Detecting Anomalous Server Behavior using Isolation Forest

Course: Capstone Project for Würth Phoenix
Instructors: Francesco Melchiori and Claus Pahl
Student: Bilal Mahmood

Abstract

The main idea of this paper is to build an unsupervised machine learning pipeline based on an anomaly detection algorithm, Isolation Forest, and measure its performance in automatically identifying anomalous periods of operation of one of the servers hosting an ERP application. Flagging when the server has a problem and fixing it quickly can reduce the downtime of the applications running by the customers. The model built detected all the anomalies in the test set, with a decent precision, and marked periods when the machine was behaving differently.

Introduction

Detecting anomalies in time series have many practical applications, ranging from monitoring health of large collection of machines, to finding fraudulent activities in bank transactions, to as basic as improving quality of data in many real-world applications. Observing anomalies in hundreds of measurements generated every second from a network of connected servers is difficult and automation of that process makes it efficient and scalable. With these machine warning systems in place, the complicated systems can be monitored and run with little friction so end customers face as little down time as possible.

Würth pheonix is an IT branch of the popular Würth group that provides numerous high quality business management software products. One of their famous products, NETEYE, provides 360 degree monitoring of customer business applications with anomaly detection as a feature. Following is an example scenario handled by the NETEYE: Microsoft Dynamics AX is a popular enterprise ERP and it is usually virtualized and streamed to an employee PC through a network of layered hosts. However, because of a variety of reasons, end users suffer latencies or even downtime using the application. One or more hosts running the application can be affected directly or indirectly. NETEYE measures and tracks a number of system features from each host to automatically detect those anomaly periods. The goal of this project is, instead of the clustered based approach currently used for the anomaly detection, but to use a different approach and measure its performance. Below diagram shows 5 measurements, among the total of 18, which were collected for a week from a single host to build such an anomaly detector that could flag periods of defective behavior.

Figure 1: Measurements generated from a single host from 4th to 10th February 2019 with the anomaly happening on 8th February from midnight to 8am in the morning

In order to build such an anomaly detector, Isolation Forrest(IF) was applied, an unsupervised machine learning approach from Lui, Ting and Zhou (Lui et al., 2012), which considers anomalies as observations that are "few and different"(Lui et al., 2012:2). The core of the detector relies on the idea of randomly partitioning of the instances, and since anomalies are "few and different", they require less partitioning steps before isolation. This step is accomplished using building several random trees until all the sampled data points become isolated (become leaf nodes) and average length of the data point, leaf node, from the root, is used to define the anomaly score – as the anomalies are easier to isolate, they have less average distance from the root than normal "many and similar" instances which on average get a larger average distance. The pipeline thus built using IF had the recall rate of 100%, precision rate of 15% and had the area under the receiver operating curve (AUC) score of 96% on the test set. The simplicity of this idea along with its potential scalability, since unlike distance and density based methods, the Isolation Forest algorithm has low computational cost (Lui et al, 2012), that makes it an attactive solution for real time anomaly detection.

Data Description:

18 time series were tracked every few seconds from one DB host for the period spanning 4th February 2019 to 10th February 2019 and comprised several metrics. The metrics included details about system memory, processor usage, disk read/write statistics, transaction details, SQL statistics and numerous others. At the start of 8th February, an anomaly was recorded for 8 hours, as can be seen by unusually high processor statistics. Figure below shows the data tracked from the server:

