**Detection of Anomalies in a Time Series Data using InfluxDB and Python**

Anih John. T1, Bede Chika Amadi2, and Festus Chima Umeokpala3

Institute für Data Science, Engineering, and Analytics.

Technical University of Cologne (TH köln)

February 2020

**Abstract:** Analysis of water and environmental data is an important aspect of many intelligent water and environmental system applications where inference from such analysis plays a significant role in decision making. Quite often these data that are collected through sensible sensors can be anomalous due to different reasons such as systems breakdown, malfunctioning of sensor detectors, and more. Regardless of their root cause, such data severely affect the results of the subsequent analysis. This paper demonstrates data cleaning and preparation for time-series data and further proposes cost-sensitive machine learning algorithms as a solution to detect anomalous data points in a time-series data. The following models: Logistic Regression, Random Forest, Support Vector Machines have been modified to support the cost-sensitive learning which penalizes misclassified samples thereby minimizing the total misclassification cost. Our results showed that Random Forest outperformed the rest of the models at predicting the positive class. Other methods like data oversampling seem to provide little or no improvement to the Random Forest model. Interestingly, with recursive feature elimination we achieved a better model performance thereby reducing the model complexity. Finally, with Influxdb and Kapacitor the data was ingested and streamed to generate new data points to further evaluate the model performance on unseen data, this will allow for early recognition of undesirable changes in the drinking water quality and will enable the water supply companies to rectify on a timely basis whatever undesirable changes abound.

**Keywords:** Imbalanced Classification, Time-series, Anomaly detection, Cost-sensitive Learning, Over-sampling, Machine learning

1. **Introduction**

With the level of technological advancement in the world today, increasingly, wireless sensors and actuators have become predominant in different fields of life. They are largely used in numerous monitoring systems and devices applied in manufacturing and machinery, airplanes and aerospace, cars, health care, robotics, environmental monitoring, and many other aspects of our day-to-day activities. Most applications that make use of sensors rely strongly on their accuracy. However, this is difficult to ascertain [[1]](#book1).

In addition to this, many sensors are also deployed under potentially harsh weather condition, making their breakage even more likely [[1]](#ref1). When a sensor becomes faulty in any form, it starts generating erroneous readings which could eventually lead to data corruption during transmission, hence, it is crucial to incorporate a means of checking the correctness of the generated data so as to smoothen out any irregularity therein. These irregularities could be referred to as an anomaly.

Anomaly detection can simply be referred to as the task of identifying the observations that do not conform to an expected pattern within a dataset [[2]](#ref2). Typically, these anomalous observations possess the ability of getting translated into problems such as structural defects, errors, or frauds, depending on the field of context from which they are being analysed. The deployment of effective machine learning models on a system for the purpose of anomaly detection plays an essential role in enhancing the quality of such systems, hence saving it from further damage due to the abnormality.

This research paper presents the application of Machine learning in a time series data. Since time series data is a data that changes over time, it is important we treat this differently from the conventional machine learning problems. Machine learning is all about finding patterns in data; since time series data change over time, it turns out to be a useful pattern to utilize. Time series data consists of at least two things: first, an array of numbers that represents the data itself, and second, another array that contains a timestamp for each datapoints.

This paper is structured as follows:

* **Materials and methods:** This section presents the business description, the problem we are trying to solve using Machine learning., data preparation methods for time-series data and approach to the problem.
* **Results:** This section presents performance evaluation of the models and further methods to improve the best performing model’s performance.
* **Discussion and Conclusion:** This section summaries this paper, limitations and proposes ideas for future work in this area.

1. **Materials and methods**

This section will introduce the Business, and the problem as well as several techniques to prepare the data for time series analysis and machine learning.

* 1. **Business Understanding and Problem**

The earth’s surface is made up water to a toll of 71% approximate proportion and this is very crucial for all forms of life [[3]](#ref3). Owing to the essential nature of water for sustainable living, water requires preservation knowing it is highly sensible to all kinds of contaminations. The provision of clean and safe drinking water is an essential task for water supply companies around the world [[3]](#ref3).

