IST-Africa 2023 Conference Proceedings
Miriam Cunningham and Paul Cunningham (Eds)
IST-Africa Institute and IIMC, 2023
ISBN: 978-1-905824-71-7

Anomaly Detection in IoT Data

Jason N. KABI[1], Ciira wa MAINA[1, 2], Edwell T. MHARAKURWA[2]
[1]Centre for Data Science and Artificial Intelligence, Dedan Kimathi University of Technology. P.O. BOX - PRIVATE BAG – 10143, Dedan Kimathi - Nyeri, Kenya.
[2]Department of Electrical and Electronic Engineering, Dedan Kimathi University of Technology. P.O. BOX - PRIVATE BAG – 10143, Dedan Kimathi - Nyeri, Kenya.
Email: jason.kabi@dkut.ac.ke, ciira.maina@dkut.ac.ke, edwell.tafara@dkut.ac.ke

Abstract: This work describes the performance-evaluation of various unsupervised classical machine learning algorithms in time series outlier detection. The aim is to test the robustness of known classical models that act as baselines in anomaly detection. IoT offers flexibility for various anomalies detection algorithms to be tested since the data collected is voluminous and the types of anomalies found are diverse. By deploying fine-tuned, long-established models, researchers can improve on the quality of the data they release from or use in various studies. This work also provides an insight into how time series data properties such as non-stationarity can affect anomaly detection and how operations such as windowing can be used to mitigate the effects and achieve desirable results. The experiments done show that, with some fine-tuning and data pre-processing, classical outlier detection methods' performance can be enhanced and utilized in IoT data quality control.

Keywords: Outlier Detection, Sensor Nodes, Time Series, Windowing, F1-Score, Internet of Things

1. Introduction

In IoT, the success of all monitoring or data collecting practices is highly dependent on the proper operation of the sensor nodes deployed, specifically, the sensing element. In scenarios, such as outdoor deployment, the desired operation is strenuous to guarantee since the sensing elements are fragile and prone to damage which can in turn lead to a malfunction. Ecosystem monitoring is one of the applications where outdoor deployment of sensor nodes to collect data such as river water level is required. The sensing hardware on the nodes is constantly exposed to undesired factors such as severe weather conditions, invasion by the insect population and vandalism which make damage very likely [1, 2].

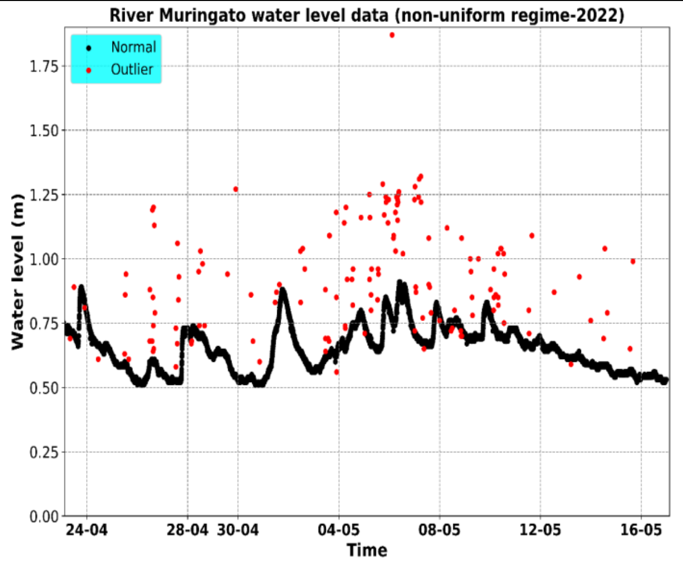
Combined with the external factors mentioned, other internal factors such as power limitations may lead to the presence of anomalous data points in the data collected. In science, anomalies are observations that are dissimilar to the presumption produced by a scientific experiment such as measuring. The presence of outliers in raw data raises the need of ensuring that high quality data is being obtained from sensor nodes. Since the operation of sensor devices is automated, time based and the intervals involved are short, they generate large amounts of data that cannot be processed manually, this dictates that the data control methods employed have to be automated, extensible and quick enough to be suitable for real time data [2].

