Expertise
and insight
for the future

Sara Al Kafri

Early identification of anomalies from user experience data

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

1 November 2020

PREFACE

Looking back, on the bumpy road towards this dissertation, a mixture of feelings, memories and thoughts overwhelm me when considering my life as a Master student. I really feel privileged for having been able to bang my head against several brick walls, to discover valuable knowledge along the way, and to be surrounded by so many strong people who all wanted, always kept believing, and made sure that I would reach this milestone successfully, it was very interesting to go back to school, study again and having assignments and deadlines.

First, I would like to thank my former managers Janne Heino and Jarkko Jussila for believing in my abilities and giving me the golden opportunity to join their team and contribute in some tasks with the freedom to explore several roads.

For this, and other insights during inspiring discussions, I would like to thank NESC monitoring team who helped me connecting the dots, in addition to constant assist whenever I pinged them to ask questions.

Naturally, I'd like to thank my supervisor: Peter Hjort, for reminding me that machine learning is not necessarily a complicated model, he brought my attention for small details and supported me always with useful references.

Moreover, I would like to thank all other people that have crossed my path the past years for putting my mind either at work or at ease.

Finally, I want to express my ultimate gratitude towards my husband, my mom and my kids for their love and support to make this achievement happened.

Thanks!

Helsinki, 01.Nov.2020
Sara Alkafri

Author(s) Title Number of Pages Date	Sara Alkafri 40 pages 02 Nov 2020
Degree	Master of Engineering
Degree Program	Information Technology
Specialization option	Data analysis and machine learning
Language	English
Instructor(s)	Peter Hjort

Abstract
Detecting anomalies in time series data is a critical task in areas such as cloud health monitoring. This Thesis proposes a proof of concept for forecasting and detecting anomalies in time series data. The proposed approach is based on Facebook Prophet model which is an open source library built on decomposable (trend+seasonality+holidays) models. It gives the user the power to perform time series predictions using simple intuitive parameters with acceptable prediction result. Moreover, the architecture helps the concerned team to detect outliers and understand what kind of problems that they may have. The results on Cloud Functional Testing data show the ability of the proposed model to detect anomalous patterns in time series from different fields of application.
This study presents the capability of accurately forecasting future cloud health with expected level of reliability in our forecast.

Keywords	Time series, forecasting, data science, machine learning, Facebook Prophet, anomaly detection

Contents

List of Figures

List of Tables

1 Introduction

1.1 Research context and scope

Nowadays, cloud services are the dynamic choice of companies to run their business processes on. Clouds allow business owners to improve their productivity by accessing, developing, and deploying their services from anywhere anytime. Data is always available across the cloud(s), the cutting-edge technologies are applied easily.

Large amount of data will be produced and collected within a cloud. Collected data might be facts, numbers, or pictures that will be analysed and interpreted into useful information.

Data analysis is to discover meaningful information from the collected data in such a way that might innovate the running business and to eliminate unnecessary services that cause abnormal behaviors. Cloud architects and engineers are welling to keep the cloud in best shape, they will create or adapt technologies to utilize extracted information properly to improve their products quality, a simple defect can lead to services failure, or cause a huge transfer of data. As result, keeping the cloud up and running is the main concern of the cloud monitoring team. Estimating future performance will help the team to capture cloud problems well in advance.

Companies are interested in predicting performance by using forecast processes to make important business decisions. Both past data log and trends in cloud behavior could be a base of the forecast.

Predictive performance and cloud health are critical for getting more high-quality services in any business. Improvement and crucial decisions might increase the revenues and stabilized the business. Several studies have been presented in the literature with variety of methods. These models are designed to model selected metrics to predict future performance and detecting anomalies, however, if anomalies are not well-timed detected and handled, it would lead to bad performance consequences which effect business and cloud efficiency.

Cloud platforms is hosting running applications, collecting data operation logs, metrics therefore apply anomaly detection method.

This thesis aims to utilize 6 month of user's experience test result logs to investigate the possible occurring anomalies and forecast the future process execution time. Forecasting and anomaly detecting results will contribute positively to improve cloud health and performance.

Figure1 shows real data logs coming from company's data source, they simulate user's behavior in a cloud, denoted as 'TestX' for confidentiality reasons, at Nokia. By looking closely at time series in figure 1, it displays TestX execution time against time, TestX data set for 6 months, execution time might be longer, and this abnormal behaviour could be an anomaly.

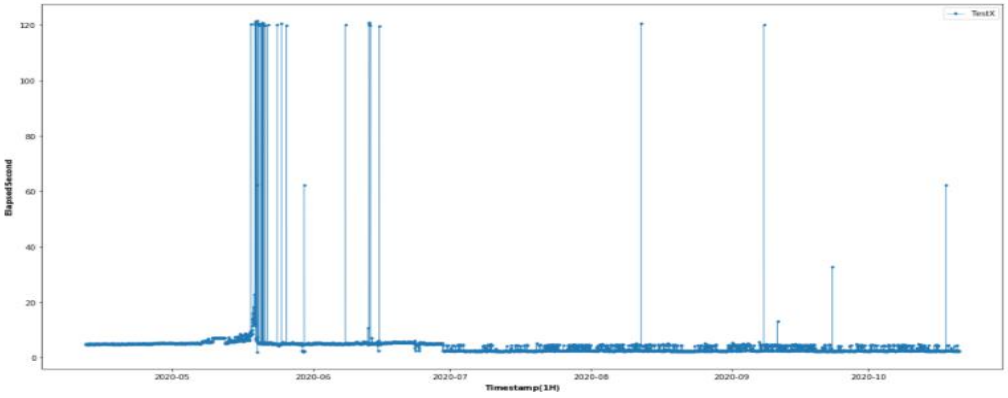

Figure 1 Timeseries shows execution time of a test over time t.

Mostly anomalies differ from application to application, it becomes difficult to generalize normal and abnormal behavior that covers different data types and domains.
It is difficult to come up with a definition of an anomaly that accounts for every deviation from a normal or standard behavior.
This study's aim is to enhance cloud services quality with the help of early detection of anomalies.

1.2 Problem Statement

Nokia is one of the biggest private cloud in the world that has billions of transactions (CGIFinland, 2019). Enormous amount of data has been gathered from these clouds in daily basis. Monitoring team is responsible of cloud performance, they ensure the cloud is always healthy and deliver to the customer best experience ever, as a result monitoring team should analyze all the collected logs and trigger the needed alerts to keep the cloud up and running properly.
This study considers the collected monitoring event logs that are produced from the process of simulating user experience in the cloud.
User experience simulator generate large range of data logs that helps to monitor cloud's performance and health. Log files contain collection of measured cloud features.

Features are usually collected every 3-5 seconds and over time (Time series). Monitoring team experts write alerting rules to keep an eye on cloud performance, however, it is not the optimal method as it relies on every “abnormal” action that’s happened in the cloud, this will lead to trigger new trend in the cloud or annoyable alerts. As a result, critical alerts are left untracked due to the not ended stream of alerts.
Figure 2 shows how the process flow that produce data logs.

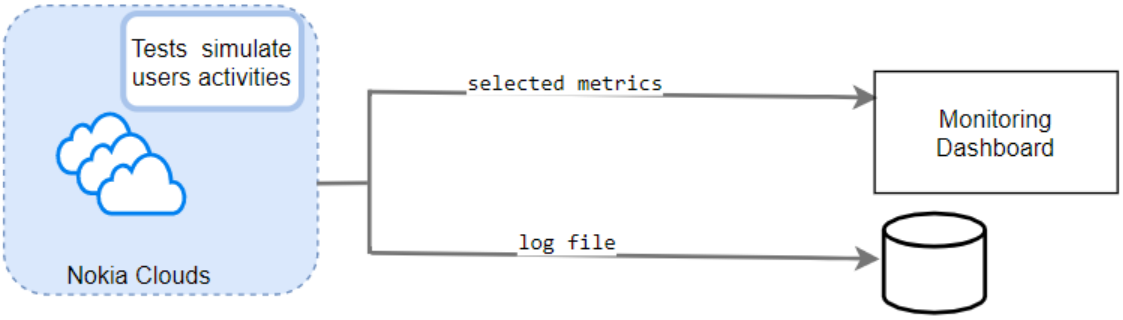

Figure 2 Framework Architecture

1.3 Objective and Result

This thesis objective is to forecast test’s execution time features and develop an anomaly detection model.
This study is a proof of concept which helps monitoring team in future to build their own machine learning tool, the results will be a list of predicted values of test’s execution time for the next 3 days in addition to detected anomalies points in the historical data.
These Results are presents as table and chart.

1.4 Real Data

This study has been applied to real data coming from production process of cloud monitoring logs – denoted as ‘CloudX’ for confidentiality reasons – at Nokia. The data set consists of X number of tests results metrics. Each test is related to certain category denoted as ‘CategoryX’. Here anomaly detection is applied over only one of monitored cloud logs, however the experiment could be applied across all clouds. Out of the X test metrics we have selected 2 training sets, both of sets are consisting of approximately 2000000 data point.

1.5 Thesis Outline

This thesis contains the following chapters.

Chapter 2: Methods: Introduce forecasting and anomaly detection methods that this study is used.

Chapter 3: Data: an overview of data statistical features. Inspect how data looks like and extract useful information about data properties

Chapter 4: Experiment: Reporting how this thesis has been applied over a real data set, experiments, and results.

Chapter 5: Conclusion: The chapter concludes the results and observation regarding to the applied methods in this study, in addition to improvement suggestions for future work.

2 Methods

Time series data is a sequential flow of data point observations collected as result of a certain process. Time series forecasting is to predict data values and behaviour overtime to enhance process performance. Forecasting method relies on identification the underlying trends, measure patterns that exist in the past. The results assume the future potential patterns and trend in data. There are several techniques to perform time series forecasting, however, it is impossible to know in advance which model will fit into data well.

Along with time series forecasting anomaly detection is a trend nowadays. Anomaly detection (also known as outlier detection) is the search for items or events which do not conform to an expected pattern.

This chapter purposes to give an overview of the concepts and methods that are used in this thesis.

2.1 Facebook Prophet Forecasting Model

This model has been developed in the last few years. Developers describe the model as following: “Prophet is an open source software that is available in python and R for forecasting time-series data, Implements a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects”. (Facebook, 2020).

Prophet was produced by Facebook’s Core Data Science team and it is using Python. Prophet works the best where data has of hourly, daily, or weekly trends with at least a couple months (preferably a year) of historical data.