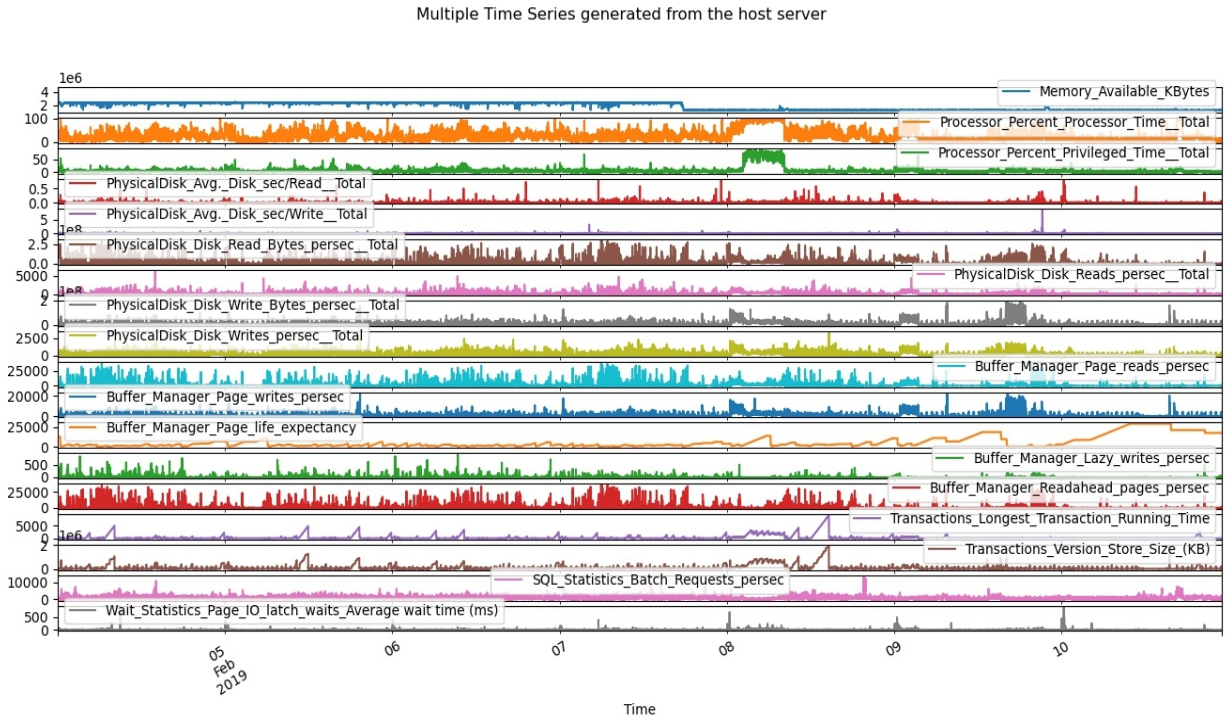


Figure 2: All the data from the server including the 8 hours of anomaly observed on the midnight of 8th February 2019

Methodology:

