Journal of Intelligent Systems and Internet of Things
Vol. 12, No. 02, PP. 44-64, 2024
DOI: https://doi.org/10.54216/JISIoT.120204

Comparative Analysis of ML-Based Outlier Detection Techniques for IoT-Based Smart Energy Management Systems

Parh Yong Wong[1], Nayef A. M. Alduais[1,*], Nurul Aswa Omar[1], Salama A. Mostafa[1], Abdul-Malik H. Y. Saad[2], Antar Shaddad H. Abdul-Qawy[3], Abdullah B. Nasser[4], Waheed Ali H. M. Ghanem[5]

[1] Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, Johor 86400, Malaysia;
[2] College of Engineering, University of Buraimi, 512, Al Buraimi, Oman
[3] Department of Mathematics and Computer Science, Faculty of Science, Abdulrahman Al-Sumait University, Zanzibar, Tanzania;
[4] School of Technology and Innovation, University of Vaasa, 65200 Vaasa, Finland;
[5] Faculty of Computer Science and Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu Terengganu, Malaysia;

Emails: hi220037@student.uthm.edu.my; nayef@uthm.edu.my; nurulaswa@uthm.edu.my; salama@uthm.edu.my; abdulmalik.h@uob.edu.om; antarabdulqawy@sumait.ac.tz; nabdulla@uwasa.f; waheedghanem@umt.edu.my

Abstract

With the development and advancement of ICST, data-driven technology such as the Internet of Things (IoT) and Smart Technology including Smart Energy Management Systems (SEMS) has become a trend in many regions and around the globe. There is no doubt that data quality and data quality problems are among the most vital topics to be addressed for a successful application of IoT-based SEMS. Poor data in such major yet delicate systems will affect the quality of life (QoL) of millions, and even cause destruction and disruption to a country. This paper aims to tackle this problem by searching for suitable outlier detection techniques from the many developed ML-based outlier detection methods. Three methods are chosen and analyzed for their performances, namely the K-Nearest Neighbour (KNN)+ Mahalanobis Distance (MD), Minimum Covariance Determinant (MCD), and Local Outlier Factor (LOF) models. Three sensor-collected datasets that are related to SEMS and with different data types are used in this research, they are pre-processed and split into training and testing datasets with manually injected outliers. The training datasets are then used for searching the patterns of the datasets through training of the models, and the trained models are then tested with the testing datasets, using the found patterns to identify and label the outliers in the datasets. All the models can accurately identify the outliers, with their average accuracies scoring over 95%. However, the average execution time used for each model varies, where the KNN+MD model has the longest average execution time at 12.99 seconds, MCD achieving 3.98 seconds for execution time, and the LOF model at 0.60 seconds, the shortest among the three.

Received: August 13, 2023, Revised: November 26, 2023 Accepted: April 09, 2024

Keywords: Internet of Things (IoT); Smart Energy Management System; Outlier Detection Techniques; Comparative Analysis; K-Nearest Neighbor (KNN); Minimum Covariance Determinant (MCD); Local Outlier Factor (LOF).

1. Introduction

Through continuous advancements in communication and information technology, the Sci-Fi concept of Smart Technologies has become a reality in recent years. The perfect fusion between the orthodox manual systems and various technologies, aiming to serve humans better, the Smart Energy Management System (SEMS) is among the intelligent applications that first get the most attention and are fully realized. Combining with the Internet-of-Things (IoT) technology, modern SEMS, under optimal circumstances, can deliver many functions such as managing and controlling electrical appliances, automatic assignment, and deliverance of energy through grids and power stations, managing renewable energy sources, intelligent metering and outsourcing the energy, can be achieved. One must always remember, though, that data-driven fields such as IoT-based SEMS are susceptible to data quality problems and will suffer seriously from them [1].

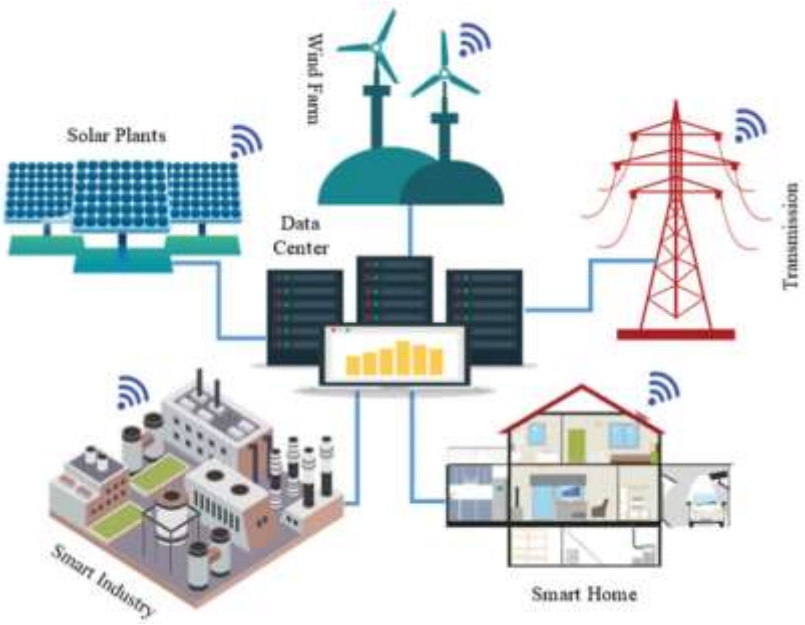

Figure 1: Applications of IoT-based Smart Energy Systems in Different Fields [1]

Figure 1 above shows some IoT-based SEMS applications in different areas. All the shown applications require data from various sources, mainly from sensor-collected data, for their complete functionality. Thus, data quality plays an essential role in ensuring the efficiency and effectiveness of these systems. Data quality can be summarised as the fitness or suitability for usage by a system or a user, meeting their requirements for different purposes [2]. The impact of data quality in a modern data-driven environment can be explained by a phenomenon known as "Garbage In, Garbage Out", which states that in AI technologies when AI "learns" or trains with data with poor quality, it will result in a highly ineffective AI model with inaccurate decision-making and has many flaws [3]. While this occurrence is often used to describe the effect of poor data quality in the fields of AI, the same occurrence can also be used to explain the impacts of poor data quality in IoT-enabled SEMS, which then causes excessive or insufficient energy to be distributed into the electrical and grid systems, further causing more destructive impacts such as city-wide blackouts and damaged electrical appliances. Thus, it is crucial to state the various data quality problems when working on an IoT-based SEMS.

One way to deal with data quality problems such as outliers is to deploy Machine Learning (ML)-based models to analyze and identify the anomalies in datasets quickly [4]. This research uses three sets of sensor-collected datasets for training and testing the selected supervised learning (SL) models: 1) K-Nearest Neighbour (KNN) + Mahalanobis Distance (MD), 2) Minimum Covariance Determinant (MCD), and 3) Local Outlier Factor (LOF). Three datasets are used in the analysis, and the onerous requirement of the datasets is that they are sensor-collected datasets used in an IoT-based Smart Energy System or related to it. Once the datasets are collected and pre-processed, including removing empty, invalid, and absurd-valued data and manually injecting outliers into the datasets, the models are trained with the training datasets to learn the patterns of the datasets. The models will then be tested with the testing datasets and evaluated for their accuracy and execution time.

The sections in the rest of the paper are organized as follows: Section 2 is the Literature Review, explaining several aspects of the paper and noting the related works. Section 3 is the Methodology, where the flow of the analysis, details of the datasets, and procedures are detailed. Section 4 is the Implementation, where important

information on implementing the Methodology in this research is detailed. Section 5 is the Result and Discussion, where analysis of the testing outcomes is recorded and discussed in detail. Lastly, in Section 6, a conclusion to this analysis is made, and the limitation of this paper is concerned.

2. Literature Review

This section gives general information and details about the ML models used with references to different sources. The related works are also discussed in this section, providing clarity on the research gaps of this analysis.

2.1 Mahalanobis Distance (MD)

Mahalanobis Distance (MD) is a general metric that aims to identify the non-isotropic (not comparable) properties of a feature space with J-dimensions, weighting the distance calculation referring to statistical differences of each component using the covariance matrices of observed samples [5].