To handle this situation, highly effective sensors monitor relevant water bodies and environmental data at several measuring points, daily. The monitored data can be analysed with aims to identify any kinds of anomalies. This will allow for early recognition of undesirable changes in the drinking water quality and will enable the water supply companies to rectify on a timely basis whatever undesirable changes abound [[3]](#ref3).

* 1. **Data Understanding**

The data that was used for this research has been measured at different stations near the outflow of a designated waterworks. The data contains time series denoting water quality data. Table 1 gives an overview of the data provided [[3]](#ref3) . The data contains 122,334 observations and 10 features, where *EVENT* would be used as the target variable. Inspecting the target variable, we see that it consists of approximately 98.5% as normal data points or majority class and approximately 1.42% as anomalies or minority class, clearly, this is a case of an *Imbalance Classification problem*.

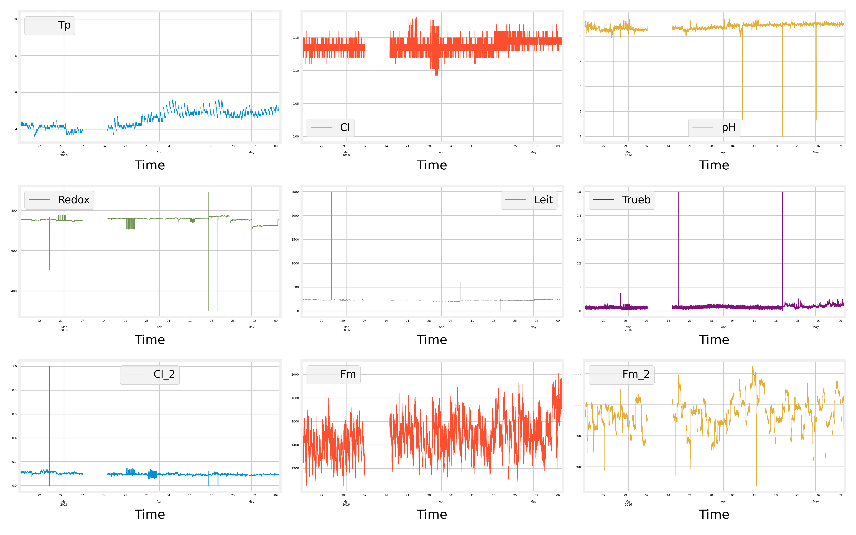
**Table 1.** Description of the time series data



* 1. **Data Cleaning**

Data can be vast and thus always messy; it requires fine tuning and cleaning before fitting it to the models. In timeseries, a messy data is often a product of failing sensors or human errors in logging the data, database failure and more.

The (Fig. 1). below show the values of various features of the dataset from 15-02-2016 to 10-05-2016. There seems to be missing values between the periods 07-03-2016 and 17-04-2016, as well as some period where the data does not fluctuate at all.



**Fig 1.** Data Visualization of the time series data

Sadly, most predictive modeling techniques cannot handle any missing values. Thus, the problem of missing values must be addressed prior to data modeling. To tackle this problem, we have handled the missing values using zero interpolation method.

* 1. **Data Preparation**

For a time-series model, its trend is a very important feature. A positive trend is a line that generally slopes up - the values increases with time, similarly; where the value decreases, it is termed a negative trend. Another important concept in a time series is white noise: this is a series of measurements where each value is uncorrelated with previous values. To effectively model a time series, it must be stationary- meaning that the distribution of the data does not change with time.

For a time-series to be stationary, it must fulfill three criteria:

1. The series has zero trend, it is not growing or shrinking
2. The variance is constant.
3. The autocorrelation is constant.

The most common test for identifying whether a time series is non-stationary is the *augmented Dicky-Fuller test*. This is a statistical test where the null hypothesis is that the time series is non-stationary due to trend. Using the augmented Dicky Fuller test shows that our data is stationary but visualizing the data proves otherwise, taking a difference of the data using equation (1) shows a more stationary plot of our data points in (Fig. 2). The data is now roughly centered at zero and the periods of high and low changes are easier to spot.

