Anomaly Detection for Raw Water Quality – A Comparative Analysis of the Local Outlier Factor Algorithm and the Random Forest Algorithms

Nahshon Mokua
School of Engineering,
Dedan Kimathi University of Technology
Private Bag 10143
Nyeri - Kenya

Ciira wa Maina
Center for Data Science and Artificial Intelligence,
Dedan Kimathi University of Technology
Private Bag 10143
Nyeri – Kenya

Henry Kiragu
Electrical and Communication Engineering Department,
Multimedia University of Kenya
P.O BOX 15653 - 00503
Nairobi – Kenya

ABSTRACT
The increased use of real-time water quality monitoring using automated systems with sensors demands and makes it possible to identify unexpected values in time. Anomalies are brought by technical issues that are likely to prevent detection of problematic data manually at the incoming data rate. Use of machine learning approaches to detect anomalies for water quality data is the main focus of this article. There is analysis of four time series machine learning anomaly detection techniques: the local outlier factor, the isolation forest, the extended isolation forest and robust random cut forest. A subset data collected from deployment of sensors in a water treatment plant (Nyeri-Kenya) was used to carry out extensive analysis of experiments of the afore-mentioned techniques; for turbidity and pH parameters. There was successful correct detection of all outliers for both subsets by the local outlier factor algorithm, contrary to the rest of the other algorithms considered. As per the primary experiment, the local outlier factor emerged the fastest. Also, it was easier use as long as there was selection of optimum parameters. Moreover, analysis of the four techniques demonstrated that with or without training, it is a powerful tool for water quality anomaly detection and hence a feasible approach.

General Terms
Water quality; anomalies; machine learning; data; algorithm

Keywords
Water quality monitoring; anomaly detection; local outlier factor; isolation forest; extended isolation forest; robust random cut forest.

1. INTRODUCTION
For survival, almost all aspects of life essentially need water, which occupies more than 70% of earth's surface [1]. Water quality can be safeguarded by water quality monitoring and management. However, determining the quality of a water body may be challenging majorly because water is a huge network consisting of linked parts such as lakes, rivers, creeks, estuaries and wetlands. Varying pollution levels in these linked parts is mainly the reason for the difficulty in assessing quality of water.

The chemical, physical, biological and other contents of water that vary through the seasons and geographic parts are described by water quality [2]. Environmental factors and human activities impact on the water quality. The major determinants of available water quality and quantity are climate, geological and hydrological factors. Conversely, impact of human activities on water quality is vast and the degree to which they disrupt the ecosystem and restrict use of water varies [3]. Human and environmental health is affected by quality of water, therefore, determination and prevention of contamination issues is possible through constant monitoring of water. A problem in posterior analysis caused by anomalies in water quality are likely to cause decisions and conclusions that are faulty [4]. Causes of anomalies include phenomena in the ecology such as floods or rainfall. Anomalies are a constant expectation that also likely result from unexpected human and technical errors. For example, errors in communication between server and sensor node may occur, dirt in the sensor probe, pulling the sensor out of water for cleaning, malfunctioning of equipment, just to mention a few.

In collaboration with Department of Electrical and Electrical Engineering, and the Center for Data Science and Artificial Intelligence (DSAIL) both in Dedan Kimathi University of Technology (DeKUT), alongside the Nyeri Water and Sanitation Company (NYEWASCO) all in Nyeri-Kenya, a raw water quality monitoring system was developed and deployed at the NYEWASCO water treatment plant [5]. The focal point of this research was finding possible approaches to detect anomalies automatically in time series water quality data. There has been development of new technologies for remote autonomous sensing of water systems. The main aim of this system was to monitor pH and turbidity for raw water.

2. LITERATURE

2.1 Anomalies
Instances or situations where subsets are considered different from the information are regarded as novelties, anomalies, noise or outliers [6]. There are three classifications of anomalies: An individual data point away from the rest in a subset is referred to as point or global anomalies. For conditional or contextual outliers, information or data is anomalous in specific contextual framework, but not otherwise. This research is concentrating on this anomalies. Lastly, a collection of information or data in a particular subset is referred to as collective anomalies. This is illustrated in Figure 1 below. Finding out the sequences or patterns in information data that do not replicate expected output or trend is a process referred to as anomaly detection [7].

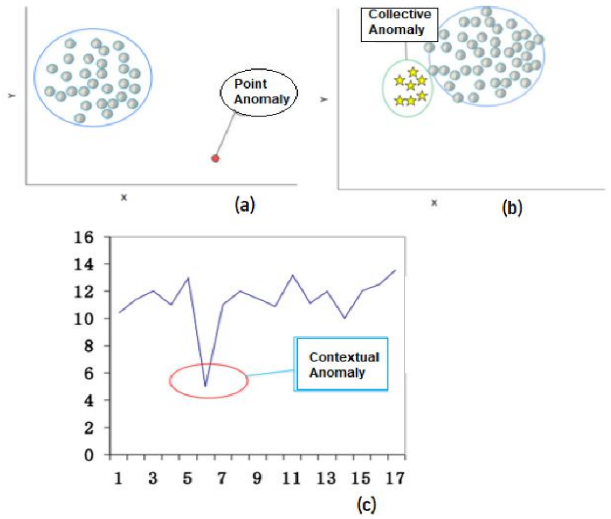

Figure 1: Types of anomalies [7]

2.2 Machine Learning Algorithms

Machine learning originated from pattern recognition and is a technique of data analysis that explicitly gives computers the capability to learn minus any form of programs. Algorithms that can learn from data, identify patterns and make decisions are explored, examined, and developed. The main classes of machine learning algorithms include supervised, unsupervised or semi-supervised learning. Labelled data for training is required in supervised learning while unsupervised learning does not entail either desired classified or labeled test data. Its algorithms can infer a function to describe hidden data structures from unclassified test data short of any guidance. On the other hand, semi-supervised learning lies between supervised and unsupervised learning [7]. These are illustrated in Figure 2 below.