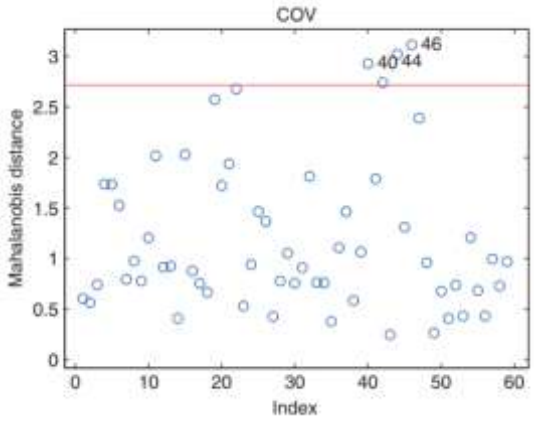

Figure 2: Example Plotting of MD [6]

Figure 2 shows an example scatter plot of Mahalanobis Distance. It functions similarly to normal Boxplot as Gaussian-based outlier detection; however, it works better than normal boxplots in a multivariate data environment [7]. An example of detecting outliers in multivariate datasets with MD is demonstrated in [8]: let ō be the mean vector, for object o in the datasets, the MD from o to ō is equal to

$$MD(o, \bar{o}) = (o - \bar{o})^T S^{-1} (o - \bar{o}) \quad (1)$$

Where S is the covariance matrices, calculating MD through Grubb's Test is possible since MD is a univariate value. Simplifying the steps, for determining an outlier value, we will first need to calculate the mean vector of ō from the dataset, then for each object of o, calculate the MD from o to ō. With the value of MD obtained, we can detect the outlier value of MD in the transformed univariate MD set:

$$\{MD(o, \bar{o}) | o \in D\} \quad (2)$$

From this point on, if an MD value is determined as an outlier, the original object o is also classified as an outlier [8].

In this analysis, MD is deployed as a part of the K-Nearest Neighbour (KNN) model. This SL model can classify objects based on the "neighbors" of the object or the closest data to the object to use MD as a supervised learning method. A comparison is made between the prior and current tested data. MD can be used as the metric for KNN, calculating the distance between inputs and outputs from the training datasets, and then be fitted to the testing set for detecting outliers [9].

2.2 Minimum Covariance Determinant (MCD)

Highly robust and resistant to outlying observation, Minimum Covariance Determinant or MCD is one of the first affine and equivariant estimators for datasets with multivariate location and scattering points.

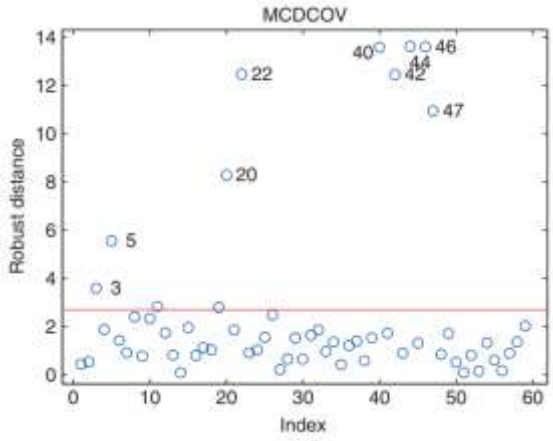

Figure 3: Example Plotting of MCD [6]

Figure 3 shows an example scatter plot of MCD. It has been instrumental in outlier detection since its introduction in 1984. Still, its popularity only surged after Rousseeuw and Van Driessen introduced the FAST-MCD algorithm, a more efficient algorithm for computation, in 1999. MCD becomes more valuable when the dataset has a higher dimension or has more variables, making it harder to illustrate. The theorem for computationally efficient FAST-MCD is as follows [6]:
Matrices $\{x_1, \ldots, x_n\}$ and let $H_1 \subset \{1, \ldots, n\}$ be an h-subset, that is $|H1| = h$. Put $\hat{\mu}_1$ and $\hat{\sum}_1$ the empirical mean and covariance matrices of the data in H_1. If $\det(\hat{\sum}_1) \neq 0$ define the relative distances:

$$d_1(i) := \sqrt{(x_i - \hat{\mu}_1)^t \hat{\sum}_1^{-1} (x_i - \hat{\mu}_1)} \qquad for\ i = 1, \ldots, n \qquad (1)$$

Now take H2 such that

$$\{d_1(i); i \in H_2\} := \{(d_1)_{1:n}, \ldots, (d_1)_{h:n}\}\ where\ (d_1)_{1:n} \leq (d_1)_{2:n} \leq \cdots \leq (d_1)_{h:n} \qquad (2)$$

are the ordered distances, and compute $\hat{\mu}_2$ *and* $\hat{\sum}_2$ *based on* H_2*. Then*

$$det(\hat{\sum}_2) \leq det(\hat{\sum}_1) \text{ with equality if and only if } \hat{\mu}_2 = \hat{\mu}_1 \text{ and } \hat{\sum}_2 = \hat{\sum}_1 \qquad (3)$$

Figure 3 is meant to be compared with Figure 2: Example Plotting of MD to see differences in performance between the two models. MCD is also known as robust MD and is better at dealing with cellwise outliers (in a multivariate dataset, cell wise outliers indicate outliers in individual cells, which do not affect data in other cells that are still worth preserving) [10]. The normal MCD is relatively heavy compared to MD and other statistical outlier detection methods. Hence, the development of FAST-MCD is an excellent alternative for programming environments [11]. The programmed-based FAST-MCD is like MD, where a pattern is first calculated from the inputs and outputs of the training datasets and then fitted to the testing datasets to predict outliers in existing datasets.

2.3 Local Outlier Factor (LOF)

The Local Outlier Factor, or LOF, is an algorithm usually implemented as an unsupervised learning model that excels in univariate data outlier detection. A density-based algorithm classifies data into clusters relative to the density of its "neighbors" (data points with the same features). This allows LOF to be an excellent outlier detection method for datasets with possible global and local outliers, as apart from the advantages above, it also works by assigning outlier factors to each data point and comparing them to determine the outliers [12].
LOF deploys a density model known as "local reachability density" (LRD). It is computed based on the local context of the object. This prediction of LRD is more advanced than simply dividing the number of objects by the volume [13]:

$$LRD(o) = \frac{|context(o)|}{\sum_{p \in context(o)} reachability - distancek(o,p)} \qquad (1)$$

Where the reachability-distance is given by:

$$\sum_{\mathrm{p}\in\mathrm{context(o)}} \mathrm{reachability-distance}_k(\mathrm{o},\mathrm{p}) = max\{kNN - dist(o), dist(o,p)\} \quad (2)$$

2.4 Related Works

The problem with data quality in IoT-based smart environments does not go unnoticed; where many researchers in the past have done their research and proposed many different methods for outlier detections in IoT environments. Table 1 below is the summary of details of related works. These associated works deploy other AI and ML-based outlier detection models in an IoT-enabled smart environment or SEMSs. These related works are essential references and guidance for this analysis paper, especially when choosing suitable models and testing.

Table 1: Summary of Related Works/Research

REF.	APPLICATIONS	OUTLIER DETECTION METHOD	DATASET(S) USED
[14]	Anomaly Detection in Power Consumption Data in Smart Buildings	One-class Support Vector named UAD-OCSVM and SL-based Anomaly Detection based on Micromoments (SAD-M2)	Real-world Collected Dataset from Qatar University's Energy Lab
[15]	Anomaly Pattern Detection for Energy Theft Detection in Advanced Metering Infrastructure	Support Vector Machine (SVM)	CER Smart Metering Project
[16]	Anomaly Detection in Industrial IoT (IIoT) for Smart Environments	Graph Neural Networks (GNNs)	Many datasets are used in the original research, kindly refer to the original research for references
[17]	Real-time Anomaly Detection in Smart Meter Data through Distributed Fog Computing Architecture and Multi-layered ML Methods	Linear Regression (LR), Support Vector Regression (SVR), Random Forest Regression (RFR), and Gradient Boosting Regression (GBR)	UCI Machine Learning Repository.
[18]	Multi-attributes Monitoring and Anomaly Detection for Power-limited IoT devices through Deep Reinforcement Learning (DRL) algorithm and Unsupervised Learning (UL) reward	Deep Reinforcement Learning (DRL) and UL Reward	Real-time Collected Datasets by the Authors
[19]	DDoS Detection through Deployment of Outlier Exposure (OE)-based Cross Sile Federated Learning at the Edge	Outlier Exposure-Based Cross-Silo Federated Learning, FedOE	Simulated DDoS Attacks by the Authors as Real-time Training and Testing Data
[20]	Anomaly Detection for IoT Devices with Mean-shift & Local Outlier Factor (LOF)-based Ensemble ML Technique	Mean-Shift & LOF-based Ensemble ML	UNSW-NB15 Network Dataset
[21]	Outlier Detection for IoT-enabled Indoor Localization through Various ML methods	Isolation Forest (iForest), SVM, KNN, and Random Forest (RF)	UCI Open Repository Datasets

The critical gaps between this analysis research and the related works come from three aspects. This analysis focuses on outlier detection of sensor-collected datasets related to SEMSs. Most related works focused on the Smart Environment, where their solutions apply to smartest systems. The few studies that relate to SEMS focus on power consumption data, which are not necessarily collected through a sensor system or Wireless Sensor

Network (WSN). At the same time, the three datasets used for this analysis are sensor-collected data specifically for IoT-based SEMS. Lastly, while the LOF and KNN methods are popular models for classification problems such as outlier detection, MD and MCD have rare usage in many smart applications, let alone the IoT-enabled SEMS. This analysis can contribute to analyzing the three specific datasets, which do not have past usage in this paper, using three different models that are also rare to use with SEMS.

