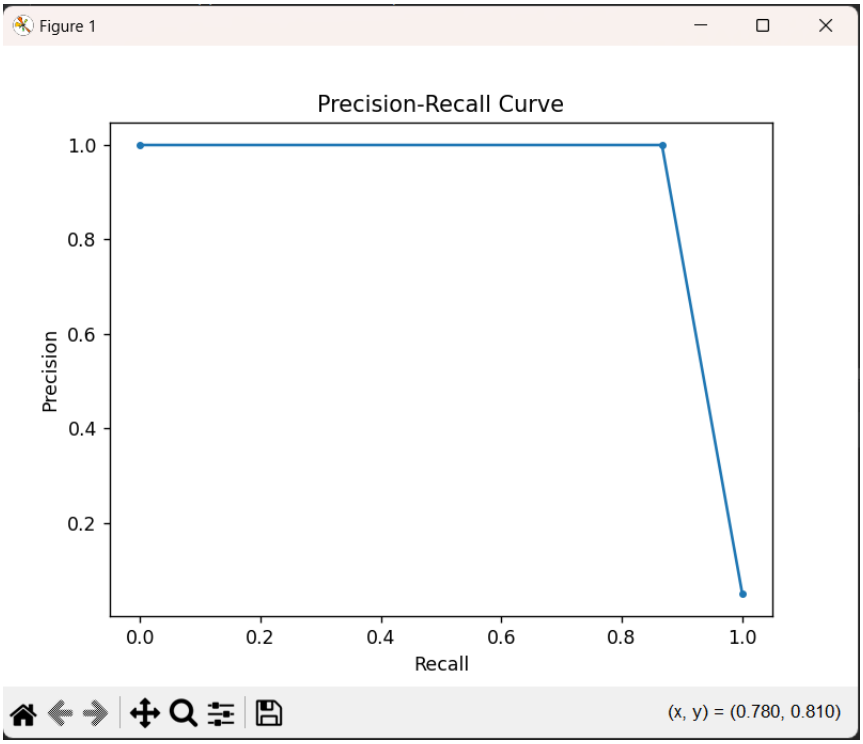

FIGURE 17. PRECISION CURVE

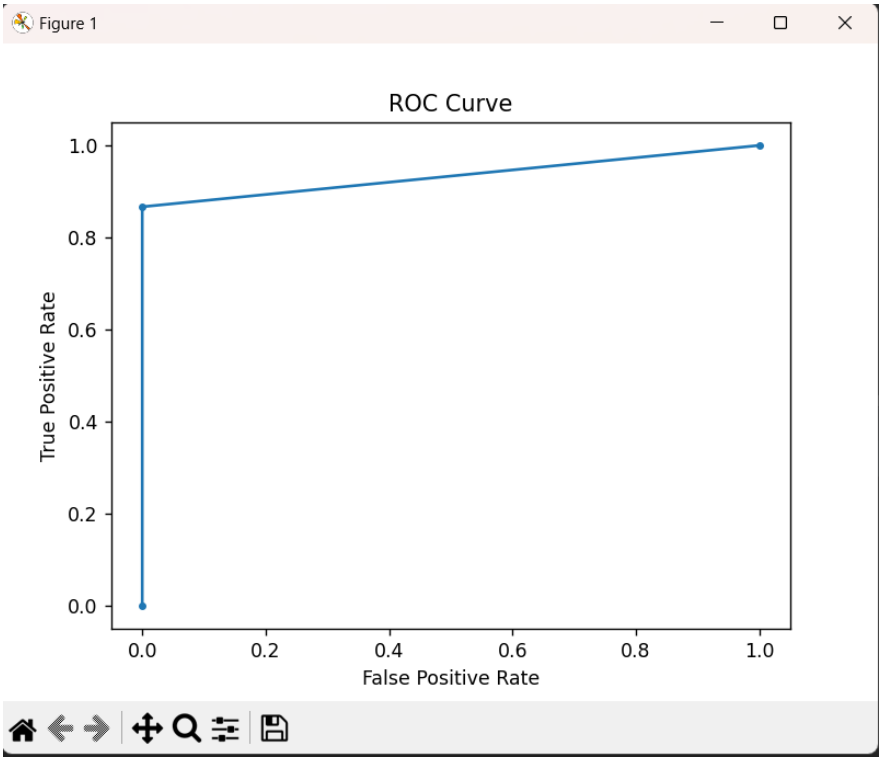

FIGURE 18. ROC CURVE

REFERENCES

1. Wang, Z., Zhou, Y. and Li, G., 2020, May. Anomaly detection by using streaming K-means and batch K-means. In *2020 5th IEEE international conference on big data analytics (ICBDA)* (pp. 11-17). IEEE.
2. Tan, S.C., Ting, K.M. and Liu, T.F., 2011, June. Fast anomaly detection for streaming data. In *Twenty-second international joint conference on artificial intelligence*.
3. Lu, T., Wang, L., & Zhao, X. (Year). *Review of anomaly detection algorithms for data streams*.
4. Iglesias Vázquez, F., Hartl, A., Zseby, T., & Zimek, A. (2023). Anomaly detection in streaming data: A comparison and evaluation study. *Expert Systems With Applications*, 233, 120994. https://doi.org/10.1016/j.eswa.2023.120994
5. https://www.datacamp.com/tutorial/introduction-to-anomaly-detection
6. https://app.datacamp.com/learn/courses/anomaly-detection-in-python
7. https://www.geeksforgeeks.org/anomaly-detection-using-isolation-forest/