Anomaly Detection In Time Series Data With Various Algorithms

Ayberk Tunca

CONTENTS

ABSTRACT

Within the scope of the project, research was conducted on algorithms to detect anomalies in time series data, along with their current uses and variations. To avoid having two separate documents for the research and algorithm experimentation, both aspects have been included in this document. The foundation of the project revolves around detecting anomalies in a graphical representation rather than classifying them as pattern-based or point-based anomalies.

As part of the research, the algorithms' F1 scores were calculated. However, due to the inability to determine the accuracy of these scores against actual anomaly results, they were excluded from further use.

The project successfully obtained the desired outputs, and the algorithms have been interpreted in a way that allows for a general analysis. The obtained results have been used to make comprehensive comments and observations about the performance of the algorithms.

The interpretations of the algorithms were based on the outputs they produced when applied to the time series data. These interpretations helped in gaining insights into how well the algorithms were able to detect anomalies in the data and how effectively they displayed these anomalies on graphical representations.

As a result, the project has achieved its goal of identifying and analyzing various algorithms for anomaly detection in time series data. The interpretations and observations made during the evaluation process have provided valuable information about the strengths and weaknesses of the algorithms. This will be beneficial in determining the most suitable algorithm or combination of algorithms for future anomaly detection tasks.

1. INTRODUCTION

I. Literature Survey

During the literature review, time series data was examined, and various examples were explored. The research encompassed both multi-variable and single-variable datasets without making a distinction between them. Considering the limited research time and the need for experimentation, priority was given to studies focusing solely on timestamps and data values.

This focus was based on the understanding that timestamps and data values play a crucial role in comprehending the content and temporal changes within time series data. The complexity and time-consuming nature of multi-variable analyses were taken into account, leading to a prioritization of studies concentrating on timestamps and data values.

This approach aimed to yield quicker results for time series anomaly detection and to swiftly focus on data analysis. However, it is essential not to overlook that multi-variable datasets could contain valuable insights for anomaly detection as well.

In the obtained data, it was observed that anomaly detection techniques are broadly categorized into 'Classic ML', 'Signal Analysis', 'Data Mining', 'Deep Learning', 'Outlier Detection', and 'Statistics'. During the research, the 'Classic ML' and 'Deep Learning' approaches were given priority.

Among the libraries used during the research, scikit-learn, Keras, TensorFlow, and various plotting libraries were found to be the most commonly utilized ones.

Furthermore, it was discovered that the 'Outlier Detection' techniques are frequently employed as general anomaly detection methods in contemporary applications.

II. Objective Of The Study

In this project, publicly available time-series datasets were examined for research purposes, and it was noticed that the 'NAB' (Numenta Anomaly Benchmark) GitHub datasets were commonly used. Due to the availability of knowledge about the reasons for anomalies and the anomalies present in the data, the 'New York Taxi Dataset' was specifically chosen and named as 'nyc_taxi.csv' for use in the project. The values in the dataset represent the number of people using taxis.

The dataset contains 5 anomalies, and the reasons for these anomalies are linked to specific events in New York City, including the New York Marathon, Thanksgiving, Christmas, New Year's Day, and a snowstorm. These events might have caused significant fluctuations in taxi usage patterns, leading to anomalies in the data.

Using this dataset in the project allows for the analysis and evaluation of different anomaly detection algorithms and techniques in the context of real-world anomalies related to special events and exceptional situations in New York City's taxi usage patterns.

After finalizing the dataset selection, the necessary algorithms were decided, and the names of these algorithms will not be mentioned here, as they will be discussed in detail in the upcoming sections. However, it is essential to add a point regarding the use of the K-Means algorithm to visualize the data distribution.

The K-Means algorithm was used to visualize the data distribution in the dataset. By applying K-Means clustering, the data points were grouped into clusters, providing insights into the underlying patterns and structure of the data.

Additionally, it is worth mentioning that the LSTM algorithm was utilized not only for detecting outliers but also for predicting approximate values in the data. As LSTM can capture temporal dependencies, it was employed to predict values and identify outliers simultaneously.

For the other anomaly detection algorithms, different parameter settings were explored to obtain varied results. This led to the categorization of these algorithms into two groups: "worst-case" and "best-case" scenarios. In the subsequent sections, these scenarios will be discussed in detail, showcasing the performance of the algorithms with various parameter configurations.

DATA ANALYSIS

The project's aim is to perform anomaly detection on a given dataset using time-series data. In this context, the first step is to examine the data thoroughly and provide relevant comments and observations.

```
(10320, 2)
             timestamp  value
0  2014-07-01 00:00:00  10844
1  2014-07-01 00:30:00   8127
2  2014-07-01 01:00:00   6210
                 timestamp  value
10317  2015-01-31 22:30:00  27309
10318  2015-01-31 23:00:00  26591
10319  2015-01-31 23:30:00  26288
```

As mentioned earlier, the dataset consists of two columns, namely 'timestamp' and 'value'. To facilitate better understanding and visualization of the data, it was decided to organize the timestamps into separate columns representing year, month, day, and week.

Table 1 Shape, first and last three element of the data.

timestamp	value	anomaly	year	month	week	day	hour	minute
2014-07-01 00:00:00	10844	0	2014	7	27	1	0	0
2014-07-01 00:30:00	8127	0	2014	7	27	1	0	30
2014-07-01 01:00:00	6210	0	2014	7	27	1	1	0

Table 2 Organized Data With Timestamps Adapted

Visualizing the data is crucial for examining it effectively. In this context, the necessary methods were employed to visualize the data using the Seaborn, Matplotlib, and Holoviews libraries within the project. The obtained results were carefully analyzed, and predictions were made before proceeding with anomaly detection.

By using Seaborn, Matplotlib, and Holoviews, various types of plots and visualizations were created to gain insights into the dataset's patterns, trends, and seasonality. These visualizations allowed for a better understanding of the data distribution and potential correlations between the timestamp and value columns.

Additionally, predictions were made based on the data to estimate the expected behavior of the time series before anomaly detection. These predictions provided a baseline for comparing the actual values to identify any deviations or anomalies.

These visualization and prediction steps were essential in preparing the data for anomaly detection and for gaining valuable context before applying the anomaly detection algorithms. The analysis of these results served as a crucial step in understanding the dataset's characteristics and patterns, aiding in the subsequent anomaly detection process.

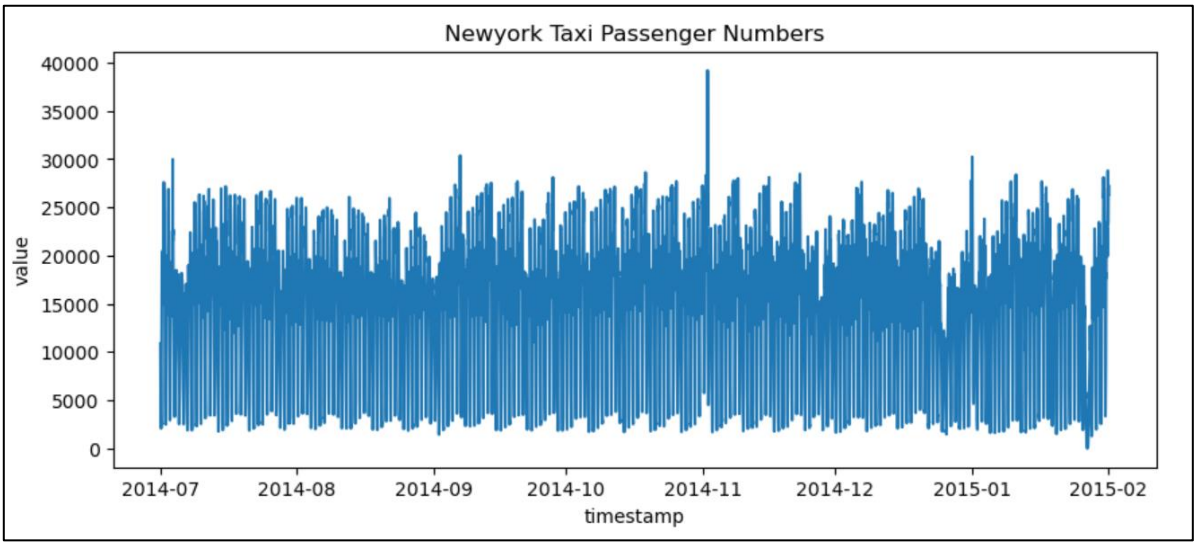


Figure 1 Newyork Taxi Passenger Data From 07/2014 - 02/2015

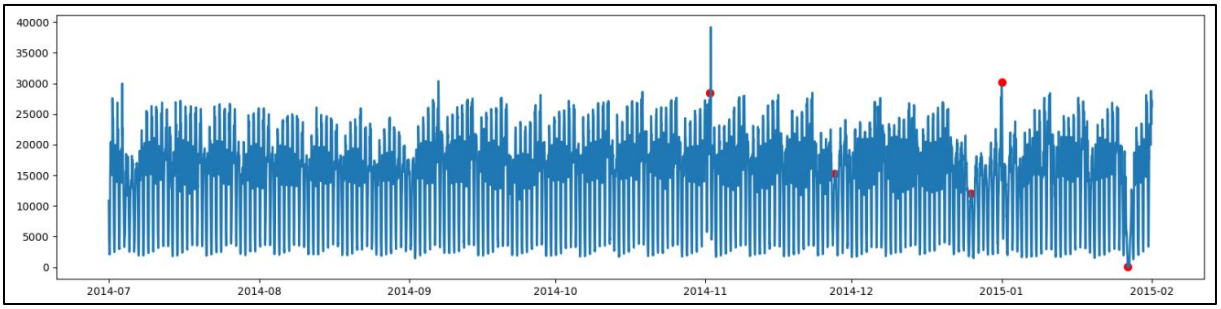


Figure 2 Newyork Taxi Passenger Data With Known Anomalies

Upon examining the data, it is evident that the data follows a certain pattern. However, there are instances of sudden spikes and drops that need to be accounted for. In this context, certain known events come to mind, such as New Year's Eve, marathons, and other special occasions. These events can explain these anomalies in the data, and the graphs also corroborate these occurrences.

We can confidently say that we are on the right track and have accurately represented the given anomaly times on the graphs to validate our findings. This careful analysis and correct representation of the data and anomalies indicate that we are making progress in the right direction. The graphs effectively confirm the presence of known anomalies and help us understand the data better, which will be instrumental in the subsequent anomaly detection process.