Difference: (1)

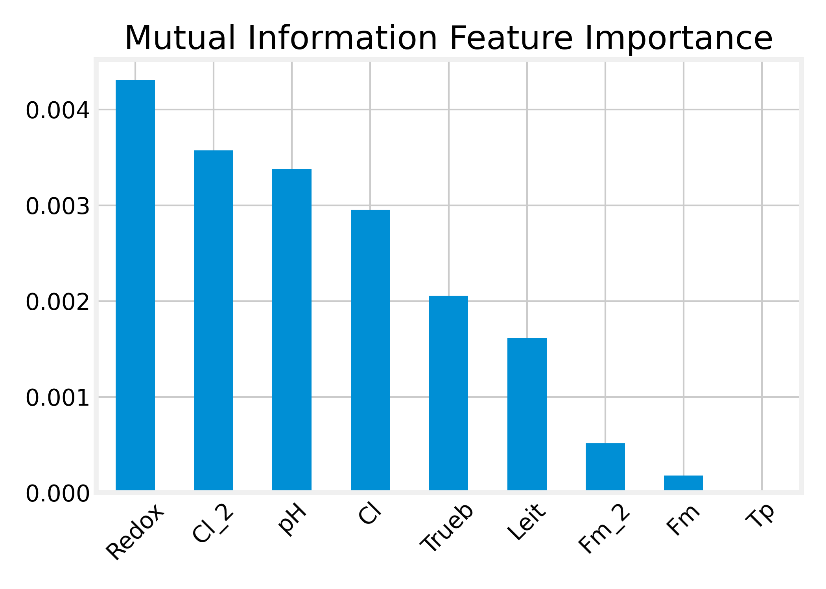
**A picture containing clock

Description automatically generated**

**Fig 2.** Data visualization of the stationary time series data

Now let us explore what input features are more likely to predict our target variable. Since our input variables are numerical, and our target variable is a categorical class, we leveraged on the mutual information statistics. Mutual information is a field of information theory that applies information gain that are typically used in the construction of decision trees to feature selection. It is calculated between two variables and measures the reduction in uncertainty for one variable given a known value of the other variable [[5]](#ref5).

The (Fig 3.) below shows the feature importance using mutual information statistics. We see the most important feature in predicting our target is “*Redox*” and the least important feature is “*Tp*”. Since there are different techniques for scoring features based on scores, which one do we rely on? In section 3, we have evaluated different models using different feature selection techniques and chosen the method that results in a model with the best performance.



**Fig 2.** Bar Chart of the Input Features using Mutual Information Feature Importance

* 1. **Proposed Machine Learning Algorithms**

Most machine learning algorithms assume that all misclassification errors made by a model are the same [[4]](#ref4). Unfortunately, this is not the case for an imbalanced classification problem. In this case, missing a positive or minority class case is worse than incorrectly classifying an example from the negative or majority class [[4]](#ref4).

Cost-sensitive learning is a branch of machine learning that takes the costs of prediction errors into account when training a machine learning model [[4]](#ref4), since it is more expensive to misclassify an actual positive or minority class into negative, than an actual negative example into positive. In this problem instance, our majority class or Negative or majority class are the normal data points which values tagged as “False” in the dataset and the minority class or positive class are the anomalies which values tagged as “True” in the data. Cost-sensitive learning for an imbalanced classification is focused on assigning different costs to the types of misclassification errors that can be made. The varying misclassification costs can be best understood by using a cost matrix [[4]](#ref4). A confusion matrix is a summary of predictions made by a model on a classification problem, whereas a cost matrix is a matrix that assigns a cost to each cell in the confusion matrix [[4]](#ref4) .