Anomaly or outlier detection is one of the operations that fall under the data quality control category. Outlier detection is a widely studied area in machine learning and data acquisition. Nowadays, the vast amount of research and information around anomaly detection in machine learning and the availability of vast, fast and fairly cheap data processing resources have made development and implementation of custom and complex anomaly detection models possible. There are deep learning anomaly detection techniques and machine learning packages for IoT time series data [3]. Despite the fact that these complex models have practically adequate results, they take a lot of time to develop and extensive processing resources to implement hence, their necessity and advantages over traditional anomaly detection machine learning models can be argued. On the other hand, traditional anomaly detection models have working principles that are easy to understand and some of these principles are utilized in the development of the complex models. Also, these long-established models are easy to implement with minimum processing resources and the results are satisfactory hence a case about their robustness can be made. In machine learning, most of the traditional methods fall under the unsupervised learning category [4].

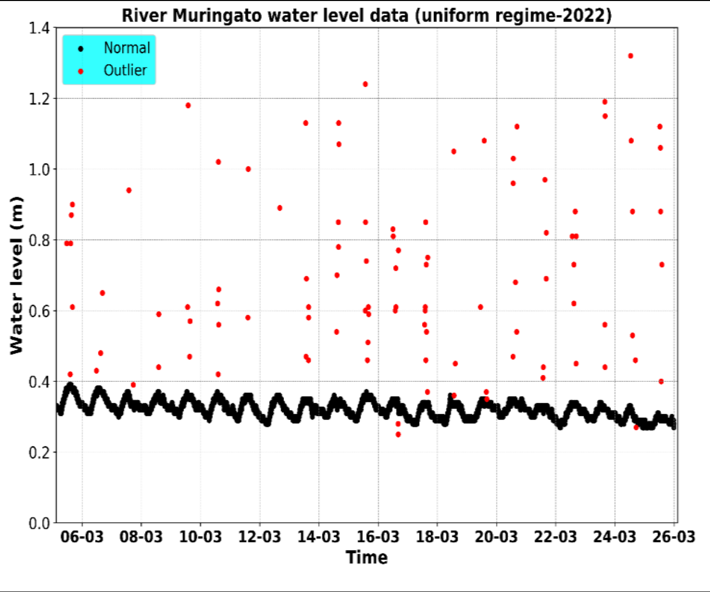
In this paper, the focus is on anomaly detection in time series data using traditional unsupervised methods in a bid to prove their utility and robustness. A time series data set is a sequence of data points indexed in time order. The resolution of time series data is usually predetermined and stated in the sensor node firmware. Anomaly detection methods are basically targeted at detecting and eliminating the erroneous data points in a given set.

The models considered in this work were all unsupervised machine learning models. In unsupervised learning, the models are classified in specific groups depending on the model's working principle e.g., Clustering based-models and proximity-based models. Unsupervised machine learning is a machine learning technique in which the dataset has no labels [2]. A model is fitted to find the hidden patterns and insights – exploratory data analysis – in the given data. Mostly unsupervised models/algorithms are used to group a set of data points into different clusters depending on a predetermined threshold. Data points in the same cluster have similar properties whereas data points in different clusters have different properties [2-4]. By employing these algorithms in anomaly detection, the outliers are grouped into one cluster whereas the normal/relevant data samples are grouped in another. The data points in these clusters are then assigned cluster labels which identify a data point with its cluster. By exploiting the label feature the outliers are eliminated from the main dataset. [5].

The time series data extracts utilized in this study were obtained from a river water level monitoring sensor node deployed along River Muringato – Muringato sub-catchment in Nyeri county, Kenya. The test data sets were divided into two classes: the uniform and the non-uniform regime as shown in Figure 1.



(a)- Test Data 1 For the Study (NUR)



(b)-Test Data 2 For the Study (UR)

Figure 1: Test Data Sets

Test data 1 shown on Figure 1a was allocated the label -non-uniform regime (NUR) due the non-uniform nature of the time series whereas Test data 2 was allocated the label – uniform regime (UR) since the water level did not change over time. The operating interval of the sensor node was 5 minutes and was in-situ for 18 consecutive months from January 2021. The node was transmitting data to a network server through a LoRaWAN[1] network [6]. From the network server the data was being re-routed to a cloud-based time series database for permanent storage. Due to the interference by weather conditions and vandalism, the dataset contained a significant number of evenly spread-out anomalies. Figure 1 shows the data sets considered in this work and Table 1 illustrates the data information. Figure 2 depicts the deployed sensor node [7].