3. Methodology

This section clarifies the Methodology used for this research. This includes the flow of conducting the analysis, the procedure in preparing and pre-processing the datasets, training, and testing the supervised ML models, and steps in evaluating the performance of datasets.

3.1 Workflow of Analysis

Figure 4 shows a general workflow of the analysis works done in this research. Generally, since the research is about the study of various supervised learning outlier detection methods, the first step of the analysis will be to collect enough datasets for training and testing the models. The datasets collected for this analysis are benchmarked datasets used in research or for using public domains for similar purposes. Three benchmark datasets are collected for this research. Once the collection is done, a pre-processing of the datasets is done, including removing the incomplete data points or data points with invalid data values, removing data points with absurd values, and statistical analysis on the datasets to find out the statistical properties of the datasets, and lastly, randomly inject outliers into the datasets through simple programs. The percentage of injected outliers is based on the dataset's number of data types or features, which is 1% for each data type. The last step before model training is dataset splitting. The injected datasets are split into two proportions: 80% as training data and 20 % for testing out the models.

With the datasets pre-processed and the code for the models prepared, the models should be trained with the prepared training datasets. For this research, three types of outlier detection models are chosen for analysis and comparison which is 1) K-Nearest Neighbor (KNN)+Mahalanobis Distance (MD), 2) Minimum Covariance Determinant (MCD), and 3) Local Outlier Factor (LOF). The training of the models is similar in many steps, namely setting the suitable threshold, which is again based on the number of data types of the dataset, and fitting the trained models to the injected training datasets. An extra step for the KNN+MD for model training involves calculating the covariance of the dataset, which is later used as a model parameter for the KNN. Once the models are successfully trained, they are immediately used for outlier detection with the testing datasets. The results are recorded and analyzed. Based on the results, the performance of the models is obtained, with the primary metrics for evaluation emphasis on accuracy and execution time.

3.2 Dataset Requirements & Preprocessing

This section introduces the detailed requirements for the used datasets, and the steps taken to preprocess the datasets. A significant point when it comes to choosing suitable datasets for the uses of research is to relate the datasets to the topic researched and the goal(s) of the research. The key focuses of this research are related to sensor-collected datasets in IoT-based SEMS. Thus, the several requirements for datasets are as follows:

- The datasets are multivariate numerical datasets (datasets with multiple types of data) related to the research of SEMS for different purposes.
- The datasets should be gathered using sensors.

Any datasets coming from the internet should be benchmarked or tied to published research related to the SEMS. Table 2 below is a summary of the basic information of the used datasets, detailing the titles of their research or usages, and the citation to the sources. Further details of the datasets are explained in the Implementation section, but in general, the datasets used are fully suitable for the usage of the research.

TABLE 2: Basic Summary of Used Datasets

Dataset	Title	Source
Dataset 1	Data for Short-Term Prediction of CO2 Concentration Based on A Wireless Sensor Network	[22]
Dataset 2	QCAT Smart Office Environment	[23]
Dataset 3	ROBOD, Room-Level Occupancy and Building Operation Dataset	[24]

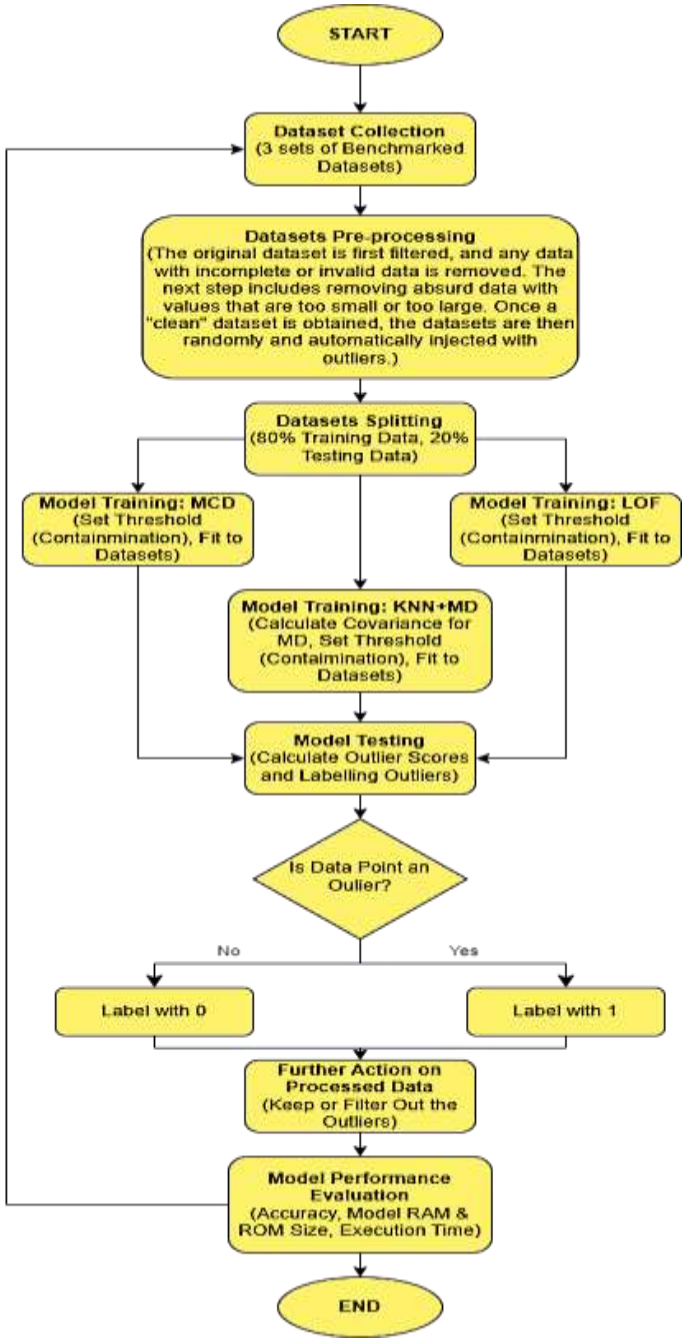

Figure 4: Flow Diagram of Analysis

Once the datasets are decided, the next step is the pre-processing of the datasets. In general, the datasets used are almost complete and can represent the data values consistently. Still, there are five steps of pre-processing of the datasets to be done. The first step is removing unwanted data types (non-numerical and unrelated) in each dataset, leaving only humidity, temperature, light intensity, CO2 count, and occupancy in numbers. Being the most common data types present in most of the existing smart energy management systems as essential, the relationship of humidity, temperature, and light intensity when taking light intensity as the primary data type is as follows: humidity is inversely proportional to light intensity. In contrast, temperature increases when light intensity increases. With this theorem established, for simplicity (more data types equal more dimensions equal to higher complexity) and accuracy in detecting outliers (more data types require more sensor types and amount, leading to higher error rate), this research only keeps these 5 data types. The second step is to remove the incomplete/missing data values, ensuring the process of calculation and evaluation during the outlier detection steps. After removing the empty and invalid data, the third step is to remove the absurdly huge or tiny values in the datasets. With these steps, a “clean” dataset, or a dataset without any data problems such as outliers and out-

of-range data, can be obtained. The fourth step is to analyse the datasets statistically to ensure that the probability of any anomalies existing is reduced to the minimum. The statistics include the count (number of data points in each dataset), mean, standard deviation, min, first quartile (25%), second quartile (50%), third quartile (75%) and max of each data type in the datasets. After obtaining a clean dataset, manually injection of outliers into the datasets are done. This is the fifth and critical step in testing the accuracy of the data, where the SL models should be able to detect the outliers in the datasets. The percentage of outliers to the original data used is 1% (0.01) for each data type.

3.3 Training and Testing the Models, Analysis & Evaluation Metrics

The development of the supervised learning models happens after the data pre-processing phase. All the programming and coding are done in Python, and the models are developed on Google Collaboratory. The models come from the PYOD Library. The models used are the MCD, KNN (Mahalanobis Distance), and LOF. All usage of the models (training the models and testing them out with test data) follows the instructions and guide from PYOD Official Documentation. Below are the explanations of each model from the PYOD Official Documentation and their use cases in this analysis paper, alongside the pseudo-code for each main process in the program.

Some general steps that are done before starting the training and testing of the models include: 1. Importing the Injected Datasets into the system as Data Frame; 2. Removing any unrelated columns (data types that are not directly related to the main data types tested for outliers) such as Index and Timestamp, and rows (data points) with any invalid, empty, or unavailable data values; and lastly, 3. Splitting the datasets into 2 parts of different proportions: the larger portion has 80% of data, which becomes the training data for the models, while the smaller portion has the remaining 20% of data, which becomes the testing data for the models. This proportion is one of the more common and most used proportions for splitting training/testing data for ML models [25]. With all these steps done, proceed to the model training and testing.

