EC2 CPU Utilization Outlier Detection Experiment

Research carried out by Liam Reid as part of Final Year Project and Dissertation: 'Proposing new methods of detecting outliers in cloud resource data'.

Abstract

This experiment tests the effectiveness of a newly implemented technique of detecting outliers. This technique involves detecting outliers using an ensemble of 'weak' classifiers that work together and vote on whether a datapoint is an outlier or an inlier. The experiment is performed on Cloud platform CPU usage. Techniques are evaluated using accuracy, recall, precision and f1. This experiment determines which voting system works best out of two implemented. Results show that the ensemble detection method implemented can detect outliers in the CPU usage data. It sometimes produces good scores but does not perform well against unstable data. This experiment determines that a 'Combined Confidence' voting system produces the best scores.

Introduction

Background Information

This document contains details of experiments carried out on cloud CPU utilization data and detecting outliers using newly implemented ensemble of outlier detection techniques.

The data being analyzed is Amazon Elastic Compute Cloud (EC2) CPU usage. EC2 is a service provided by Amazon used for on-demand cloud infrastructure. Customers use the platform for its compute power, running multiple different kinds of operating systems for various applications [1].

Amazon Web Services (AWS) provides many tools for analyzing the metrics of an EC2 instance. CPU Utilization is arguably the most important metric, since it "identifies the processing power required run an application on a selected instance [2]". Problems with an EC2 instance, or an application running on one, can often be identified by a discrepancy in CPU usage [3].

This CPU utilization data is labelled, meaning the outlier detection technique applied to the data can be evaluated based on accuracy, precision, recall and f1 score [4].

$$F_1 = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
$$= \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

Equation: Calculating F1 score [5].

How is the experiment carried out?

This experiment is carried using a webapp developed using dash. The purpose of this application is to apply outlier detection algorithms to datasets defined by the user. The application plots the data and generates scores for the chosen method. Accuracy, precision, recall and an f1 score will be calculated and displayed to the user. The time taken to analyse the dataset can also be extracted from this application.

The outlier detection methods in the ensemble are implemented using python. These methods work individually first to make a prediction with a confidence score. A voting system, also implemented using python, determines the final classification.