Table 1: Data Specifics Table

Test Data 1 (NUR)	
Variable	River Muringato water level data
Time	24th April 2022 to 17th May 2022
Outlier count	150 data points (RED)
Normal data point count	6562 data points (BLACK)
Test Data 2 (UR)	
Variable	River Muringato water level data
Time	06th March 2022 to 26th March 2022
Outlier count	101 data points (RED)
Normal data point count	5762 data points (BLACK)

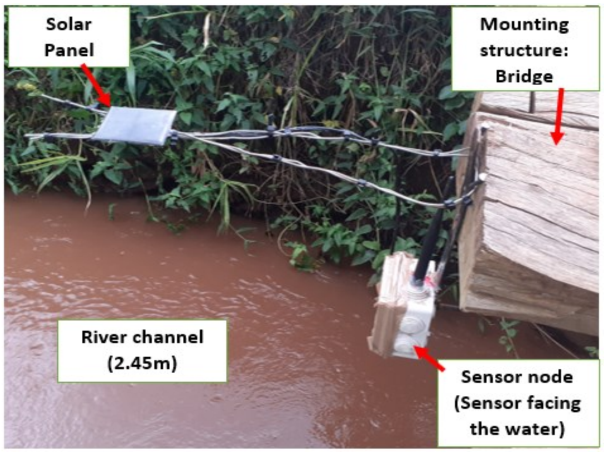

Solar Panel
Mounting structure: Bridge
River channel (2.45m)
Sensor node (Sensor facing the water)

Figure 2: Sensor Node Along River Muringato, Nyeri County, Kenya

This work describes the comparison of performance among various classical machine learning models in detecting anomalies present in time series data. The data specimens used were acquired directly from a deployed sensor node hence no modifications were made to the data extracts used. Section 2 presents the objectives, section 3 outlines the methodology used in the performance-comparison study while section 4, presents the results. Section 5 describes the business benefits while section 6 concludes the paper.

2. Objectives

2.1 Main Objective

To compare the outlier detection performance of different unsupervised, basic machine learning models in IoT time series data.

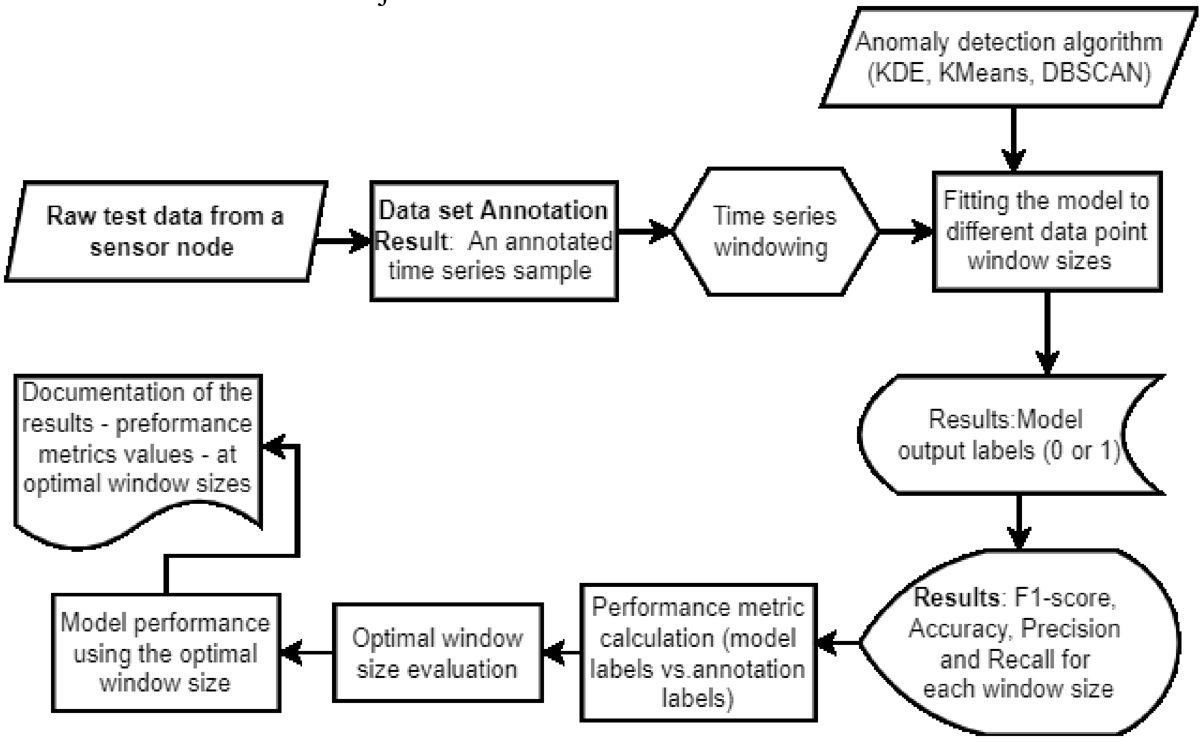
2.2 Specific Objectives

a) Manual annotation of test data samples.