3.3.1 PYOD ML Model: KNN+MD

According to the PYOD Official Documentation. Three KNN classes are supported for outlier detection, which is largest (outlier score based on max distance from neighbors), mean (outlier score based on the mean of the distance from neighbors), and median (outlier score based on median distance from neighbors). PYOD's KNN works by calculating the data point's distance to its kth nearest neighbor as the outlying score [26].

// Training of KNN+MD Model //

1. INPUT: Datasets, Inputs, Targets // Input = Testing Part of the Injected Datasets, Targets = Training Part of the Injected Datasets.
2. OUTPUT: Trained KNN+MD Model
3. BEGIN:
4. // Phase I: Dataset Pre-processing: //
5. Select Dataset → Dataset N
6. Clean Dataset N // Remove columns not included in evaluations (such as the Index column), and rows with invalid, empty and/or absurd data values.
7. // Phase II: Setup Division of Data for Training and Testing: //
8. Set Training_Percent, Testing_Percent ← (80, 20)
9. Split Dataset N → Input_N, Target_N
10. // Phase III: Setup for KNN+MD Model: //
11. Evaluate Covariance for Dataset → Cov_N // calculate the covariance of each dataset.
12. Set Contamination_Rate, Method, Metric, Metric_Param, Algorithm ← "Threshold", "Largest", "MD", "Cov_N", "BallTree")
13. // Phase IV: Train the KNN+MD Model //
14. Fit KNN+MD Model with Training Data ← Target_N
15. Predict Raw Outlier Score, Raw Outlier Label → Target_N_OS, Target_N_OL
16. Output
17. END

For this analysis, the only parameter changed when using the KNN modal is the metric used for calculating the outlying score, which, in this case, MD is used. The algorithm used for the KNN+MD is automatically set by the model based on the dataset.

// Testing the KNN+MD Model //

1. INPUT: Input_N // Testing Data Part of Injected Datasets
2. OUTPUT: Input_N_Out // Outliers in Testing Data
3. BEGIN:
4. Call Trained KNN+MD Model
5. Fit Trained KNN+MD Model ← Input_N
6. For i=1 to A do // A indicates the number of observations in Testing Data, Input_N
7. Evaluate Outlier Score → Input_N_OS
8. Predict Outlier Label → Input_N_OL
9. Determine Outliers → Input_N_Out
10. Output
11. END

The outputs needed are the outlier scores and outlier labels. Overall, with the split training and testing data, a covariance of the training data is evaluated. The covariance is then used as the metric parameter for the KNN model. The KNN model has a few parameters that are changed when used with this analysis: 1. The default number of k neighbors are used, which is 5; 2. The method of determining the k-distance of the target point to its nearest neighbors is also defaulted to using the original and unaltered values of k-distance; 3. The metric is set to use MD as a method of calculating the outlier score; 4. Metric param which is the covariance calculated for MD and lastly, 5. Contamination, or threshold, is where the value is obtained through various rounds of testing and is also used with the other models (the contamination rate targets the dataset, not the model). Fitting the KNN model into the training dataset will output the raw outlier scores and prediction labels in binary. With both the raw outlier scores and the prediction labels obtained, fitting the model to the testing data will get the result needed in the form of outlier scores for the testing data and outlier labels for the test data. These results are then imported into Data Frame columns and exported together with the data points in the form of an Excel Workbook.

3.3.2 PYOD ML Model: MCD

// Training of MCD Model //

1. INPUT: Datasets, Inputs, Targets // Input = Testing Part of the Injected Datasets, Targets = Training Part of the Injected Datasets.
2. OUTPUT: Trained MCD Model
3. BEGIN:
4. // Phase I: Dataset Pre-processing: //
5. Select Dataset → Dataset N
6. Clean Dataset N // Remove columns not included in evaluations (such as the Index column), and rows with invalid, empty and/or absurd data values.
7. // Phase II: Setup Division of Data for Training and Testing: //
8. Set Training_Percent, Testing_Percent ← (80, 20)
9. Split Dataset N → Input_N, Target_N
10. // Phase III: Setup for MCD Model: //
11. Set Contamination_Rate ← “Threshold”
12. // Phase IV: Train the MCD Model //
13. Fit MCD Model with Training Data ← Target_N
14. Predict Raw Outlier Score, Raw Outlier Label → Target_N_OS, Target_N_OL
15. Output
16. END

According to the PYOD Official Documentation, the MCD model is a robust estimator of covariance, detecting outliers in a Gaussian-distributed dataset. The MCD covariance estimator will be applied to Gaussian-distributed data but could still be used for data drawn from an unimodal, symmetric distribution [46].

// Testing the MCD Model //

1. INPUT: Input_N // Testing Data Part of Injected Datasets
2. OUTPUT: Input_N_Out // Outliers in Testing Data
3. BEGIN:
4. Call Trained MCD Model
5. Fit Trained MCD Model ← Input_N
6. For i=1 to A do // A indicates the number of observations in Testing Data, Input_N
7. Evaluate Outlier Score → Input_N_OS

8. Predict Outlier Label → Input_N_OL
9. Determine Outliers → Input_N_Out
10. Output
11. END

The outputs needed are the outlier scores and outlier labels. Overall, with the split training and testing data, a covariance of the training data can be obtained by fitting the MCD model into the training dataset which will then compute the raw outlier scores and prediction labels in binary after setting the contamination score, which is the same as in the KNN+MD model. With both the raw outlier scores and the prediction labels obtained, fitting the MCD model to the testing data will get the result needed in the form of outlier scores for the testing data and outlier labels for the test data. These results are then imported into Data Frame columns and exported together with the data points in the form of an Excel Workbook. Overall, the MCD model, in terms of users' usage and what the users can observe on the surface, works similarly to the previously explained KNN+MD model in PYOD library settings.

3.3.3 PYOD ML Model: LOF

// Training of LOF Model //

1. INPUT: Datasets, Inputs, Targets // Input = Testing Part of the Injected Datasets, Targets = Training Part of the Injected Datasets.
2. OUTPUT: Trained LOF Model
3. BEGIN:
4. // Phase I: Dataset Pre-processing: //
5. Select Dataset → Dataset N
6. Clean Dataset N // Remove columns not included in evaluations (such as the Index column), and rows with invalid, empty and/or absurd data values.
7. // Phase II: Setup Division of Data for Training and Testing: //
8. Set Training_Percent, Testing_Percent ← (80, 20)
9. Split Dataset N → Input_N, Target_N
10. // Phase III: Setup for LOF Model: //
11. Set Contamination_Rate, N_Neighbours, Metric, Algorithm ← "Threshold", "20", "Minkowski Space (MS)", "BallTree")
12. // Phase IV: Train the LOF Model //
13. Fit LOF Model with Training Data ← Target_N
14. Predict Raw Outlier Score, Raw Outlier Label → Target_N_OS, Target_N_OL
15. Output
16. END

According to the PYOD Official Documentation, PYOD LOF exists as a wrapper of the Scikit-Learn LOF Class with more functionalities. It is an unsupervised Outlier Detection using the Local Outlier Factor (LOF) which is the anomaly score of each sample. It measures the local deviation of the density of a given sample concerning its neighbors. It is local in that the anomaly score depends on how isolated the object is from the surrounding neighborhood given by the k-nearest neighbors, whose distance is used to estimate the local density. By comparing a sample's local density to its neighbors' local densities, one can identify samples with a substantially lower density than their neighbors, which are outliers [26].

// Testing the LOF Model //

1. INPUT: Input_N // Testing Data Part of Injected Datasets
2. OUTPUT: Input_N_Out // Outliers in Testing Data
3. BEGIN:
4. Call Trained LOF Model
5. Fit Trained LOF Model ← Input_N
6. For i=1 to A do // A indicates the number of observations in Testing Data, Input_N
7. Evaluate Outlier Score → Input_N_OS
8. Predict Outlier Label → Input_N_OL
9. Determine Outliers → Input_N_Out
10. Output
11. END

The outputs needed are the outlier scores and outlier. Overall, with the split training and testing data, a covariance of the training data can be obtained by fitting the LOF model into the training dataset which will then compute the raw outlier scores and prediction labels in binary after setting a few attributes for the Model, which are as follows: 1. Contamination Rate which is the same as in the KNN+MD and MCD models; 2. Nearest Neighbour (N-Neighbour) which is defaulted to 20; 3. Metric which is defaulted to using Minkowski Space (MS); and 4. Algorithm which again is defaulted to Ball Tree. With both the raw outlier scores and the prediction labels obtained, fitting the LOF model to the testing data will get the result needed in the form of outlier scores for the testing data and outlier labels for the test data. These results are then imported into Data Frame columns and exported together with the data points in the form of the Excel Workbook. Overall, the LOF model, in terms of users' usage and what the users can observe on the surface, works similarly to the previously explained KNN+MD and MCD models in PYOD library settings.