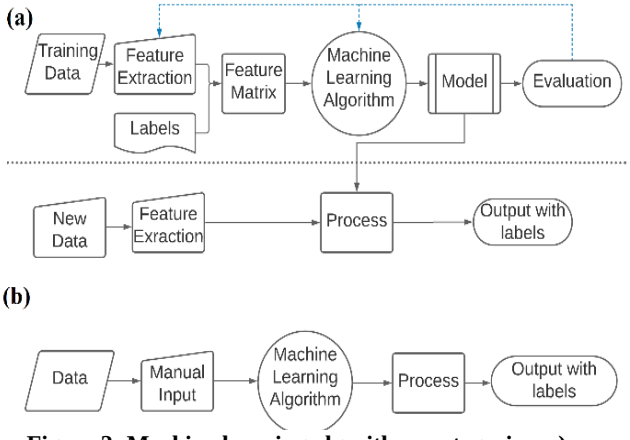

Figure 2: Machine learning algorithms categories: a) Supervised Learning, b) Unsupervised Learning

2.3 Anomaly Detection Algorithms

Local density gives a base for nearest neighbors anomaly detection techniques which are built on the algorithm of *k*-nearest neighbors algorithm. Clustering-based anomaly detection is unsupervised learning. There is an assumption by these algorithms that similar objects tend to belong to similar groups (clusters) and that distance determines the similarity. Categorization of data into different classes with labels is done by the classification technique. Anomaly detection involves only two distinct classes: normal class and abnormal class. Calculation of deviations from common statistical properties of a distribution and flagging of abnormal points with deviations above a threshold is done by statistical methods. The commonly used properties for such calculations are mean, median, mode, and quantiles. Random forests is the learning algorithm that functions by construction of decision trees' multitude at training time and class outputting (i.e. class mode-classification; prediction of mean-regression). This algorithm classifies and regresses individual trees. [7]. The local outlier factor and random forests are chosen because of the following reasons [8]:

- Time series data with only one variable such as water quality data is not suitable for clustering.
- There is a requirement of model training and labeled data for neural networks and support vector machine (SVM) algorithms.
- Training a generic model for classification is difficult and different results for the same data point may be generated by different models.
- LOF is based on K-NN and extensions of LOF are other nearest neighbors-based algorithms.

2.3.1 Local Outlier Factor

Local outlier factor (LOF) is an unmonitored outlier detection algorithm that compares the local density of information data to its neighbors [9]. This was the number one algorithm based on local density and k-neighborhood. It was the first algorithm based on *k*-neighborhood and local density. LOF is the anomaly score of each sample in the training data set and it indicates the degree of its outlier-ness as shown in figure 3.

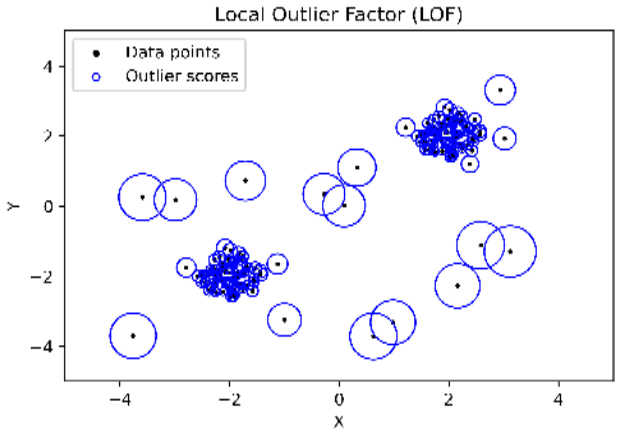

Figure 3: Local Outlier Factor (LOF) outlier-ness degree

Determination of the local neighborhood of LOF is based on the number of nearest neighbors. The following descriptions are used for the completion of the whole process by the LOF. The *k*-distance (p), is well-defined to be the distance $d\ (p, o)$ amid p and an object $o \in D$ so that for at least k instances $o' \in D\backslash\{p\}$ it stands that $d(p, o') \leq d(p, o)$, and for at most k - 1 instance $o' \in D\backslash\{p\}$ it stands that $d(p, o') < d(p, o)$. The k-distance neighborhood of instance p is a subset with instances whose distances are not greater than the k-distance from it [9]. With regard to instance o, the definition of reachability distance of instance p is:

$$\boldsymbol{reach - dist_k = max\{k - distance(o), d(p, o)\}} \quad \mathbf{1}$$

Figure 4 shows examples of reachability distance for $k = 4$. Between these two instances, the reachability distance, when they are far away from each other, is their actual distance (like o and $p2$); but, the reachability distance is $k - distance$ of o if they are close enough (like o and $p1$). Consequently, there can be a significant reduction of the statistical

fluctuations of $d(p,o)$ for all of the $p's$ close to o. The parameter k controls the strength of this smoothing effect and therefore, the reachability distances likenesses become more within the same neighborhood for higher the values of *k* [9].