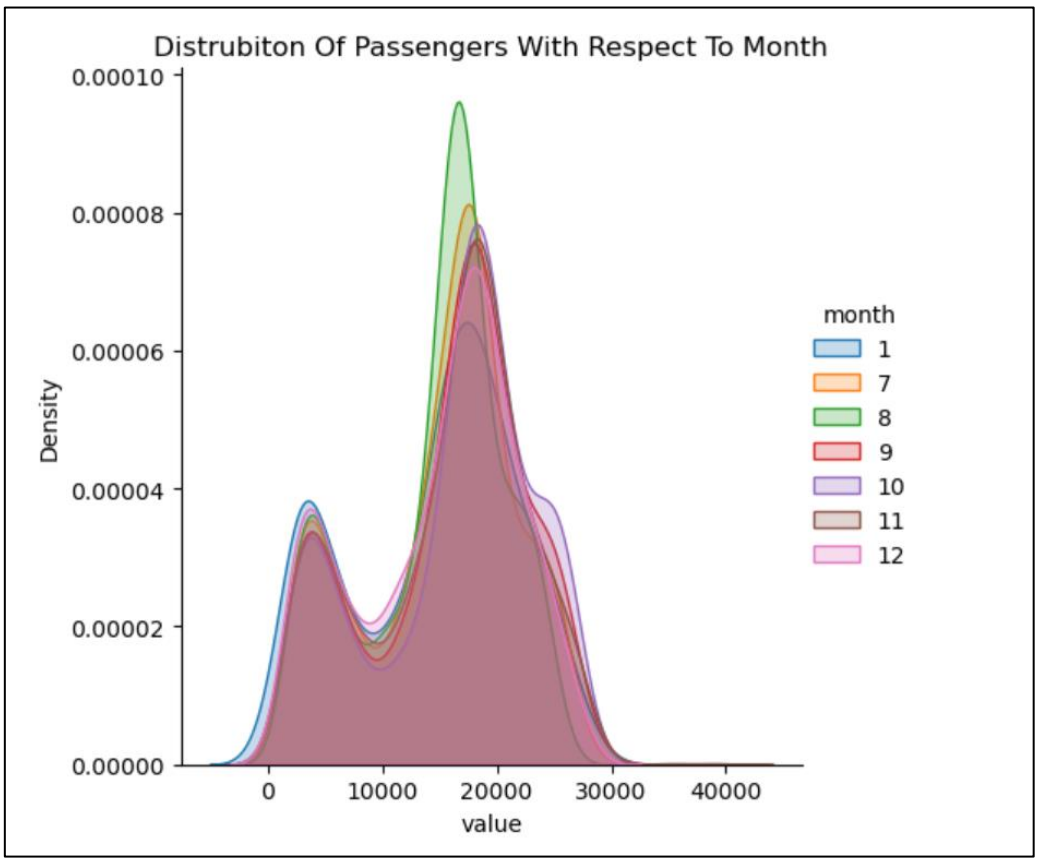


Figure 3 Data Distribution With Respect The Month

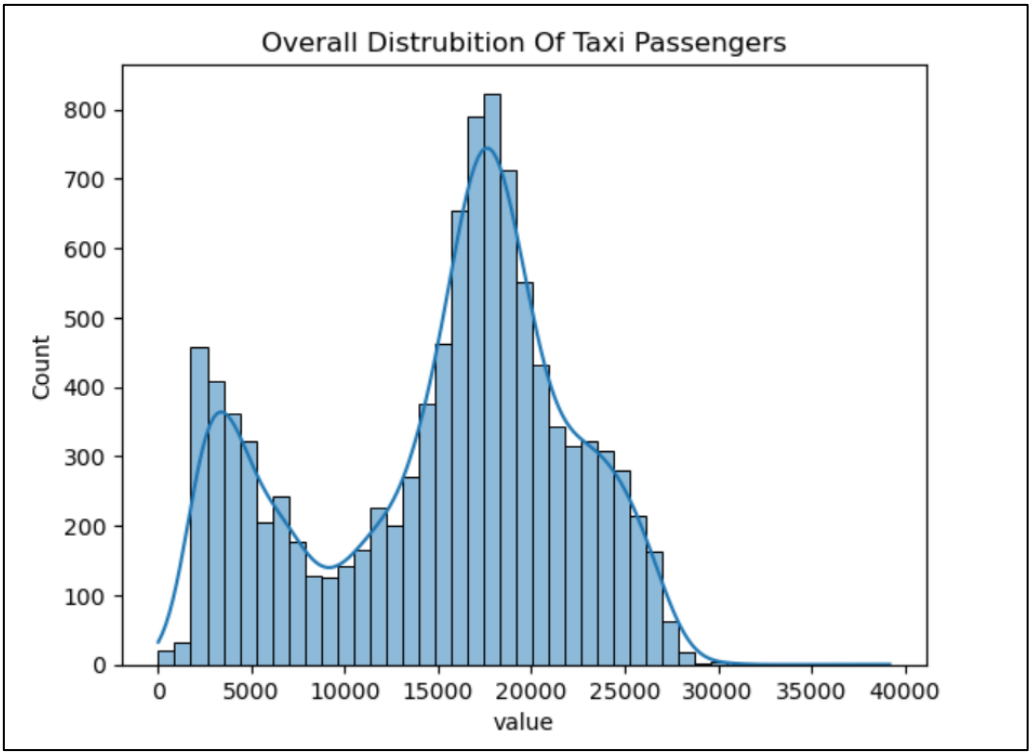


Figure 4 Overall Distribution Of The Data

When examining the general distribution of the data, it can be observed that the most common passenger count hovers around 17,000. Looking at the data on a monthly basis, this value is most frequently repeated in the 8th month.

As correctly pointed out, these patterns align with the special dates or events you mentioned earlier, such as New Year's Eve, marathons, etc. During these special periods, there are corresponding months where the passenger count increases or decreases, and this is reflected in the data.

The analysis of the data distribution and its correlation with specific events provides valuable insights into the patterns and trends within the dataset. Understanding these fluctuations will be crucial in detecting anomalies and identifying abnormal behaviour in the time series data. It allows for a more informed approach to anomaly detection and improves the accuracy of anomaly detection algorithms by taking into account the known patterns associated with special events.

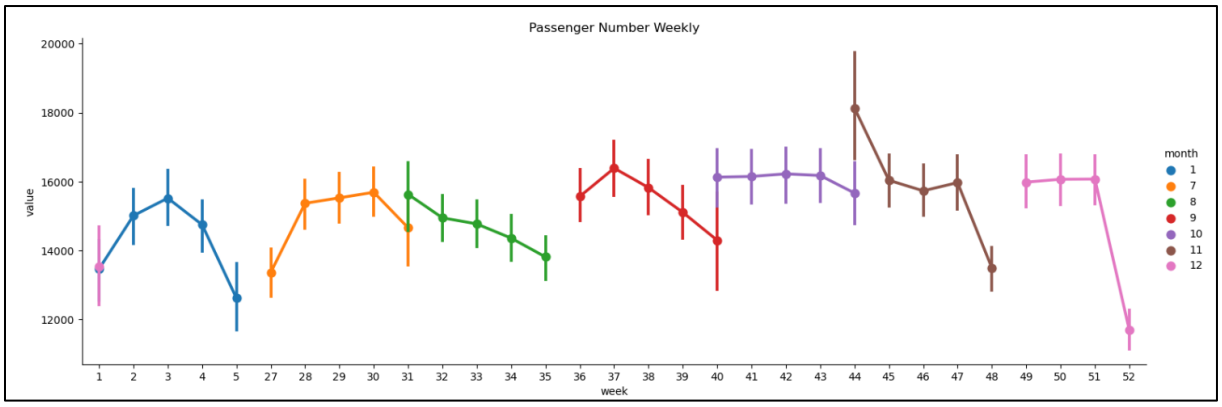


Figure 5 Passenger Number Distribution Weekly

Indeed, when examining the data on a weekly basis, we can observe the distribution of the data and its patterns more clearly. When comparing the data on a monthly basis, we can see that the most drastic changes occur during the 1st, 11th, and 12th months. Additionally, there is a noticeable change in the 9th and 10th months.

These variations in passenger counts during specific months align with the known special events or occasions you mentioned earlier. The data reflects the increased or decreased passenger count during these months, indicating that the anomalies we observed earlier are indeed linked to these events.

Understanding these weekly and monthly patterns is essential in identifying normal behavior and distinguishing anomalies effectively. This insight will be valuable in the subsequent stages of anomaly detection and will aid in the selection and fine-tuning of anomaly detection algorithms to accurately capture deviations from the regular patterns observed in the data.

REALISTIC LIMITS CONDITIONS AND CONSTRAINTS TAKEN INTO ACCOUNT IN THE DESIGN OF THE PROJECT

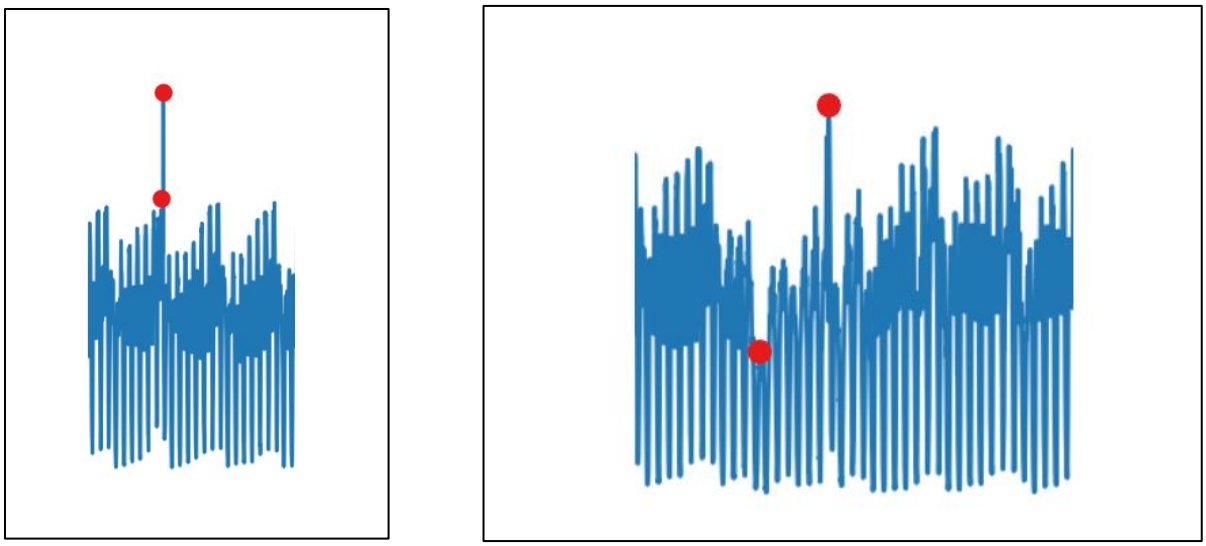
Physical Limitations and Constraints	The accessibility and quality of the data could have an effect on how well the project performs. Results for anomaly identification and forecasting may be less reliable if the data is incomplete or inaccurate. The time and effort required for hyperparameter tuning and model optimization may be constrained by project deadlines or resource limitations. The chosen machine learning models have their own limitations, and some may perform better on specific types of data than others.
Engineering Norms and Standards	The project should adhere to standard data preprocessing techniques, including data cleaning, normalization, and feature engineering, to ensure the quality and consistency of the data used for modeling. The machine learning models employed in the project should be implemented following best practices and industry standards to ensure robustness and reliability.
Economic Aspects	The selection of machine learning models and algorithms should take into account the computational resources required for training and inference, as well as any associated costs.
Environmental Aspects	To lessen the carbon footprint, energy efficiency and resource consumption should be considered while choosing machine learning models and algorithms.
Sustainability	Transparency and openness in the project's processes should be given top priority because they make it simpler for stakeholders to comprehend and believe in the outcomes. To adapt to evolving sustainable practices and technologies, it should be encouraged to develop continuously and to learn new things during the project.
Ethical Aspects	The project should place a high priority on ethical data collection and use, making sure that all data is acquired with consent and in accordance with privacy laws. Anomaly detection model development and deployment should be done fairly and responsibly, avoiding prejudice and discrimination.
Security Aspects	Data encryption should be employed during data transmission and storage to safeguard against data breaches.
Manufacturability	In a manufacturing context, the selected anomaly detection models and algorithms should be implementable and scalable. The hardware and software requirements for implementing the models in a production environment should be taken into account.

ANOMALY DETECTION AND TYPES

I. Anomaly Detection

Anomaly detection, also known as outlier detection, is a technique used to identify patterns or instances that deviate significantly from the norm or expected behavior within a dataset. In other words, it aims to find data points that are rare, unusual, or suspicious compared to the majority of the data.

The "normal" behaviour in the dataset is usually represented by a pattern or regularity, and anomalies are data points that do not conform to this pattern. These anomalies can be caused by various factors, such as errors, fraud, defects, rare events, or significant changes in the underlying process.