3.3.4 Performance Evaluation

Once the model is successfully trained and tested for the first time, a few more settings are added to the program for evaluation of the models' performances. Two performance evaluation metrics need to be considered for this analysis. The first is Accuracy, which is the amount of correct data identified and can be calculated with a simple equation:

$$\frac{TN + TP}{TN + TP + FN + FP} \tag{1}$$

Where TN is the detected True Negative, TP is the detected True Positive, FN is the detected False Negative, and FP is the detected False Positive, this metric can be easily calculated manually or through programming [26]. The second metric is the execution speed in time, indicating how long the model takes to complete its execution. It can be traced with the Performance Counter (perf_counter).

4. Implementation

This section explains certain important details when implementing the procedures explained in the Methodology for this analysis. In general, the procedures are followed fully to the end of conducting the analysis. There are certain changes and basic rules that are done and followed in this analysis to obtain the results as efficiently as possible.

4.1 Explaining the Datasets

This subsection is a collection of sub-subsections that introduce and explain the datasets chosen to be used for this research. Again, all the datasets used have fully met the requirements of this analysis.

4.1.1 Dataset 1: Data for Short-Term Prediction of CO2 Concentration Based on A WSN

A set of datasets was collected from the 8th of February 2018 to the 8th of November 2018; the data was collected on the 8th day of each month during 2018 by the research team for predicting the concentration of CO2 and the environment through a Wireless Sensor Network. In this research, the concentration of Carbon Dioxide gas (CO2) in the tested environment is also measured as a part of the Indoor Air Quality (IAQ) index, which will affect modern Heat, Ventilation, and Air Conditioning (HVAC) systems which will account for the optimal and recommended concentration of CO2 in indoor environments. The justification for the use of this dataset is that it suits the requirements of this research with the related topic of study in which the dataset is collected, where the light, humidity, and temperature readings of an IoT-based Wireless Sensor Network (WSN)-enabled SEMS are collected and used for prediction of indoor CO2 concentration level. This set of datasets is also trustworthy as it is tied to a published study [22]. Table 3 shows snippets of the collected Dataset 1:

Table 3: Dataset 1: Data for Short-term Prediction of CO2 Concentration based on a Wireless Sensor Network

Record ID	Timestamp (Unix)	CO2	Temperature (°C)	Humidity (%)	Light Intensity (Lux)	Node ID

4678229	1533243314	2944	30.5	61	89	9
6642641	1533365453	2.78E+09	27.3	0	667	73
8267680	1533199750	347	-100	63.8	24	0
8267681	1533199751	287	27.6	63.8	25	0
8267682	1533199752	296	27.6	63.8	24	0
8267683	1533199753	301	27.6	63.8	24	0
8267684	1533199754	297	27.6	63.8	23	0
8267685	1533199755	315	27.7	63.9	23	0
8267686	1533199756	317	27.7	63.9	23	0
8267687	1533199757	321	27.7	63.9	25	0

4.1.2 Dataset 2 – QCAT Smart Office Environment

A set of benchmark datasets was collected by the Commonwealth Scientific and Industrial Research Organisation (CSIRO) in Australia and gathered in a Smart Offices Environment of the CSIRO office, starting on the 19th of February 2012, and ending on the 16th of September 2012. The justification for this dataset's usage is that it meets the requirements of being sensors-gathered multivariate numerical datasets collected for and through an IoT-enabled SEMS in an IoT-based Smart Office. It is also counted as benchmarked datasets as they are collected by an International Government Agency for public use. In the references, only one of the data types, Humidity (%) is referred to, the other data types are collected with the same system too, though, in the source, they were output and exported as separate data frames and thus are in different web links [23]. Table 4 shows snippets of the collected Dataset 2:

Table 4: Dataset 2: QCAT Smart Office Environment Datasets

Fleck	Temperature (℃)	Humidity (%)	Light Intensity (Lux)	Time	Quality
Z2 3E09-W Climate	25.40	55.80	36.75	52:17.1	0
Z2 3E09-W Climate	25.54	55.33	39.98	52:17.2	0
Z2 3E09-W Climate	25.57	54.79	38.02	54:02.6	0
Z2 3E09-W Climate	25.64	54.45	26.08	54:02.7	0
Z2 3E09-W Climate	25.64	55.26	37.56	55:48.1	0
Z2 3E09-W Climate	25.68	54.42	2.69	55:48.2	0
Z2 3E09-W Climate	25.77	54.48	10.72	57:33.6	0
Z2 3E09-W Climate	20.38	81.98	9.92	57:33.7	0
Z2 3E09-W Climate	25.83	54.51	16.00	59:19.4	0
Z2 3E09-W Climate	25.85	54.20	16.24	59:19.5	0

4.1.3 Datasets 3 - ROBOD, Room-Level Occupancy, and Building Operation Dataset

Room occupancy, or the number of users present in a room at one time, is one of the more critical attributes in an SEMS, alongside the concentration of CO2. It is shown there is a correlation between the increased number of humans in a room and the CO2 concentration, temperature, effective light intensity, and humidity in an indoor environment without proper ventilation. This set of benchmark datasets was collected for measuring and processing room occupancy, building environment, and operations; these datasets were collected between the 7th of September 2021 and the 23rd of December 2021. The datasets are managed by a group of researchers funded and supported by the National Research Foundation, Singapore, and the Ministry of National Development, 252 Singapore, under its Cities of Tomorrow R&D Programme (CoT Award COT-V4-2020-5). The dataset suits the requirements of this research with the related topic of study in which the dataset is collected. The datasets are trustworthy as they are sensor-collected by researchers employed by the Government of Singapore when researching smart environment development in Singapore [24]. Table 5 shows snippets of the collected Dataset 3.

Table 5: Dataset 3: ROBOD, Room-level Occupancy, and Building Operation Datasets

timestamp	VOC [ppb]	sound_pressure_level [dba]	indoor_relative_humidity [%]	air_temperature [Celsius]	illuminance [lux]	pm2.5 [mu_g/m3]	indoor_co2 [ppm]	wifi_connected_devices [number]	ceiling_fan_energy [kWh]
2021-09-07 00:00 +08:00	20	48.8200	72.755	26.668	0	4	430.1	0	0
2021-09-07 00:05 +08:00	20	48.4333	70.092	26.448	0.9866	3.766	427.266	0	0.0050
2021-09-07 00:10 +08:00	20	51.7533	70.67	26.281	6.7700	3.533	424.466	0	0.004882
2021-09-07 00:15 +08:00	20	49.6900	74.743	26.492	6.9833	3.7	422.4	0	0.0050
2021-09-07 00:20 +08:00	20	49.2206	74.72	26.356	6.9793	3.241	422.517	0	0.0048
2021-09-07 00:25 +08:00	20	49.8600	71.931	26.179	8.3500	3.066	424.733	0	0.0050
2021-09-07 00:30 +08:00	20	50.1566	74.321	26.453	7.4066	3.1	422.966	0	0.0050
2021-09-07 00:35 +08:00	20	50.6533	75.308	26.431	8.4633	3.033	425.166	0	0.0048
2021-09-07 00:40 +08:00	20	50.6214	72.041	26.098	10.0500	3.25	424.178	0	0.0050
2021-09-07 00:45 +08:00	20	50.4300	73.167	26.302	8.1633	3.2	425.4	0	0.0048

4.2 Statistical Analysis of the Datasets

After cleaning up the original dataset following the steps in Methodology, the next step is a statistical analysis of the datasets. There are 2 types of statistical analysis: descriptive and numerical, each of them giving different functions and working differently. Since ML models are trained and tested with the datasets, descriptive analysis should do the required functions nicely. Descriptive analysis is a simple summarization of the numbers in a set of data, providing the basic characteristics of the data. Pairing up with graphical analysis, descriptive analysis can do wonders to form the basis of almost all quantitative data analysis [28]. The descriptive analysis is done for each data type (column) of each data types, and the results are shown in Table 6 below:

Table 6: Statistical Analysis of "Clean" Datasets

Data Types	Descriptive Statistical Analysis: Features							
	COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
Dataset 1 (Clean)								
TemperatureCelcius	23369	34.532	2.617	26.7	33.1	34.8	35.6	50
HumidityPercent	23369	50.362	7.208	32.4	46.9	50.7	53.1	65.4
LightIntensityLux	23369	26.566	9.807	2	19	27	32	412
CO2PPM	23369	161.032	109.510	2	82	166	215	1141
Dataset 2 (Clean)								
TemperatureCelcius	25840	25.223	2.243	21.91	23.65	24.76	25.99	35.72
HumidityPercent	25840	62.262	8.239	38.048	56.825	62.352	68.257	82.616
LightIntensityLux	25840	17.343	28.673	0	0	0	35.189	99.976
Dataset 3 (Clean)								
TemperatureCelcius	24999	26.564	1.460	20.442	25.769	26.841	27.568	33.995
HumidityPercent	24999	69.264	9.852	45.385	61.035	68.658	78.063	93.003
LightIntensityLux	24999	39.098	60.201	0	0	1.840	72.886	2015.545
CO2PPM	24999	468.039	67.995	400	427.166	447.266	486.864	1489.357
OccupantsCount	24999	1.457	3.784	0	0	0	1	38

In general, the analysis of the columns of the clean dataset shows normal data, although the standard deviation (std) says otherwise for a few columns, mainly for the measurements of the CO2 concentration in Dataset 1, Light Intensity in Dataset 2, Light Intensity and CO2 concentration in Dataset 3. This is caused by the larger differences between the minimum values and maximum values in the data columns, which causes large deviations in those columns. Research referred to in the source of the datasets indicates the normality of those numbers.