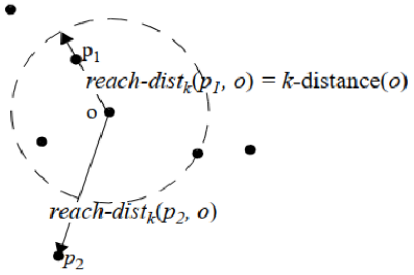

Figure 4: Example of reachability distance for k=4 [9]

For object $o \in N_{MinPts(p)}$, the local reachability density of point *p* is defined as:

$$\boldsymbol{lrd_{MinPts}(p) = 1/(\frac{\sum o \in N_{MinPts(p)} reach-dist_{MinPts(p,o)}}{|N_{MinPts(p)}|})} \quad \mathbf{2}$$

Where;

- $MinPts$ stipulates a least numeral of objects
- $reach-dist_{MinPts(p,o)}$ represents the reachability distance of object *p* with respect to object *o*

For object $o \in N_{MinPts(p)}$, the definition of LOF of *p* is:

$$\boldsymbol{LOF_{MinPts(p)} = 1/(\frac{\sum o \in N_{MinPts(p)} \frac{lrd_{MinPts}(0)}{lrd_{MinPts}(p)}}{|N_{MinPts}(p)|})} \quad \mathbf{3}$$

Where,

- $lrd_{MinPts}(p)$ is the local reachability density of *p*
- $lrd_{MinPts}(o)$ represents the local reachability density of *p*'s $MinPts$ -nearest neighbors

For a given dataset, the following five calculations are obtained in the order outlined below:

i. The distances between every two instances.
ii. The distances between the k^{th} nearest neighbors to *p*.
iii. All the *k*-nearest neighbors of *p.*
iv. The reachability density (lrd) of *p.*
v. The LOFs (anomalies) of *p.*

2.3.2 Isolation Forest

Isolation Forest (IF) used in this research refers to an unsupervised algorithm that can be used to determine the existence anomalies in a dataset [10]. The algorithm was designed to detect anomalies depending on their isolation, and not density or distance measures. Besides, the IF analyzes mobile time series data to identify change points and outliers. The algorithm uses normal data specific anomalies and few anomalies in each dataset as quantitative attributes detecting outliers.

The IF algorithm begins with data training that includes tree diagrams construction [10]. First, an *N*-dimension dataset leads to a random subsample that constructs a binary tree labelled as Algorithm 1. During the process, branching happens by selecting the dimension $\boldsymbol{x_i}$ randomly where with $\boldsymbol{i \in \{1, 2, \ldots, N\}}$. Then, the algorithm selects another random value *v* from the range of random values. The point is branched leftwards for a smaller value than *v* for the stated dimension. If it exceeds *v*, the point is branched rightwards in the tree. The tree is split twice on the current node using a similar procedure. A single data point is isolated or a specific depth limit is achieved by this recursive branching course. This is repeated to construct another random tree for another sub-sample. A large ensemble of trees is created to complete a process that is collectively termed as forest training. The algorithm runs a contestant data argument selected from the trees after the process moves onto the scoring step. An outlier score is given to each data point depending on the depth reached by each candidate data point on the tree, as illustrated in figure 5. A radial line signifies each tree in the model: red characterizes an outlier whereas the blue radial line represents a nominal point [10].

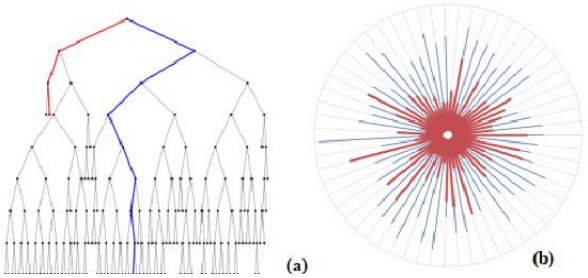

Figure 5: The schematic diagram of a particular tree (a), and the forest (b) [11]

Every occurrence *x* in the anomaly detection is assigned an outlier score *s* useful in analysis. The outlier score *s* and occurrence *x* can be formulated as:

$$\boldsymbol{s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}} \quad \mathbf{4}$$

Where, $E(h(x)$ represents the depth mean-value every datapoint *x* reaches in every tree and $c(n)$ represents the normalizing factor (mean depth for Binary Search Tree (BST) searches that are not successful).

$$\boldsymbol{c(n) = 2H(n-1) - (\frac{2(n-1)}{n})} \quad \mathbf{5}$$

Where $H\,(i)$ in this case represents the harmonic number $ln(i) + 0.5772156649$ (Euler's constant) and *n* is the total sum of change points used in building the trees [10].

Figure 2.9 (a) below represents the anomalous datapoint branching process where branching occurs until the point in question (red point) is isolated. Three random cuts were used to arrive at the desired isolation point. In Figure 6 (b), the branching process for a nominal is illustrated, where the branching process requires multiple cuts to identify the point and isolate since it sits deep in the initial dataset. The tree depth limit is achieved before the point is reached. The line numbers in the figure demonstrate the order of the branching process.

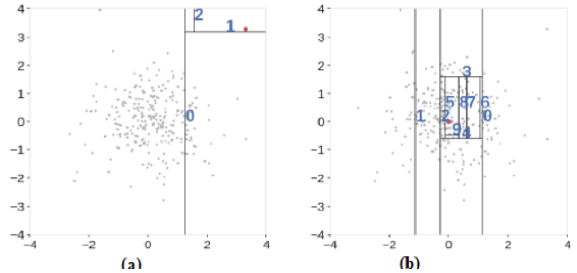

Figure 6: Branching process for an anomalous data (a) and a nominal point (b) [11]

The IF algorithm utilizes two input parameters: ψ (size of the sub-sample) and *t* representing the sum of trees. The parameter ψ determines the data size used for training. Three algorithm procedures are used to determine the anomaly score as: the $iForest(X, t, \psi)$ [10], the $iTree(X, e, l)$ [10], and the $PathLength(x, T, e)$ [10].