The Prophet procedure related to Generalized Additive Model (GAM) family (Taylor, S. and Letham, B, 2018). Generalized additive model (GAM) (Hastie, T. & Tibshirani, R. , 1987) , is a regression models with non-linear smoothers applied to the model variables. Prophet is using time predictor. The GAM formulation is suitable to be adapted in this model because it is decomposing easily and accommodating new components as necessary, for example when new seasonality is detected. GAMs highly fits by applying back-fitting or L-BFGS (Byrd, R. H., Lu, P. & Nocedal, J., 1995) so it gives the user high flexibility to be interactive and change the model parameter as needed. So by using Prophet, user does need to have a strong math background.

Prophet is customized flexibly, essential extensions can be handled to apply time series data modeling easily. All these features are wrap up inside the Prophet model with high simplicity.

The strength of Prophet model shows up when the fitted data has strong variant seasonality such as: day of week, business days, national/important holidays, a number of missing observations, epidemic trend changes like we are experiencing right now, and non-linear growth of trend. All the previous examples might cause data irregularity and it would be considered an outlier.

Facebook prophet is used in this study as forecasting model because it supports the user with the advantages of the Bayesian approach.

Bayesian approach is a mathematical method to solve statistical statements that requires solutions by using probabilities, it unites the input information and sources of uncertainty into a predictive distribution for the future values to produce the forecasted interval.

Prophet use the advantages of Bayesian algorithms as following:

1. It makes Prophet model to be handy to use and easy to explain periodic structure.
2. Prophet prediction output supports the confidence interval derived from the complete posterior distribution, in addition to risk estimate.

Prophet strives to provide a simple to use model that is sophisticated enough to provide meaningful insights and results. The modeling solution provides numerous parameters that analysts and data scientists alike can alter easily to suit their modelling requirements. Our implementation is on Ubuntu and using PyPI which is the Python Package Index and known as the official third-party software repository for Python. (Facebook, 2020)

Prophet is expecting columns to have specific names, ds for the temporal part and y for the value part, so we adhere to that. It is a powerful model, there is no need to interpolate missing values. Prophet is defined in terms of regression-like model (Taylor, S. and Letham, B, 2018)

Analyst-in-the-Loop is the framework that Prophet Method uses as shown in figure 3 below (Taylor, S. and Letham, B, 2018)

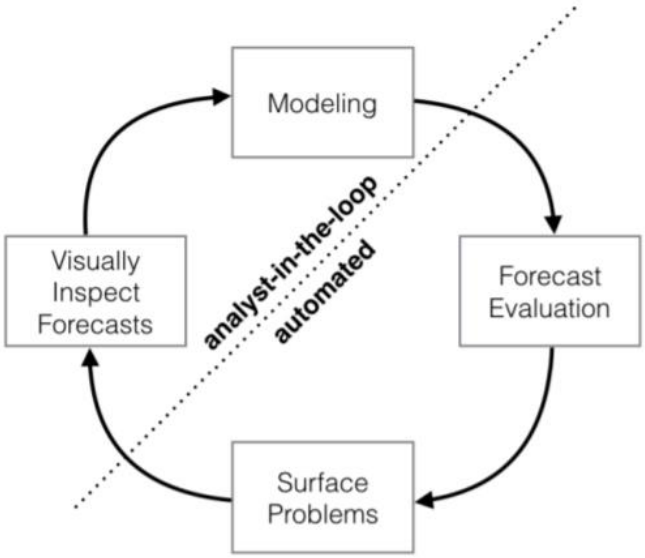

Figure 3 Analyst-in-the-Loop

The “Analyst-in-the-Loop” framework has two sides where on first side is to automate the model by supposing that the user has no prior knowledge of statistics, while on second side this framework qualify this same user to interact with model in such an easy way to feed the model with input data taking into consideration user’s domain/industry knowledge.

Prophet is an additive regression method which contains of the following components and functional form:

$$y(t) = b(t) + s(t) + f(t) + \epsilon_t. \qquad (1)$$

Equation (1) the component is proposed as following: b(t) represents trend changes in time series, s(t) represents seasonality in timeseries, f(t) represents holidays and εt is a the normal distributed noise factor that is often used when modelling with Bayesian approach, this noise term clarifies abnormal changes that are not accommodated by the model. Unlike other model terms, εt is always present in any instance of Prophet Model, remaining terms may not always be presented, as the user decides what the needed terms for his model instance are.

2.2 Anomaly

The Most common asked question we would like to know when we talk about the outlies is: "What are Anomaly?".

Anomaly is introduced as an action that differ from usual or expected behavior. As an example, suppose an e-Shop has a sudden drop in their selling rates while their application is still up and running normally, statistical charts shows that they usually have higher selling rate and no abnormal action happened before. This is irregular activity for sales rate is an anomaly for the e-shop. The anomaly is sort of incompatible observation in the usual known data behavior.

It is important to know how an outlier is described for the purpose of this research.

The outlier is described as a Statistical data which is extremely different from the others in the same sample, as a result the keyword here is "different". An outlier is a point that is significantly different from other data points in the set. In figure 4 an example of outliers, red dots represent an example of abnormal behavior in a time series, TestX does a certain task, usually task's execution time takes 0.3-1.2 seconds, and for certain circumstances testX takes more than 1.5 sec to execute the assigned task. After anomalies is detected, the monitoring team managed to understand the cause problem and take the proper action.

For this research an outlier is defined as an observation that is out of the ordinary. In general, there are several reasons why outliers may occur in a time series. The first explanation would be data errors. Secondly, the data point that is an outlier could be generated by a different population and thus be generated by a different distribution than the rest of the data. This could be related to changes in behavior of people or a system. A third explanation would be that there are unusually high residuals, (Greene, 2012), Residuals are is the difference between the observed value and the mean value that the model predicts for that observation and adjust accordingly.

Outlies might also be caused by human mistakes or tool error.

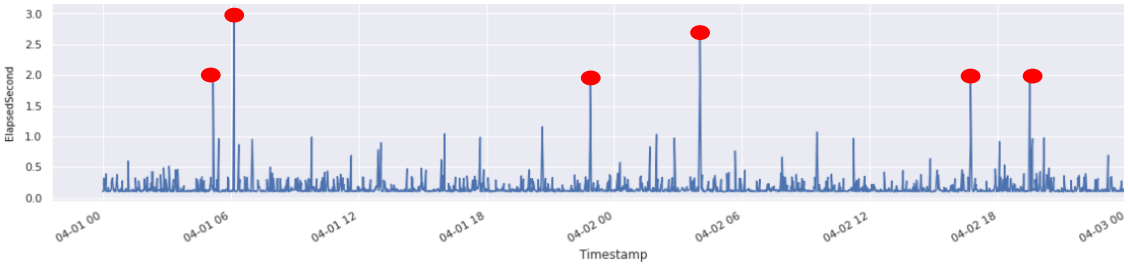

Figure 4 Possible Outliers in TestX time series

2.2.1 Anomaly Analysis

This section aims to clarify the different types of anomalies, since the type of data and anomaly has considerable effects on the quality of anomaly detection methods.
Anomaly indicates the data points that could be of interest due to the strong deviation from other data points, it could be considered as a special type of outlier. The present task of anomaly detection consists of making a distinction between noise and anomalies. Anomalies could be categorized as following: point anomalies, contextual anomalies, and collective anomalies. These types are defined as following:

- Normal anomaly is an isolated data point that considered as odds comparing to the rest of data.
- Contextual anomaly is a data point that inside the normal known distribution range but are anomalous in comparison with the seasonal common pattern.
- Collective anomaly a collection of data shows different shape compared with the regular and usual patterns that appear in time series.

Anomaly detection has several approaches, it falls into the following approaches:

- Supervised learning model: an algorithm that is provided with labelled dataset to learn from, these labels is used by the algorithm to evaluate its accuracy on training data.
- Unsupervised learning model: This model is the opposite model of supervised learning. It is an algorithm that is provided with unlabeled data. The algorithm will extract the existing features and on its own.
- Semi-supervised learning: It is a hybrid model that is a mix between supervised and unsupervised algorithm. It combines a smaller group of labelled with large group of unlabelled data, the algorithm will the smaller labelled dataset as training set to learn the features of data, the test set will be the large unlabelled data, the result will be two data samples: unknown samples which are considered as outliers and known or normal samples that are classified according to the known labels from training set. Unknown samples are considered as anomalies because of their irrelevant behavior is a way from that of the known normal samples.

2.2.2 Outline

It is difficult to define a known data point as anomaly in practice because of either lack of information or technique. Therefore, in reality it will be more applicable to perform

unsupervised methods, and this study will only handle unsupervised method with collective anomaly.

2.2.3 Z-Score for finding outliers

Z-Score is a functional method to be applied as a benchmark in the unsupervised machine learning which groups varied algorithms for the ultimate anomaly scores (J. Gustafsson, F. Sandin, 2016).

This algorithm needs a time series as input. Furthermore, it needs three user inputs: a lag values (hereafter called lag), a threshold and an influence. The lag indicate the number of previous observations that are taken into account to smooth the data, the threshold, τ , is used to dene what is an outlier and what is not, and the influence tells the influence that an outlier has on the smoothed standard deviation and average. The algorithm itself works as follows: the difference between a real data point, x, and the smoothed average from the last lag observations, μ, is calculated and compared to τ times the standard deviation from the last lag observations, σ. If the difference is higher, then the new data point is considered an outlier and it will be saved as an outlier. After, the new smoothed average and standard deviation are calculated, with the influence value if the real data point is considered an outlier (P.,Kiselev., R., 2020).

In other words, The Z-score method leans on mean and standard deviation of set of data to calculate central tendency and dispersion, it will test if the number places outside the three standard deviations, this is the base rule, if the value is outlier, the method will return true, if not, return false.