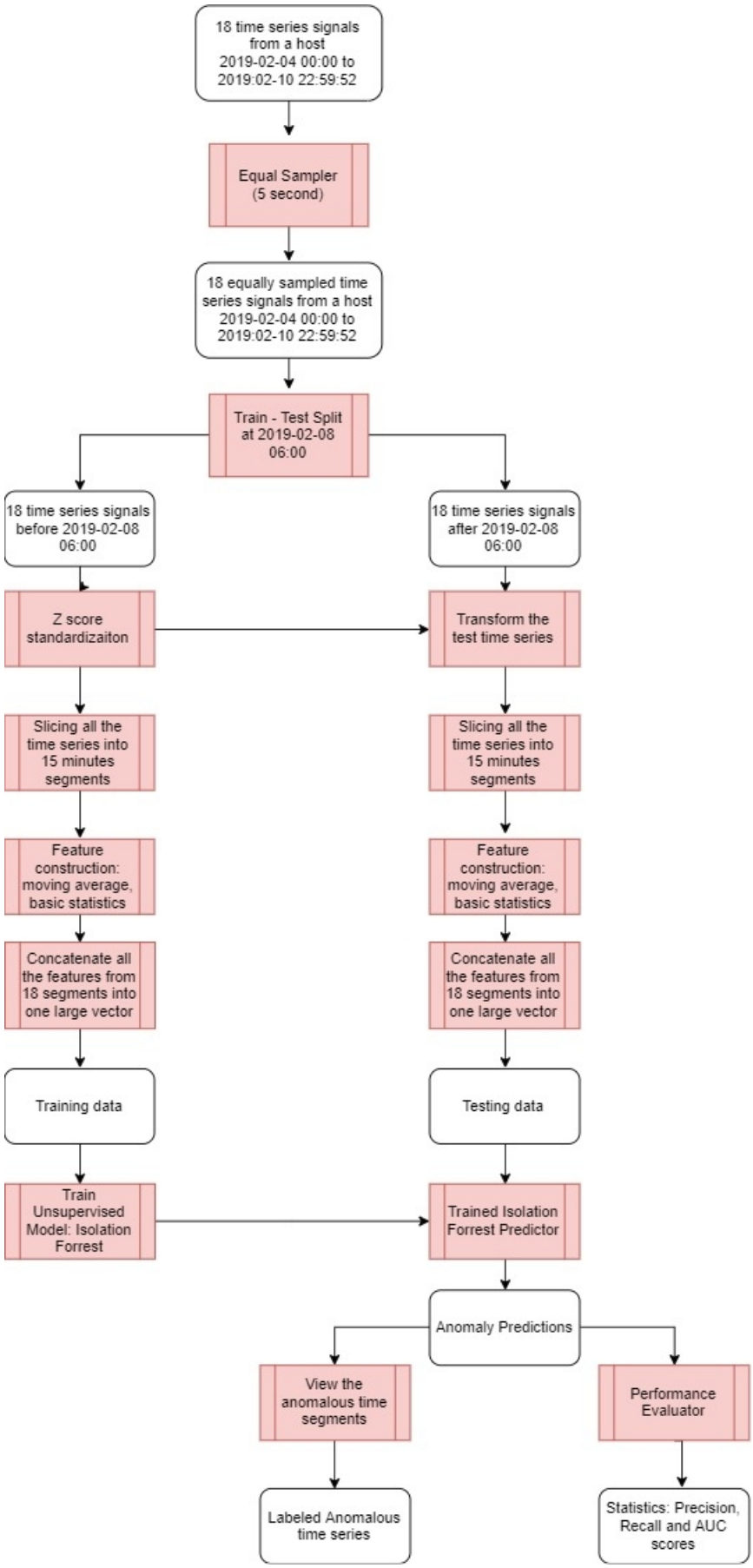


Figure 3: Summary of the whole methodology for building and evaluating the anomaly detector

In order to build such a, pipeline several steps were performed – the above figure summarizes the core steps taken. The first one was to solve the problem of unequal sampling. The time series were sampled at 5 seconds intervals, but because of reasons unknown, some measurements were not collected at exactly 5 seconds marks. In order to solve this issue, measurements which were delayed or arrived a bit soon were dropped, and the most recent past value was used to assign value to 5 seconds tick mark, so that all the data is sampled at the same frequency. This step was essential for constructing same sized feature vector later in the pipeline.

After fixing the problem of unequal sampling, the data was broken into train and test set. The time used for this was chosen to be 6am on 8th February 2019. This moment was chosen so that first 6 hours of the anomaly could be found in the training set and later two hours in the testing set. Next the training data was normalized to z scores with all the 18 time series had their values subtracted by their respective mean values and divided by the standard deviation. The statistics learned from the training data were used to normalize the testing time series. This was done to make the training and testing steps exclusive of each other without any information leaking to happen between the two parts, an essential step required for evaluating machine learning models correctly.

Moving along the pipeline, collection of time series in train and test were cut into 15 minutes segments – the time window selected by the domain expert. Each segment was then used to extract the essential features. We used mean value of the segment, its standard deviation, the minimum and maximum value (4 segment statistics) along with the smoothed original signal itself as the features extracted from each segment (169 features). Altogether we got 173 features from each segment. To provide all the information to the machine learning model, features derived from each segment were all concatenated into one large vector, resulting in 3114 dimensional vector, which would be used as an instance that would capture the state of the host at a particular time window. Below figure shows four 15 minute time segments capturing different readings from the server, normalized by the global mean and global standard deviation of the time series in the training data set.