2.3.3 *Extended Isolation Forest*

The extended isolation forest (EIF) facilitates the improvement of outlier score consistency and reliability. The EIF identifies various slopes for making branching cuts and then randomly assigns intercept values within the training dataset. This EIF phenomenon is different from the usual random attribute-random value method used by Isolation Forest (IF) [11].

Figure 7 (a) below demonstrates the branching process of determining an outlier. As earlier stated, branching continues till the desired point is determined, and this process took three cuts to isolate the required point. The next figure 7 (a) shows how the branching process is used to arrive at the nominal point. The point is nearly at the center of the dataset, and therefore, several random cuts are required to isolate it. However, the depth limit is achieved before the isolation of the point for this scenario.

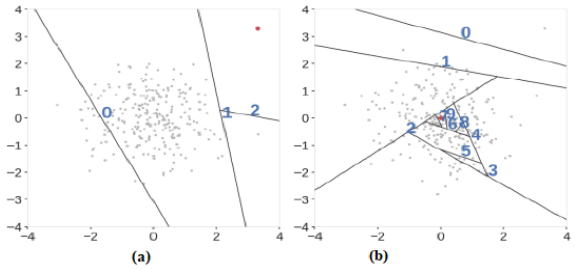

Figure 7: Branching in the EIF [11]

In the normal IF algorithm, two types of information are necessary for branch cut to be achieved: the coordinates, and the random value from the dataset. Conversely, the EIF branch cut needs two pieces of information: the random slope-intercept from the training dataset and the branch cut slope. Choosing a random slope from a branch cut in an *N*-dimension dataset is like selecting a normal vector $\vec{n}$ uniformly per unit of an *N*-sphere. This can be achieved through drawing random numbers for every *n*-coordinate from a normal distribution $\mathcal{N}(0,1)$ and uniform *N*-sphere selection points are reached as a result. The $\vec{p}$ intercept can be obtained from a uniform dataset used at each point of branching [11]. When the two types of information are received, the branching process for splitting data for a particular point *x* proceeds as follows:

$$(\vec{x} - \vec{p}) \cdot \vec{n} \leq 0 \qquad 6$$

The data point $\vec{x}$ is passed to the left branch of the process if condition is achieved. However, it is passed to the right branch if that condition is not fulfilled. The algorithm procedure of anomaly score determination resembles that of IF. However, the $iTree(X, e, l)$ [10] is advanced to become the $iTree(X, e, l)$ [11]

2.3.4 *Robust Random Cut Forest*

This anomaly detection algorithm on streaming data was proposed in 2016 [12]. Developing the machine learning model is done using current records in the stream. Neither older records nor statistics from previous executions are used by RRCF. The common procedure of anomaly detection using RRCF is as follows:

i. A bunch of random instances is taken by RRCF (Random).
ii. It then cuts them into the same number of instances and creates trees (Cut).
iii. Finally, all of the trees together are considered by determining whether a particular instance is an anomaly (Forest).

On a point set S, a robust random cut tree (RRCT) is generated as follows:

i. A random element relational to $\frac{l_i}{\sum_i l_j}$ where $l_i = max_{x \in S^{x_i}} - min_{x \in S^{x_i}}$ is chosen.
ii. Choose $X_i \sim Uniform[min_{x \in S} x_i,\ max_{x \in S} x_i]$
iii. Let $S1 = \{x | x \in S, x_i \leq X_i\}$, $S2 = S/S1$ and recurse on $S1$ and $S2$

Figure 8 shows how RRCF cut instance happens into pieces recursively. When each point is isolated, the cutting is stopped.

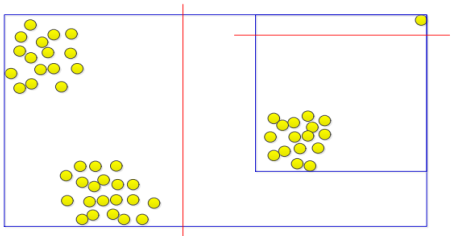

Figure 8: Random Cut Tree [12]

Deletion (*ForgetPoint Algorithm*) [12] and insertion (*ForgetPoint Algorithm*) [12] operations can be used to dynamically maintain robust random cut trees when anomalies on stream data are detected using RRCF. For deletion; if T were drawn from the distribution $RRCF(S)$ then the *ForgetPoint* algorithm produces a tree T' which is drawn at random from the probability distribution $RRCF\ (S - \{p\})$. On the other hand, for insertion; given T drawn from distribution $RRCF(S)$ and $p \in S$ produce a T' drawn from $RRCF\ (S \cup p)$,the *InsertPoint* algorithm is used.

3. METHODOLOGY

The adopted methodology is represented by the block diagram in Figure 9. In this section, the anomaly detection techniques of the LOF, IF, EIF and the RRCF were evaluated thoroughly. A subset of the dataset, with 447 records was extracted considering a region with graphically notable anomalies, and used as the ground truth. A section of the dataset between 25th November and 5th December 2020 (10 days) was used. It was manually examined and all the outlier instances identified and later analyzed using the algorithms as depicted in Figure 9. The findings for every parameter, with each algorithm are discussed in the results section.

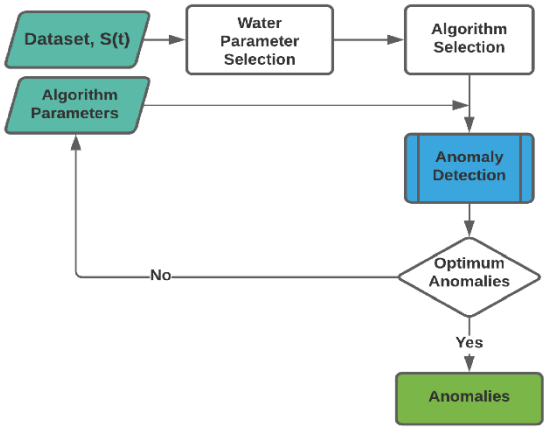

Figure 9: Anomaly Detection Algorithms Evaluation Process

4. RESULTS AND DISCUSSION

4.1 The Turbidity and pH Datasets

Table 1 below shows a subsection of the 2,658 records of both turbidity and pH data that were collected in the period of 60 days.