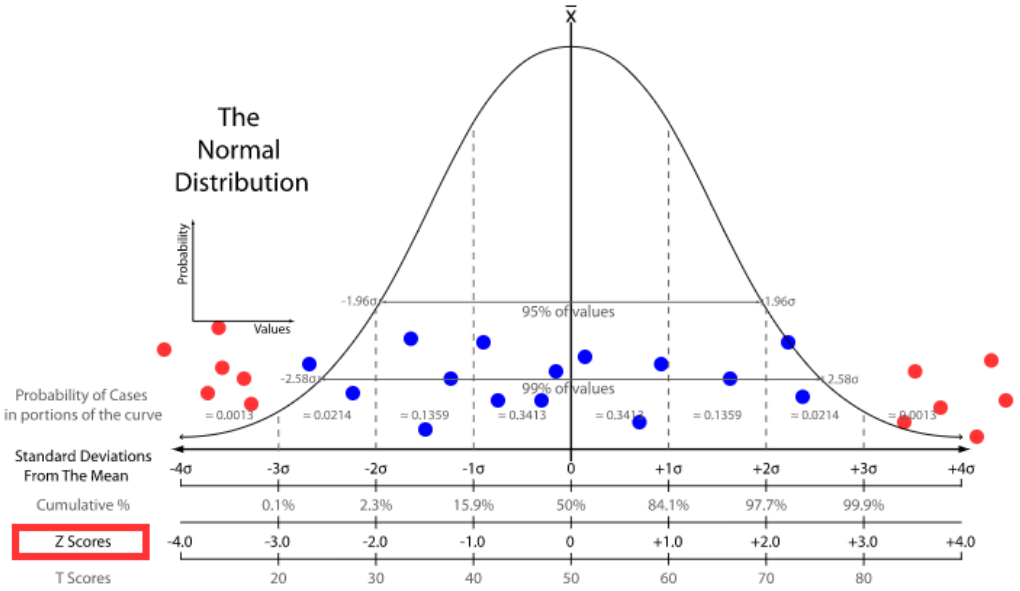

Figure 5 Z-score Algorithm

Z-Score fits well with Facebook prophet that's because prophet has uncertainty intervals feature which is given upper yhat and lower yhat, it captures the un-modeled variance of the time series and not producing false positives in settings where the model just doesn't have a clear fit for what's happening (Prophet, 2020). Width of them the uncertainty intervals can be configurable however default width had given a good result.

3 Data

Time series is an important instrument to model, analyze and predict data collected over time.

This study suggested a successful performance forecasting model built on real live data as suggested in Figure3.

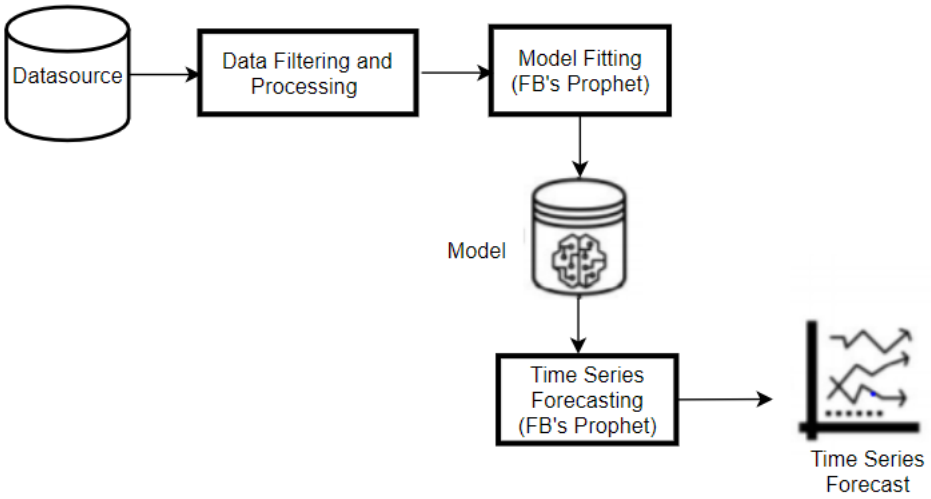

Figure 6 Process Workflow.

3.1 Data source

In this study, our data source provides us a cleaned dataset that helps to spend the time on actual analysis and not sorting out typical problems of bad data.

In this chapter the experiment will applied across one cloud instance.

3.2 Data Preparation

Python is used as programming language, it is essential for data analysis because of the statistic libraries that python offers, it is easy to learn, an open source and Python is well supported.

The test's time series data is extracted from the related data source (database), dates are clean and in right format, but still different pre-processing procedures are needed to be applied to data right before starting analysis study is done. The aim of data pre-processing is to achieve a combination of an aggregated timestamp and execution time at same time frequency.

3.2.1 Choosing Features

Data feature assessing is very important, it contributes in model prediction performance. In order to pick the meaningful features, it is needed to understand deeply each field in the data source, for data confidentiality it would not be possible to mention the steps of this section in details.

In the beginning in this study it has been chosen 3 different features that team believed it might add meaning to the model, eventually we nominated 1 feature to be predicted and drop the others.

The used model in this study applied on single observation (test execution time) that occurs sequentially over time (test duration) and this is called univariate time series.

Univariate time series term means a time series that contains one single variable varies over time, this variable is recorder in sequential order over time.

The predicted value is the future execution time for the test and detecting anomalies in historical data.

3.2.2 Dataset Indices

Dataset's timestamp is used as an index. Execution time order is important in this study case, it remarks the order of running test cases in the cloud.

Notice that:

1. This case is not numeric indexing, For instance, if we index a list as df[:4] then it would return the values at indices – [0,1,2,3], but here the index '2020-06-01' was included in the output.
2. The indices must be sorted associated with it in ascending order. Here workflow is sequential, so ordering the indices will allow us to identify the out-of-sequence data point and indicate possible irregularities. If randomly the indices are shuffled it will definitely give wrong results.

   Facebook prophet has a feature that the output of `m.predict` will always be sorted (Facebook/Prophet., GitHub., 2020).

Prophet requests data with YYYY-MM-DD format, as input for the model instance, data frame should always has two columns: ds and y, our index is timestamp in the required date format YYYY-MM-DD HH:MM:SS.

The y column has to be numeric and represents the execution time we wish to forecast.

3.2.3 Timeseries Extraction

The used data is presented in the form of tables, each table row is presenting the value of a specific execution time for a specific test at a specific timestamp. In this phase, all data values that are related to the same test and it will be extracted to a separate data set. The following tables describe how data looks like before and after pre-processing. Table1 shows the main Test results log where comes directly from the main data source, it has all executes tests over a cloud. In order to get each test time series separately (Test1 timeseries, Test2 timeseries, etc.), the main log is introduced in Table1 will be processed in the python code, as a result the univariate time series will be ready to be predicted.

Timestamp	Test	Elapsedseconds
Timestamp1	Test1	0.19980033
Timestamp1	Test2	2.16232874
Timestamp2	Test1	0.81007643
Timestamp2	Test2	0.23498004
Timestamp2	Test3	0.33432324
Timestamp3	Test1	1.03132135
Timestamp3	Test2	2.09763213
Timestamp3	Test3	0.72332434

Table 1 data source before pre-processing

Test1	
Timestamp	ElapsedSeconds
Timestamp1	1.71279312
Timestamp2	0.567776545
Timestamp3	0.915279332

Table 2 Timeseries for Test1 on cloudX

Test 2	
Timestamp	ElapsedSeconds
Timestamp1	0.01279312
Timestamp2	1.56700015
Timestamp3	0.01279312

Table 3 Timeseries for test2 on CloudX

Test 3	
Timestamp	ElapsedSeconds
Timestamp2	5.32620331
Timestamp3	0.05404146

Table 4 Timeseries for Test3 on CloudX

Additionally, unneeded fields have been removed from the data.

3.2.4 Time Aggregation

Timeseries aggregation is done to merge a set of periods into similar groups. The execution time data could be saved and collected every x number of seconds, it is required to unify data frequency to perform aggregation over our extracted data set. The aggregation unit for this data set is one-hour frequency. When test's execution time field has several values in an hour, then the statistical methods that are applied in our case will be either the median or mean of the values is used for that hour, these statistical aggregation function with depends on the type of data distribution whether it is normal data distribution or not. The following tables describe how timeseries data looks like before and after aggregation step:

TIMESTAMP	ELAPSEDSECONDS
2020-06-16 01:59:56	0.554521
2020-06-16 01:59:57	2.606284
2020-06-16 01:59:58	0.939185
2020-06-16 02:00:03	0.358077
2020-06-16 02:03:00	1.022024
2020-06-16 02:03:04	0.358077
2020-06-16 02:03:05	6.565416
2020-06-16 02:05:08	0.244675

Table 5 Timeseries before aggregation

TIMESTAMP	ELAPSEDSECONDS
2020-06-01 01:00:00	**Median() = 0.208773**
2020-06-01 02:00:00	**Median**() = 0.289091

Table 6 Timeseries after aggregation

In addition, prophet assumes a continuous y with normal noise, which can work poorly for small count data, so time aggregation will be ideal to our case.

3.2.5 Missing Data

Handling missing values in data set is common. The applied model in this study is a regression model on continuous times. So, the time set consist of times time1, ..., timeN, and observed values are y Value1, ..., yValueN at those times, y values is estimated by y = f(t) function. Since the applied model is continuous time, there's no problem with having a day missing and data set may not have value at every possible value of t. Our

missing data has been filled by using .fillna(method='ffill') functionality that pandas library offers.

3.2.6 Change in variance

In this section the aim is to find out the most fluctuated and non-steady, this information will help the monitoring team to understand test performing.
Variance is the statistical property that determine the fluctuation within a time series.
Suppose initial data value is X_i, then define $Y_i = (X_i - \mu)^2$, where μ is the mean.
Then the change in mean of Y_i will be a change of variance in X_i.

$$\frac{\sum_{i=0}^{n} Y_i}{n} = \frac{\sum_{i=0}^{n} (X_i - \mu)^2}{n} = \mathrm{Var}(X_n) \quad (2)$$

Change of values evaluates tests with in same category by calculating each metric's variance value and compare it with each other, then highest top 3 variance value will be picked to be monitored, these values could be measured every 3-4 month. The results has been verified a cross x number of clouds.

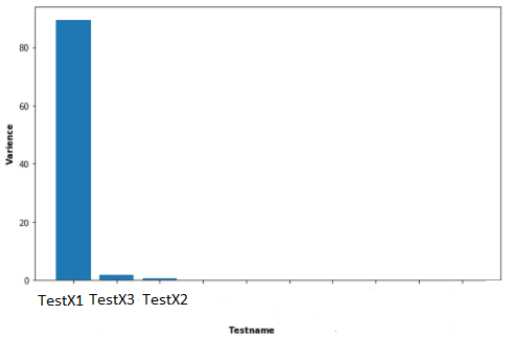

Figure 7 Top3 test with highest variance values for categoryX in cloudX

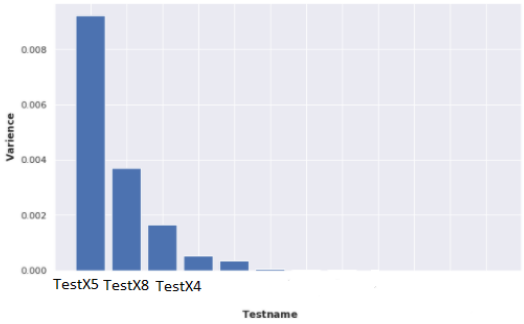

Figure 8 Top3 test with highest variance values for categoryY in cloudX

In Figure 8, testX1 and testX3 have the highest variance value across all examined X clouds, third nominated test varies from cloud to another. As conclusion, almost all may behave in same way.