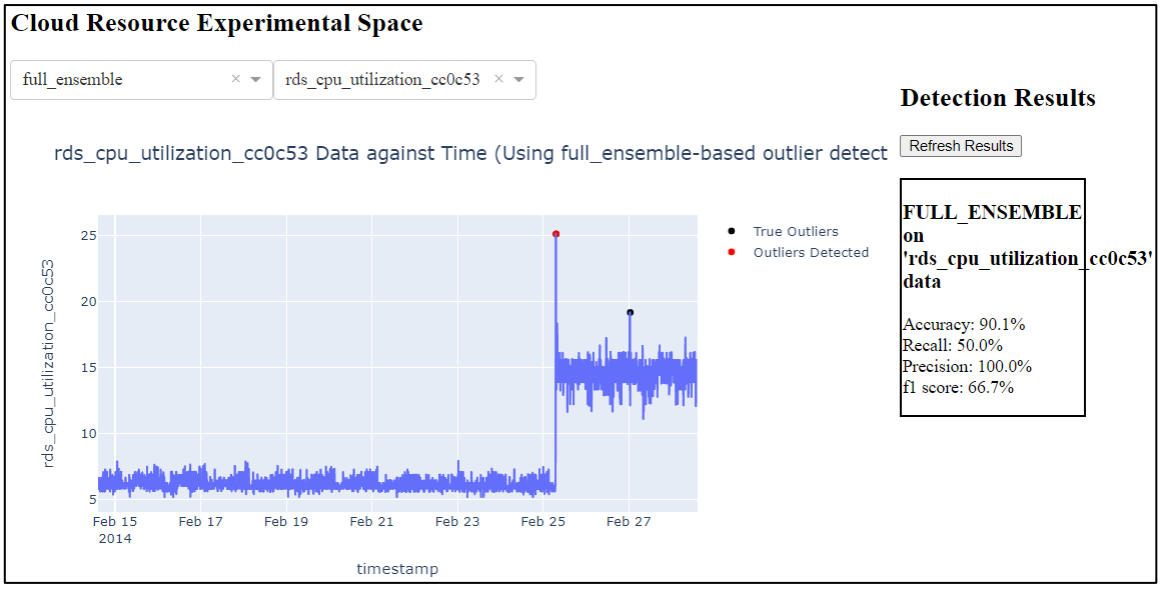


Fig. 1 Screenshot showing application used to apply outlier detection.

Hypothesis

The implemented outlier detection method (ensemble) is an effective outlier detector for labelled datasets and will generate good scores for accuracy, precision, recall and f1.

The proposed scoring method for the ensemble of detectors will generate better scores and the previously implemented solution (Combined confidence will outperform majority classification).

Methods

The combined predictions of an ensemble of weak classifiers are used as the method to detect outliers in this experiment. These detectors generate a prediction for a piece of data (outlier or inlier), the combined predictions are combined to produce a final classification. Two methods of voting are used to produce the final classification to see which method produces the better score.

A quick observation shows that the time-of-day affects the data. I.e., CPU usage is higher during typical office hours compared with night-time. The idea of concept drift must be considered since the CPU usage can vary throughout the day but also from day to day (due to factors such as new applications being deployed to these EC2 instances) [6].

There are 4 classification techniques in the Ensemble.

- **Moving Average**
  This technique uses the average of the previous data points in the time series to classify the next. After the average is calculated, the standard deviation of the previous datapoints is calculated. The standard deviation is used as a threshold, if they next datapoint is less than or greater than the average calculated +/- the threshold then the datapoint is classified as an outlier [7]. The graph shows this technique in practice, the red lines represent the boundaries, and the red dots are the outliers detected.

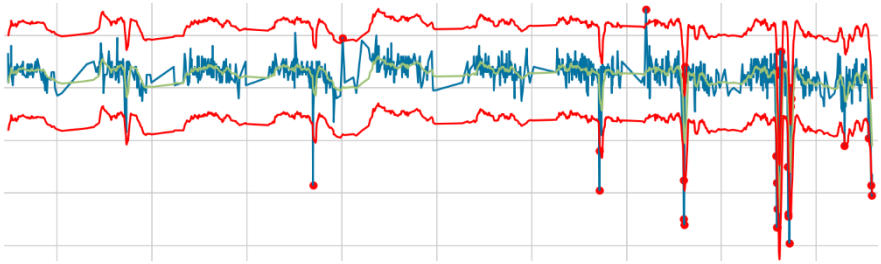


Fig. 2 Moving Average outlier detection showing boundaries

- **Moving Median**
  This technique follows the same steps as the previous except a median is calculated instead of the average [8]. The graph below shows this technique in practice, observe how the boundaries are like moving average but different outliers have been detected.

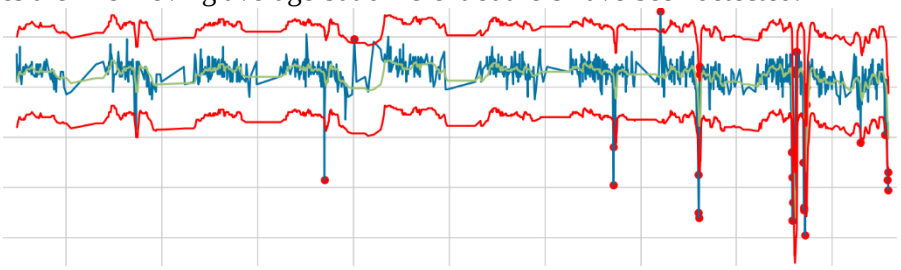


Fig. 3 Moving Median outlier detection showing boundaries

- **Moving Boxplot**
  This technique takes several of the previous data points and generates a boxplot. The interquartile range is combined with the upper and lower quartiles to produce a threshold (1.5 * the inter-quartile range). If the next datapoint is outside the threshold, then the datapoint is classified as an outlier [9].