Table 1: Turbidity and pH Dataset

time	turbidity	pH
2020-11-04 11:00:31.822439+00:00	21.063435	7.34
2020-11-04 11:01:22.124333+00:00	20.868153	7.33
2020-11-04 11:01:51.663062+00:00	20.584553	7.32
2020-11-04 11:02:29.373718+00:00	21.185328	7.33
...	...	...
2021-01-04 08:23:37.035804+00:00	17.975997	7.35
2021-01-04 08:53:53.104009+00:00	17.734662	7.34
2021-01-04 09:24:09.578901+00:00	15.094176	7.36
2021-01-04 09:54:25.214766+00:00	14.611506	7.36
2658 rows × 3 columns		

A plot diagram for turbidity subset data in Figure 10 shows a number of contextual anomalies.

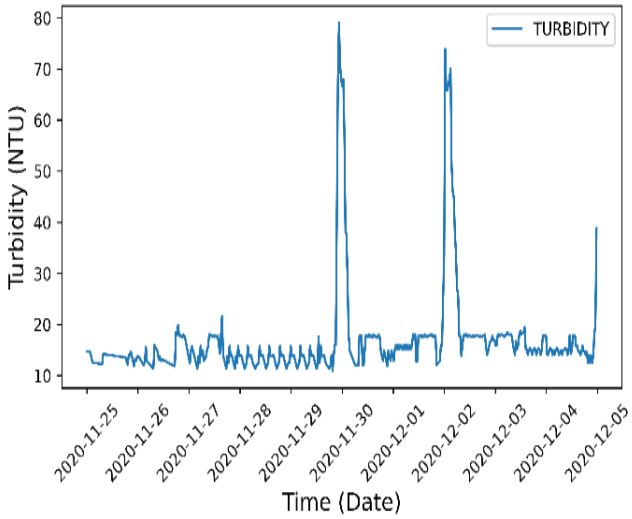

Figure 10: Turbidity subset data for the 10 days

Similarly, water pH data in Figure 11 below shows existence of contextual outliers.

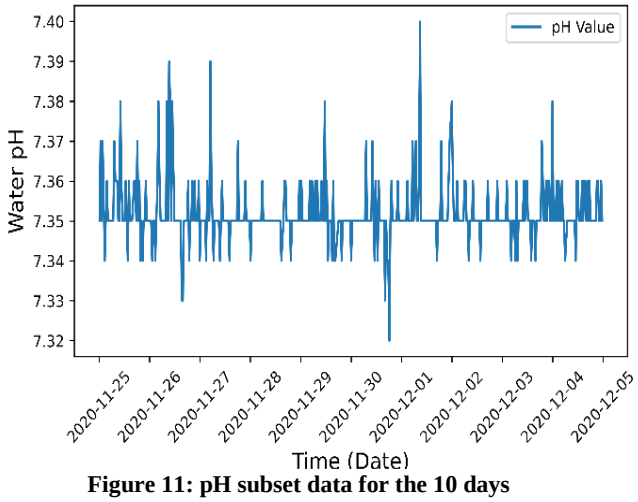

Figure 11: pH subset data for the 10 days

4.2 pH Subset

4.2.1 Local Outlier Factor

The LOF algorithm was used to detect the water pH anomalies shown in table 2 and plotted in figure 12, where the red stars diagram are the 131 point outliers with the number of neighbors, $k = 100$. The algorithm took 27 milliseconds to determine these anomalies. Choosing an optimal k was essential for detection performance. For a value too small or very large, the error went up due to under-fitting.

Table 2: pH outliers as detected by the LOF algorithm

time	pH
2020-11-25 00:36:05.490773+00:00	7.37
2020-11-25 01:36:37.639205+00:00	7.37
2020-11-25 02:37:09.730382+00:00	7.34
2020-11-25 03:37:41.850160+00:00	7.36
2020-11-25 07:09:34.270053+00:00	7.37
... ...	
2020-12-04 16:32:22.136635+00:00	7.36
2020-12-04 17:32:54.254236+00:00	7.36
2020-12-04 21:04:46.681849+00:00	7.36
2020-12-04 21:35:02.740644+00:00	7.36
2020-12-04 23:05:50.927469+00:00	7.36
131 rows × 2 columns	