Example 1 Detected Anomalies From 'nyc_taxi'

Cybersecurity, medicine, machine vision, statistics, neurology, law enforcement, and financial fraud are just a few fields where anomaly detection is used. To help with statistical analysis, such as computing the mean or standard deviation, anomalies were originally examined for obvious rejection or exclusion from the data.

There are various techniques for anomaly detection, including statistical methods, machine learning algorithms, clustering approaches, and time-series analysis. Each technique has its strengths and weaknesses, and the choice of the appropriate method depends on the specific characteristics of the dataset and the requirements of the application.

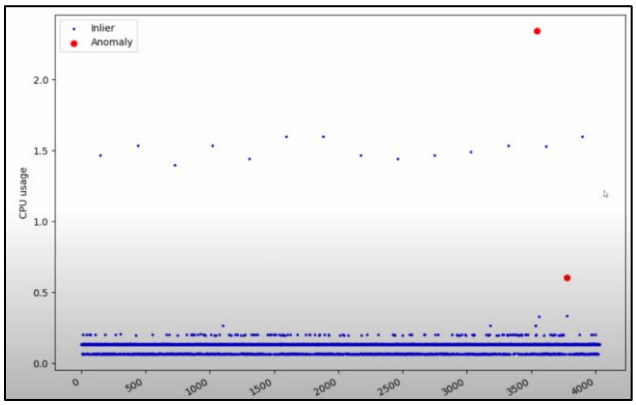
II. Types Of Anomaly Detection Tasks

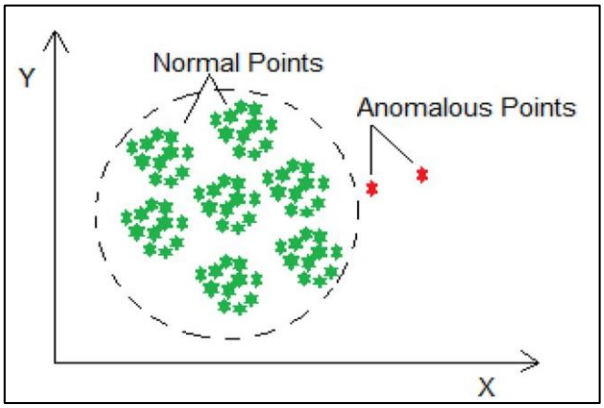
It can be appropriate to summarize the topic of Anomaly detection under two main headings. These headings can also indicate the type of data. These headings can be classified as "Point-based anomaly" and "Pattern-based anomaly."

Point Based Anomaly

The purpose of point-based anomaly detection is to locate specific data points that significantly vary from the rest of the dataset. These points are known as "outliers" because, in comparison to the majority of the data, they stand out as exceptional or unusual findings. Point-based anomalies are distinguished by their uniqueness and do not always display any particular pattern or behaviour.

Point-based anomaly detection is a powerful tool for identifying individual instances that may require further investigation due to their uniqueness or potential significance in a dataset. It can complement other anomaly detection techniques and help uncover specific outliers that might not be apparent through pattern-based methods.





Example 2 Point Based Anomalies

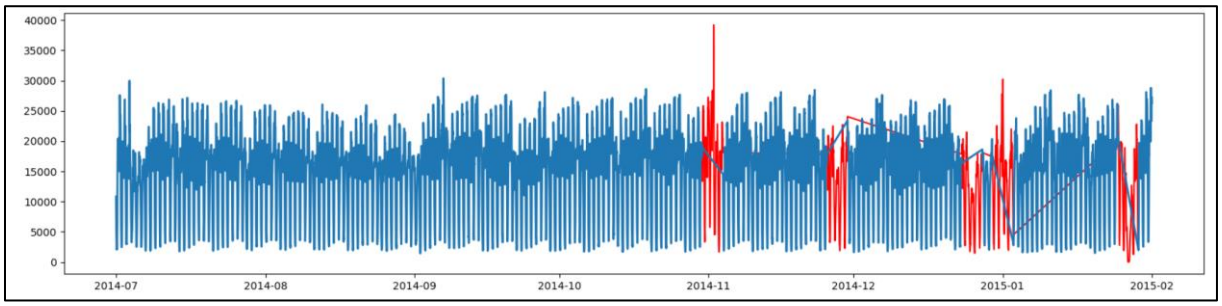
Pattern Based Anomaly

The purpose of pattern-based anomaly identification is to locate anomalies based on out-of-the-ordinary patterns or variations from the typical behaviour shown in the dataset. Pattern-based anomalies, as opposed to point-based anomalies, involve a number of data points that come together to generate unusual patterns or behaviours.

The process of finding unusual patterns or behaviours in groupings of data points that deviate considerably from the dataset's regular patterns is known as pattern-based anomaly detection.

Clustering algorithms may be used in pattern-based anomaly detection to group together related data points. The data points that do not belong to any cluster or form a separate, independent cluster are then referred to as anomalies.

To identify the common patterns in the data, machine learning algorithms and pattern recognition techniques are frequently used. Data points that do not fit the established patterns are referred to as anomalies.



Example 3 Pattern Anomaly Representation of 'nyc_taxi' Data

When we examine our data, we can actually observe that it follows a certain pattern. Instead of including the image of the data again to avoid clutter, we can say that if we had chosen to focus on pattern-based anomaly detection, we could have created a graph like this to illustrate the average pattern and how it might look with anomalies.

ANOMALY DETECTION ALGORITHMS

I. Isolation Forest

A form of unsupervised machine learning called the Isolation Forest algorithm is used to find anomalies or outliers in datasets. By separating anomalies into distinct trees within a random forest of it's operation. Isolation Forest's basic idea is that, in contrast to the vast majority of typical data points, anomalies are comparatively uncommon and distinctive. As a result, they can be isolated with fewer random splits. The procedure starts by picking a portion of the data from the original dataset at random. When every data point is separated into its own leaf node, it then generates distinct decision trees by recursively splitting the data using random feature splits.

Each data point's average path length from the root to the leaf node is calculated, and anomaly scores are determined using these path lengths. Anomalies may be easier to isolate when the path lengths are shorter. Longer paths, on the other hand, reflect normal data points. The anomaly scores are used to establish a threshold for categorizing data points as anomalies or normal. Isolation Forest works well for detecting anomalies in a variety of applications, including fraud detection, network intrusion detection, and defect identification in industrial systems. It is computationally efficient, especially for big datasets.

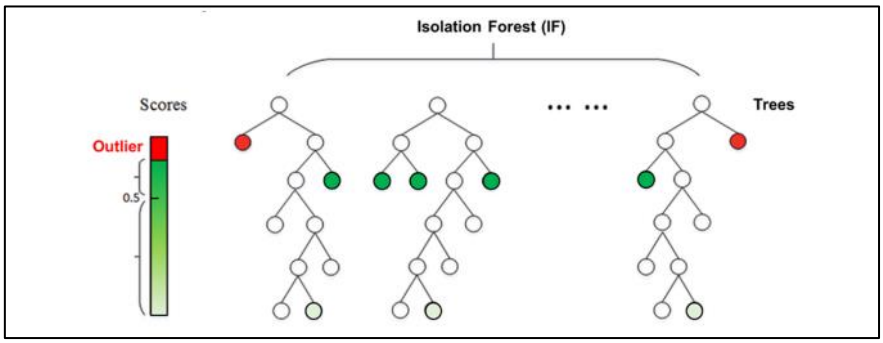


Figure 6 Isolation Forest Example

The anomaly score can be calculated as follows

Anomaly Score = 2^(-h(x) / c(n))

Here, h(x) represents the average path length from the root to the leaf node for a specific data point, and c(n) is the average path length of an unsuccessful search in a binary search tree with n nodes. The smaller the path length (h(x)) and the larger the value of c(n), the higher the anomaly score will be, indicating a higher likelihood of the data point being an anomaly. By comparing the anomaly scores of data points to a threshold value, anomalies can be identified in the dataset.

After randomly selecting a data point, the process of calculating the h(x) value involves building an isolation tree using a subset of data points (S) and determining the average depth of the nodes in the tree. The h(x) value is then calculated as h(x) = 1 + c(|S|), where c(|S|) is the average depth of nodes in a binary tree with |S| data points.

Once the anomaly scores are computed, a threshold value is determined. Data points with anomaly scores below the threshold are considered normal, while those with scores above the threshold are labelled as anomalies. The threshold can be set based on the specific requirements of the anomaly detection task and can be adjusted to achieve the desired level of sensitivity and specificity in identifying anomalies.

By comparing the anomaly scores of each data point to the threshold value, the algorithm can distinguish between normal and anomalous data points, enabling the identification of anomalies in the dataset. This threshold-based comparison is a crucial step in the Isolation Forest algorithm, as it defines the boundary between normal and anomalous behaviour in the data.

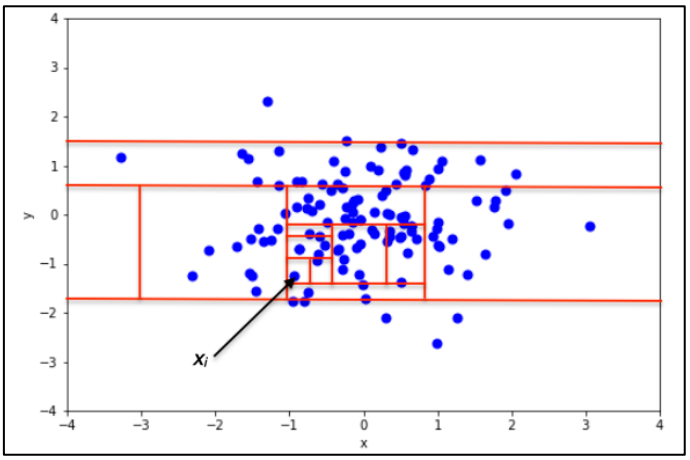


Figure 7 Isolation Forest Representation

When using the Isolation Forest library under the scikit-learn framework, there are several important parameters to consider. These parameters enable you to fine-tune the Isolation Forest algorithm and customize it according to your specific anomaly detection needs. The key parameters include:

class `sklearn.ensemble.`**`IsolationForest`**(**, n_estimators=100, max_samples='auto', contamination='auto', max_features=1.0, bootstrap=False, n_jobs=None, random_state=None, verbose=0, warm_start=False*)[source]

n_estimators: This parameter determines the number of isolation trees to be created. Increasing the number of trees can improve the accuracy of anomaly detection, but it may also increase computation time.

max_samples: It sets the number of data points to be randomly subsampled for each tree. Typically, it is set to a default value of "auto," which uses a fraction of the dataset, such as 256, or it can also be set to a float value, e.g., 0.8 (80% of the data).

max_features: This parameter specifies the number of features to be randomly selected for splitting at each node of the decision tree. It is set to "auto" by default, which means it uses all features. Alternatively, it can be an integer or a float value (e.g., 0.5) to use a fraction of features.

contamination: The contamination parameter sets the proportion of anomalies in the dataset. It is set to "auto" by default, where the estimated proportion of anomalies is derived based on the proportion of outliers in the data (approximately 0.1%).

bootstrap: It specifies whether to use bootstrapping when building each tree. The default value is "True," which means bootstrap samples are used.

n_jobs: This parameter sets the number of CPU cores to be used during training. If set to -1, it uses all available cores. For smaller datasets, it is often set to -1, and for larger datasets, it may be beneficial to set it to a specific number.

In simple use cases of the Isolation Forest algorithm, the two most commonly adjusted parameters are 'contamination' and 'n_estimators.'

II. Local Outlier Factor

An method for detecting anomalies called Local Outlier Factor (LOF) bases its capacity to find anomalies on the local density of a data point. As compared to other anomaly detection techniques, it concentrates on finding locations in less populated areas.

By comparing the density of each data point's neighbors to that point's own density, LOF determines the anomaly score for each data point. A data point is regarded as an anomaly if the density of its neighbors is significantly lower than the density of the point itself. As a result, LOF is able to find anomalies regardless of the data's overall distribution and discover local abnormalities depending on density.