**Table 2.** Cost Matrix for Binary Classification task

|  |  |  |
| --- | --- | --- |
|  | **Actual Negative** | **Actual Positive** |
| **Predicted Negative** | C(0,0) True Negative | C(0,1) False Negative |
| **Predicted Positive** | C(1,0) False Positive | C(1,1) True positive |

Equation (2) below defines the total cost of a classifier as the cost-weighted sum of the False Negative and False Positives [[4]](#ref4).

Total Cost = C(0,1) FalseNegatives + C(1,0) FalsePositives (2)

Machine learning algorithms are not often developed specifically for cost-sensitive learning. Instead, the wealth of existing machine learning algorithms can be modified to make use of the cost matrix [[4]](#ref4). The following machine learning algorithms have been used for the cost-sensitive learning: Logistic Regression, Support Vector Machines, and Random Forest.

* 1. **Evaluation Metrics**

Classification accuracy, which is the most widely used will be an inappropriate evaluation metric for the problem we are trying to because the distribution of the target variable is not equal, and this might be misleading. The following metrics have been used for evaluation: Sensitivity-specificity metrics, F0.5 score and F1 Score. Sensitivity refers to the true positive rate and summarizes how well the positive class or the minority class were predicted [[4]](#ref4). Specificity is the complement to sensitivity, or the true negative rate, and summarizes how well the negative class or the majority class were predicted [[4]](#ref4). Since we are dealing with an imbalanced classification task, the sensitivity would be important than the specificity. The F-measure is a popular metric for imbalanced classification. The Fbeta-measure is an abstraction of the F-measure where the balance of precision and recall in the calculation of the harmonic mean is controlled by a coefficient called beta [[4]](#ref4). If the False positives are more costly, the F0.5 score would be preferred. If both False positives and False negatives are equally costly, the F1 score would be preferred.

**Sensitivity-specificity Metrics**

Sensitivity = (3) Specificity = (4)

**F-measure Metrics**

F-measure = (7) Fbeta-measure = (8)

1. **Results** 
   1. **Data Modelling and Evaluation**

**3.1.1 Evaluate Cost-Sensitive Algorithms**

The following models: Logistic Regression, Support Vector Machines and Random Forest which support Cost-Sensitive Learning have been evaluated individually using *Repeated Stratified k-fold Cross Validation*. This procedure provides a good general estimate of model performance that is not too optimistically biased compared to a single train-test split [[4]](#ref4). We have used k=10, meaning each fold will contain about or about 12,233 examples. Stratified implies that each fold will contain the same mixture of examples by class, that is approximately 99 percent to 1 percent of normal to anomaly examples. Repetition indicates that the evaluation process will be performed multiple times to help avoid fluke results and better capture the variance of the chosen model. In this, we have used three repeats. That means a single model will be fitted and evaluated 10 x 3 (30) times, the mean and standard deviation of models was used. The results have been reported in Fig 3 and Table 3. We see that the Random Forest model with 1000 Decision Trees outperformed the rest of the models with an F1 Score of approximately 86% and an F0.5 score of approximately 92%. Now that we have a good first sets of results let us see if we can improve the model with data oversampling methods. Oversampling methods duplicate examples in the minority class or synthesize new examples from the examples in the minority class [[4]](#ref4). The following oversampling methods have been used with random forest to further improve the model performance: Random Oversampling (ROS), Synthetic Minority Oversampling Technique (SMOTE), BorderLine SMOTE (BLSMOTE), SVM SMOTE, and Adaptive Synthetic Sampling (ADASYN).

A picture containing star

Description automatically generated

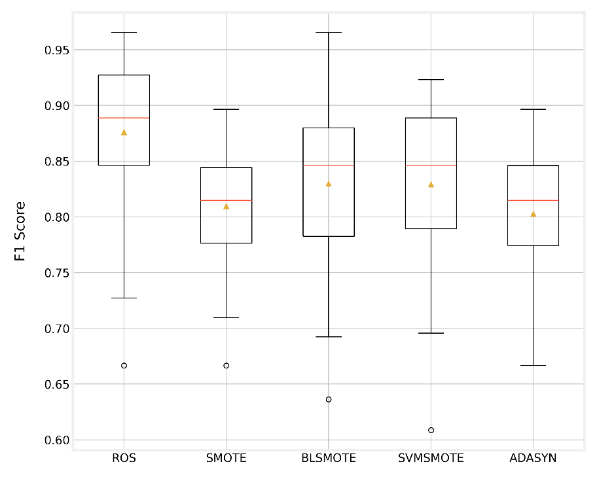
**Fig 3.** Box and Whisker Plots of Cost-Sensitive Machine Learning Models on the Imbalanced data