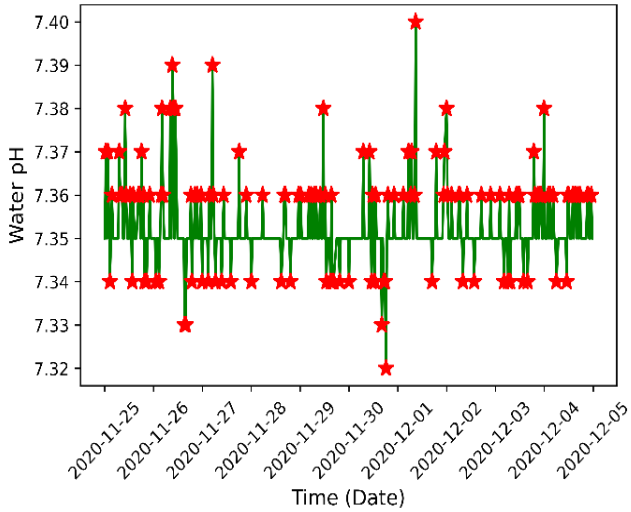

Figure 12: A plot of LOF pH outliers

4.2.2 Isolation Forest and Extended Isolation Forest

Both the IF and EIF demonstrated high precision and reliability when the ψ was considerably increased gradually to the desired value of $\psi = 200$. It was also observed that the number of trees t directly controlled the ensemble size. It was also found that the ideal paths converged at $t = 100$. Anomalies were assigned scores and over 300 points for turbidity data were marked as anomalies at 0.6. It was very difficult to find a feasible threshold for improvement. For instance, most of the anomalies were considered as inliers if 0.7 was set as the threshold score; but several of normal points were still considered as anomalies when 0.65 was set as threshold, and therefore determining top anomalies was impossible. However, a plot of top 50 instances based on the score in (Figure 13), which shows that the IF worked better than the EIF for turbidity data and found more anomalies with less false anomalies. This process took 3.66 seconds for IF algorithm and 3.99 seconds for the EIF algorithm.

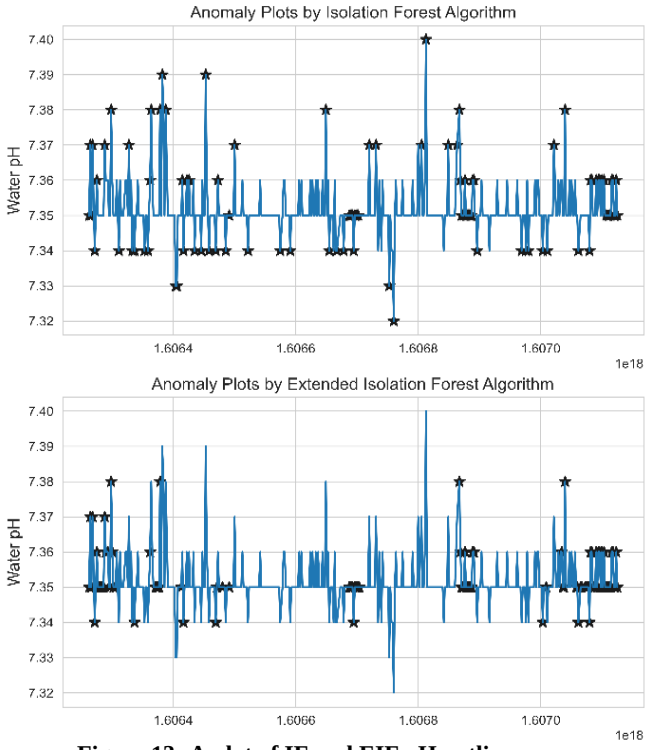

Figure 13: A plot of IF and EIF pH outliers

4.2.3 The Robust Random Cut Forest

Table 3 below shows the pH anomalies with their scores that were detected by the RRCF and were plotted as shown in Figure 14. The top 61 records having the highest outlier scores were identified since it was difficult finding a feasible threshold to split them. Some point anomalies seem to have inconsistent scores; such as the pH record 7.37 occurs twice in quick succession at [2020-11-25 00:36:05.490773+00:00] and at [2020-11-25 01:36:37.639205+00:00], only 1 hour apart, but they are assigned variant scores. This process took 11.2 seconds.

Table 3: pH outliers as detected by the RRCF algorithm with their scores

time	pH	Score
2020-11-25 00:36:05.490773+00:00	7.37	1.000
2020-11-25 01:36:37.639205+00:00	7.37	2.000
2020-11-25 02:37:09.730382+00:00	7.34	2.625
2020-11-25 03:37:41.850160+00:00	7.36	3.150
2020-11-25 07:09:34.270053+00:00	7.37	6.375
... ...		
2020-12-04 16:32:22.136635+00:00	7.36	2.325
2020-12-04 17:32:54.254236+00:00	7.36	1.925
2020-12-04 21:04:46.681849+00:00	7.36	1.625
2020-12-04 21:35:02.740644+00:00	7.36	1.625
2020-12-04 23:05:50.927469+00:00	7.36	1.625
135 rows × 3 columns		

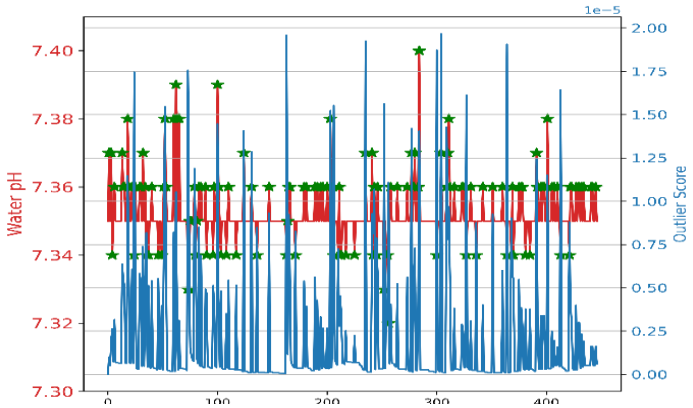

Figure 14: A plot of RRCF pH anomalies and their scores

4.3 Turbidity Subset

4.3.1 The Local Outlier Factor

The turbidity subset was subjected to the LOF algorithm and a total of 28 point outliers in table 4 were found and plotted as shown in Figure 15. The optimal value of number of neighbors $k = 80$. It took the LOF algorithm 52 milliseconds to complete this process.