3.2.7 Structural Breaks

Time series data in some cases show an unexpected behavior change at one point in time. As an example, Figure 10 shows that the execution time changed sharply in '2020-06-01' after the start of certain action known for the team. These unexpected changes are often represented as structural breaks or non-linearities.

Figure 9 Structural break in test's elapsed seconds over time.

Instability in the parameters is created by these structural breaks of a model. This, at a time, can diminish model validity and reliability.

Structural breaks in the median of time series data will show in graph as sudden unexpected shifts in the level of the data at certain breakpoints. For example, in the time series plot above there is a clear jump in the median of the data which around the start of June-2020. We also noticed drops in time series instead of jumps in other different cases.

3.2.8 Stationary in Timeseries

Exploring the state of time series stationary is important to build a model that fits the time series. Time series can be described as stationarity if its statistical properties does not change over time. The mean (average), standard deviation and auto-covariance are the statistical properties that usually been checked (Kraft, C. H., 1967). For a time, series to be stationary, it should have a covariance that is not time dependent (analyticsvidhya, 2015). Below are examples of stationary and non-stationary time series (analyticsvidhya, 2015):

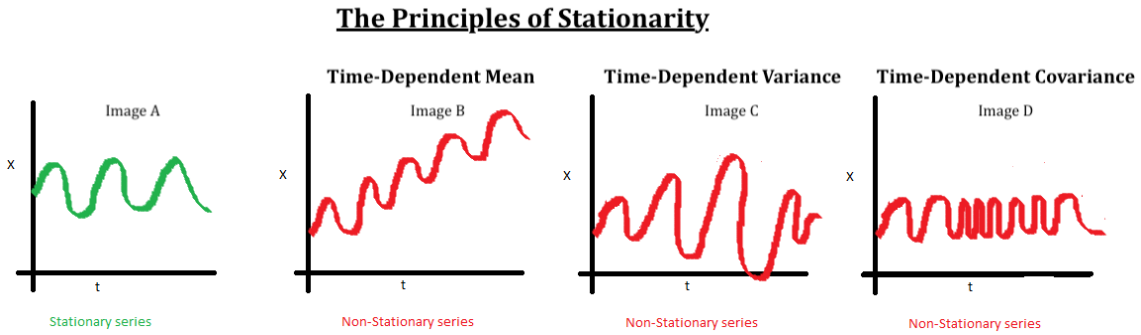

Figure 10 Stationary and non-stationary time series

Time series is called as non-stationary when the mean does not stay constant over time so the time series will show a certain type of behavior that called as trend. The trend might be upward, downward, negative or positive depending if the values of the time series increase or decrease over time.

Facebook prophet can deal with non-stationary data depending on what type of non-stationarity it is. Trend is non-stationary, so if a piecewise linear trend is enough to capture the non-stationarity then Facebook prophet will work well. Otherwise (including if the non-stationarity is in seasonality), the model may not fit the time series as expected (Facebook/Prophet, 2017).

The prophet model handles baseline trend changes so there's no stationary assumption. Used data set contains cases with stationary data and data with upward trend (non-stationary), most of tests' data set are stationary so the median of the metrics value does not change over the chosen observation time.

3.2.9 Correlation in time series

Timeseries features might be have a relation between each other, one variable might be a result to another variables, or just be of the variables are relevant, this relationship is called time series correlation.

To check the correlation within time series we took the numeric feature (execution_time and fail) for one of the tests(testX). We found out that there are no strong correlations between the testX's features.

	Execution_time	**Fail**
Execution_time	1.00000	0.00783
Fail	0.00783	1.00000

Table 7 Correlation between test's 2 features

3.2.10 Data seasonality

Another attribute of time series is seasonality, According to Aarshay it is defined as “regular changes that occur in time series at specific frequency of time. For example, the values of a time series could peak around the mid of the day and dip into low values around the mid of the night. The seasonality could have different frequency like hourly, daily, weekly, monthly, …, etc (AarshayJ.,analyticsvidhya, 2016).
Figure12 shows that data has no cycles neither yearly nor monthly seasonality. Upward trend was found in some test.

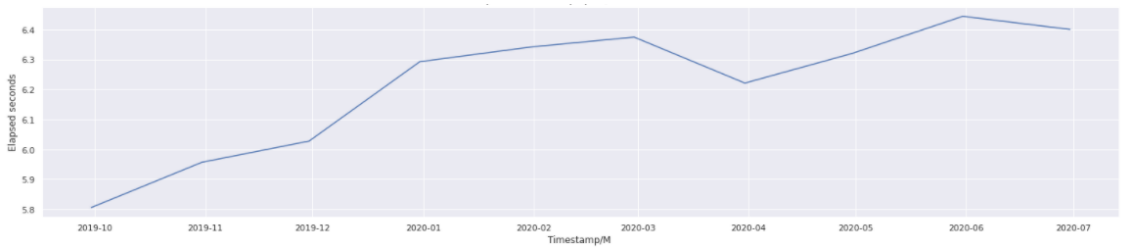

Figure 11 TestX elapsed seconds over time

As Figure 13 shows us there is no captured seasonality or correlation between tests within same category.

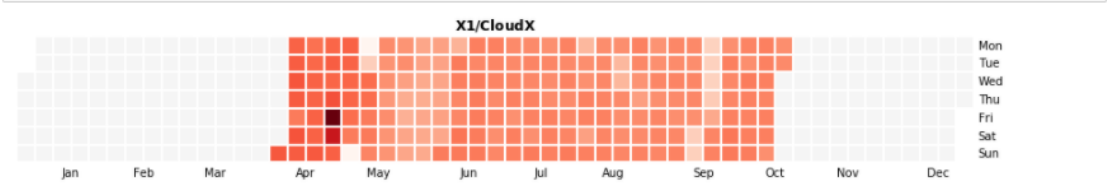

Figure 12 Calendar Heatmap for Tests in ComponentX

Daily seasonality was found for some test.

3.2.11 Data Distribution

Data distribution is a method or a way to list the existing values of the data variables, it is also useful and easy way is to represent data graphically so it would be easy to know how data looks like.

3.2.12 Normal Data Distribution

Normal distribution also called Gaussian distribution which means statically the data is distributed near the mean, the far values from the mean are less occurring and this distribution is symmetric about the mean (Walck, 1996). Normal distribution plot in Figure 14 shows the distribution have one peak looks like a bell curve.

Majority of study cases have normal execution time distribution and since Prophet uses a normal noise model, so the model can fit the data successfully.

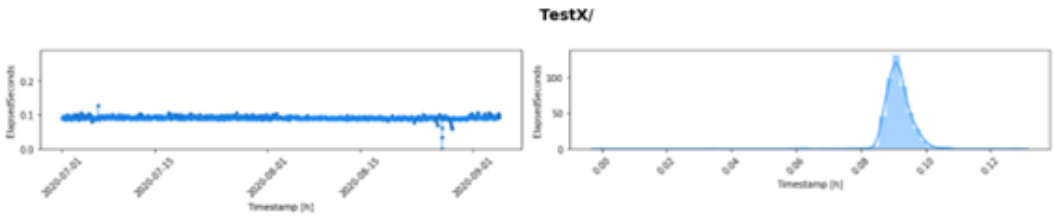

Figure 13 Normal Data Distribution for TestX

3.2.13 Bimodal Data Distribution

Bimodal distribution means statically the data can have two or more peaks
Bimodal data is a common situation; this distribution called a bimodal/mixture model.
We can notice from this use case example that generally execution time value falls into two categories:

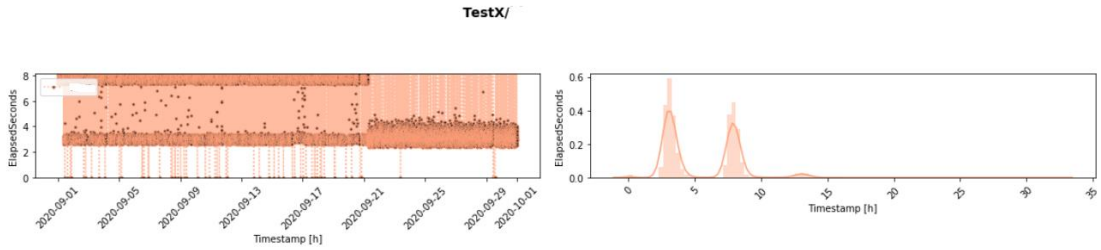

Figure 14 Bimodal Data Distribution for TestX

Figure 15 shows range of 2.8-3.5 sec (or very close to 3) or somewhere in the range of 7-9 sec. There are a couple of entries somewhere in the middle of 3-9, but the majority fall into one of these two categories, we can't consider these data points as outliers because they are the distribution is always between two range and the outlies is an extraordinary behavior in data which is not in our case.
Part of this study was to explore what are the right avenues to deal with this type of data distribution, we have tried two approaches:

1. Classify each data point into one of the two groups then run two separate prophet models on each of these groups this solution would be impossible in our case, that's because we are dealing with univariant time series not having any extra information or label that help us to classify and mark each data point is related to which right group. Figure15 shows data points distribute between two separate groups without an extra regressor that explains the difference between the two data groups.

2. Second approach would be to take the sample mean of values by considering Central Limit Theorem (CLT) probability theory.

3.2.13.1 Central Limit Theorem probability theory

This theory assumes that the sample mean will have approximately a normal distribution across large sample sizes, regardless of the distribution from which we are sampling.