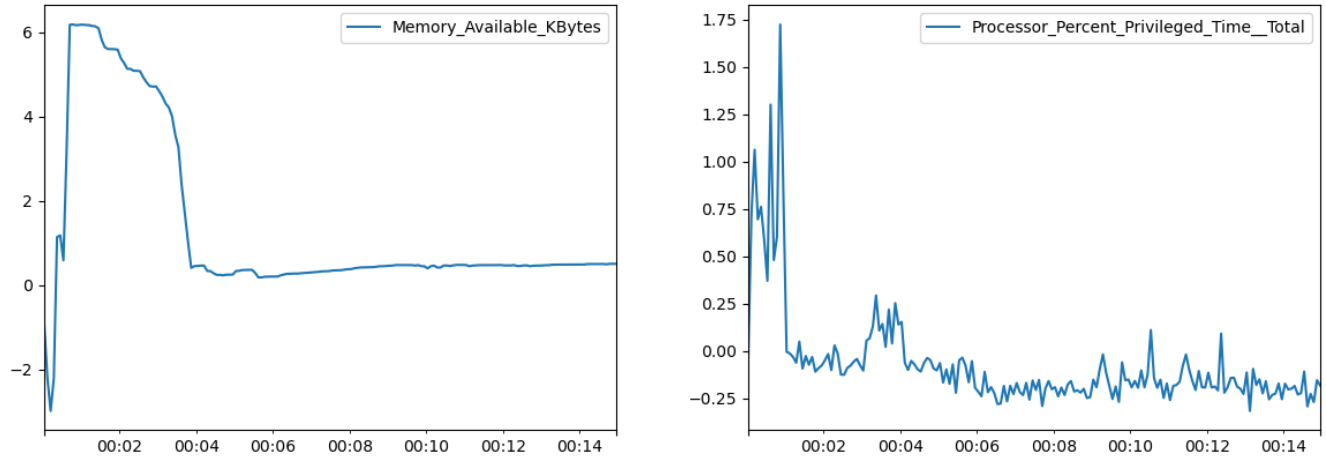
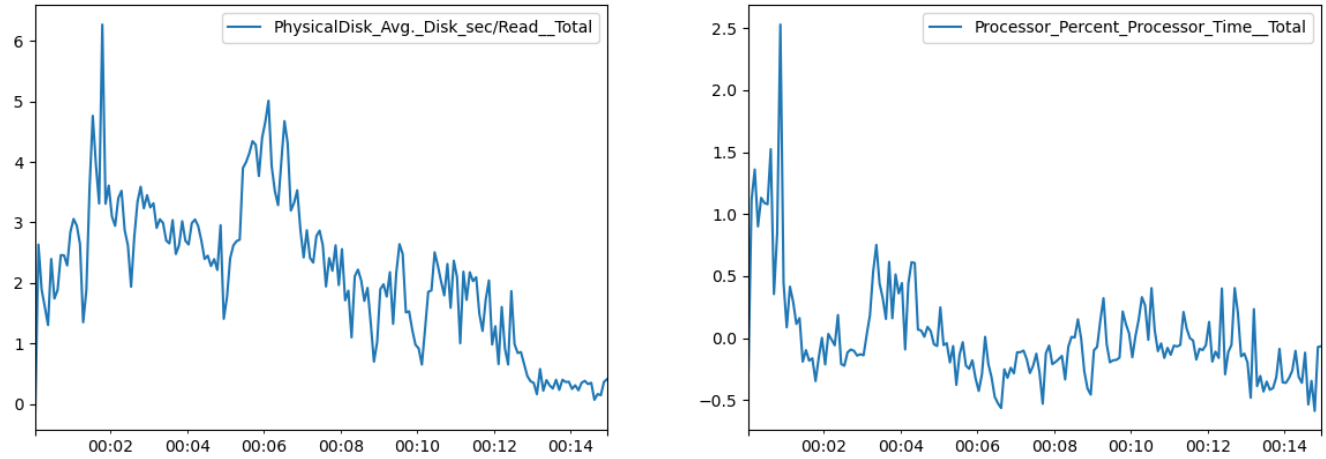
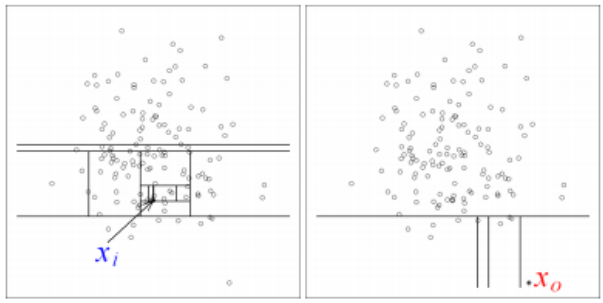


Figure 4: 15 minutes segment of several metrics that were used to extract important features, and later concatenated to