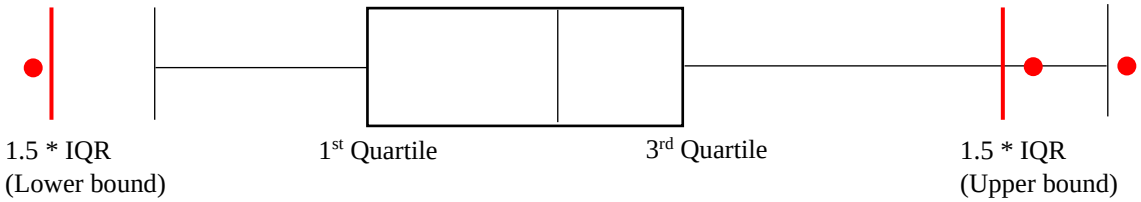


Fig. 4 Boxplot Outlier Detection Example

- **Histogram**
  This technique plots histograms of subsets of the data. If a range in the histogram has a height less than a defined threshold, then the range is said to contain outliers [10]. If a range has a height below the threshold, but the ranges beside it have a height higher than the threshold then it is considered a borderline inlier.

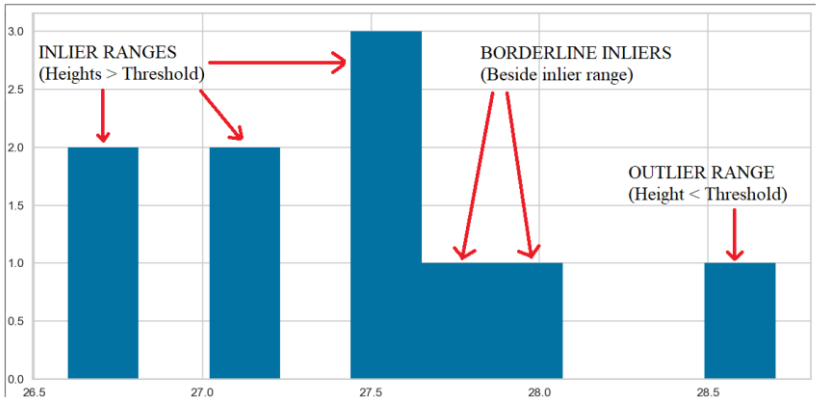


Fig. 5 Histogram Based Outlier Detection

Voting on a Final Classification

Voting Method 1 – Combined Confidence

These techniques run individually first, generating a prediction and a 'confidence' score. The confidence scores are combined to generate a final prediction. The formula behind this voting mechanism is described below.

Confidence is calculated by the distance between a datapoint and the threshold.

$$Confidence = max\left[\frac{datapoint\ distance\ from\ threshold}{threshold}, \quad 1\right]$$

$$Outlier\ if: \left(\left(\sum_{i=0}^{n} Detector_i(prediction) \cdot Detector_i(confidence)\right) < 0\right)$$

For predictions, -1(outlier) and 1(inlier), the above equation computes a minimum prediction of -n and a maximum of n. By visualising possible outputs on a spectrum, it can be said that an outlier score < 0 is likely to be an actual outlier.

Voting Method 2 – Majority Classification

A datapoint is labelled as an outlier a pre-defined number of detectors predict it to be, as described below.

$$Outlier\ if: \left(\sum_{i=0}^{n} Detector_i(prediction)\right) \geq t$$

where 'n' is number of outliers in the ensemble and 't' is the predefined number of detectors that must classify as an outlier.