b) Detecting anomalies in the test samples using the considered algorithms (labelling using models).
c) Performance evaluation of the model by leveraging the manual annotation and model labels.

3. Methodology

The performance comparison of the unsupervised anomaly detection machine learning algorithms described in this paper followed the procedure outlined in Figure 3. The programming was done using Python. Since the used-model frameworks are provided in the python library named; Scikit-learn, the main task was in tuning/adjusting the models' parameters accordingly. These parameters play a major role in dictating the performance of the models. Other Python data preparation and manipulation packages were also utilized in order to achieve the stated objectives.



Anomaly detection algorithm (KDE, KMeans, DBSCAN)
Raw test data from a sensor node
Data set Annotation Result: An annotated time series sample
Time series windowing
Fitting the model to different data point window sizes
Results:Model output labels (0 or 1)
Results: F1-score, Accuracy, Precision and Recall for each window size
Performance metric calculation (model labels vs.annotation labels)
Optimal window size evaluation
Model performance using the optimal window size
Documentation of the results - preformance metrics values - at optimal window sizes

Figure 3: Methodology Block Diagram

3.1 Test Data

The main reason behind the consideration of the non-uniform regime (NUR) shown on Figure 1(a) was the non-uniform profile of the time series. The profile was caused by the changes in water level after heavy downpours in and around the Muringato watershed during the time stretch indicated on the x-axis. The specimen also contained anomalies at interesting points of the profile which called for a sophisticated approach for detecting them. Uniform regime (UR) shown on Figure 1(b) was also considered due to the uniform profile/regime. The data shows lack of rainfall during the time stretch outlined on Table 1. Before the extraction of the two specimens from the 18-month time series, the whole data was validated by comparing it to the rainfall data in the Muringato watershed. During the sensor node deployed period, manual measurements which helped in the validation process were also being taken.

3.2 Annotation of the Data Specimens

Before feeding the selected test data sets to the models, the extracts underwent labelling to identify the anomalous data points and the normal/relevant data points. The labelling was done manually using a one-day labelling window. The relevant data points representing the main water level trend were identified with the binary label 0 whereas the outliers were identified with the binary label 1. The ratios of the outliers to the total count in both cases were small which indicated that the sensor node was efficient and performing well. The short interval data collection produced a high-variance, low-bias time series as shown on Figure 1.

3.3 Windowing

During the fitting of test samples to the models, a windowing process was adopted. The procedure was adopted after realization that the models were classifying some relevant data points at the peaks and troughs as anomalies (clipping) – Figure 4-a. Also, some models were not able to detect the outliers in between two peaks due to the models' architectures. By windowing, the NUR was broken down to sections therefore reducing the effect of the non-uniformity and improving the model performance. During the fitting of the UR, the windowing process was not utilized for some of the models since the data was uniform enough for it to be fitted as a whole. To score the model after fitting and label generation, the original test set was being reconstructed (stitching-back) from the window-length data points. In order to determine the optimal window size for a particular model, different window sizes were considered and model performances at each of these window sizes were analyzed.

3.4 Unsupervised Machine Learning – Model Fitting – Anomaly Detection

Highlighted in Table 2 below are the unsupervised models that were considered in this work. Unlike supervised machine learning models, unsupervised frameworks are not provided with target labels for the model fitting and prediction process. Instead, the models learn using the fundamental concepts that define them. As shown on Figure 4, the test sets (NUR and UR) were fitted and the results (labels) were generated via prediction and stored. The label feature aided in identifying the points labelled as anomalies by the various models and also helped in scoring the models.