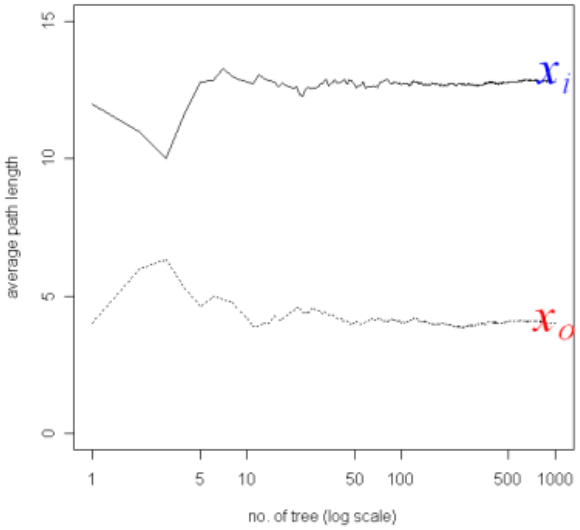


produce one large vector representing the state of the server

After preparing the data for training and testing, we trained the isolation forest on the training set and tested how many of the anomalies it was able to retrieve in the test set. Since the main idea of isolation forest is that anomalies can easily be isolated from rest of the data points because they are rare and few. This is achieved by randomly selecting a feature and randomly partitioning the feature space between that features maximum and minimum values. This process is repeated several times by growing several binary trees, and because anomalies get isolated first compared to normal points, on average they have a shorter average distance from the root. A score is assigned based on this distance, shorter the distance higher the anomaly score. Below figure, borrowed from authors original paper, how anomalous data point needs less partitions to get isolated and have lower average length than normal-
many data points. Furthermore, during the testing phase, new data points are run through several isolation trees and their average path from the root to the terminal nodes are used to assign anomaly scores to each.



(a) Isolating x_i (b) Isolating x_o



(c) Average path lengths converge

Figure 5: Anomalous instances need less partitions to get separated from the rest of the data points, and hence, on average, are closer to the root of the isolation binary trees (iTrees)

Evaluation:

To evaluate the performance of the constructed unsupervised machine learning pipeline, the original time series were broken into two segments, one before 8th February 2019 before 6 am, comprising first 6 hours of the anomaly period, and the testing period which included 2 hours of anomaly, from 6 am to 8 am on 8th February 2019. The performance was evaluated based on three numbers: precision, recall and area under the receiver operating curve (AUC). The constructed pipeline had recall of 100%, i.e. the model detected 2 hours anomalies present in the test set, with a precision of about 16%. Below figures show the confusion matrix, receiver operating curve and the anomalies marked by the model.

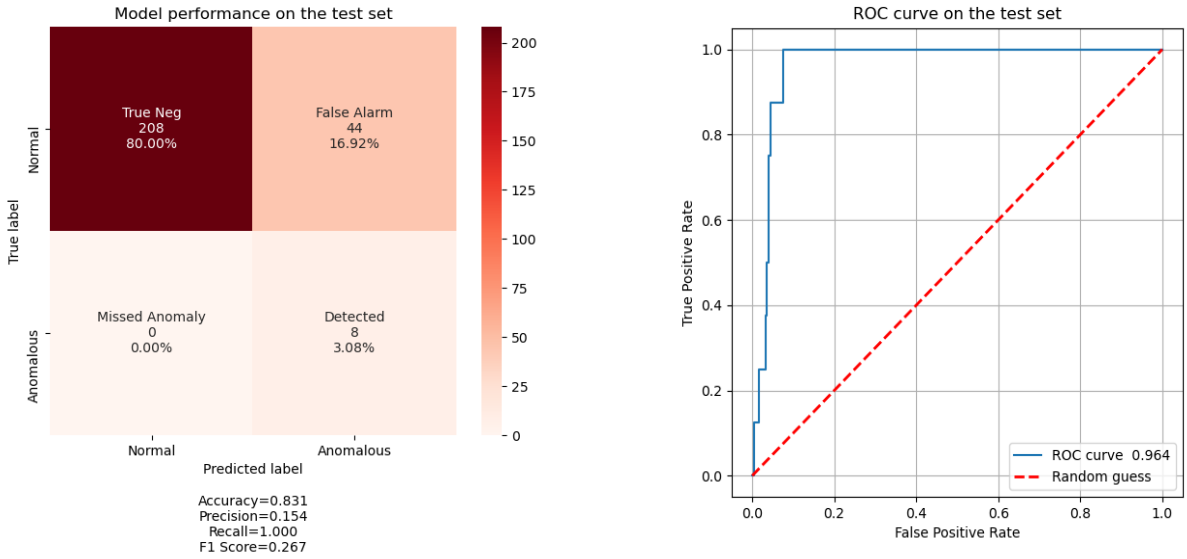


Figure 6: Figure on the left shows the confusion matrix, showing that the model detected all the anomalies, but had a lot of "false alarms" too, whereas the right plot shows the ROC curve with the AUC score