LOF determines the k-nearest neighbors for each data point. The proximity between data points is often determined in that phase using a distance metric, frequently the Euclidean distance.

After data point's local density score is calculated. The density of a data point's k-nearest neighbors may be represented by this score. LOF is calculated by dividing a data point's local density by that of its k-nearest neighbors. A point is deemed an anomaly if its LOF value is more than 1 and it has a lower density than its surrounding areas.

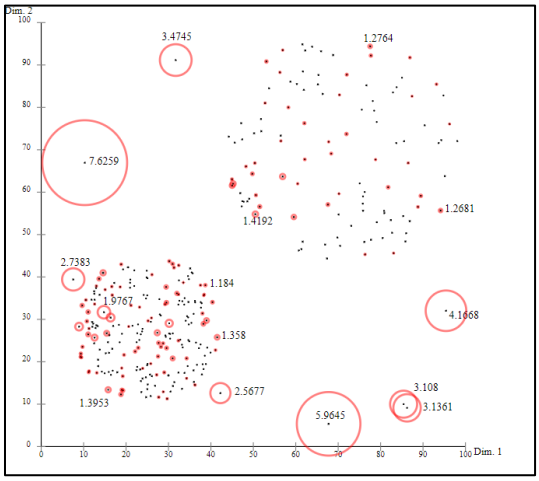


Figure 9 Example Of LOF

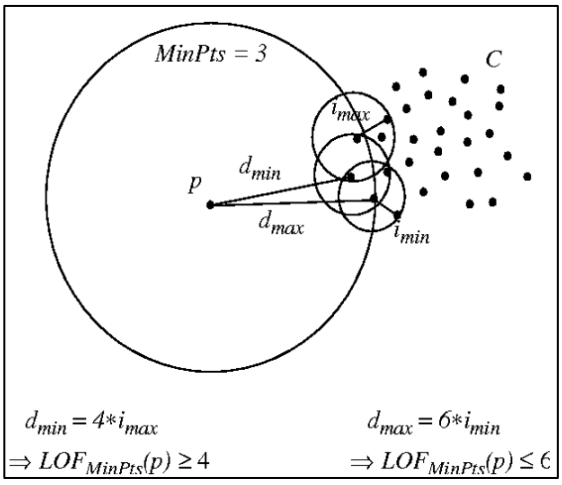


Figure 8 Finding Neighbors Of LOF

The Euclidean Distance can be calculated as the square root of the squared difference between the temporary points x and y. The reason for squaring this operation is to avoid potential negative values that could disrupt the algorithm. This method is commonly used in mathematics and statistics. As mentioned in the previous text, the reachability distance is defined as the length between the selected data point and its neighbor within the limits of k-nearest neighbors.

In the Local Outlier Factor (LOF) algorithm, there are several parameters that can be tuned to customize the behavior of the algorithm. Here are the key parameters in the scikit-learn library for the LOF algorithm:

n_neighbors: This parameter defines the number of neighbors to consider when calculating the local density of a data point. It determines the size of the neighborhood around each point. A higher value will result in a smoother density estimation, while a lower value may capture more local variations.

algorithm: This parameter specifies the algorithm used to compute the nearest neighbors. The options include "auto," "ball_tree," "kd_tree," or "brute." The "auto" option automatically chooses the best algorithm based on the input data and other factors.

metric: The distance metric used to calculate the distance between points. The default is "minkowski," which corresponds to the Euclidean distance. Other options include "manhattan" for Manhattan distance and "chebyshev" for Chebyshev distance, among others.

contamination: This parameter represents the proportion of outliers in the data. It is used to define the threshold for classifying data points as outliers. The default value is "auto," which estimates the contamination based on the proportion of outliers in the data.

novelty: If set to "True," the LOF algorithm is used for novelty detection, where it is trained on normal data and then used to detect outliers in new, unseen data. If set to "False" (default), it performs outlier detection on the given data.

n_jobs: The number of parallel jobs to run when computing the k-nearest neighbors. Setting this parameter to -1 uses all available CPU cores for computation.

III. Robust Z- Score

The Median Absolute Deviation (MAD) Z score, commonly referred to as the Robust Z score, is a statistical technique for identifying outliers. It is a variant of the common Z score, which is employed to spot data points that considerably disappear from the dataset's mean.

The amount of standard deviations a data point is from the data mean is used to compute the standard Z score. However, when the dataset contains extreme values, the conventional Z score is particularly susceptible to outliers.

Through the use of the Median Absolute Deviation (MAD) rather than the standard deviation, the Robust Z score resolves this sensitivity to outliers. The variability or dispersion of the data is measured by the MAD, which is less impacted by extreme values compared to the standard deviation.

The Robust Z score addresses this sensitivity to outliers by using the Median Absolute Deviation (MAD) instead of the standard deviation. The MAD is a measure of the variability or dispersion of the data, which is less affected by extreme values compared to the standard deviation.

The formula for calculating the Robust Z score for a data point x is as follows:

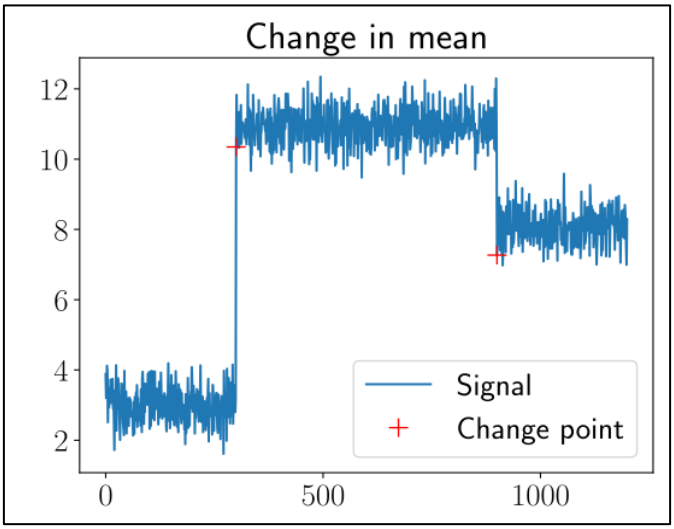
Robust Z score (RZ) = 0.6745 * (x - Median) / MAD

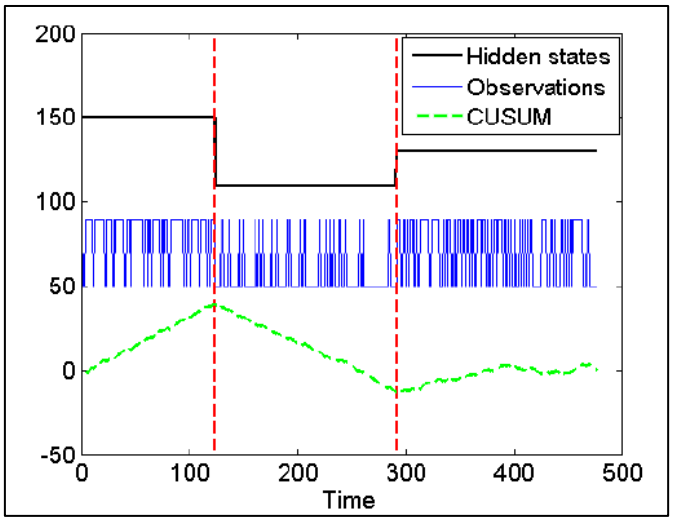
$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD}$$

IV. Change Point Detection

A statistical technique named change point detection, also known as change detection or change point analysis, is used to spot sudden changes or shifts in a time series or sequence of data. Finding the data points where the underlying statistical features, such as mean, variance, or distribution, significantly change is the objective.

There are various methods for change point detection, and the choice of the method depends on the characteristics of the data and the specific application. One of the common technique is Cumulative Sum Algorithm which is what used in the project. It basically tracks the cumulative sum of the differences between observed and reference value. It outputs anomaly as when cumulative sum exceeds the defined threshold.





V. Hotelling T^2

A statistical technique for multivariate analysis called Hotelling's T-squared (T2) is particularly useful for comparing the means of two or more groups of data. To handle many variables at once, it expands the univariate t-test.

To ascertain whether the means of two groups (or more) in a multivariate context are substantially different from one another is the main objective of Hotelling's T2 test. In other words, it evaluates the impact of the difference between the means of two groups when a number of factors are taken into account simultaneously.

The degrees of freedom associated with the sample sizes and number of variables are used to compare the computed T2 statistic to a critical value from the F-distribution. When the T2 value is more than the critical value, there is a considerable difference between the group means.

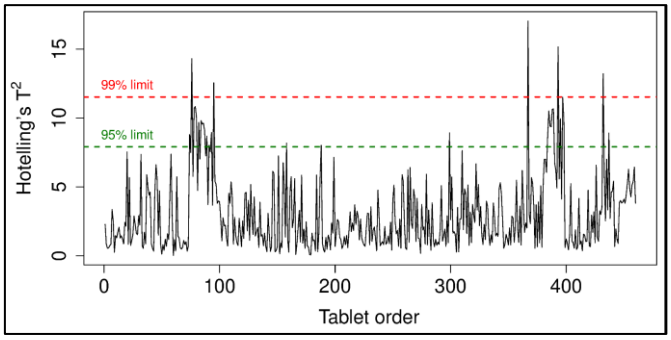


Figure 10 Hotelling T^2 Confidence Limit (r)

Numerous disciplines, including statistics, quality assurance, and pattern recognition, frequently employ Hotelling's T2 test. It is a crucial tool for comparing group means in multivariate data analysis and is particularly helpful when working with data containing numerous associated variables.

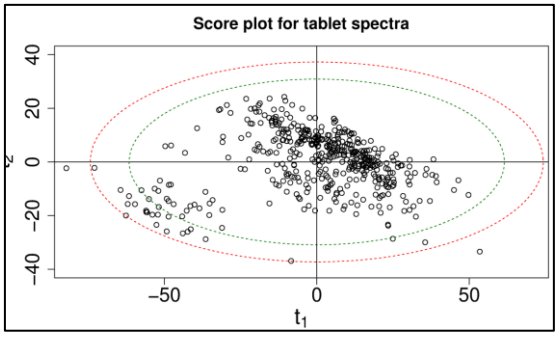


Figure 11 Scatter Plot Representation of Confidence Limit

For our case it is also important because an algorithm with the group means and standard deviation helps a ton about our data. It is more and more effective when the pattern may found.

VI. One Class SVM

One-class SVMs (Support Vector Machines) are a type of machine learning technique that are used for anomaly detection, especially when working with data that contains anomalies that are unusual or hard to obtain for training. It's a variant of the traditional SVM, which is mainly employed for binary classification jobs.

The one-class SVM's objective is to locate a region in the feature space that primarily contains data points (inliers) while excluding as many anomalies (outliers) as is practical. The method accomplishes this by identifying the ideal hyperplane that best deviates from the origin (i.e., the center) of the feature space to isolate the data points. The hyperplane is chosen to have the greatest distance from the origin and to have the greatest number of inliers inside its bounds.

One-class SVM's main principle is that normal data points should be grouped together in the feature space and should be closer to the origin, whereas anomalies, which are few and far between, should be located far from the origin. As a result, the data points' hyperplane, which divides them from the origin, effectively identifies outliers as points outside the inlier cluster's bounds.

In order to maximize the margin while minimizing the number of outliers (data points outside the margin), the mathematical formulation of one-class SVM entails solving an optimization problem. Convex optimization methods can be used to resolve this optimization issue.