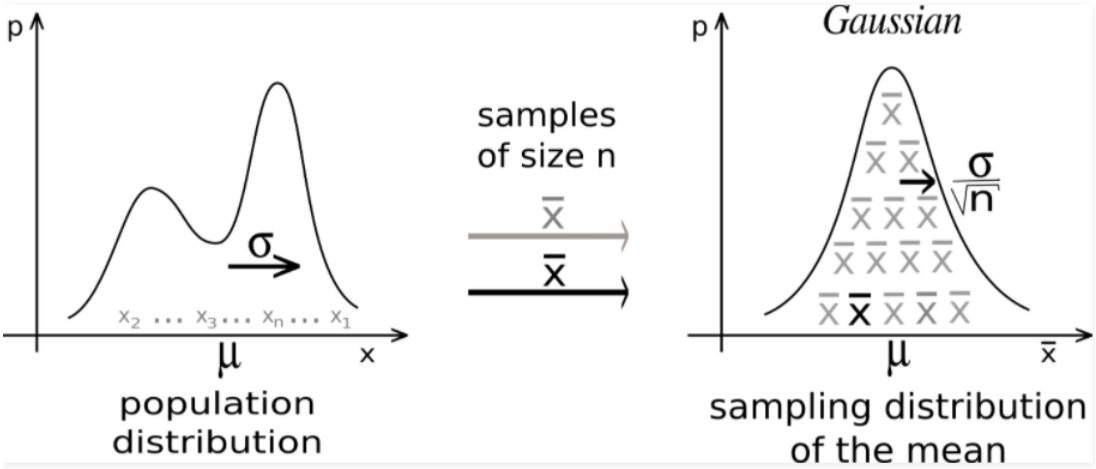

Figure 15 Central Limit Theorem

In the used data, it has been found a bimodal/mixture distribution of TestX timeseries (figuer15) which considered as population with an unknown density distribution function, it select a random sample of N observations and calculate their mean. By Repeating this procedure many times, data will be resembled as normal distribution then the time series can be applying our forecasting model pretty easily (statistics4u, 2010).

This approach is one of the possible solutions to reform the data distribution of time series from bimodal into normal distribution. After applying this approach over bimodal tests, this solution was successful and acceptable for part of tests but not for others.

3.2.14 Data Resampling

Data resampling is used in forecasting to rebalance the data, it contributes in the process of learning about data. (Moniz, N., Branco, P. & Torgo, L., 2017).

In this study, the data set interval is in second. Variance in the history should be reduced by aggregating hourly across days, as a result, forecasting model is expected to show a better forecast with lower uncertainty. In this study, median will be used over mean.

Mean and median are often described as descriptive statistics, in the table below we have been collected the characteristics of both statistic methods:

Median	Mean
1. The median is the middle score for a set of data that has been arranged in order of magnitude. 2. The median is less affected by outliers and skewed data.	1. The Mean is the result of a probability model over errors. 2. it is particularly susceptible to the influence of outliers.

Table 8 Median vs. Mean

Therefore, we'd use the median over the data.

3.2.15 Data Scaling

This type of processing is handled by prophet itself, prophet scale Y by its maximum (absolute) value.

4 Experiment

After a thorough investigation of the properties of the available dataset in chapter 3 and the explanation of the theory behind the Facebook prophet forecasting technique in chapter 2, this chapter represents the implementation of these techniques in the dataset. As with most datasets, there are mainly two types of problems: the missing values and the outliers.

In this chapter, a certain amount of noise was added for the result in order to keep the confidentiality of the original data

The result of the experiments will be into 2 parts:

1. Forecasting Test's execution duration over time for next 3 hours.
2. Detecting the anomalies in the historical data.

The following step must be done before applying the forecasting model:

1. Every time series is univariate and aggregated hourly.
2. Data continuity is ensured by recovering any missing values in the metrics.
3. Calculating the median or mean of time series depending on time series distribution as we discussed in chapter 3.

Let's make the following brief work break down:

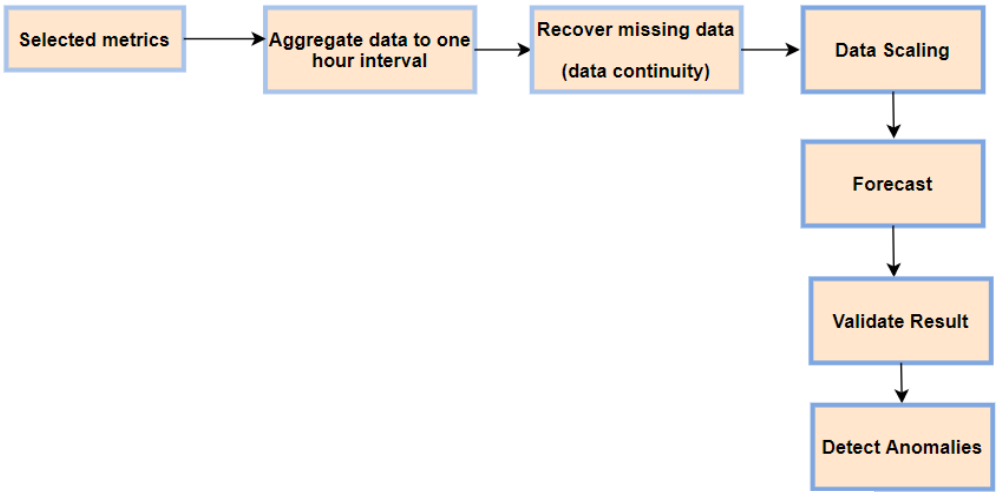

Figure 16 Proposed forecasting model process

4.1 Forecasting with prophet

Our goal is to calculate an effective forecasting with acceptable accuracy, so we build a model using historical data to forecast test's execution time, actual data would have predicted by using the following model:

$$y(t) = trend(t) + weekly\_seasonality(t) + noise \quad (3)$$

By consider that during data analyzing we have not detected any yearly or monthly seasonality, which was the main reason to limit the minimum length of observation horizon to 2-3 months. When prediction is made on the history as is done by default within the prophet model, the model forecasted values on historical data points and these data were used to fit the model. They thus do not provide an accurate assessment of the model's ability to forecast. That's because the model can do a much better job at predicting values for which it already knows the true values. The model is applied over 6-month data, prediction result will be test's execution time for next 3 days.

In both below examples, Prophet catches the trends and most of the time gets future values right.

Prophet prediction is a data frame, where target prediction sits in 'yhat' column, prediction interval bounded by 'yhat_lower' and 'yhat_upper', the rest columns contain predictions of trend, additive and multiplicative terms and copy of features that were argument in the prediction method. The black dots show the actual y values that we gave as the input data, the dark blue line is yhat. The light blue at the top is yhat_upper, and the light blue at the bottom is yhat_lower, we can notice outliers will exist outside prediction boundaries.

Tested data has an hourly frequency, Prophet will model the structure of trend, daily and weekly seasonality, yearly seasonality has been turned off. Prophet provides us with helpful feature is model component visualization.

Let's have a look forecast and component visualization looks like:

4.1.1 Example 1

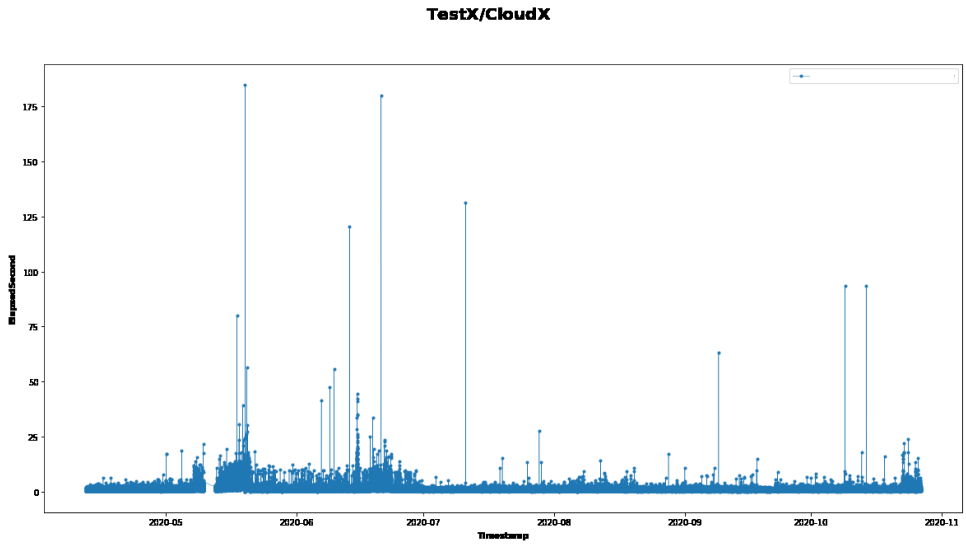

Figure 17 TestX timeseries

By looking closely at time series in figure 18, it displays TestX execution time against time, TestX data set for 6 months, execution time might take longer duration sometimes, and this abnormal behaviour could be an anomaly.

Below are examples of prophet forecasting result for TestX.

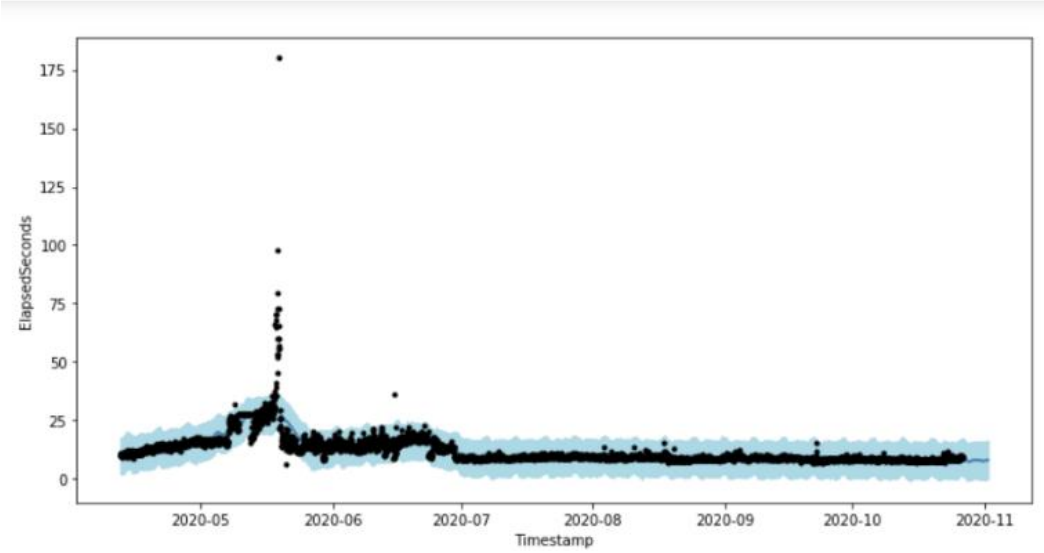

Figure 18 Prophet to forecast elapsed seconds of TestX for the next 3 hours.

Figure19 shows forecasting results, dark blue line is forecasting spend numbers, black dots are actual execution time values. The light blue shade is 95% confidence interval around the forecast. TestX has an increase in execution time between May and first June otherwise TestX seems to be stable. Prophet fits the model to data well.