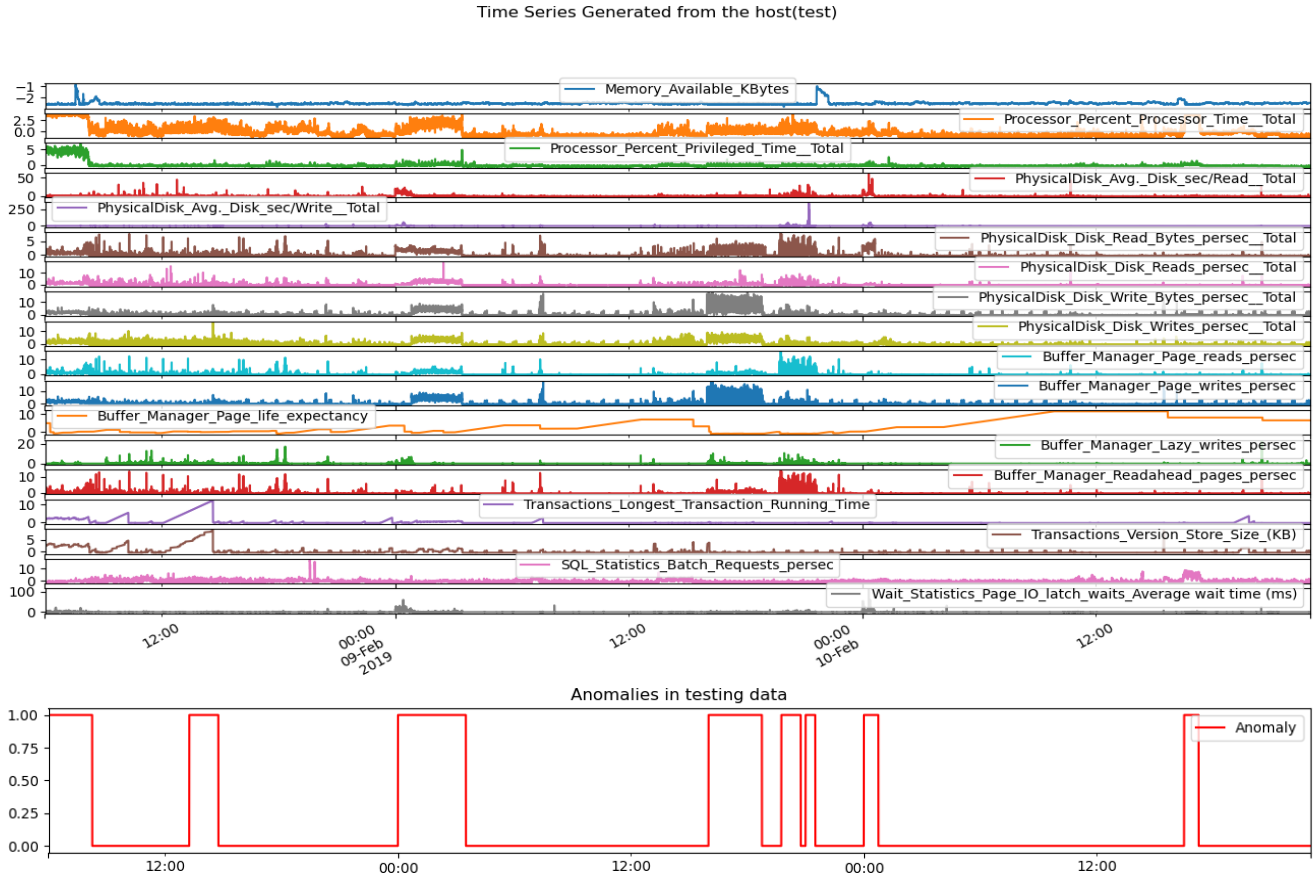


Figure 7: The model detected 2 hours of anomaly before 8:00 am of 8th February. In addition, it marked other time periods where the server was behaving in abnormal ways. For example, during the beginning hours of 9th February, processor, physical disk and buffer manager operations were unusually high, likewise, in the later half of the day, there was also unusual activity that has been captured

Conclusion:

In this project a simple unsupervised machine learning pipeline was constructed using Isolation Forest that detected anomalous server behavior from 18 metrics tracked over a period of a week from a single DB server. The algorithm proved useful in terms of simplicity and utility in detecting all the anomalies present in the testing days and identifying periods in which the database server was operating strangely. Next logical steps would be to test the developed pipeline on new observations and label anomalous points in large training data for building anomaly forecasting system that would alert future server failure.

References:

Liu, Fei Tony, Ting, Kai Ming and Zhou, Zhi-Hua. *"Isolation-based anomaly detection."* ACM Transactions on Knowledge Discovery from Data (TKDD) 6.1 (2012)

Jan van der Vegt (2019, June, 24). A walk through the isolation forest [Video]. YouTube. https://www.youtube.com/watch?v=RyFQXQf4w4w