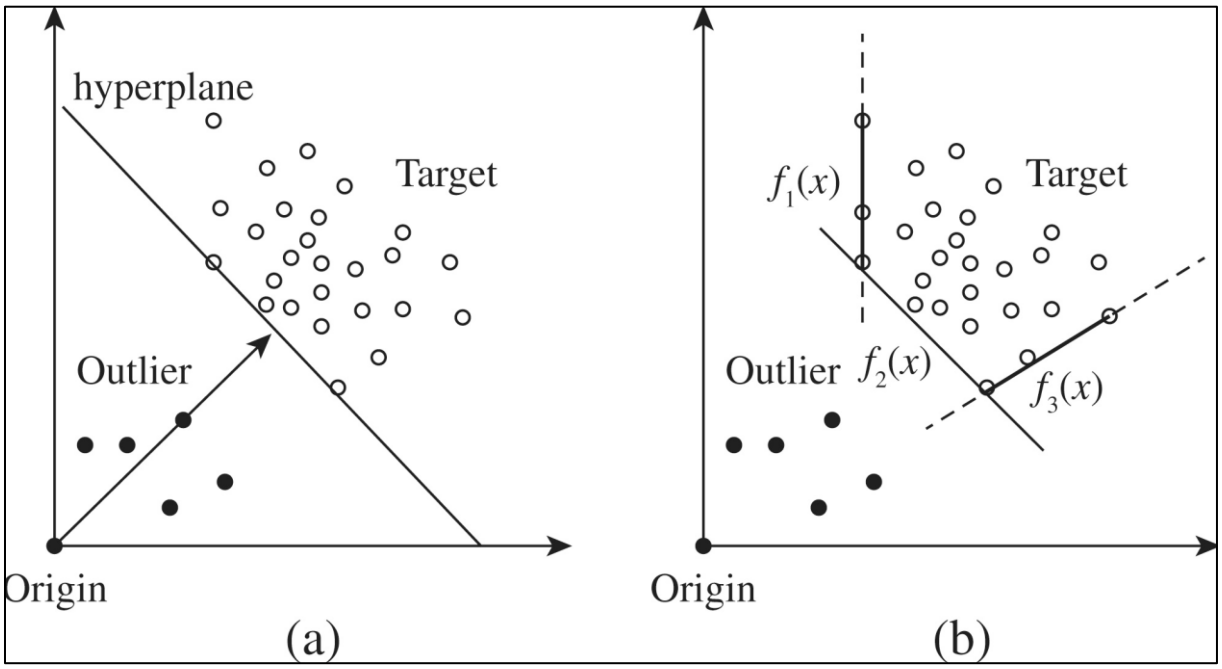


Figure 12 Representation of Support Vectors

There are three key parameters in the library of the SVM algorithm and mainly in the equations.

Nu: It is a hyperparameter that regulates the maximum allowed percentage of outliers. More outliers can occur when nu is smaller, whereas fewer outliers can occur when nu is greater. (nu takes values between 0 and 1)

Kernel: In order to translate the data into a higher-dimensional space, the kernel function type is specified. Linear, polynomial, radial basis function (RBF), and sigmoid are examples of common kernel functions.

Gamma: This hyperparameter regulates the decision boundary's smoothness. A smoother boundary is produced by a smaller gamma value, whereas a bigger value results in a more complex boundary.

VII. Auto Encoder

The technique known as "Autoencoder-based Anomaly Detection" makes excellent use of autoencoders for anomaly detection. The primary idea behind this method is to utilize an autoencoder to rebuild fresh data samples after training it on a dataset of typical, non-anomalous data. The likelihood of an anomaly increases if the autoencoder cannot reconstruct a data sample appropriately.

Auto Encoder works with three principles as listed.

1. Training Phase

Training phase data set should contain only normal data, non-anomalous data. Designing auto encoder should relay on this protocol. Encoder should have a bottleneck layer with lower dimensions than the input data. The goal is to minimize the difference between input and constructed output mostly MSE loss usage is typical. The last part of the training phase is keeping in mind that validation is not overfitting.

2. Anomaly Detection Phase

After encoder trained it used to reconstruct the new data samples. Calculation the difference between the input and the constructed output should be the next step of this phase. When construction complete an anomaly threshold should be chosen with respect to statistical methods to consider a data point as anomaly.

3. Anomaly Identification Phase

Comparing is done in this phase with the errors of the new data with respect to predefined threshold. If the error exceeds the threshold it should be identified as anomaly.

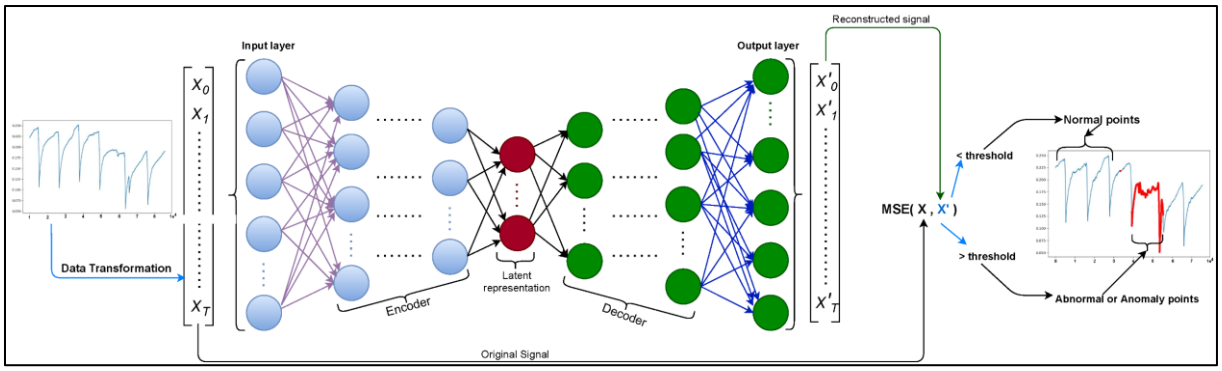


Figure 13 Autoencoder Representation

Autoencoder-based anomaly detection has the advantage of being an unsupervised approach, meaning it does not require labeled anomaly data during the training phase. Instead, it learns the normal data distribution from the training data and identifies anomalies as data samples that deviate significantly from this learned distribution.

It is essential to choose the appropriate architecture and hyperparameters for the autoencoder, as well as setting the threshold for anomaly detection, to achieve effective results. Fine-tuning and experimentation may be necessary to optimize the performance of the autoencoder for the specific anomaly detection task at hand.

VIII. LSTM

Recurrent neural networks (RNNs) of the LSTM (Long Short-Term Memory) kind are particularly effective at processing time series data and other sequential data. LSTM networks are useful for tasks that call for modeling long-term dependencies in data because of their capacity to store information for extended periods of time.

The LSTM model must be trained on normal, non-anomalous time series data in order to learn the patterns and temporal dependencies present in the data for anomaly identification. Future data points are then predicted using the learned LSTM model. A new data point is labeled as an anomaly if it significantly deviates from what the LSTM model anticipates based on the training data.

LSTM has few phases/methods and these methods may shown as:

1. Data Preparation

The most suitable format usually divide data into two sequences, first sequence is input sequence which is time steps and the second sequence is corresponding output in other name labels.

2. *Designing LSTM Model Architecture*

LSTM model usually designed with one or more LSTM layers followed by fully connected layers. LSTM layer's duty is learning temporal dependencies in the data. Which helps the remember the changeable pattern.

3. *Training*

LSTM model uses only normal values as mentioned in 'Data Preparation' phase. During training session model learns to predict next data point in the sequence based on the historical data.

4. *Anomaly Detection*

After the training session model can be used to identify anomalies. To do that model predict future data points in real time and when a new data point comes and compared with the actual data if the prediction deviates significantly from actual data model predicts as an anomaly point.

5. *Thresholding*

When the prediction deviated model should compare the difference between the actual and predicted values. So, finding the appropriate threshold value has significant role in that phase.

After all the phases done correctly the visualization part take role. When visualization complete accuracy of the data may be discussed.

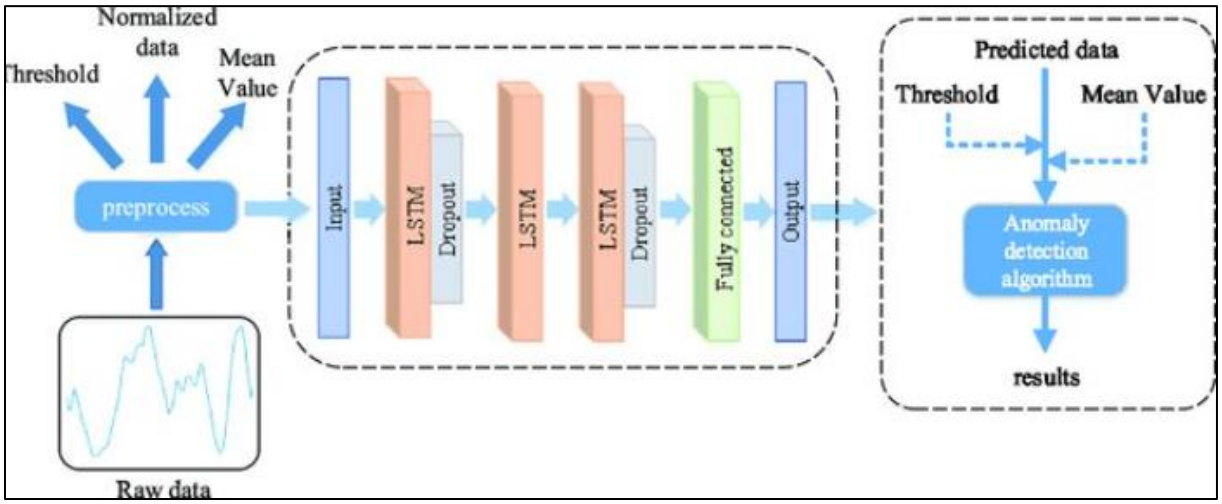


Figure 14 LSTM Process

TEST RESULTS

In the test result section, various anomaly detection methods mentioned and explained in the project were evaluated. In this context, F1 scores were obtained, but the real anomaly values that should be compared with the F1 scores in the test data were found to be uninformative. Therefore, in this context, test analyses were performed by visually analyzing the graphs of the models, focusing on both pattern-based and, if necessary, point-based approaches.

In this context, since the data and its anomaly values have already been analyzed in the "data analysis" section of the project, it is considered appropriate not to make further comments on the data itself. Instead, the comparison between the anomaly values provided by the data owner and the model results is emphasized as the focus of the evaluation. This approach allows for a more meaningful assessment of the performance of the models and their ability to detect anomalies based on the given ground truth anomaly values.

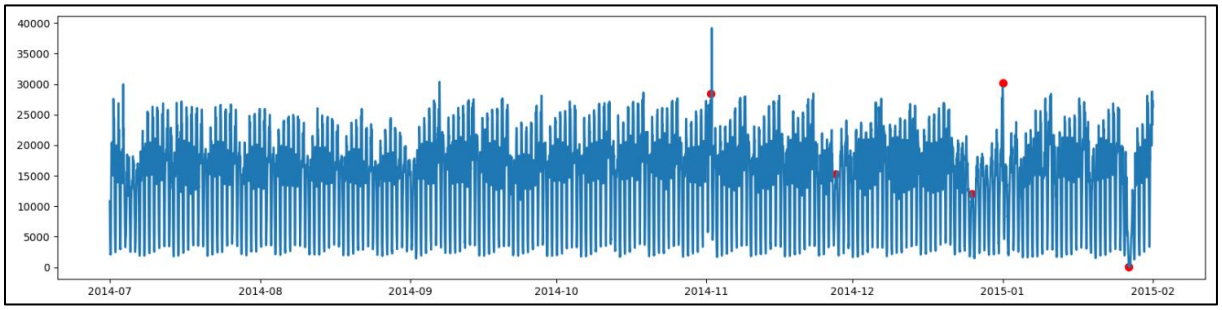


Figure 15 Real Identified Anomaly Points for Reference

As the codes for processing the models and data are included in the appendix section, we will proceed with providing comments on the code and parameters without displaying the actual code and visuals here. This approach allows us to present a concise and organized analysis of the models and data processing without overwhelming this section with detailed code snippets . If necessary, readers can refer to the appendix section to review the code under the respective headings for a more comprehensive understanding.