Table 2: Models Considered in The Study

S/N	Model	Fundamental operating concept
1	Local Outlier Factor (LOF)	Proximity/density-based clustering model [2]
2	Interquartile range (IQR)	Statistical clustering model [2]
3	KMeans	Proximity based clustering model [2]
4	Isolation forest	Isolation based clustering model [2]
5	Kernel Density Estimation	Density based (probabilistic) clustering model [2]
7	DBSCAN	Density based clustering model [2]
8	OneClassSVM	Proximity/density-based clustering model [2]
9	Gaussian mixture model (GMM)	Density based (probabilistic) clustering model [2]
10	Elliptic Envelope	Density based clustering model [2]

3.5 Model Performance Evaluation

Model evaluation or model scoring is realized using performance metrics. Since the anomaly detection problem in this work was a classification task, the metrics considered were; the F1-score. As a scoring metric, the F1-score is the harmonic average of the precision and recall score. Confusion matrices were also used in result visualization. The common Python library utilized in the calculation of the metric listed was the Sklearn.metrics().

3.6 Optimal Window-Size Evaluation

To evaluate the optimal window-size, the performance metric (F1-Scores) results for various window sizes were first visualized. By visualizing the performance curves across the window sizes, it was easy to identify the window size with the highest performance metric figures and isolate it. The performance metrics that matched the optimal data point window were also identified and recorded for use in model comparison analysis.

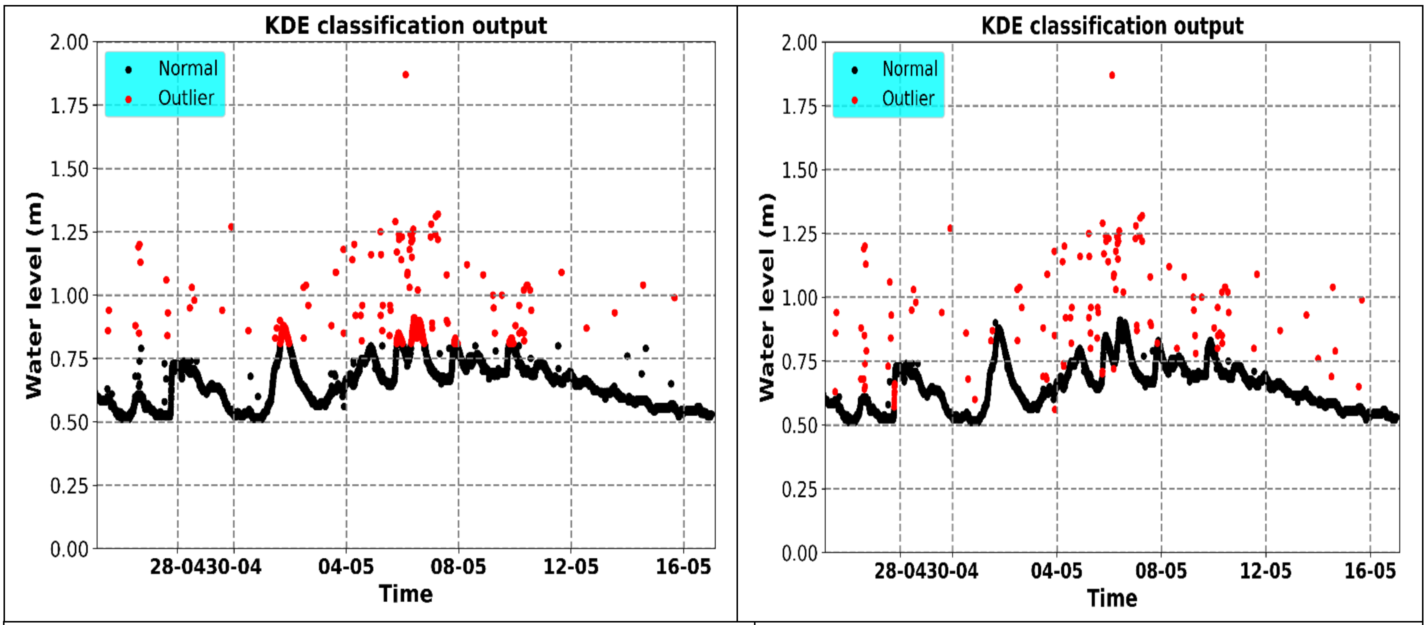
4. Results

To outline the results of the windowing process and the determination of the optimal window size, the Kernel Density Estimation (KDE) algorithm was considered. KDE utilizes the probability distribution of the data in clustering and anomaly detection. Table 3 highlights the performance of KDE on the NUR using an optimal window size figure and without using a window.