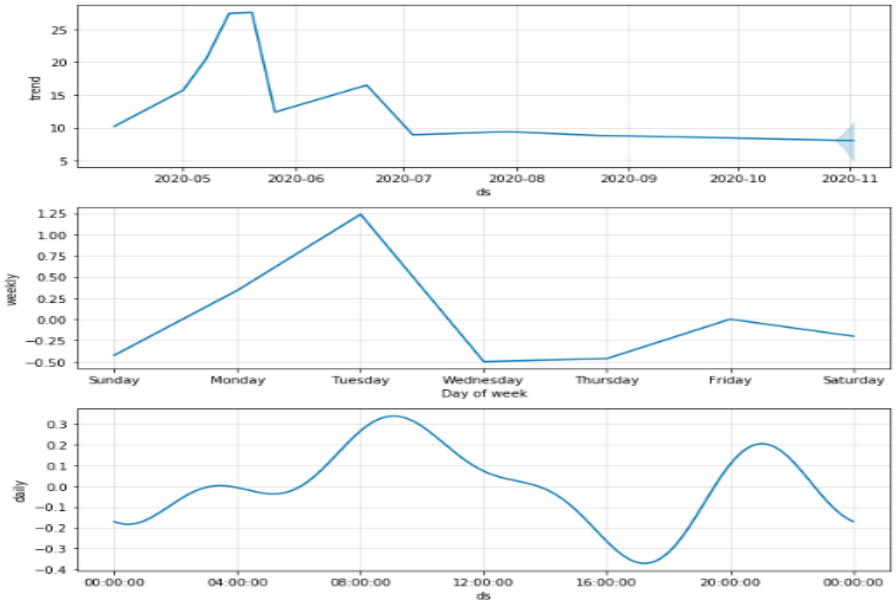

Figure 19 Prophet component plot of TestX

Figure 20 shows that there are a few clear takeaways such as higher activity during is on Tuesdays or less activity on Wednesday & Thursday, usually morning hours TestX have higher execution time. Additionally, it appears to have bigger jumps in execution duration towards morning period of the day.

4.1.2 Example 2

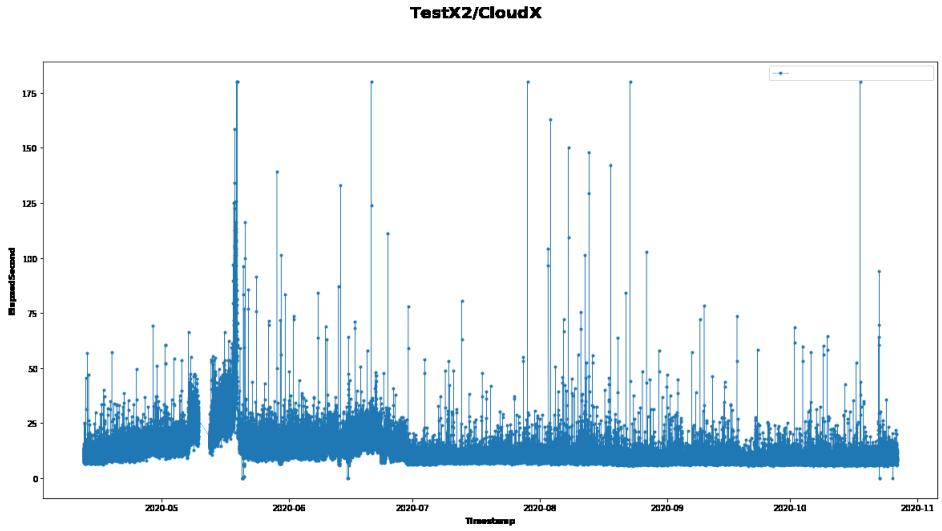

Figure 20 TestX2 time series

By looking closely at time series in figure 21, it displays TestX2 execution time against time, TestX2 data set for 6 months, execution time might take longer duration sometimes, and this abnormal behaviour could be an anomaly.

Below is an example of prophet forecasting result for TestX2:

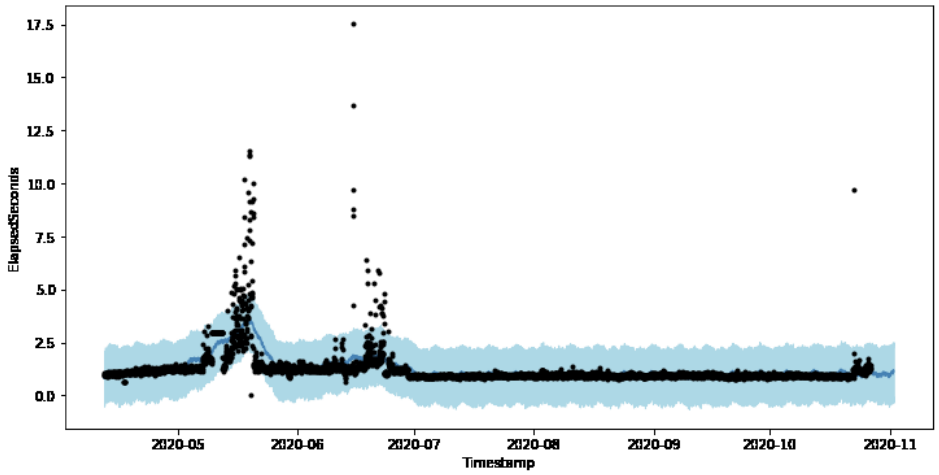

Figure 21 Prophet to forecast elapsed seconds of testX2 for next 3 hours

Figure22 shows forecasting results, dark blue line is forecasting test's elapsed seconds, black dots are actual execution time values. The light blue shade is 95% confidence interval around the forecast. TestX2 has 2 slight peaks in May and July, otherwise TestX2 seems to be stable. Prophet catch data trend very nicely and fits the model to data well.

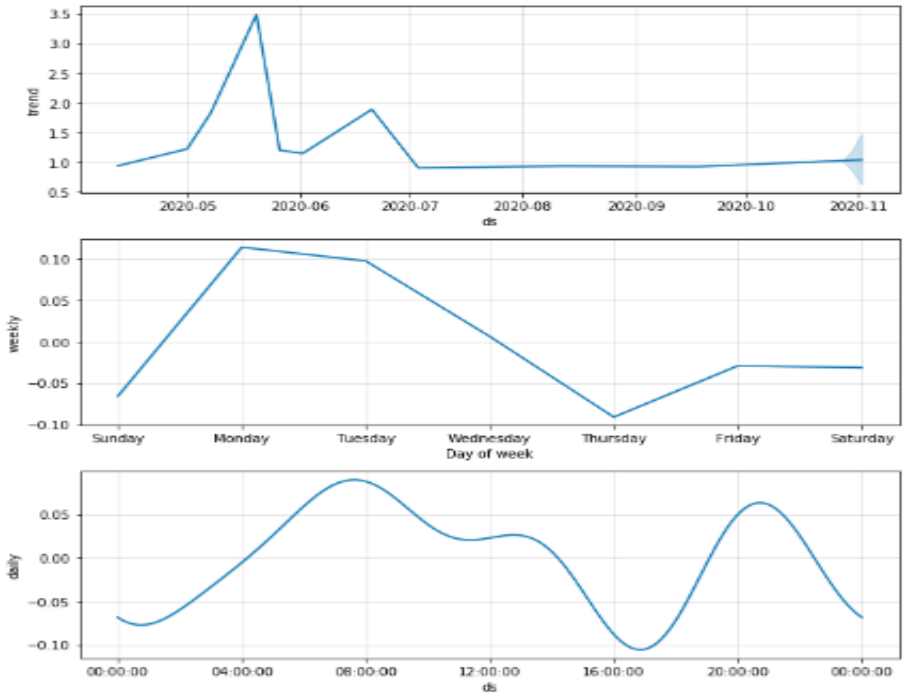

Figure 22 Prophet component plot

Figure 23 shows that the execution time starts to increase from Sunday to Wednesday, it reaches the peak on Monday. Tuesday morning test execution time starts to decrease again. TestX2 has big fall offs on Thursday.

4.2 Cross Validation

Cross validation is mainly applied in machine learning to give an estimation about model's performance and skill on unseen data. It simulates how the model will predict on data that's not used before in making prediction. Model results should be verified in order to be used, so we should find a way we can check how accurate the model is, it could be either by eye or with error metrics such as Mean Absolute Error (MAE) measures the distance of predicted values from the observed value, MAPE (Mean Absolute Percent Error) calculates the average value of the percentage error, Root Mean Square Error (RMSE) is the value of standard deviation residuals value (errors of the prediction), it shows how residual values are expanded. Residuals are is the difference between the observed value and the mean value that the model predicts for that observation and adjust accordingly. Cross validation automates the previous steps.

Cross validation function for timeseries has included in Prophet to calculate forecast error using historical data.

The first parameter of the function is our trained model not the observation data, the next parameter is the prediction horizon which means how frequently we want to predict, then an optional parameter: initial, which express how long to train before starting the tests, and finally period parameter which means how frequently to stop and do a prediction. In case of not feeding the function with the initial, Prophet will assign defaults of initial = 3 * horizon, and cutoffs every half a horizon, that will lead to have a long running series of validations, each time predicting forward and calculating the error using performance_metrics, so Basically we just need to run this method in a loop for a range of values of horizon, and then compute the MAPE for all of the results for each value of horizon.

After looking at the output of cross_validation data frame (df_cv), ds column is the timestamp of the point being predicted, true values y, the out-of-sample forecast values yhat, and cutoff that is the cutoff with which that prediction was made.

In the following example, cross validation is used to assess prediction performance on a horizon of 3 days, starting with 10 days of training data in the first cut-off and then the predictions is made every 4 days. Blow figure shows part df_cv data frame.

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	2020-07-12 00:00:00	389.807773	309.125235	475.737722	306	2020-07-11 23:00:00
1	2020-07-12 01:00:00	396.308295	316.856368	483.150660	332	2020-07-11 23:00:00
2	2020-07-12 02:00:00	352.647003	272.760219	432.772391	300	2020-07-11 23:00:00
3	2020-07-12 03:00:00	351.011822	271.733821	430.581355	306	2020-07-11 23:00:00
4	2020-07-12 04:00:00	344.367558	260.667334	426.953362	301	2020-07-11 23:00:00

Figure 23 Cross validation dataframe

In particular, cross validation is used for computing error metrics using performance_metrics function (FacebookProphet, 2020) and its data frame can be used to compute error measures of yhat vs. y, as we will explain in the next section.