Method of Scoring the Detector

The datasets being analyzed have been labelled by the Numenta Anomaly Benchmark (NAB). NAB is a platform for testing detection techniques on timeseries data. It contains 58 labelled datasets used for scoring algorithms [11]. Labelling data is expensive [12], but these datasets are available for free from NAB and are crucial to this experiment.

The software used to perform the experiments uses the labels provided by NAB to plot the outliers. The graph below shows some timeseries data with classifications. These colors are used to show how the true positive, false positive and false negative outcomes.

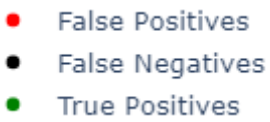


Fig. 6 Outlier detection classification key

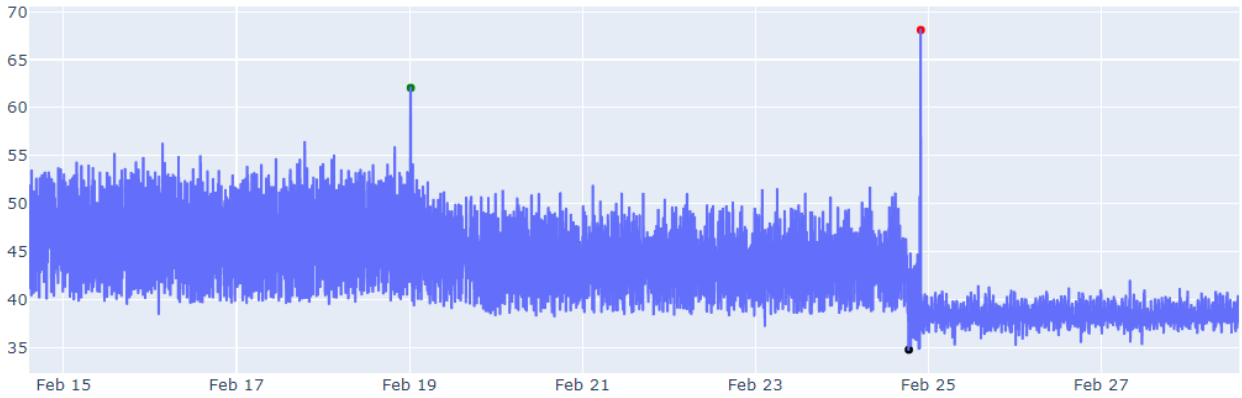


Fig. 7 Graph demonstrating how the application represents the detection data.

The software calculates four metrics to determine the overall performance of the detector, accuracy, precision, recall and f1 score. These are calculated using the following equations.

$$Accuracy = \frac{True\ Postives + True\ Negatives}{No.\ Datapoints} \qquad Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$Precision = \frac{True\ Positives}{True\ Positives + Flase\ Positives} \qquad F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Equations used to evaluate the detector against the dataset

Evaluating Results

Evaluating the Voting Methods

The ‘Combined Confidence’ voting system produces better scores than the ‘Majority Classification’ voting system concluding that ‘Combined Confidence’ is a better solution as hypothesized. Tables 1 and 2 in Appendix A provide detailed scores for each dataset, table 4 provides a side-by-side comparison. ‘Combined Confidence’ produces better scores for precision, recall and f1. Table 3 shows that the ‘Majority Classification’ system cannot outperform moving average (one of the detectors in the ensemble) proving this method is ineffective. ‘Majority Classification’ has a higher average accuracy, but false negatives are crucial when analyzing CPU usage [13], therefore this metric is less useful than precision, recall and f1.

‘Majority Classification’ is more efficient than ‘Combined Confidence’, the simplicity of its voting system results in a lower execution time (less than half the time taken compared to the confidence technique). Some improvements may be required to improve the detection time so that this technique can work for real time outlier detection.

Evaluating the Effectiveness of the Ensemble

The ensemble technique of detecting outliers is sometimes very effective. In Fig. 7, graphs I, V, VIII and IX show that the detection is working and good scores for recall, precision and f1 are generated.