After obtaining a clean dataset, outliers are injected into the datasets. After the injection, another descriptive statistical analysis is done on the injected dataset columns. This allows the tracking of changes to the original clean datasets after outlier injection. Table 7 below is the descriptive statistical analysis of the datasets injected with outliers.

Table 7: Statistical Analysis of Injected Datasets

	COUNT	MEAN	STD	MIN	25%	50%	75%	MAX
Dataset 1 (Injected)								
TemperatureCelcius	23369	34.5422	4.8193	-19	33.1	34.8	35.7	86
HumidityPercent	23369	50.4023	8.7008	-28	46.9	50.7	53.1	126

LightIntensityLux	23369	26.7246	26.0999	-382	19	27	32	439
CO2PPM	23369	161.0401	130.039	-1064	81	166	215	1845
InjectedOuliers	**935**							
Dataset 2 (Injected)								
TemperatureCelcius	25840	25.2303	3.5530	-12	23.64	24.76	26	66
HumidityPercent	25840	62.2869	10.4125	-25	56.763	62.352	68.347	152
LightIntensityLux	25840	17.3554	29.2830	-99	0	0	35.433	183
InjectedOuliers	**778**							
Dataset 3 (Injected)								
TemperatureCelcius	24999	26.5652	3.0086	-8	25.759	26.842	27.5756	61
HumidityPercent	24999	69.2521	12.1672	-43	60.9593	68.659	78.1278	168
LightIntensityLux	24999	39.108	60.3105	-69	0	1.9933	72.89	2015.545
CO2PPM	24999	468.9581	124.5091	-1000	427.0501	447.2857	487.7667	2168
OccupantsCount	24999	1.4577	3.7848	0	0	0	1	38
InjectedOuliers	**1218**							

From Table 7, it is immediately noticeable that despite the major changes in the min and max values of each data type, having extreme and absurd values in each column, the mean of each column stays the same as before, while the standard deviation has risen for almost all the columns, except the Occupants Count column for Dataset 3. The increase of standard deviation in each column is of different margins. The Injected Dataset 1 contains 935 outliers, the Injected Dataset 2 has 778 outliers, and the Injected Dataset 3 has 1218 outliers. When the models are used for outlier detection, the closer the number of correctly detected outliers, the more accurate the outlier detection method should be for a multivariate dataset. To see the difference between the injected and clean datasets, Figures 5, 6, and 7 are formed of the first 1000 samples of the Temperature data type of each pair of clean and injected datasets:

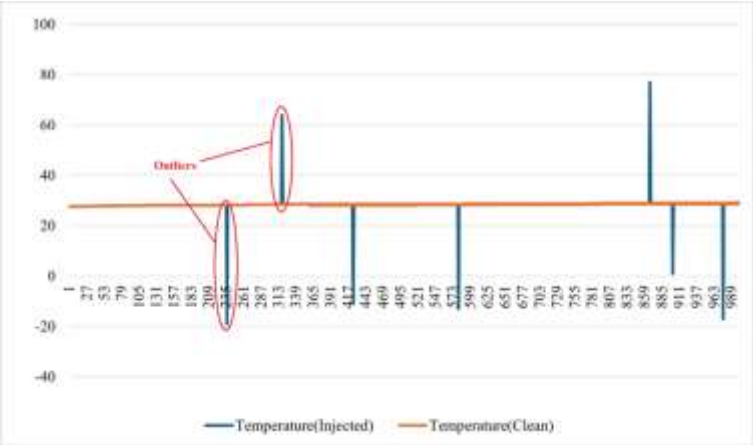

Figure 5: Comparison of Values of Temperature in the Cleaned & Injected Dataset 1

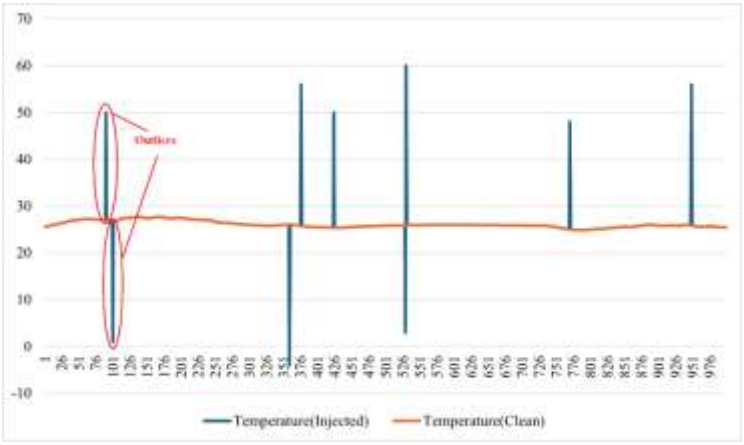

Figure 6: Comparison of Values of Temperature in the Cleaned & Injected Dataset 2

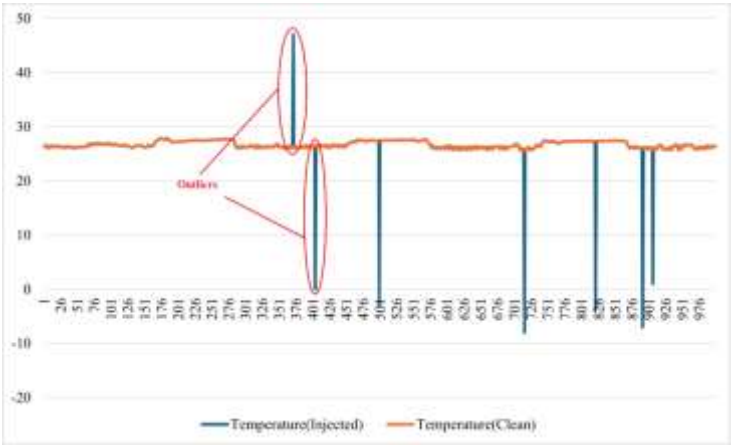

Figure 7: Comparison of Values of Temperature in the Cleaned & Injected Dataset 3

From Figures 5, 6, and 7, there are clear observations of existing outliers in the first 1000 data points of each dataset after the outlier injection, as the absurd protrusions of orange color seen in the figures (examples are marked with red ovals). The datasets are then used for training and testing of the models, and it is the models' goal to identify the outliers as shown in these figures.

4.3 Contamination Rate (Threshold) in Model Training

In the process of training and testing out the models, there are no big changes from what has been discussed in the Methodology, in terms of steps and processes. However, there is one element involved in the training of the models that requires a few rounds of experiments to get it right, which is the contamination rate or threshold in setting up the model. As explained in the methodology, the contamination rate setting in each model is closely related to how much of anomalies, in this research's case, outliers, exist in the selected datasets. After a few rounds of try and error, it is found that for Dataset 1, a contamination rate of 0.20 can score a near-perfect accuracy for all the models; for Dataset 2, it is 0.15; and for Dataset 3, it is 0.25. As introduced in the previous sections, Dataset 1 has 4 data types, Dataset 2 has 3 data types, and Dataset 3 has 5 data types. Also from the Methodology, the outlier rate for each data type of each dataset is 0.01 or 1%. A simple calculation can be used to evaluate the contamination rate for the three datasets when each of their data types is injected with 1% of outliers, which is:

$$Contamination\ Rate = (Number\ of\ Data\ Types\ in\ Dataset) * 0.05 \quad (1)$$

It is difficult to determine whether this calculation can work with other datasets or the same datasets but with more data types considered in calculations or injected with a higher percentage of outliers, which will likely be unusable. However, as of the conduction of the experiment and analysis, this formula stays true.

4.4 Performance Evaluation

The evaluation of the performance of each model is done once an output is successfully obtained from the testing phase of the models. The models will then be used to run several more times to obtain the average execution time and accuracy. Once the models are ensured to obtain a successful output every time they run, their performance in terms of accuracy and execution time is recorded. The metric of evaluating the performances is accuracy. It is easily calculated with the equation below:

$$Accuracy\ (\%) = \frac{NDO}{NIO}\ (where\ NDO\ \leq NIO), or\ \frac{NIO}{NDO}\ (where\ NIO\ \leq NDO) \quad (2)$$

Where NDO indicates the number of detected outliers by each ML model for each dataset, and NIO represents the number of injected outliers of each dataset. For the analysis, both the accuracy of an ML model in each dataset and the average accuracy of an ML model across all datasets are evaluated and recorded as results.