Table 4: Turbidity outliers as detected by the LOF algorithm

time	turbidity
2020-11-26 18:51:43.794443+00:00	19.856159
2020-11-27 15:32:42.207766+00:00	21.606544
2020-11-29 22:01:36.359777+00:00	67.975997
2020-11-29 22:31:52.410155+00:00	78.975997
2020-11-29 23:02:08.469977+00:00	70.856159
....	....
2020-12-02 05:31:02.868901+00:00	33.975997
2020-12-02 06:01:18.928168+00:00	26.975997
2020-12-02 06:31:34.994928+00:00	25.975997
2020-12-04 23:05:50.927469+00:00	19.849407
2020-12-04 23:36:06.982840+00:00	38.768413
28 rows × 2 columns	

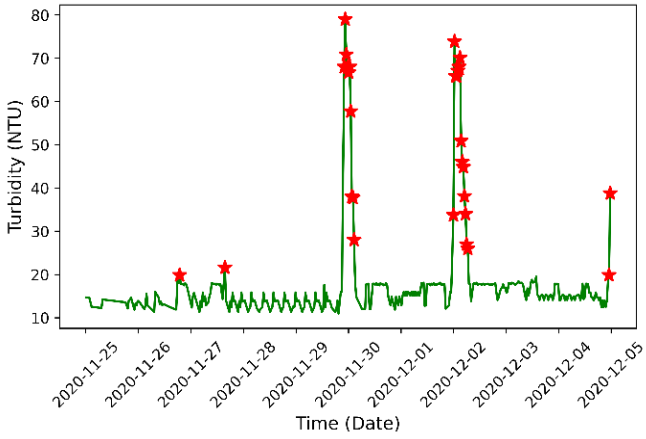

Figure 15: A plot of LOF turbidity outliers

4.3.2 The Extended Isolation Forest and the Extended Isolation Forest

A desired value of $\psi = 200$ was used with ideal paths converging at $t = 100$. Anomalies were assigned scores and over 200 points for pH subset data were processed as anomalies above 0.65 and in this case it was very difficult to find a feasible threshold for improvement and therefore

determining top anomalies was impossible. A plot of top 50 instances based on the score in (Figure 16), which shows that the EIF worked better than the IF and it found more anomalies with less false anomalies. This process took 4.48 seconds for IF algorithm and 5.08 seconds for the EIF algorithm.

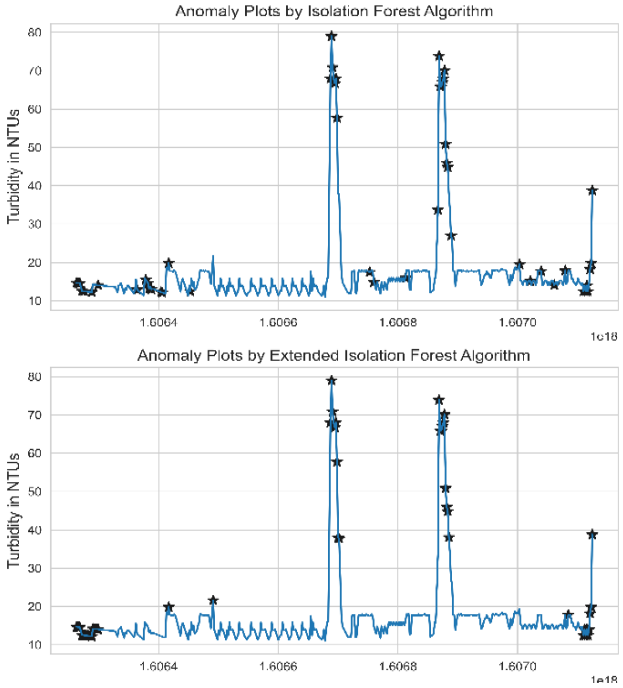

Figure 16: A plot of IF and EIF turbidity outliers

4.3.3 The Robust Random Cut Forest

A feasible threshold to split the outliers was difficult to find and therefore the top 25 outlier records having the highest scores were listed in table 5, taking 11.2 seconds. These results were plotted as shown in Figure 17. Similar to pH subset results, some point anomalies seem to have inconsistent scores; such as the turbidity records [2020-12-03 05:43:53.865925+00:00] 18.4387 and [2020-12-03 11:47:06.617958+00:00] 18.7684 are almost equal and occur in the same day but have scores highly variant from each other (51 and 44 respectively).

Table 5: Turbidity outliers and their scores as detected by the RRCF algorithm

time	turbidity	Score
2020-11-26 03:43:41.930323+00:00	15.6115	29.2252
2020-11-26 17:51:11.650345+00:00	18.5503	44.6488
2020-11-26 18:51:43.794443+00:00	19.8561	29.2479
2020-11-27 15:32:42.207766+00:00	21.6065	45.5557
2020-11-29 19:30:16.090586+00:00	10.9759	43.5028
…..	…	….
2020-12-02 21:39:36.851682+00:00	16.7346	34.9220
2020-12-03 05:43:53.865925+00:00	18.4387	51.3195
2020-12-03 11:47:06.617958+00:00	18.7684	44.9016
2020-12-03 13:48:10.853753+00:00	19.4795	57.0429
2020-12-04 23:05:50.927469+00:00	19.8494	46.4757
25 rows × 3 columns		