This technique is sometimes ineffective, especially against unstable datasets. In Fig. 7, graphs II, III and VII show that the detection has failed, and the ensemble of detectors are ineffective. Although the ensemble produces weak scores in these datasets, table 3 shows that the ‘moving histogram’ detector produces good scores. An improvement to the voting system, by potentially adding weighted confidences, could produce better detection in these graphs.

Observations of Fig. 7 show that the detector is very nearly producing perfect scores for some datasets. Graph VI shows that a false positive detection was made 1 datapoint in the timeseries away

from the false negative. This would have produced a perfect score for this dataset. Similarly, in graphs VIII and IX, the detectors would have produced perfect scores if they had correctly classified the second false negatives.

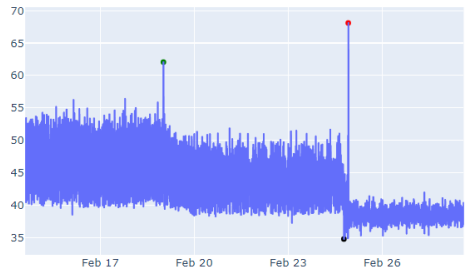
Conclusion

This experiment evaluated the newly implemented Ensemble technique of detecting outliers and determined that it is effective in detecting outliers and producing good scores in some datasets.

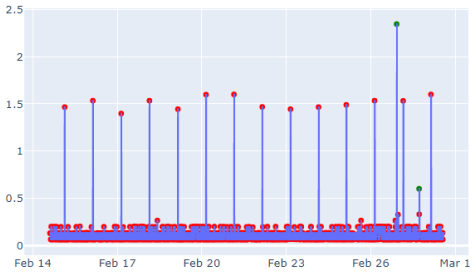
It was found that this technique is ineffective in detecting outliers in unstable datasets, but some detectors within the ensemble are more effective in unstable datasets than others. Meaning improvements to the voting system could improve upon this issue. In other datasets, perfect scores are almost achieved.

Results show that ‘Combined Confidence’ produces better scores than ‘Majority Classification’ as hypothesized. The intricacy of the voting system means that it takes longer to perform detection, but some optimization techniques could improve this score.