4.3 MAPE diagnostic

In this case study performance_metrics is used as utility to compute the Mean Squared Error, Root Mean Squared Error,Mean Absolute Error, Mean Absolute Percentage Error(MAPE) and the estimation coverage of the yhat_lower and yhat_upper values.
MAPE method calculates the outputs accuracy of individual forecasting for each hour which is produced by the Prophet forecast model, the mean absolute percentage error (MAPE) is calculated as following:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y^{i}_{forecast} - y^{i}_{true}}{y^{i}_{true}}\right| \cdot 100\% \quad (4)$$

When Prophet is computing the performance metrics (like MAPE here in particular), it computes the APE for every row in df_cv, and then computes MAPE by averaging over a rolling window based on the horizon (df_cv['ds'] - df_cv['cutoff']).

	horizon	mse	rmse	mae	mape	mdape	coverage
0	08:00:00	13.769178	3.710684	1.996601	0.166634	0.087715	0.936170
1	09:00:00	13.647834	3.694298	1.997844	0.164542	0.089926	0.933778
2	10:00:00	13.982713	3.739347	2.053749	0.168267	0.094022	0.930820
3	11:00:00	13.399229	3.660496	2.044589	0.164701	0.097436	0.935604
4	12:00:00	13.178240	3.630185	2.042087	0.161758	0.101093	0.942087

Figure 24 Performance metrics for TestX1

Forecasting results are excellent in this case, because that the MAPE value is less than 10%.

The performance metrics is visualized as it shows in the following plot, performance metrics that we are generating is computing the metrics with moving averages with a window size of 10%, and this is what is plotted in the blue line. The maximum of the blue line is the 0.83 that is the max in the table as shown in Figure 26, gray dots come from doing the same calculation with a rolling window of 0 which is no averaging at all, so each represents the absolute percent error (no mean involved) for each pair (predicted value y_hat, observation value y) that was forecasted during the cross validation (FacebookProphet, 2020)

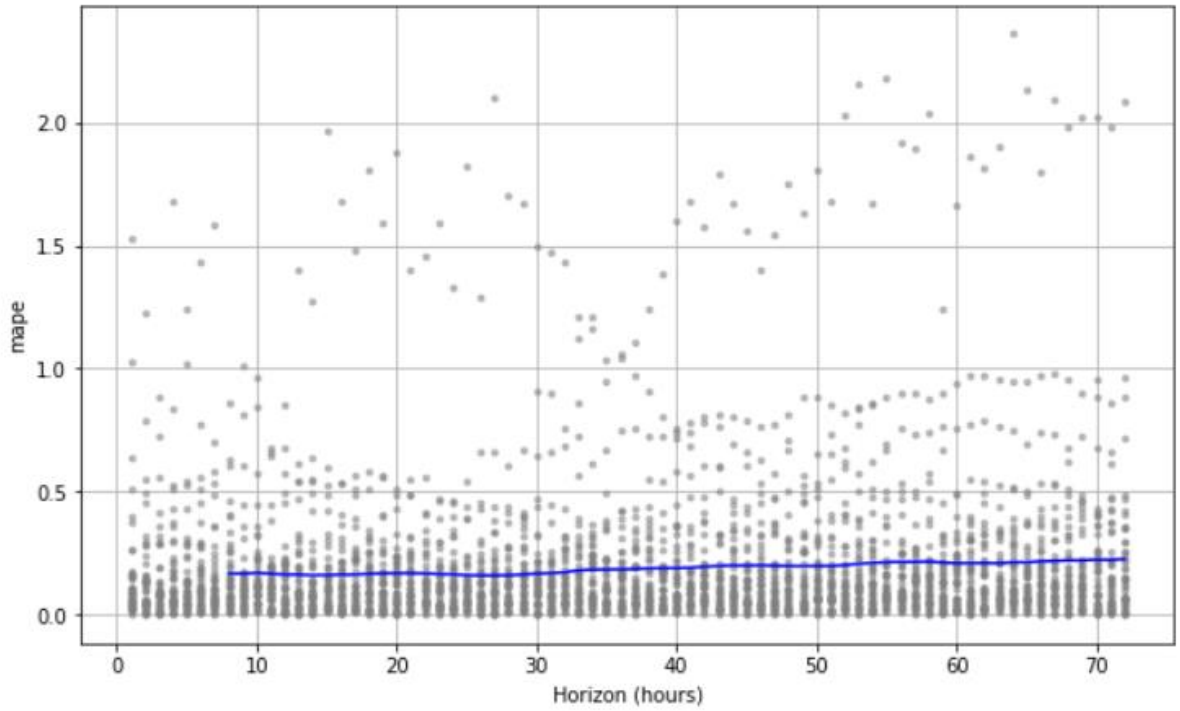

Figure 25 Performance Metrics Visualization for TextX1

4.4 Changepoints

Changepoints is method to detect the trend when data is changing. Facebook prophet can automatically detect changepoint, by default prophet can produce grid of potential changepoints as it is shown in figure 27.

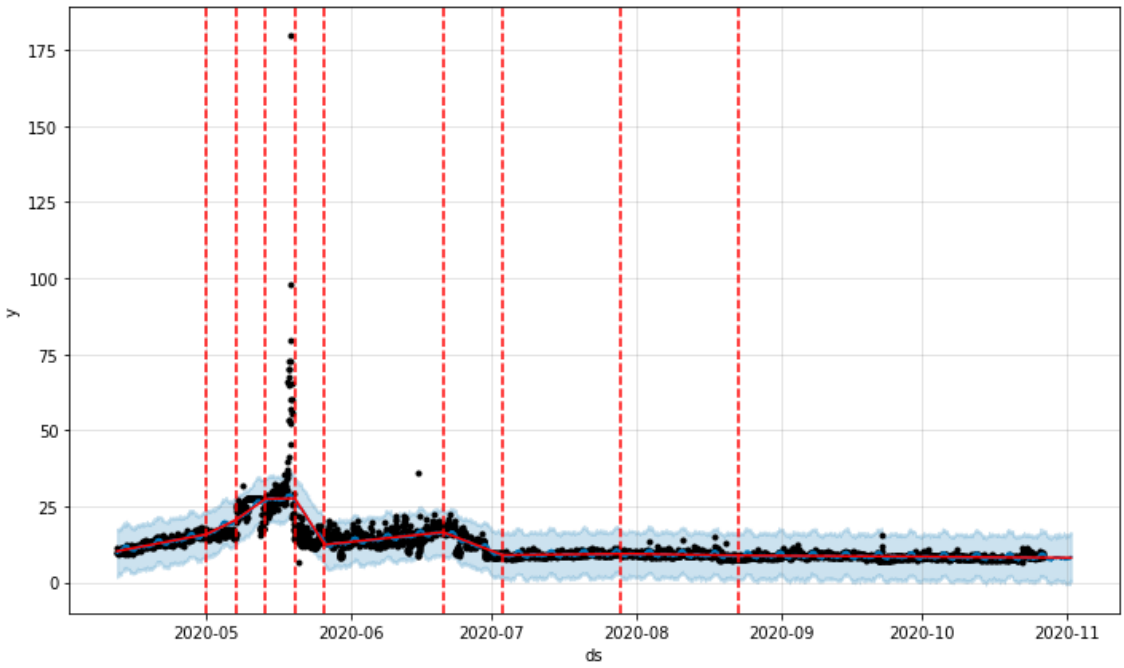

Figure 26 Change points in TestX1 captured by Prophet

All the measured changepoints in data are included in the model whether these changes are significant or not. Prophet will fit into the model only the changepoint that have significant values, and this is called automatic changepoint selection.

Facebook prophet has a nice feature which allow the user to list custom changepoints. Custom changepoints are treated in the same way as before: All of the changepoint whether are customed or detected by Prophet are included in the model, but the actual change at any of them will be fitted.

In this study there isn't any specified any custom change points, in order to specify a custom change point all you need to pass all of these into Prophet with the changepoints argument, it will do the automatic selection on that grid as usual, plus our added point. Monitoring team can easily do change points analysis to understand how data interact with changes in clouds.

4.5 Anomaly

Data point has been plotted against time to find outliers, and as we can see in the plot below, the anomaly is colored with red.

Any value between yhat_upper and yhat_lower boundaries is considered as a normal behavior. The Anomaly will be any value(y) is greater than yhat_upper and less than yhat lower.