**Table 3.** Results from the Cost-Sensitive Machine Learning Models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | Sensitivity | Specificity | F1-Score | F0.5 Score |
| Random Forest (RF) | 78.6% | 100% | 86% | 92% |
| Logistic Regression (LR) | 49% | 97% | 4% | 2% |
| Support Vector Machine (SVM) | 31% | 56% | 0.2% | 0.1% |

**3.1.2 Evaluate Data Oversampling Algorithms**

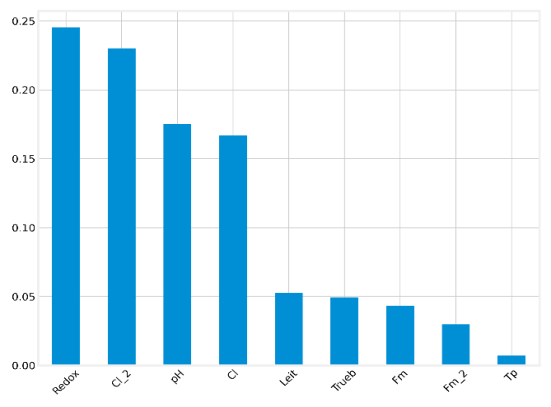
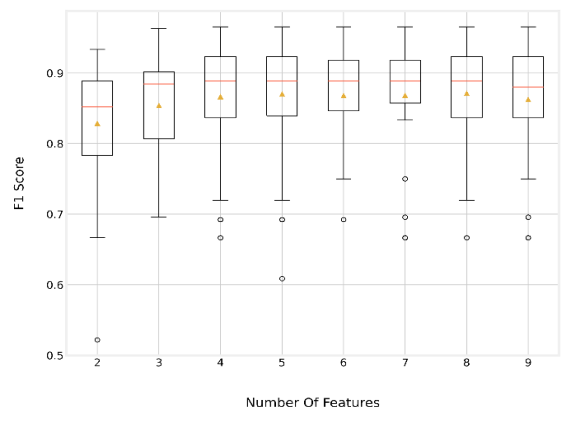
Data sampling provides a way to better prepare the imbalanced training dataset prior to fitting a model. The easiest oversampling technique is to duplicate examples in the minority class, called random oversampling [[4]](#ref4). Each technique will be tested with the best performing algorithm from the previous section, specifically Random Forest. Fig 4 shows the results of the F1-scores. We see a little improvement of the F1 score with Random sampling of the data from 86% to 87.6% and an F0.5 score of 91.5% which is approximately the same with the baseline. Now let us see how we could improve the model further.



**Fig 4.** Box and Whisker Plot of Random Forest models With Data Oversampling on the Imbalanced Data.

**3.1.3 Recursive Feature Elimination**

Recursive Feature Elimination (RFE) is a popular feature selection algorithm [[5]](#ref5). RFE is very effective at selecting features in the training dataset that are most relevant in predicting the target variable [in this the anomalies]. In section 2.4, we discussed briefly about mutual information statistics and the problem of selecting appropriate scoring features. Using the best performing model, with Recursive Feature Elimination RFE to get rid of features that do not increase the model performance, by so doing we increase the model’s F1 score and reduce the model complexity. Fig 5 (a) shows that the box plot with the mean samples of the F1 Scores with the number of features selected. We see that the F1 Score decreased after 8 features + i.e. when *“Tp”* was added. Fig 5 (b) shows the feature importance in descending order. We can therefore conclude that the feature “*Tp*” does not constitute any information towards our target variable so therefore at this point, we exclude “*TP*” from our input feature to reduce the model complexity and improve the model performance.