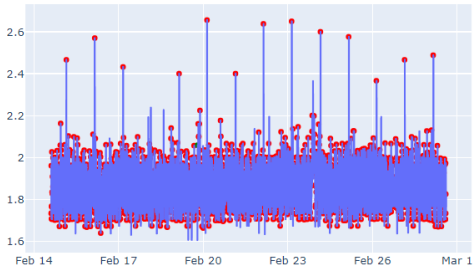
Appendix A – Experiment Results



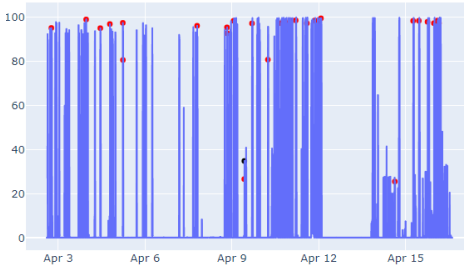
I. Detection result for Numenta VM1



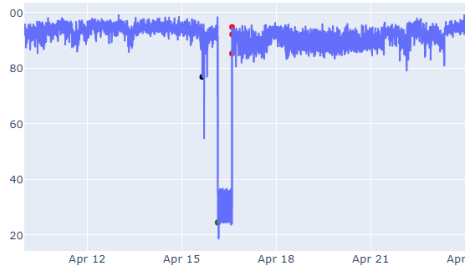
II. Detection result for Numenta VM2



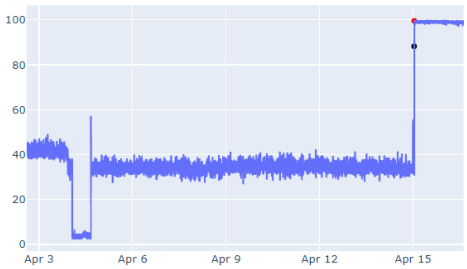
III. Detection result for Numenta VM3



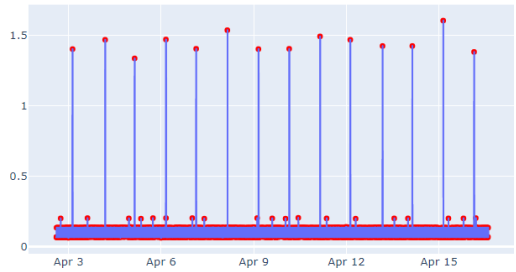
IV. Detection result for Numenta VM4



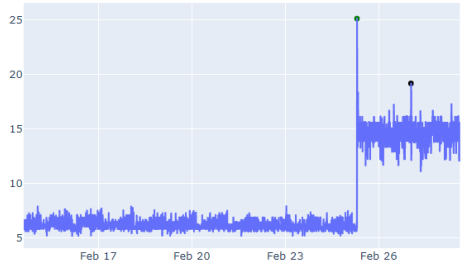
V. Detection result for Numenta VM5



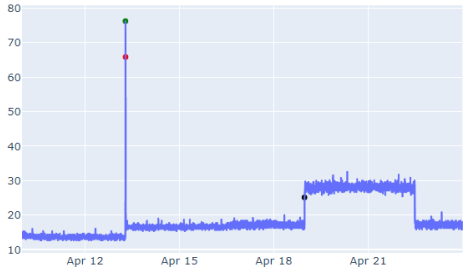
VI. Detection result for Numenta VM6



VII. Detection result for Numenta VM7



VIII. Detection result for Numenta VM8



IX. Detection result for Numenta VM9

Fig. 8 Graphs showing CPU utilization over time with outliers detecting using the Ensemble and 'Combined Confidence' voting.

Table 1 Results for ensemble detection with ‘Combined Confidence’ voting.

VM NAME	Accuracy	Recall	Precision	f1	Time to execute
Numenta VM1	99.97	50.00	50.00	50.00	14.3014
Numenta VM2	66.10	100.00	0.15	0.29	21.6323
Numenta VM3	43.23	100.00	0.09	0.17	25.6607
Numenta VM4	99.20	0.00	0.00	0.00	14.0828
Numenta VM5	99.90	50.00	20.00	28.57	14.4061
Numenta VM6	99.98	0.00	0.00	0.00	14.2985
Numenta VM7	70.26	0.00	0.00	0.00	18.5497
Numenta VM8	100.00	50.00	100.00	66.67	13.734
Numenta VM9	99.98	50.00	50.00	50.00	13.8773
Average	86.51	50.00	27.53	24.46	16.7270

* Numenta VM7 is excluded from the average since there are no true positives and an f1 score cannot be calculated *

Table 2 Results for ensemble detection with ‘Majority Classification’

VM NAME	Accuracy	Recall	Precision	f1	Time to execute
Numenta VM1	99.98	50.00	50.00	50.00	5.3486
Numenta VM2	77.75	100.00	0.22	0.44	7.2271
Numenta VM3	78.17	100.00	0.23	0.45	8.7147
Numenta VM4	99.93	0.00	0.00	0.00	5.4163
Numenta VM5	99.88	50.00	16.67	25.00	5.7832
Numenta VM6	100.00	0.00	0.00	0.00	4.8424
Numenta VM7	76.81	0.00	0.00	0.00	7.2695
Numenta VM8	100.00	50.00	100.00	66.67	5.0816
Numenta VM9	99.85	50.00	14.29	22.22	6.8909
Average	92.49	44.44	20.16	18.31	6.2860

* Numenta VM7 is excluded from the average since there are no true positives and an f1 score cannot be calculated *