Table 3: KDE Performance on Test Data 1

Performance Metric	Without windowing	Optimal window (160 data points)
Accuracy score	0.9328	0.9943
Precision score	0.2008	0.9117
Recall score	0.6733	0.8268
F1-score	0.3093	0.8671

Figure 4a shows the performance of the KDE on the NUR without the utilization of a data point window while Figure 5 shows the results with an optimal data point window in place. In Figure 4 the clipping phenomena described in the methodology is evident and as stated some of the outliers between the peaks were not detected. Figure 4b shows an improvement in performance after windowing hence some of the setbacks of having non-uniformity were mitigated.



(a) *Without a window* | (b) *With an optimal window (160 data points)*

Figure 4: KDE Performance (NUR)

4.1 Optimal Window Size Determination

The plot on Figure 5 shows the change in performance against a varying data point window size for the KDE algorithm. The optimal window was discovered to be 160 data points since only at the 160 mark the recall score goes above the 0.8 mark. The spike in recall corresponded to a spike in the other metrics as shown in Figure 5 and Table 3. The accuracy curve was not affected by the windowing. This was as a result of the small outlier: total data point tally ratio. This also meant that the accuracy was not a desirable metric in comparing the performance of the models. The F1-score combines the Recall and the Precision; hence it was considered to be the deciding metric.

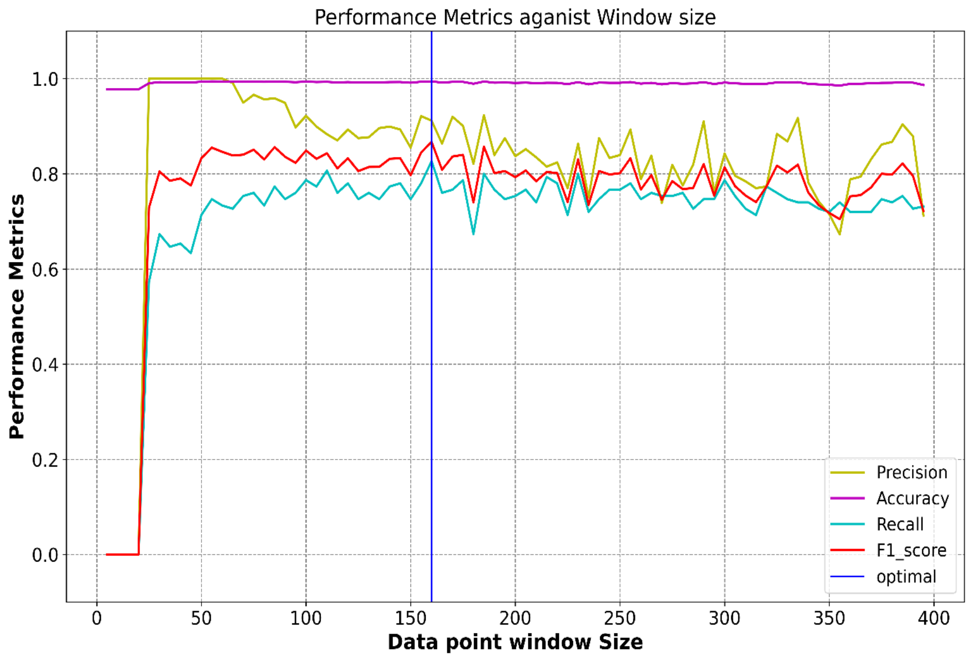


Figure 5: Performance Metrics Against Data Point Window Size (Optimal Window Size 160 Data Points – F1-Score = 0.8671) - NUR

4.2 KDE Results for Test Data 2

Figure 6 shows the KDE performance on the UR. The data had a uniform regime; lack of change in water level. As stated, the data did not require a windowing process.