(a)

(b)

**Fig 5.** BoxPlot of RFE Number of Selected Features vs. F1 Score.

* 1. **Data Ingestion and Streaming with InfluxDB and Kapacitor**

InfluxDB is an open source time series database that is part of the TICK (Telegraf, InfluxDB, Chronograf, Kapacitor) stack [[6]](#ref6). Like every other database system, it was designed to write high quality queries to query data and store data. Kapicitor which is a part of TICK, was designed to stream data points in real time. Kapacitor utilizes a Domain Specific Language (DSL) named TICKscript to define tasks involving the extraction, transformation and loading of data (ETL), likewise, the tracking of arbitrary changes and the recognition of events within data [[7]](#ref6). The steps followed to achieve this are discussed as follows:

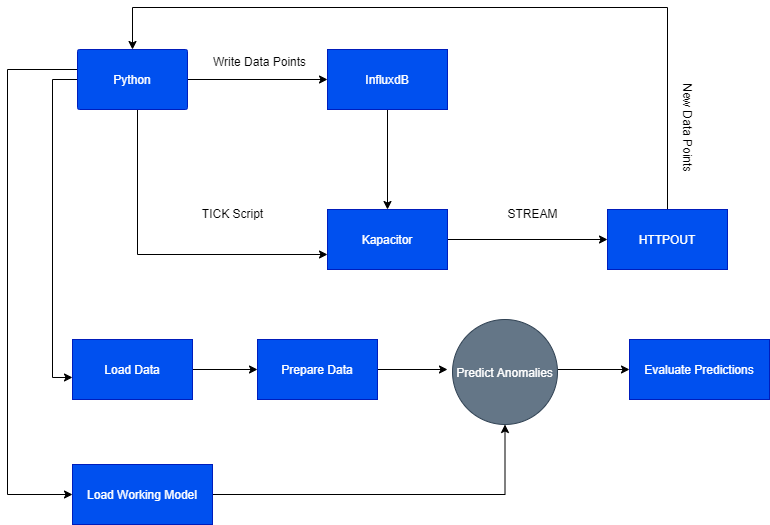
First, we first define the alerts. TICKscript is used in .tick files to define ETL pipelines for processing data [[7]](#ref6). The TICKscript language is intended to chain together the invocation of data processing operations defined in nodes [[7]](#ref6). In the TICK script, we specify a period of 5 days and 2 hours for the generation of new data points i.e. every 2 hours, 7200 data points are generated. Additionally, the TICK script contains two edges which are:

* stream→from()– which defines the processing type of the task and the data stream [[7]](#ref6).
* from()→httpOut()– which passes the data stream to the HTTP output processing node [[7]](#ref6).

Secondly, with python, we write the raw data to influxdb and enable the TICK script with a type stream to stream the data in real time to generate new data points.

Finally, with the httpout specified in our TICK Script, we retrieve the new data points. The new data are streamed to the HTTPOUT specified in the TICK script and its available in a JSON structure. With python, we read the data from the HTTPOUT and load the data with pandas json method, and also load our working model and make predictions on the new data points to capture outliers in the data points.

The Fig. 6 shows the overall picture of the process described.



**Fig 6.** Flow diagram of the data and working process

1. **Discussions and conclusion**

This paper proposed cost-sensitive machine models as a solution to an imbalanced time-series classification task to detect anomalies from data generated through sensors.

First, we started by cleaning the data and preparing the data for machine learning, since we were dealing with a time-series data, we ensured that the data was stationary to avoid long-term drift.

Moreover, we evaluated and compared the performance of our cost sensitive machine learning models using a boxplot that shows the mean of the repeated stratified k-fold cross validation approach. Our results demonstrated that Random Forest outperformed the rest of the algorithms. However, we investigated other methods like data oversampling to see if we could further improve the model performance. We notified that the methods did not make any significant increase in the performance or our models and it was computationally expensive.