I. Isolation Forest Test Results

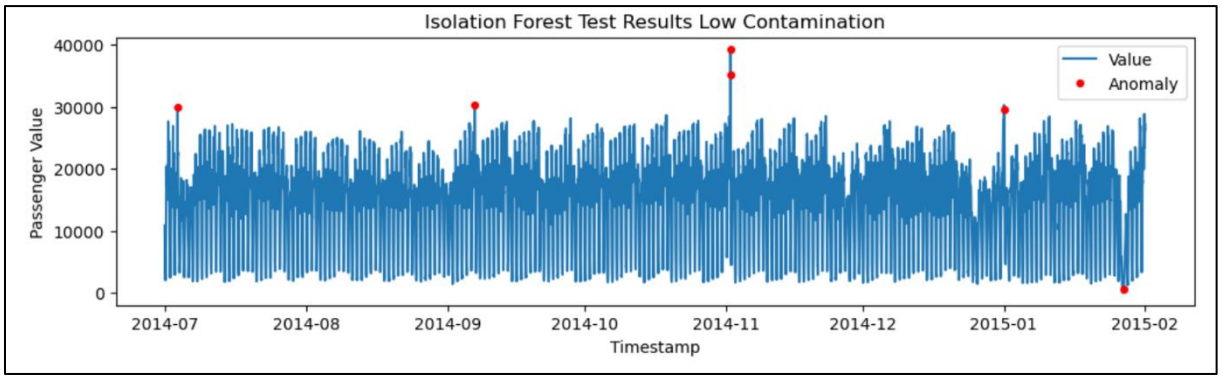


Figure 16 Test Results For Isolation Forest With Low Contamination

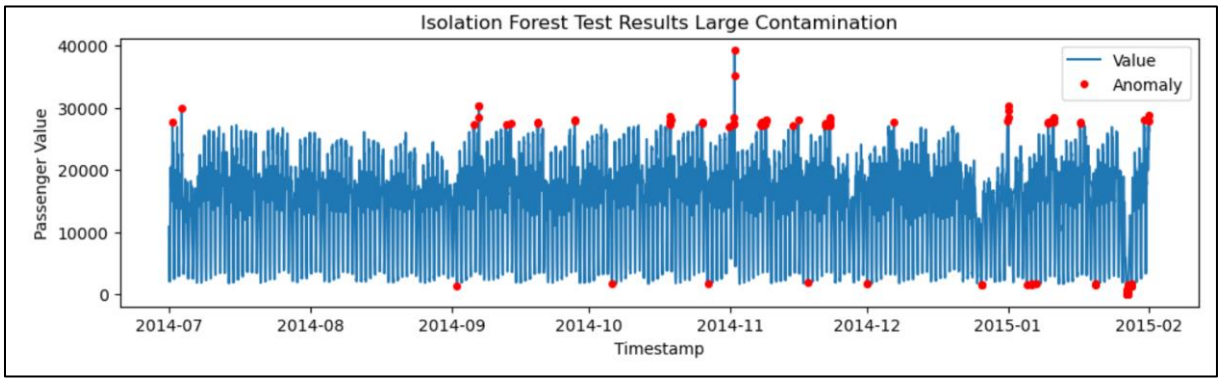


Figure 17 Test Results For Isolation Forest With Large Contamination

When examining the graph, we observe that sudden drops are not being captured as anomalies, even though peak values are easily detected. Even the most recent anomaly value, which is supposed to represent the latest detection, can be found when max_samples is set to nearly the size of the entire dataset. However, in other scenarios, even this cannot be achieved.

Regarding the contamination parameter, it was set equal to the proportion of anomalies in the given dataset. However, in Figure 17, when the contamination rate is increased, both upper and lower peak values can be seen.

These observations suggest that Isolation Forest is more effective at detecting extreme peak values rather than subtle anomalies like sudden drops. Moreover, the choice of hyperparameters such as max_samples, contamination, and others may have a significant impact on the model's performance. It's essential to fine-tune these hyperparameters and evaluate the model's performance using appropriate metrics to achieve better results in detecting different types of anomalies.

II. Local Outlier Factor Test Results

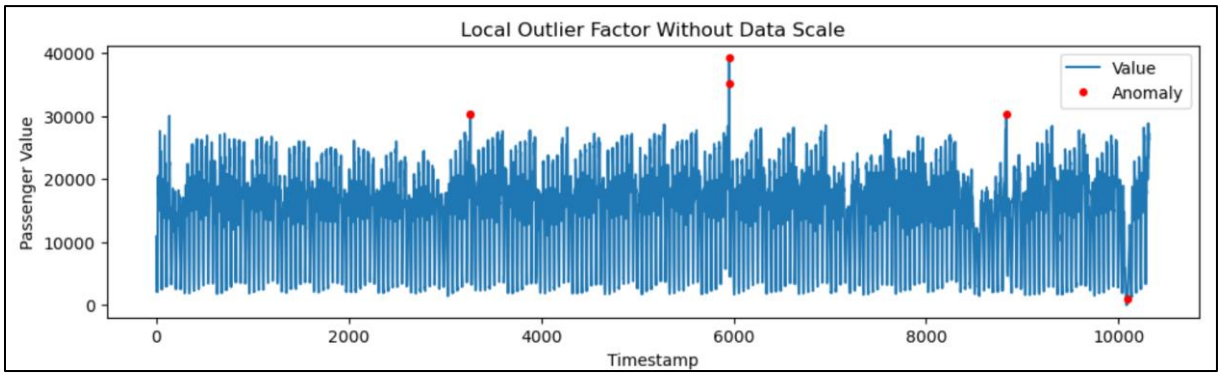


Figure 18 Local Outlier Factor Without Data Scaling

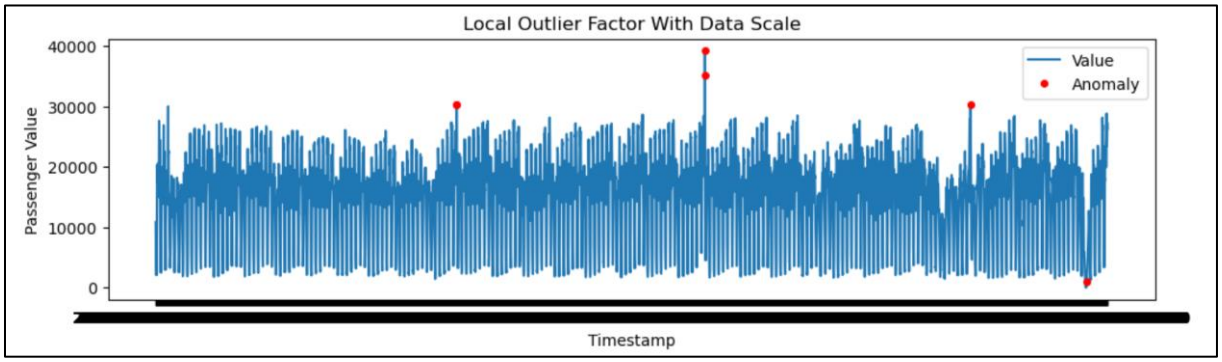


Figure 19 Local Outlier Factor With Data Scaling

Local Outlier Factor (LOF) may not be successful in capturing sudden drops due to several reasons. Two main factors that come to mind are n_neighbors and contamination rate. When we inspect the contamination rate, it seems to produce similar results to what we obtained with the isolation forest, which indicates that LOF is not providing a significant advantage in this context. Another possible factor to consider is data scaling, as LOF is sensitive to data scaling. However, as we can see from the plots, scaling the data did not yield noticeable improvements either.

In conclusion, while both isolation forest and LOF are useful anomaly detection algorithms, they may have limitations in capturing sudden drops in some cases. Anomaly detection is a challenging task, and the choice of algorithm and its hyperparameters should be carefully considered based on the specific characteristics of the data and the anomalies you are trying to detect.

III. Robust Z Score Test Results

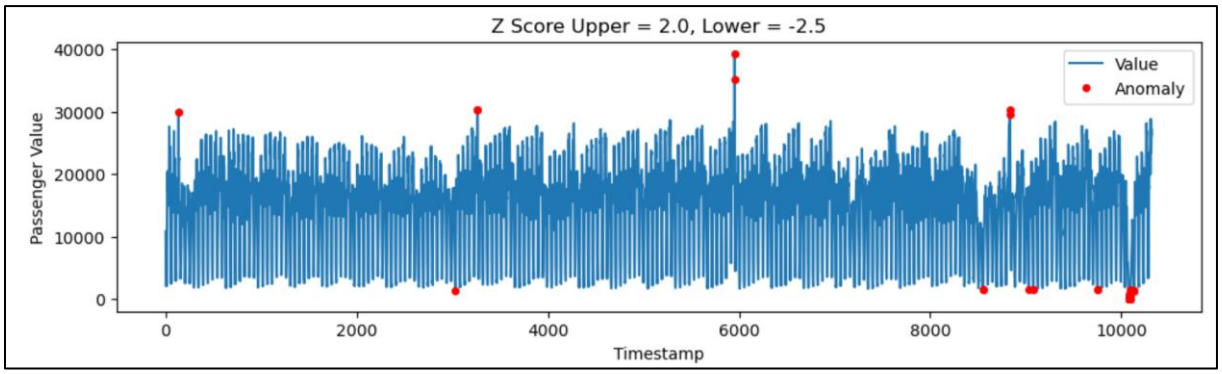


Figure 20 Z Score Test Results

When analyzing the Z-score model, it is observed that although the reaction to drops can be more adjustable, it doesn't significantly improve the sensitivity. In an attempt to improve the results, Time Window Analysis was conducted, but it was found to deviate from the desired outcome. Therefore, the decision to include this method in the test results section was abandoned. Combining specific threshold values with an additional model that considers contamination could potentially enhance the performance.