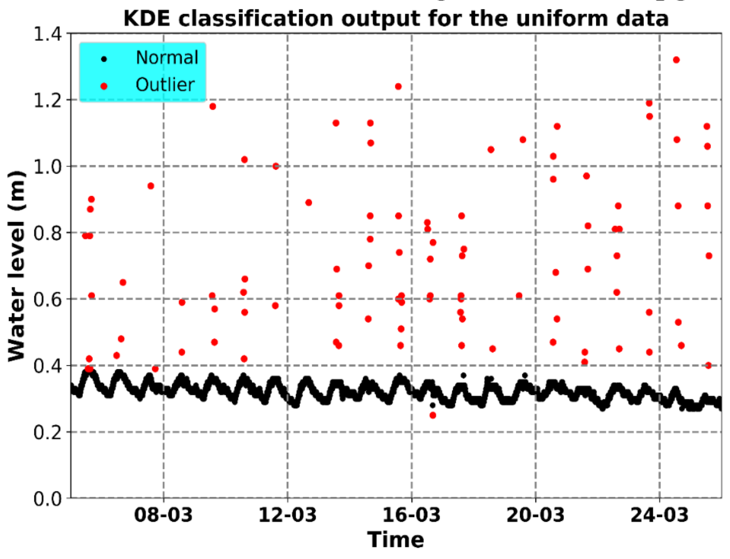
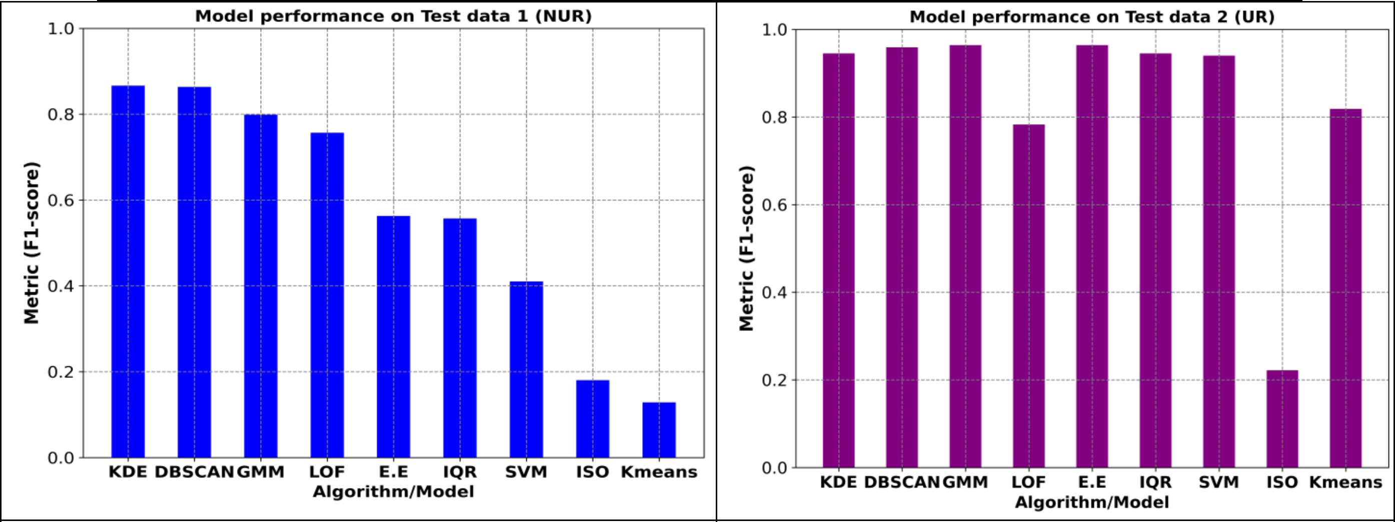


Figure 6: KDE Performance - UR (F1-Score = 0.9453)

Table 4 shows the performance (F1-scores) of the other algorithms that were considered in the study. The parameters column indicates the tuning model parameters which were used to achieve that indicated results/score. The algorithms with the high F1-score (close to 1) in both test datasets were able to detect and eliminate a large percentage of the anomalies from the test data in both regimes.

Table 4: Models' Performance

	Test Data 1 (non-uniform regime)-NUR			**Test Data 2 (uniform regime) -UR**		
Model	F1-Score	Parameters	Window size	F1-Score	Parameters	Window size
Local Outlier Factor (LOF)	0.7567	n_neighbors = 246 metric= Euclidean	280	0.7826	n_neighbors = 14, metric= Euclidean	288
Interquartile range (IQR)	0.5571	-	124	0.9452	-	None
KMeans	0.1285	K = 2	268	0.8187	K = 2	576
Isolation forest	0.1802	n_estimators = 95	None	0.2220	n_estimators = 60	None
Kernel Density Estimation	0.8667	Kernel = Gaussian BW = 0.00635	160	0.9453	Kernel = Gaussian BW = 0.00635	None
DBSCAN	0.8636	eps = 0.07, min_samples= 15	25	0.9588	eps = 0.015, min_samples= 60	None
OneClassSVM	0.4104	kernel= 'linear' gamma='scale' nu=0.948	610	0.9400	kernel= 'linear' gamma='scale' nu=0.948	None
Gaussian mixture model (GMM)	0.7986	-	18	0.9641	-	None
Elliptic Envelope	0.5628	Contamination = 0.04	249	0.9641	Contamination = 0.016	None