The second metric is the execution speed in time, indicating how long the model takes to complete its execution. It can be traced with the Performance Counter (perf_counter). A direct quote from the official documentation of the performance counter explains how it works: “Return the value (in fractional seconds) of a performance counter, i.e., a clock with the highest available resolution to measure a short duration. It does include time elapsed during sleep and is system-wide. The reference point of the returned value is undefined, so only the difference between the results of two calls is valid.” [29]
The performance counter (perf_counter()) is started before the computation process of each model for each dataset and stopped before starting the other. To find the time of execution time of each process, just deduct the ending time (tf_stop) from the starting time (tf_start). Although according to the official documentation of the perf_counter, it does not have a time unit, it will just suit itself to the system's highest available way to measure short time duration, however, the unit of time can still be deduced as seconds when deployed in Google Colab, since it normally calculates execution time of its programs in seconds too.

5. Results & Discussion

This section includes the recording and explanation of the results of testing the models. Further evaluation of the models based on the performance matrices and discussion of the result are also done. A clarification for the results is that how well each model will perform is highly dependent on both the datasets and the operating system. With the injected outlier rate and threshold mentioned in Section III (outlier injection: 1% for each measured data type; contamination rate: Dataset 1, 4 data types, 0.2 contamination rate; Dataset 2, 3 data types, 0.15 contamination rate; Dataset 3, 5 data types, 0.25 contamination rate), all the three datasets can achieve the accuracy stated in the tables above. As for the execution time, the models are deployed in Google Colaboratory, which has a disk size of 107.72 GB, a RAM size of 12.67 GB, and the Python 3 Google Compute Engine. With the training and testing of the models done, the results are shown in Table 8:

Table 8: Detected Outliers Count by Each ML Model in Each Dataset

ML Models	**Detected Outliers Count**		
	Dataset 1	**Dataset 2**	**Dataset 3**
KNN+MD	929	779	1184
MCD	949	807	1205
LOF	953	794	1230

Table 6 shows the total amount of outliers detected by each model for each dataset. The KNN+MD model detected 929 outliers in Dataset 1, 779 outliers in Dataset 2, and 1184 outliers in Dataset 3. The MCD model determined 949 data points in Dataset 1, 807 data points in Dataset 2, and 1205 data points in Dataset 3, as outliers for each dataset respectively. The LOF model observed 953 outliers, 794 outliers, and 1230 outliers in Dataset 1, Dataset 2, and Dataset 3 respectively.

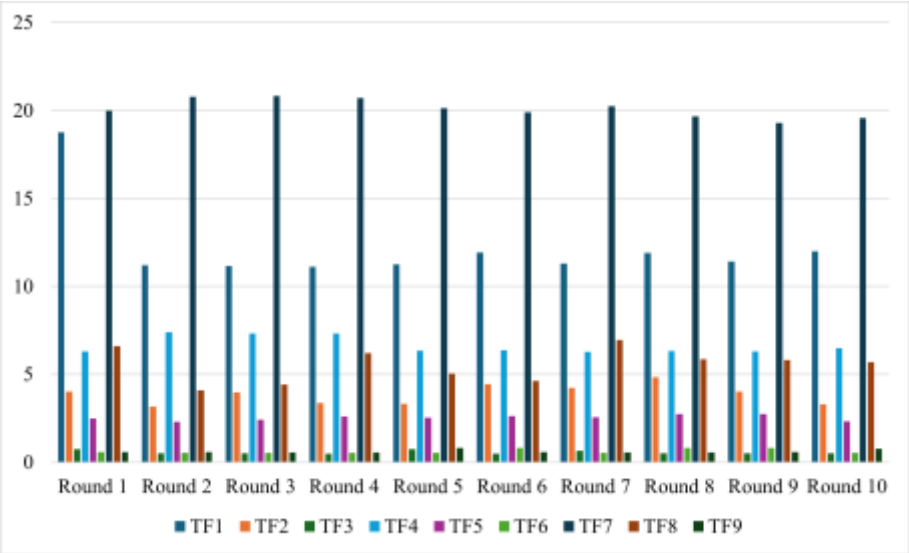

Figure 8: Execution Time for 10 Rounds of Experiments by Each Model for Each Dataset

Figure 8 above shows the execution time of each model running the train-test procedure for each dataset 10 times. In general, the KNN+MD model uses the most time to complete the training-testing procedure, MCD is in the second place and LOF uses the least time to do so.

Table 9: Performance Evaluation Metrics (For Each Dataset)

ML Models	Dataset 1		Dataset 2		Dataset 3	
	Average Accuracy (%)	Average Execution Time (Seconds)	Average Accuracy (%)	Average Execution Time (Seconds)	Average Accuracy (%)	Average Execution Time (Seconds)
KNN+MD	99.35829	12.20057042	99.87163	6.643623682	97.20854	20.11529351
MCD	98.52476	3.864896964	96.53036	2.533458346	98.93268	5.530711583
LOF	98.11123	0.568371086	98.11083	0.621414729	99.02439	0.614110351

Table 7 shows a clearer picture of both Table 6 and Figure 8, averaging out the values for the accuracy and execution time over the 10 times of experiments by each ML model for each Dataset. For accuracy, all the models achieve a similar result with little differences between them. For Dataset 1, KNN+MD achieved the highest accuracy at 99.36%, MCD in second place at 98.52% and LOF with the lowest accuracy, but still scoring an accuracy of 98.11%. In Dataset 2, KNN+MD is still the first place with an accuracy of 99.87%, LOF is now second place at 98.11% and MCD last this time, achieving only an accuracy of 96.53%. As for Dataset 3, KNN+MD is scoring last in accuracy at only 97.21%, MCD is back in second place again, with an accuracy of 98.93% and lastly, LOF has the highest accuracy at 99.02%.

For the execution time though, there are major, clearly noticeable differences between the three ML models. Across all three Datasets, KNN+MD has the longest execution time, scoring 12.20 seconds in Dataset 1, 6.64 seconds in Dataset 2, and 20.12 seconds in Dataset 3. MCD has the second fastest execution time amongst the three models, achieving 3.86 seconds, 2.53 seconds, and 5.53 seconds for Dataset 1, 2, and 3 respectively. LOF has the fastest execution speed out of the three ML models, where its execution time for Dataset 1 is only 0.57 seconds, Dataset 2 at 0.62 seconds, and Dataset 3 at 0.61 seconds, outing the other two models by a huge margin.

Table 10: Performance Evaluation Metrics (Average Across Datasets)

ML Models	AVERAGE ACCURACY (%)	AVERAGE EXECUTION TIME (Seconds)
KNN+MD	98.81282	12.98649587
MCD	97.99593	3.976355631
LOF	98.41548	0.601298722

Table 8 above concludes the average accuracy and execution time for the three models across the three datasets. From the table above, the three models have achieved similar accuracy in the outlier detection of the three datasets. Among these three models, KNN(MD) has the highest accuracy at 98.81%, followed closely by the LOF model at 98.42% accuracy. At the same time, MCD has the lowest accuracy compared to the other models

but also achieves a high average accuracy of 98.00%. All models can achieve a high accuracy at more prominent than 95% accuracy. A lot of existing research using these methods for outlier detection of complex multivariate datasets also supports and proves the high accuracy of these methods in such datasets, though often, the MCD model would achieve a higher accuracy than MD-based methods [8, 30].

However, there are significant differences between the three models in terms of execution time. LOF has the shortest execution time in both average and per database, clocking at an average of 0.60 seconds. MCD comes in second place, with an average execution time of 3.98 seconds. The KNN+MD has the longest execution time overall and per case, with an average execution time of 12.99 seconds.

The possible cause for the long execution time of the KNN(MD) model could be calculating the distances between each data point, which consumes more resources and is much costlier than other models. This weakness of KNN also causes the model to have lower efficiency when calculating large datasets and datasets with high dimensionality (or with many features/data types) [31]. Another trend in the datasets also shows that MCD gains more accuracy than the MD-based KNN in datasets with higher dimensions, becoming more accurate while keeping a quick execution speed. This is caused by both the ineffectiveness of KNN in processing datasets with many features and MCD's insensitivity to the masking effect of multivariate data as a robust distance estimator [6].