IV. Change Point Detection Test Results

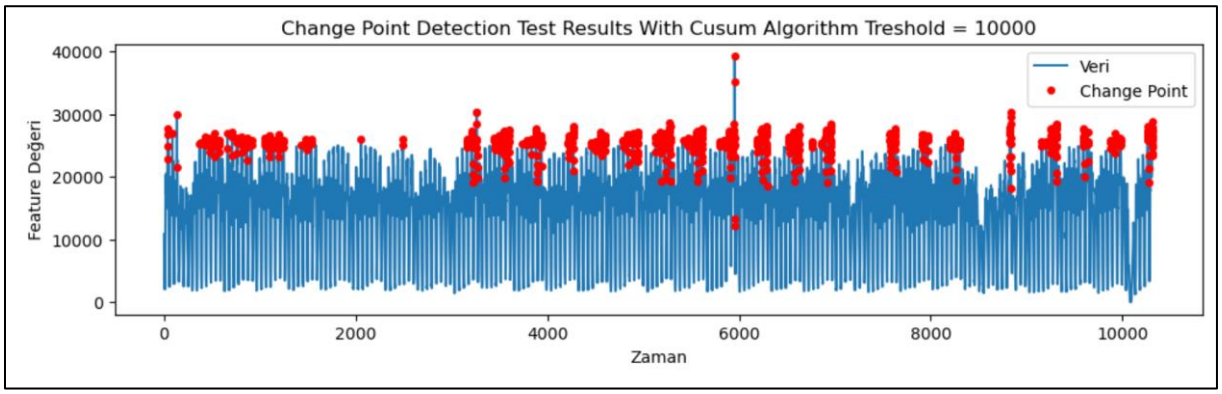


Figure 21 Change Point Detection Test Results With

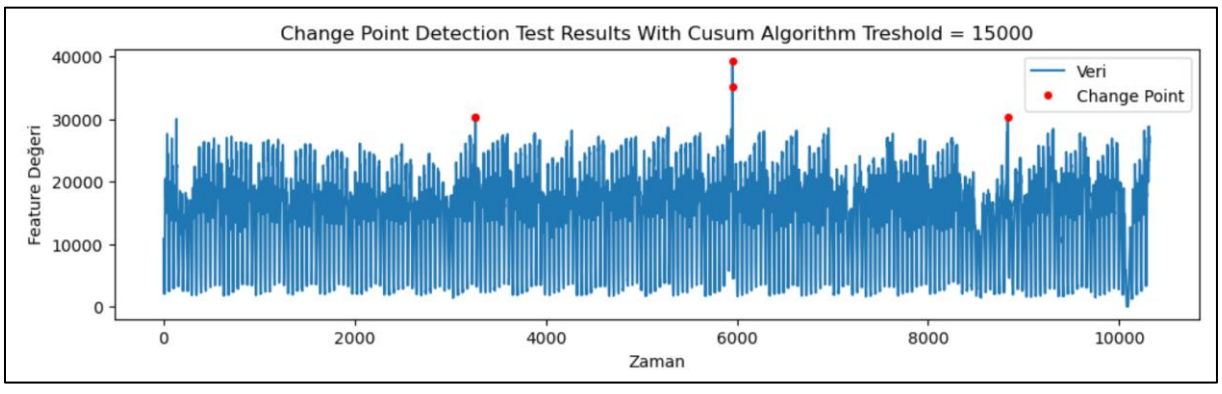


Figure 22 Change Point Detection Test Results With Cusum Algorithm With High Threshold

The CUSUM algorithm, which is implemented based on the change point detection method, relies on threshold-dependent variables and produces results from both graphs and the algorithm itself. From the project, it can be observed that this algorithm performs pattern tracking and value comparisons without the need for additional information. However, adding such an approach could demonstrate the importance of grouping and comparing mean values of these groups while detecting anomaly values in the given data. This enhancement aims to make the results more informative and interpretable.

V. Hotelling T2 Test Results

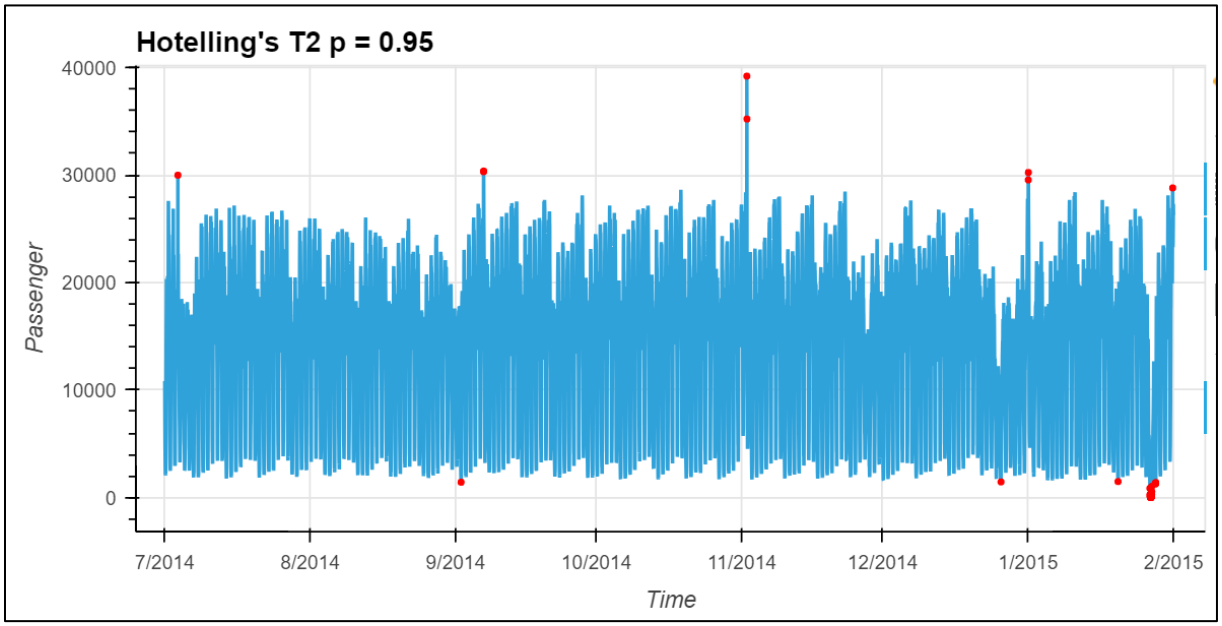


Figure 23 Hotelling T2 Test Results With Higher P Value

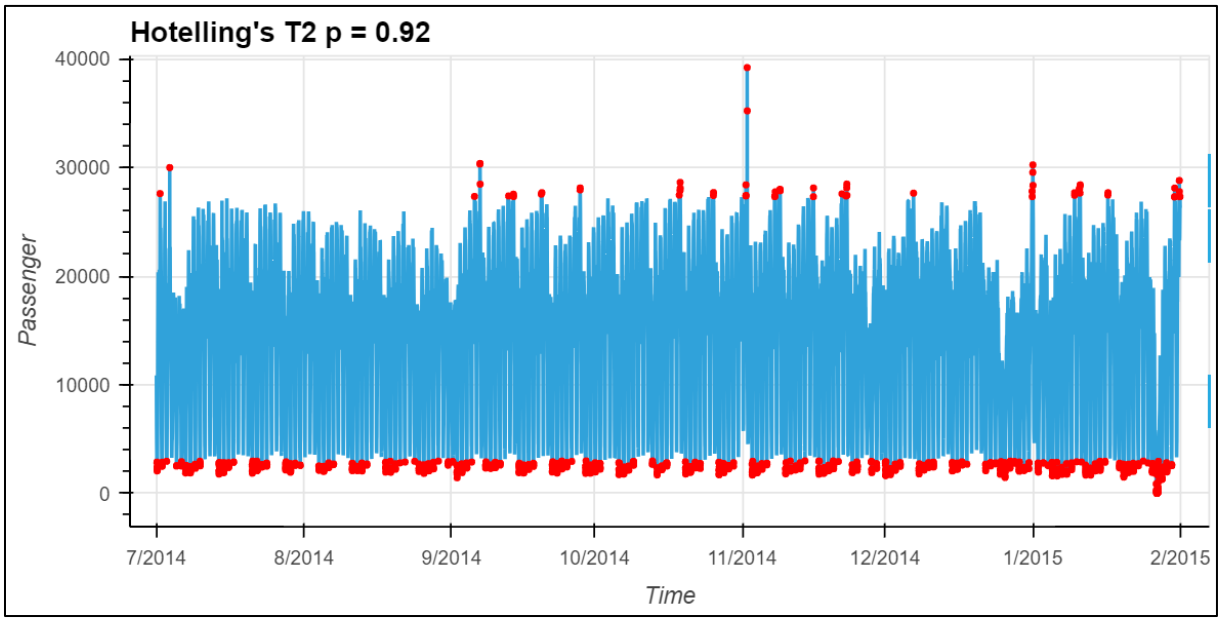


Figure 24 Hotelling T2 Test Results With Lower P Value

When examining the Hotelling T^2 model, it can be observed that the data is constrained within a certain interval and this constraint is evaluated under the name "significance level" to study the error rates. Based on these investigations, when the test result is set at p = 0.95, the selected pattern boundaries will be higher, and accessing the desired anomalies will be easier. However, when the p-value is taken in the range of 0.92, a large portion of the data, which actually forms a pattern, may not be recognized as a pattern, and instead, it will be considered as a continuous anomaly, leading to errors. This phenomenon can be better observed in the provided graphs.

Choosing the appropriate significance level (p-value) is crucial in Hotelling T^2 model. A higher p-value results in tighter boundaries, making it harder to detect anomalies, while a lower p-value leads to looser boundaries and increased chances of false positives. Finding the right balance by considering the characteristics of the data and the desired anomaly detection sensitivity is essential for the model to provide accurate and reliable results.

VI. One Class SVM Test Results

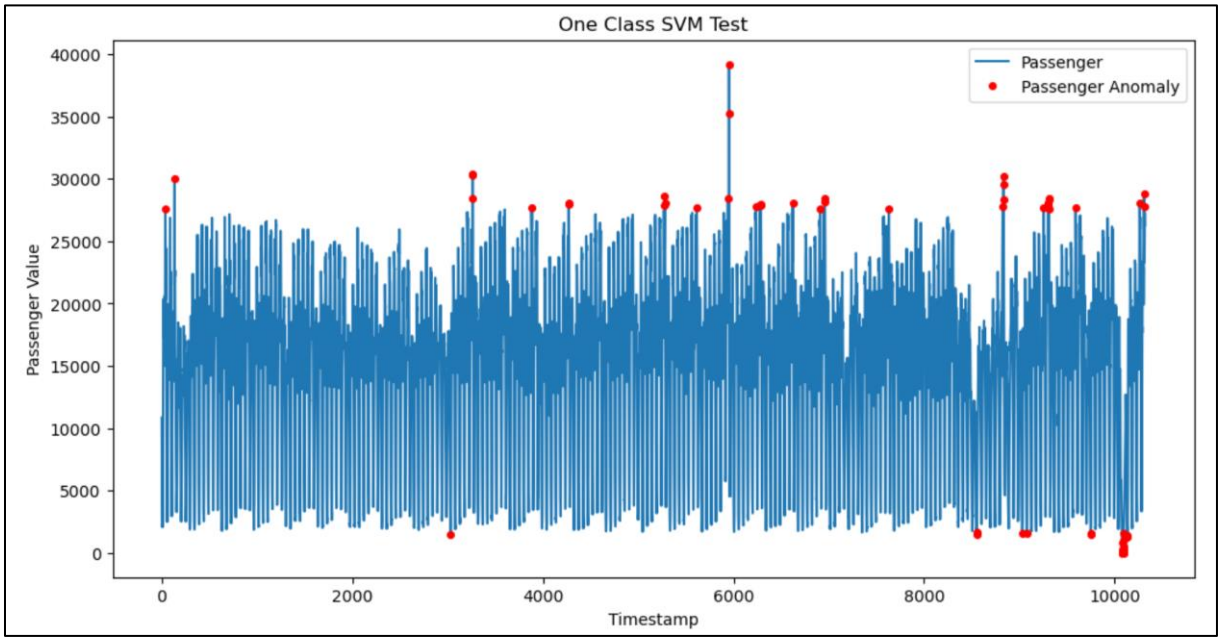


Figure 25 One Class SVM Test Result With nu = 0.95 gamma = auto

When examining the One-Class SVM model, the importance of the "nu" and "gamma" values becomes apparent. In this context, it is crucial to group the data within itself and determine the contamination rate. From this perspective, choosing a high "nu" value will restrict the training set's data and, consequently, lead to fewer anomalies being detected. Conversely, decreasing the "nu" value will allow more anomalies in the training set, resulting in the detection of more anomalies but potentially increasing the risk of false positives and overfitting.

Regarding the "gamma" value, it also plays a significant role in determining the anomaly score threshold. A high "gamma" value creates a narrower kernel, aiding in making more precise decisions about anomalies and normal data separation. However, this might increase the risk of overfitting and misclassifying anomalies. On the other hand, a low "gamma" value creates a wider kernel, leading to a smoother decision boundary and potentially less overfitting. However, this could also result in less precise separation between anomalies and normal data, causing some anomalies to be misclassified.

In conclusion, finding the right balance for "nu" and "gamma" is essential for achieving optimal performance in One-Class SVM. It requires careful consideration of the data's nature, the level of contamination, and the desired trade-off between sensitivity to anomalies and avoiding false positives. Experimentation with different parameter values and evaluating the model's performance on a validation set will help make an informed decision about the best settings for the specific anomaly detection task.