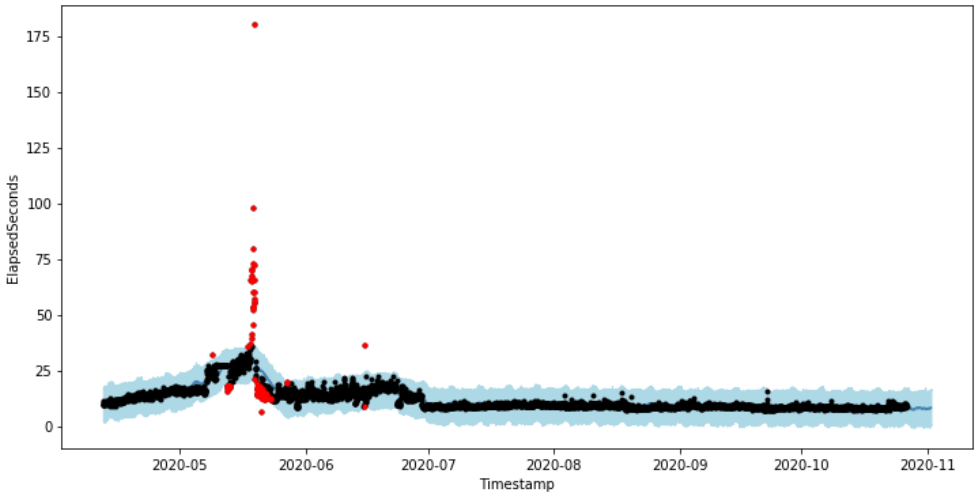

Figure 27 Detected Anomalies in TestX1

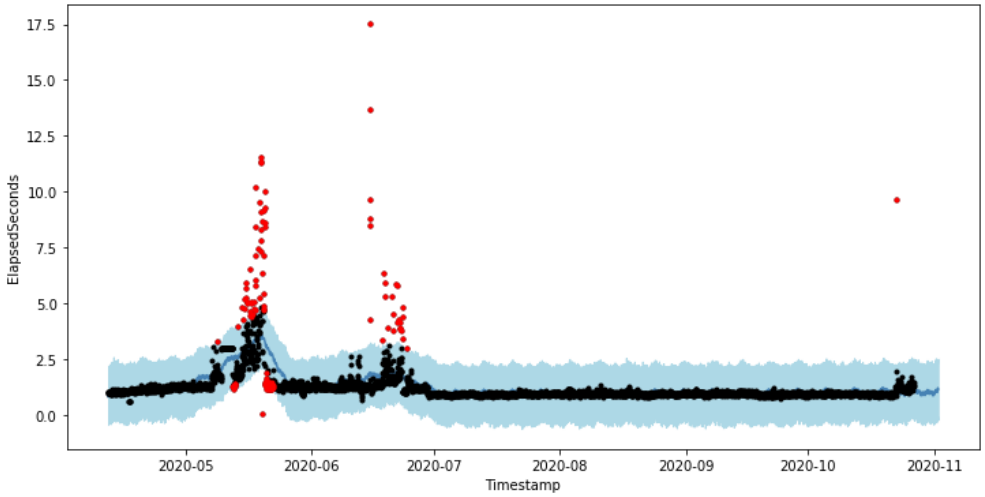

Figure 28 Detected Anomalies in TestX2

4.6 Failing Scenario

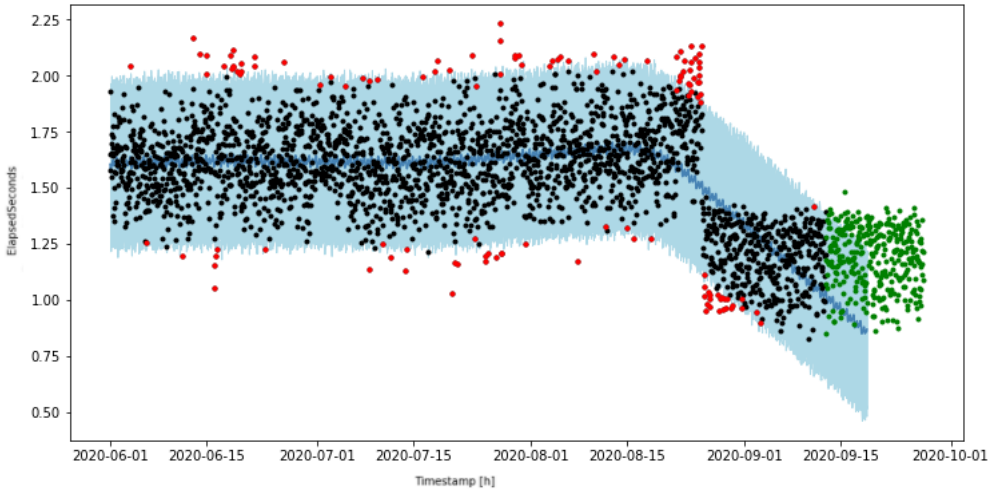

Figure 29 Prophet forecast underfit

As it was mentioned previously, our data has an hourly frequency, Prophet will model the structure of trend, seasonality, and any specified. The chart in figure 27 shows that data trend is the only thing that has been captured now; there's an obvious weakness in seasonality effect modeling, so basically, it is non seasonal data, Since there is no clear seasonality that we can see in this test's time series, The light blue shade outlines the confidence interval around the forecast 95%. As the Figure 28 shows, the model tries to fit all data point smoothly, but it fails to catch any seasonality.

5 Conclusion and future improvements

5.1 Conclusion

What has been done in this study can be considered as first step toward building forecasting and anomaly detection framework, data was analyzed in such a way that helps the monitoring team look to the dataset from another aspect.

In this study, and particularly with huge our data set, it is concluded that anomalies might happen in test's time series, however it does not necessarily mean that this anomaly reflects a problem in test execution. The key focus in this case is the following:

1. Understand each test's nature and know what the aim of running the test case is.
2. Understand all the circumstances led to cause the abnormal observation that caught by the anomaly detector, then it will be decided case by case whether this behavior is really an anomaly or not.
3. Abnormal behavior is not necessarily considered an anomaly, cloud upgrade is a regular task and it may cause a fluctuation execution time.

As a result, understanding where the anomalies happened might be a good indicator to the team to detect and analyze all the actions that are happing in the cloud. Data variables should be studied in depth in order to help in identifying anomalies and that happens simply by using visualizing methods, it helps to provide the user with information about abnormal data points. Finding anomalies could be challenging by using Z-score, it assumes that data follows normal distribution, however, this simple method has been used to show abnormal values as a lead values need further investigation. It's just the first step to deal with anomalies. Prophet has done good job in forecasting for most of test cases, but in other hand, there are some tests that Prophet failed to fit the model using their data. However, prediction results are represented with a confidence interval around the forecast, which can often be more helpful than the forecasted data values itself when the team make decisions about how the test may perform for near future. Prophet supposed that the seasonality independent of each other. That is, the daily seasonality (seasonality within one day) is the same every day, the hourly/daily seasonality is stronger on some days than on others for some tests. Having interactions, there is a solution there, but it is still needed to do more studies to find the perfect model tuning for some test cases. Facebook prophet is an acceptable model for part of our test's data, but it did not fit well for other challenging metrics that it did not fit with the current model.

5.2 Future improvement

All research questions asked in chapter 1 have been answered in this thesis. However, the models presented can possibly be improved.
Suggestions for possible improvement would be:

- Including cloud upgrade log data to tune Facebook prophet with extra regressor and adjusting trend flexibility.
- Predict different feature other than test's execution time
- Forecast multivariant time series.
- For Z-score could use a different lag value for.

6 References

AarshayJ.,analyticsvidhya, 2016. *A comprehensive beginner's guide to create a Time Series Forecast.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python
[Accessed 01 10 2020].

Aggarwal, C. C., 2016. *Outlier Analysis Second Edition.* second ed. s.l.:Springer.

analyticsvidhya, T. S., 2015. *A Complete Tutorial on Time Series Modeling in R.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/
[Accessed 10 09 2020].

Anon., 2014. *Numerical Optimization: Understanding L-BFGS. Limited-memory bfgs,.* [Online]
Available at: http://aria42.com/blog/2014/12/understanding-lbfgs
[Accessed 12 09 2019].

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell, 2017. Stan: A Probabilistic Programming Language. *Stan: A Probabilistic Programming Language.*

B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell, n.d. *Stan: A probabilistic.*

Bourg, D. M. & Seemann, G., 2004. *AI for Game Developers.* Sebastopol(California): O'Reilly Media.

Byrd, R. H., Lu, P. & Nocedal, J., 1995. A limited memory algorithm for bound. *SIAM Journal on Scientific and Statistical Computing,* p. 1190–1208.

CGIFinland, 2019. *Verkoista pilveen.* [Online]
Available at: https://www.cgi.fi/fi/ratkaisu-lehti/2-2019/verkoista-pilveen
[Accessed 17 November 2020].

cloudera, 2017. *Taking Prophet For A Spin..* [Online]
Available at: https://blog.fastforwardlabs.com/2017/03/22/taking-prophet-for-a-spin.html
[Accessed 17 10 2020].

En.wikipedia.org., 2020. *Multicollinearity.* [Online]
Available at: https://en.wikipedia.org/wiki/Multicollinearity
[Accessed 22 10 2020].

Facebook/Prophet., GitHub., 2020. *GitHub. 2020. Should "Ds" Be Sorted (In Order Wrt Time)? · Issue #1412.* [Online]
Available at: https://github.com/facebook/prophet/issues/1412
[Accessed 26 10 2020].
Facebook/Prophet, 2017. *GitHub. 2020. Stationary And Non-Stationary Timeseries · Issue #234.* [Online]
Available at: https://github.com/facebook/prophet/issues/234
[Accessed 21 10 2020].
Facebook, P. F. A. S. -., 2020. *Prophet.* [Online]
Available at: https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale
[Accessed 2020].
FacebookProphet, 2020. *Diagnostics.* [Online]
Available at: https://facebook.github.io/prophet/docs/diagnostics.html
[Accessed 30 10 2020].
Greene, W. H., 2012. *Econometric Analysis.* 5 ed. New york: Pearson Education, Inc., Upper Saddle River,.
Hastie, T. & Tibshirani, R. , 1987. Generalized additive models: some applications. *Journal of the American Statistical Association 82(398),* p. 371–386..
Hawkins, D., 1980. *Identification Of Outliers..* London: Angleterre: Chapman and Hall..
J. Gustafsson, F. Sandin, 2016. *12 - District heating monitoring and control systems.* Luleå: Woodhead Publishing.
Kraft, C. H., 1967. *Stationary and related stochastic processes : sample function properties and their applications, by Harald Cramer and M.R. Leadbetter.* Canadian Mathematical Bulletin, Cambridge University Press11 ed. New York: John Wiley and Sons.
Moniz, N., Branco, P. & Torgo, L., 2017. Resampling strategies for imbalanced time series forecasting. Int J Data Sci Anal 3. *Resampling strategies for imbalanced time series forecasting.,* p. 161–181.
P.,Kiselev., R., 2020. *Peak Signal Detection In Realtime Timeseries Data.* [Online]
Available at: data, P. and Kiselev, R., 2020. Peak Signal Detection In Realtime Timeseries Data. [online] Shttps://stackoverflow.com/questions/22583391/peak-<signal-detection-in-realtime-timeseries-data>
[Accessed 2020 10 01].
Portnoy, L., 2000. Intrusion detection with unlabeled data using clustering (Doctoral dissertation, Columbia University).

Prophet, 2020. *Uncertainty Intervals.* [Online]
Available at: https://facebook.github.io/prophet/docs/uncertainty_intervals.html
statistics4u, 2010. *Fundamentals of Statistics.* [Online]
Available at: http://www.statistics4u.info/fundstat_eng/cc_central_limit.html
[Accessed 20 10 2020].
Steven L. Scott and Hal Varian., 2014. *Predicting the present with bayesian structural time series. International Journal of Mathematical Modelling and Numerical Optimisation,* p. 4–23.
T. J. Hastie and R. J. Tibshirani, 1990. *Generalized additive models.* London: Chapman & Hall.
Taylor, S. and Letham, B, 2018. Forecasting at Scale. *Forecasting at Scale. The American Statistician.*
V. Chandola, A. Banerjee, and V. Kumar., 2009. Anomaly detection: A survey. ACM computing surveys (CSUR). *Anomaly detection: A survey. ACM computing surveys (CSUR),* p. 41(3):15.
Walck, C., 1996. *Hand-book on.* Stockolm: Particle Physics Group.