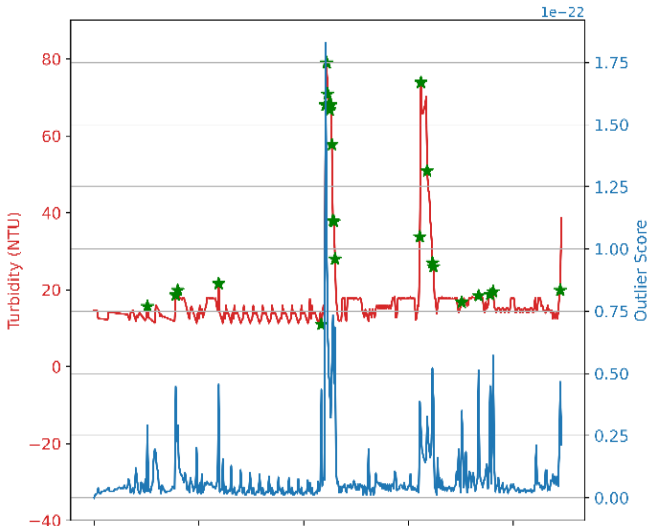

Figure 17: A plot of RRCF turbidity anomalies and their scores

From table 6 and table 7, the LOF algorithm successfully detects all the 63 anomalies in the time series water pH subset data as well all the 75 anomalies in the time series turbidity data. The RRCF algorithm suffers from 19 false anomalies as well as missing 27 outliers in the turbidity subset. The case is similar for 2 undetected point anomalies in the pH subset data. Additionally, the LOF algorithm is much faster than the RRCF algorithm on detecting anomalies for both turbidity and pH data. While the LOF algorithm only took milliseconds, the RRCF algorithm consumed a number of seconds, for both the subsets.

Table 6: pH Dataset Algorithms Performance Evaluation

Algorithm	Anomalies	False Anomalies	Undetected Anomalies	Execution Time
LOF	131	0	0	21 ms
IF	-	-	-	1.88s
EIF	-	-	-	1.25s
RRCF	135	4	0	2.51s

Table 7: Turbidity Dataset Algorithms Performance Evaluation

Algorithm	Anomalies	False Anomalies	Undetected Anomalies	Execution Time
LOF	28	0	0	38.9 ms
IF	-	-	-	3.66s
EIF	-	-	-	3.91s
RRCF	25	7	10	7.1s

5. CONCLUSION

This paper presents a comprehensive evaluation of four different machine learning anomaly detection algorithms on two parameters from a water sensor node at a water treatment plant raw water section. A subset data that was used in algorithm evaluation had 447 records for both parameters, extracted from the 2,658 that were collected over the deployment period. The LOF algorithm emerged superior to the IF, EIF and RRCF algorithms in contamination or anomaly event detection and hence a practical water contamination detection algorithm that can trigger alarms to alert the analyzers when contamination is detected. It is faster and involves less resources like memory and hence less computational resources as well. Moreover, the IF, the EIF and the RRCF algorithms exhibited inconsistent results

(scores) for some specific points in the dataset.

More water quality parameters can be subjected to the analysis done besides turbidity and water pH; these include: Total Dissolved Solids, Oxygen Reduction Potential, Temperature, Electrical Conductivity, Dissolved Oxygen, Free Residual Chlorine, Nitrates, to mention just but a few. Additionally, more anomaly detection algorithms can be assessed too to give a more robust and detailed report on anomaly detection algorithms.

6. ACKNOWLEDGMENTS

The first author would like to thank the Dedan Kimathi University of Technology for their sponsorship award towards his studies and research. He also acknowledges Dr. Ciira wa Maina, the Director of the Center for Data Science and Artificial Intelligence, Dedan Kimathi University of Technology and Dr. Henry Kiragu, Department of Electrical and Communication Engineering, Multimedia University of Kenya, for their valuable comments and supervision towards this research. Additionally, he is highly indebted to the Nyeri Water and Sanitation Company (NYEWASCO) water quality laboratory staff, more so Mr. Mohamed Yunis Ali (Head of the Lab) and Mr. Paul Michuki (Head of Operations) for their collaboration, advice and support towards system development, deployment and data collection.

7. REFERENCES

[1] S. Jyoti, Y. Priyanka, K. Ashok and M. PalVishal, "Water Pollutants: Origin and Status," Sensors in Water Pollutants Monitoring: Role of Material, pp. 5-20, 2020.

[2] I. Joshua and G. A. Adewale, "A comprehensive review of water quality monitoring and assessment in Nigeria," Chemosphere, vol. 260, p. 127569, 2020.

[3] B. Jamie and B. Richard, Water quality monitoring: a practical guide to the design and implementation of freshwater quality studies and monitoring programmes, 1996.

[4] F. E. Grubbs, "Procedures for Detecting Outlying Observations in Samples," Technometrics, vol. 11, no. 1, 2012.

[5] M. Nahshon, W. M. Ciira and K. Henry, "A Raw Water Quality Monitoring System using Wireless Sensor Networks," International Journal of Computer Applications, vol. 174, no. 21, pp. 35-42, 2021.

[6] L. Alexander and A. Subutai, "Evaluating Real-Time Anomaly Detection Algorithms -- The Numenta Anomaly Benchmark," in 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), Miami, 2015.

[7] Q. Elena, C. Francesco, G. Giulio Di and P. Riccardo, "Machine learning for anomaly detection and process phase classification to improve safety and maintenance activities," Journal of Manufacturing Systems, vol. 56, pp. 117-132, 2020.

[8] L. Jie, W. Peng, J. Dexun, N. Jun and Z. Weiyu, "An integrated data-driven framework for surface water quality anomaly detection and early warning," Journal of Cleaner Production, vol. 251, 2020.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander, "LOF: Identifying Density-Based Local Outliers," Association for Computing Machinery, p. 93–104, 2000.

[10] T. L. Fei, M. T. Kai and Z. Zhi-Hua, "Isolation-based Anomaly Detection," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 6, pp. 1-39, 2012.

[11] H. Sahand, C. K. Matias and R. Brunner, "Extended Isolation Forest," IEEE Transactions on Knowledge and Data Engineering, pp. 1 - 1, 2019.

[12] S. Guha, N. Mishra, G. Roy and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," International conference on machine learning, pp. 2712-2721, 2016.