VII. LSTM

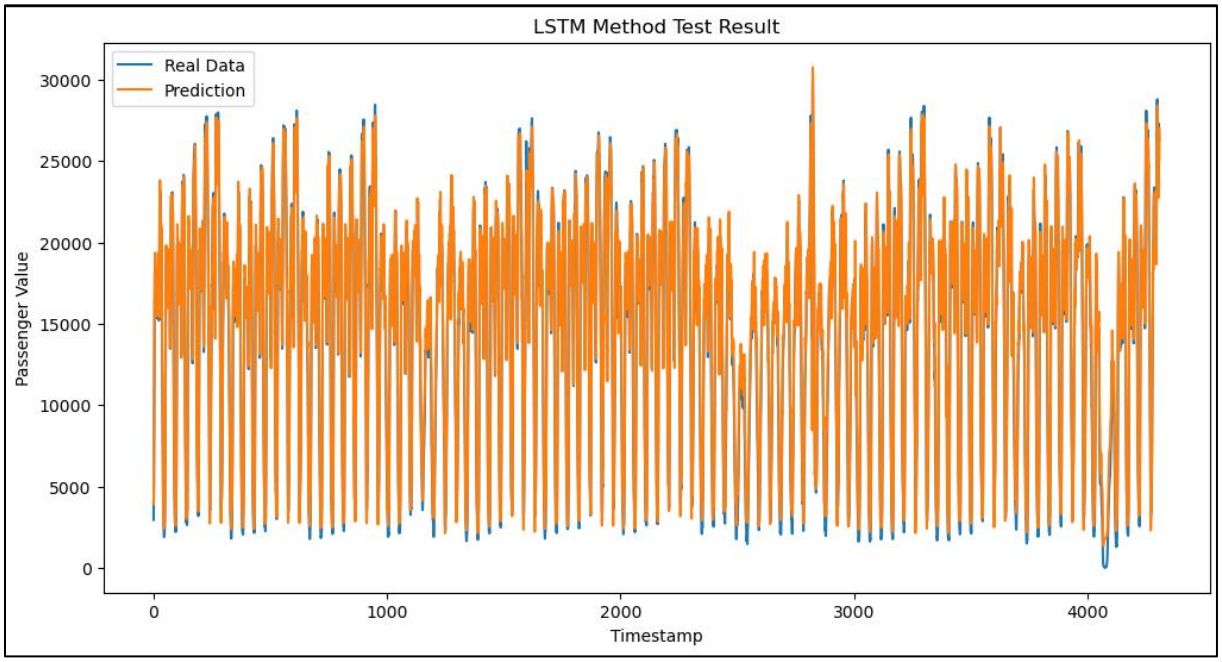


Figure 26 LSTM Method Test Result Prediction and Real Data

When examining the obtained results, we can observe that the LSTM model can effectively mimic the fluctuations around the mean value. However, there are noticeable errors in mimicking the lower extreme values. The primary reason behind this is that the model can easily adapt to the fluctuations around the mean since the predictions for intermediate values are relatively close to each other, especially with the help of dropout. However, it becomes more challenging to prevent errors in predicting sudden jumps or extreme points in the data.

This phenomenon is not uncommon in LSTM models. To overcome this issue, you can experiment with adjusting hyperparameters such as batch size, epoch count, and seq_length. Modifying these hyperparameters can potentially reduce the error and improve the model's ability to capture extreme points accurately.

Keep in mind that LSTM models can be sensitive to their hyperparameter settings, and finding the right combination of parameters that works best for your specific dataset can be an iterative process. By fine-tuning these hyperparameters and possibly trying different architectures, you may be able to achieve more accurate predictions for extreme values and enhance the overall performance of the LSTM model in your time series forecasting task. Additionally, performing model evaluation using appropriate metrics will help you quantitatively assess the model's performance and guide further improvements.

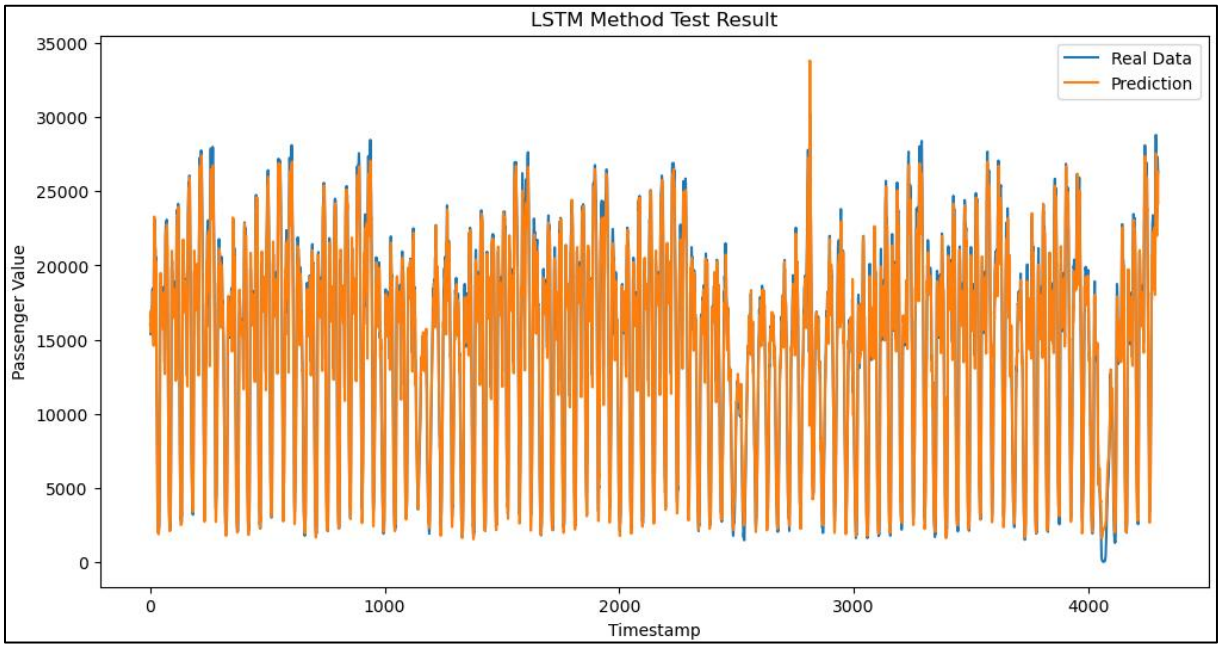


Figure 27 LSTM Method With Higher seq_length and Batch Size

When increasing seq_length and batch_size, as seen in Figure 27, the model's ability to predict lower values has improved, but there is some loss in predicting the upper values. To address this, you can experiment with different seq_length values to find the appropriate one that helps to better capture the patterns in the upper range and improve the overall performance.

It is important to be mindful of overfitting while making these adjustments. Overfitting occurs when the model becomes too specific to the training data and performs poorly on new, unseen data. To prevent overfitting, you can use techniques like dropout, L2 regularization, or early stopping during model training.

Another approach that can be considered is the Windowed Approach, where instead of using a fixed seq_length, you use a range of seq_length values that can change over time. This way, the model can adapt to different patterns at different scales in the data. You can have a set of pre-defined seq_length values or dynamically adjust them based on the characteristics of the data.

By experimenting with different seq_length values and considering the Windowed Approach, you can improve the model's ability to predict both lower and upper values in the time series data. However, always keep in mind the trade-off between capturing complex patterns and avoiding overfitting. Regular validation and evaluation techniques can help you choose the most suitable hyperparameters for your specific dataset and model architecture.

CONCLUSION

In this research, we investigated a number of anomaly detection techniques, such as Hotelling T2, Isolation Forest, Local Outlier Factor (LOF), and One-Class SVM. Each of these techniques has strengths and weaknesses when it comes to spotting anomalies in time series data.

A tree-based method called Isolation Forest is effective in finding anomalies in data that have a small percentage of anomalies. It is excellent for huge datasets since it effectively identifies anomalies in shorter pathways. In areas of high density, though, it might have trouble picking up abnormalities such sudden drops in time series data.

In contrast, LOF is sensitive to data scaling and is capable of spotting anomalies in both high- and low-density areas. It can recognize intricate patterns and adjust to various data distributions. However, it could struggle to spot abnormalities in areas with different densities or in situations with highly dimensional data.

A multivariate statistical technique called Hotelling T2 takes into account feature correlation. It is capable of capturing small changes in the data distribution and is efficient at finding abnormalities in high-dimensional data. However, it may struggle with non-linear anomalies and significantly depends on selecting adequate threshold values for performance.

One-Class SVM is an effective technique for finding anomalies, particularly when the data is complicated and multidimensional. It accurately produces findings and efficiently detects anomalies in both high- and low-density areas. However, it's important to pick the appropriate hyperparameters, like nu and gamma, to prevent overfitting or underfitting.

We also investigated LSTM, a deep learning-based technique for detecting time series anomalies. In time series data, LSTM demonstrated promising results in capturing intricate patterns and variations. For accurate predictions and to prevent overfitting, hyperparameters like batch_size and seq_length must be changed.

When working with time series data, it is crucial to properly scale the features, handle missing values, and preprocess the data. The trade-off between detecting abnormalities and preventing false positives must also be carefully taken into account. Finding the best strategy for the particular dataset and anomaly detection task requires testing several hyperparameters, model topologies, and assessment metrics.

To sum up, anomaly detection is a difficult task that calls for careful technique, hyperparameter, and data preparation step selection. The ideal method to use depends on the features of the data and the particular requirements for anomaly detection, each of which has strengths and disadvantages.

REFERENCES

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

https://neptune.ai/blog/anomaly-detection-in-time-series

https://matplotlib.org/stable/tutorials/colors/colors.html

https://seaborn.pydata.org/generated/seaborn.catplot.html

https://learnche.org/pid/latent-variable-modelling/principal-component-analysis/hotellings-t2-statistic

https://www.geeksforgeeks.org/ml-auto-encoders/

https://www.geeksforgeeks.org/machine-learning-for-anomaly-detection/

https://spotintelligence.com/2023/03/18/anomaly-detection-for-time-series/

https://www.kaggle.com/code/joshuaswords/time-series-anomaly-detection

https://www.youtube.com/watch?v=qy41dXGbAxY

https://www.kaggle.com/code/koheimuramatsu/industrial-machine-anomaly-detection/notebook

https://github.com/numenta/NAB/tree/master/data

https://www.kaggle.com/code/victorambonati/unsupervised-anomaly-detection/notebook

https://github.com/marcopeix/youtube_tutorials/blob/main/YT_02_anomaly_detection_time_series.ipynb

https://www.kaggle.com/code/joshuaswords/time-series-anomaly-detection#Anomaly-Detection-in-Time-Series-Data

https://dspace.mit.edu/bitstream/handle/1721.1/123129/1128282917-MIT.pdf?sequence=1&isAllowed=y

https://dataamber.medium.com/use-clustering-to-detect-time-series-anomaly-a83311cde3ce

https://scikit-learn.org/stable/modules/clustering.html#k-means

https://www.kaggle.com/code/caesarlupum/outlier-detection-change-finder

https://openreview.net/pdf?id=r8IvOsnHchr

APPENDIX

In this section, we include the code snippets used for implementing the various anomaly detection methods and the corresponding graphs generated during the analysis. Due to the large size of the code and the graphs, we have provided them in a separate file named "Appendix" This file contains the detailed code, along with the graphs for each method, including Isolation Forest, Local Outlier Factor, Hotelling T2, One-Class SVM, and LSTM.

Please refer to the "Appendix.pdf" for a comprehensive view of the code and the generated graphs. This approach allows us to keep the main document concise and focused while providing the necessary details and visuals in a separate file.