Table 3 F1 scores (%) of Individual Detectors

VM Name	Average	Median	Boxplot	Histogram
Numenta VM1	50.0	50.0	0.0	0.0
Numenta VM2	0.1	0.2	0.5	66.7
Numenta VM3	0.1	0.1	0.0	0.0
Numenta VM4	0.0	0.0	0.3	0.0
Numenta VM5	25	25.0	8.2	0.0
Numenta VM6	0.0	0.0	4.3	0.0
Numenta VM7	0.0	0.0	0.0	100.0
Numenta VM8	66.7	50.0	1.8	0.0
Numenta VM9	50.0	50.0	9.1	0.0
Average	24.0	22.0	3.0	8.3

Table 4 Comparison of Voting Systems

Voting System	Accuracy	Recall	Precision	f1
Combined Confidence	86.51	50.00	27.53	24.46
Majority Classification	92.49	44.44	20.16	18.31

Table 5 VM Data Naming Convention

S/No	VM ID	VM NAME
1	ec2_cpu_utilization_5f5533	Numenta VM1
2	ec2_cpu_utilization_24ae8d	Numenta VM2
3	ec2_cpu_utilization_53ea38	Numenta VM3
4	ec2_cpu_utilization_77c1ca	Numenta VM4
5	ec2_cpu_utilization_825cc2	Numenta VM5
6	ec2_cpu_utilization_ac20cd	Numenta VM6
7	ec2_cpu_utilization_c6585a	Numenta VM7
8	rds_cpu_utilization_cc0c53	Numenta VM8
9	rds_cpu_utilization_e47b3b	Numenta VM9

References

[1] AWS Amazon. (2022). *Amazon EC2 – Secure and resizable compute capacity for virtually any workload* [Online]. Available: https://aws.amazon.com/ec2/

[2] AWS Amazon (2022). *Monitor Amazon EC2* [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring_ec2.html

[3] Ionos (2020, Feb. 24) *High CPU usage: What does this mean?* [Online]. Available: https://www.ionos.com/digitalguide/server/know-how/cpu-usage/

[4] Y. Sasaki. (2007). The truth of the F-measure. Teach Tutor Mater. Available: https://www.researchgate.net/publication/268185911_The_truth_of_the_F-measure

[5] T. Wood. (n.d.) Machine Learning Glossary and Terms – F-Score [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/f-score

[6] A. Tsymbal. (2004). *The problem of concept drift: definitions and related work*. Computer Science Department, Trinity College Dublin 106.2: 58. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.9085&rep=rep1&type=pdf

[7] Dr. Dataman. (2021, Apr, 18). *Anomaly Detection for Time Series. (1) Simple Moving Average.* [Online]. Available: https://medium.com/dataman-in-ai/anomaly-detection-for-time-series-a87f8bc8d22e

[8] Anomaly. (2016, Jan. 12). *Detecting Anomalies with Moving Median Decompsition*. [Online]. Available: https://anomaly.io/anomaly-detection-moving-median-decomposition/index.html

[9] A. Kliton, G. Shevlyakov and P. Smirnov. (2013). *Detection of outliers with boxplots*. 141-144. https://www.researchgate.net/publication/261173084_Detection_of_outliers_with_boxplots

[10] M. Goldstein, A. Dengel. (2012). *Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm*. Available: https://www.goldiges.de/publications/HBOS-KI-2012.pdf

[11] Numenta. (2015). *The Numenta Anomaly Benchmark – White Paper*. 1. Available: https://numenta.com/assets/pdf/numenta-anomaly-benchmark/NAB-Business-Paper.pdf#:~:text=The%20Numenta%20Anomaly%20Benchmark%20%28NAB%29%20is%20an%20open,NAB%3A%20the%20labeled%20dataset%20and%20the%20scoring%20system.

[12] Cloudfactory. (n.d.). *The Ultimate Guide to Data Labelling for Machine Learning* [Online] Available: https://www.cloudfactory.com/data-labeling-guide

[13] P. Huilgol. (2019, Aug. 24) *Accuracy vs. F1-Score* [Online]. Available: https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2