6. Conclusion

In this paper, three different ML-based outlier detection models, namely KNN+MD, MCD, and LOF, are analyzed for their performance in the outlier detection of 3 sets of multivariate SEMS-related datasets. The datasets are pre-processed before manually injected with outliers. The models are then trained with the training data and further tested with the testing data, both obtained from splitting the injected datasets. All the models achieve a high accuracy, higher than 95%, where the lowest accuracy still averages 98%. The KNN model has proven ineffective against multivariate datasets, needing an average execution time of 12.99 seconds. Meanwhile, the other two ML models have lower execution times, at 3.98 seconds for MCD and 0.60 seconds for LOF. It is vital to take note of the detected outliers, as the generated outliers in this analysis have unknown values and which data point it changed. Thus, they can be removed as outliers. However, for a real-time dataset, it is crucial for an outlier detection method to consider possible events that can cause specific data quality problems, such as out-of-range data and outliers. While being outliers, they should not be unnoticed and removed. Instead, extra attention should be paid to monitoring these abnormal data, as they might indicate various dangers and accidents, such as fires and break-ins. There are still limitations in this analysis paper, where having so many types of ML-based Outlier Detection Models developed and used, only 3 models are chosen to be tested out and experimented. In the future, we can pursue a more accurate analysis of the models by analyzing many other ML-based outlier detection models such as One-class Support Vector Machine (One-class SVM), Random Forest (RF), Isolation Forest (iForest), k-Means et cetera.

Acknowledgment

"The authors would like to express their thanks to Universiti Tun Hussein Onn Malaysia (UTHM) for support. Communication of this research is made possible through monetary assistance by Universiti Tun Hussein Onn Malaysia through GPPS (vot Q596) and the UTHM Publisher's Office via Publication Fund E15216."

Funding: "This research is supported by Universiti Tun Hussein Onn Malaysia (UTHM) through GPPS (vot Q596) and the UTHM Publisher's Office via Publication Fund E15216."

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] Saleem, M. U., Usman, M. R., & Shakir, M. (2021). Design, Implementation, and Deployment of an IoT-Based Smart Energy Management System. IEEE Access, 9, 59649–59664. doi:10.1109/access.2021.3070960

[2] Wang, R. Y., & Strong, D. M. (1996). Beyond Accuracy: What Data Quality Means to Data Consumers. Journal of Management Information Systems, 12(4), 5–33. doi:10.1080/07421222.1996.11518099

[3] Geiger, R. S., Cope, D., Ip, J., Lotosh, M., Shah, A., Weng, J., & Tang, R. (2021). 'Garbage In, Garbage Out' Revisited: What Do Machine Learning Application Papers Report About Human-Labeled Training Data? doi:10.48550/ARXIV.2107.02278

[4] Yassine, S., & Stanulov, A. (2024). A comparative analysis of machine learning algorithms for the purpose of predicting norwegian air passenger traffic. International Journal of Mathematics, Statistics, and Computer Science, 2, 28–43.
[5] Viteri, M. C., Aguilar, L. R., & Sánchez, M. (2012). Statistical Monitoring of Water Systems. In 11th International Symposium on Process Systems Engineering (pp. 735–739). doi:10.1016/b978-0-444-59507-2.50139-6
[6] Hubert, M., & Debruyne, M. (2009). Minimum covariance determinant. WIREs Computational Statistics, 2(1), 36–43. doi:10.1002/wics.61
[7] Smiti, A. (2020). A critical overview of outlier detection methods. Computer Science Review, 38, 100306. doi:10.1016/j.cosrev.2020.100306
[8] Han, J., Kamber, M., & Pei, J. (2012). Outlier Detection. In Data Mining (pp. 543–584). doi:10.1016/b978-0-12-381479-1.00012-5
[9] Lubis, A. R., Lubis, M., & Khowarizmi, A.-. (2020). Optimization of distance formula in K-Nearest Neighbor method. Bulletin of Electrical Engineering and Informatics, 9(1), 326–338. doi:10.11591/eei.v9i1.1464
[10] Raymaekers, J., & Rousseeuw, P. J. (2023). The Cellwise Minimum Covariance Determinant Estimator. Journal of the American Statistical Association, 1–12. doi:10.1080/01621459.2023.2267777
[11] Leys, C., Klein, O., Dominicy, Y., & Ley, C. (2018). Detecting multivariate outliers: Use a robust variant of the Mahalanobis distance. Journal of Experimental Social Psychology, 74, 150–156. doi:10.1016/j.jesp.2017.09.011
[12] You, L., Peng, Q., Xiong, Z., He, D., Qiu, M., & Zhang, X. (2020). Integrating aspect analysis and local outlier factor for intelligent review spam detection. Future Generation Computer Systems, 102, 163–172. doi:10.1016/j.future.2019.07.044
[13] Abuzaid, A. H. (2020). Detection of Outliers in Univariate Circular Data by Means of the Outlier Local Factor (LOF). Statistics in Transition New Series, 21(3), 39–51. doi:10.21307/stattrans-2020-043
[14] Himeur, Y., Alsalemi, A., Bensaali, F., & Amira, A. (2021). Smart power consumption abnormality detection in buildings using micromoments and improved K‐nearest neighbors. International Journal of Intelligent Systems, 36(6), 2865‐2894. doi:10.1002/int.22404
[15] Park, C. H., & Kim, T. (2020). Energy Theft Detection in Advanced Metering Infrastructure Based on Anomaly Pattern Detection. Energies, 13(15), 3832. doi:10.3390/en13153832
[16] Wu, Y., Dai, H.-N., & Tang, H. (2022). Graph Neural Networks for Anomaly Detection in Industrial Internet of Things. IEEE Internet of Things Journal, 9(12), 9214–9231. doi:10.1109/jiot.2021.3094295
[17] Jaiswal, R., Chakravorty, A., & Rong, C. (2020, August). Distributed Fog Computing Architecture for Real-Time Anomaly Detection in Smart Meter Data. 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService). doi:10.1109/bigdataservice49289.2020.00009
[18] Frikha, M. S., Gammar, S. M., & Lahmadi, A. (2021, November). Multi-Attribute Monitoring for Anomaly Detection: a Reinforcement Learning Approach based on Unsupervised Reward. 2021 10th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN). doi:10.23919/pemwn53042.2021.9664667
[19] Pourahmadi, V., Alameddine, H. A., Salahuddin, M. A., & Boutaba, R. (2023). Spotting Anomalies at the Edge: Outlier Exposure-Based Cross-Silo Federated Learning for DDoS Detection. IEEE Transactions on Dependable and Secure Computing, 20(5), 4002–4015. doi:10.1109/tdsc.2022.3224896
[20] Gulhare, A. K., Badholia, A., & Sharma, A. (2022, July). Mean-Shift and Local Outlier Factor-Based Ensemble Machine Learning Approach for Anomaly Detection in IoT Devices. 2022 International Conference on Inventive Computation Technologies (ICICT). doi:10.1109/icict54344.2022.9850880
[21] Bhatti, M. A., Riaz, R., Rizvi, S. S., Shokat, S., Riaz, F., & Kwon, S. J. (2020). Outlier detection in indoor localization and Internet of Things (IoT) using machine learning. Journal of Communications and Networks, 22(3), 236–243. doi:10.1109/jcn.2020.000018
[22] Wibisono, A. (2020). Data for: Short-term Prediction of CO2 Concentration based on a Wireless Sensor Network. doi:10.17632/6D798DKHPZ.1
[23] Kusy, B., Hovington, L., Hu, W., & Rana, R. (2012). QCAT Smart Office environment - Humidity. doi:10.4225/08/50629B0DE50C7
[24] Tekler, Z. D., Ono, E., Peng, Y., Zhan, S., Lasternas, B., & Chong, A. (2022). ROBOD, room-level occupancy and building operation dataset. Building Simulation, 15(12), 2127–2137. doi:10.1007/s12273-022-0925-9
[25] Pawluszek-Filipiak, K., & Borkowski, A. (2020). On the Importance of Train–Test Split Ratio of Datasets in Automatic Landslide Detection by Supervised Classification. Remote Sensing, 12(18), 3054. doi:10.3390/rs12183054

[26] yzhao062. (2024, February). PYOD Official Documentation. Retrieved from https://pyod.readthedocs.io/en/latest/pyod.models.html#pyod.models.knn.KNN
[27] Varoquaux, G., & Colliot, O. (2023). Evaluating Machine Learning Models and Their Diagnostic Value. In Neuromethods (pp. 601–630). doi:10.1007/978-1-0716-3195-9_20
[28] Rashid, C. A. (2021). The importance of statistical analysis in accounting research. Journal of Global Social Sciences, 2(7), 71–84. doi:10.58934/jgss.v2i7.26
[29] Foundation, P. S. (2023, October). time — Time access and conversions. Retrieved from https://docs.python.org/3/library/time.html
[30] Dashdondov, K., & Kim, M.-H. (2021). Mahalanobis Distance Based Multivariate Outlier Detection to Improve Performance of Hypertension Prediction. Neural Processing Letters, 55(1), 265–277. doi:10.1007/s11063-021-10663-y
[31] Boateng, E. Y., Otoo, J., & Abaye, D. A. (2020). Basic Tenets of Classification Algorithms K-Nearest-Neighbor, Support Vector Machine, Random Forest and Neural Network: A Review. Journal of Data Analysis and Information Processing, 08(04), 341–357. doi:10.4236/jdaip.2020.84020