(a) (b)

Figure 7: (a) Models' Performance on Test Data 1(NUR) (b) Models' Performance on Test Data 2(UR)

5. Business Benefits

Most time series datasets obtained from instruments which interface the real world often contain anomalies. Recently, the demand for environmental datasets such as river water level, temperature and rainfall has increased due to factors such as climate change studies. Therefore, producers of such datasets have to invest in data cleaning pipelines. By setting up automatic and efficient data cleaning pipelines which highly consist of anomaly detection, reliable time series datasets can be produced. This practice improves and maintains the third-party data user confidence in the data available which in turn translates to more revenue if the datasets are commercialized. Also, user confidence can be improved by granting access to these data pre-processing pipelines in terms of model selection and output visualization.

Due to the diversity of the time series data being collected nowadays, various organizations are hiring developers with an aim of building data quality control pipelines that can process real-time data from deployed nodes. Also, various developers are working on prototype anomaly detection models that can be commercialized.

6. Conclusions

The work presented in this paper involves evaluating the performance of various classical machine learning algorithms for detecting anomalies in different sets of time series data. By testing the performance of various models, researchers are able to evaluate which model is fit for their use case and what model parameters need optimization. As indicated in the results, the model performances for the two test cases were diverse. The superior performance of the KDE, DBSCAN, GMM and LOF in both cases was credited to the factors such as, algorithm properties that govern the clustering, the fine tuning of the parameters and the windowing process that proved to be vital. The windowing operation greatly improved the performance of some of the models, proving that basic data pre-processing techniques can enhance the performance of some traditional outlier detection models. Based on the performance of the models highlighted in this work it can be concluded that fine-tuned classical baseline anomaly detection algorithms can still be utilized in modern fields such as IoT and by adjusting the models' parameters the models can be manipulated to fit any use case due to their robustness. Therefore, in future experts in data analysis and curation should consider the implementation of traditional outlier detection methods before the development of complex algorithms that can only be fitted to specific use cases.

Acknowledgement

We thank Stephen Githinji, a researcher at the Centre for Data Science and AI – Dedan Kimathi University of Technology for the input in data analysis and visualization.
Funding: This work was supported by the National Research Fund (NRF) Kenya through the Multidisciplinary Research Award; Canada's International Development Research Centre (IDRC) and Swedish International Development Cooperation Agency (SIDA) through the Artificial Intelligence for Development in Africa (AI4D Africa) program. We appreciate support from Data Science Africa (DSA) through the Affiliated Centre Program.

References

[1] L. An, A.-J. Tu, X. Liu, and R. Akkiraju, "Real-time Statistical Log Anomaly Detection with Continuous AIOps Learning," Proceedings of the 12th International Conference on Cloud Computing and Services Science. 2022. doi: 10.5220/0011069200003200.

[2] F. Giannoni, M. Mancini, F. Marinelli "Anomaly Detection Models for IoT Time Series Data," 2018 ArXiv preprint arXiv: 1812.00890.

[3] B. Sharma, L. Sharma, and C. Lal, “Anomaly Detection Techniques using Deep Learning in IoT: A Survey,” 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE). 2019 [Online]. Available: http://dx.doi.org/10.1109/iccike47802.2019.9004362

[4] M. Munir, S. A. Siddiqui, A. Dengel and S. Ahmed, "DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series," in IEEE Access, vol. 7, 2019, doi: 10.1109/ACCESS.2018.2886457.

[5] A. A. Cook, G. Mısırlı and Z. Fan, "Anomaly Detection for IoT Time-Series Data: A Survey," in IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6481-6494, July 2020, doi: 10.1109/JIOT.2019.2958185..

[6] J. Akoto and T. Salman, “Machine Learning vs Deep Learning for Anomaly Detection and Categorization in Multi-cloud Environments,” 2022 IEEE Cloud Summit. 2022.

[7] J. Kabi and C. Maina, "Leveraging IoT and Machine Learning for Improved Monitoring of Water Resources - A Case Study of the Upper Ewaso Nyiro River," 2021 IST-Africa Conference (IST-Africa), 2021, pp. 1-8.