The goal every machine learning model is to build a model that is not prone to overfitting and its able to generate well to unseen dataset and less complex, hence we explored various feature selection algorithms to investigate what features actually contributes towards the prediction of our target variable, it was noted that the variable “*Tp*” does not provide any information towards our target variable, hence we excluded the variable from our input feature by so doing we increased the model performance and reduced the model complexity.

The following evaluation metrics have been used for the model evaluation: Sensitivity, Specificity, F1 score and F0.5 score. We came to the following conclusions:

* If the False positives are more costly, the F0.5 score should be used for evaluation.
* If both False positives and False negatives are equally costly, the F1 score should be used for the evaluation.
* If predicting the normal data points is very important, the specificity score would be an interesting metric to look out for, similarly if the anomalous points are very costly, the sensitivity score would be an interesting metric to watch out for.

We concluded this paper by performing data ingestion and streaming with influxdb and Kapacitor via a TICK script to specify the stream interval of 5 days every 2 hours and then load the streamed data points from the HTTPOUT with python and load the working model to predict outliers. Overall, our model proved effectively at predicting outliers in the sensor data.

1. **Future Works**

This paper has captured the detection of anomalies in a time series data using cost-sensitive machine learning algorithms by approaching the problem like a supervised machine learning task. While we seem to have an effective model that does well at predicting outliers in the sensors data, there are still a lot more that could be done to further improve the model performance.

Hyperparameters are parameters we set before the model training. Performing hyperparameter tuning could further improve model performance. For this project, we built a random forest of 1000 trees. Since Random Forest is a combination of decision trees to make decisions, what if we could increase the number of trees. There are also several hyperparameter tuning algorithms that could be applied to improve the model like Bayesian optimization, Grid Search, Random Search, Coarse to Fine, since this is computationally expensive, this was one of our limitations to this project.

This paper treated this task like a supervised machine learning task; however, this problem could be treated as an unsupervised machine learning task and outlier-based machine learning algorithms could be explored. Some of the outlier-based machine learning algorithms are Isolation Forest, Local outlier factor, one class support vector machines and more.

Since data was ingested and streamed via Influxdb and Kapcitor, another thing that would be improved could be data monitoring to avoid data drift in the data at a later time.

Finally, Influxdb provides a platform for outlier detection. Interfacing this technology with the proposed machine learning algorithm and have the detection automated and happening in real time would be interesting.

**References**

1. F. Giannoni, M. Mancini, and F. Marinelli. (2018). *Anomaly detection models for IoT time series data*. Data Mining - Project Report. Accessed at https://www.researchgate.net/publication/329387705\_Anomaly\_Detection\_Models\_for\_IoT\_Time\_Series\_Data
2. Naveen Joshi, *Machine learning for anomaly detection*, *Allerin*, 01 May 2017, https://www.allerin.com/blog/machine-learning-for-anomaly-detection
3. Sowmya Chandrasekaran, Martina Friese, Jörg Stork, Margarita Rebolledo, Thomas Bartz-Beielstein. *GECCO 2017 Industrial Challenge* *Monitoring of drinking-water quality*, April 10, 2017
4. Jason Brownlee. (2020). *Imbalanced Classification with Python: Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning*
5. Jason Brownlee. (2020). *D⁠a⁠t⁠a⁠ ⁠P⁠r⁠e⁠p⁠a⁠r⁠a⁠t⁠i⁠on⁠ ⁠f⁠o⁠r⁠ ⁠ ⁠M⁠a⁠c⁠h⁠i⁠n⁠e⁠ ⁠L⁠e⁠a⁠r⁠n⁠i⁠n⁠g: Data Cleaning, Feature Selection, and Data Transforms in Python*
6. Influxdata (2020). InfluxDB Retrieved from <https://www.influxdata.com/products/influxdb-overview/>
7. Influxdata (2020). Influxdata documentation Retrieved from <https://docs.influxdata.com/kapacitor/v1.5/tick/introduction/>