LAPPEENRANTA-LAHTI UNIVERSITY OF TECHNOLOGY LUT

School of Engineering Science
Industrial Engineering and Management
Business Analytics

Joni E. Kettunen

Anomaly detection in business metric monitoring

Master's thesis

Examiners: Professor Pasi Luukka
Docent D.Sc. (Tech.) Jan Stoklasa

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Industrial Engineering and Management

Joni E. Kettunen

Anomaly detection in business metric monitoring

Master's thesis
2021
91 pages, 24 figures, 4 tables
Examiners: Professor Pasi Luukka and D.Sc. Jan Stoklasa

Keywords: Anomaly detection, Timeseries modelling, Business metrics

In the digitalizing world, the amount of data transferred exceeds the human ability to study it manually. This is also the case for business metrics since data volume and the number of metrics to monitor is rapidly increasing. The era of big data raises requirements for new methodologies and tools that can be used to leverage expert opinion in the decision-making progress. Anomaly detection is one of the most common data analysis methods that can be used to get useful and often critical actionable insights from the big data.

This study delivers a clear picture of the current state of the art in business metrics anomaly detection. First anomalies in business metrics are explained as a phenomenon and literature review is conducted on the methodologies for finding anomalies as well as for existing solutions that are used to monitor business metrics at scale. Based on the findings an anomaly detection tool was developed to monitor business metrics organization wide.

The findings of the literature review as well as evidence from the implemented automated anomaly detection tool indicate that simple timeseries modelling methods for anomaly detection can provide significant business value, when used in large scale. Primary challenge in an automated business metric anomaly detection tool is the human factor in scalability which can be overcome with an intuitive user interface and fine selection of anomaly detection algorithms. The results of this study can be leveraged across industries in every company where the number of business metrics has overpassed the analyst's capability to monitor each one separately.

Tiivistelmä

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tuotantotalouden koulutusohjelma

Joni E. Kettunen

Poikkeamien tunnistaminen liiketoimintametriikoissa

Diplomityö
2021
91 sivua, 24 kuvaa, 4 taulukkoa
Tarkastajat: Professori Pasi Luukka ja TkT Jan Stoklasa

Hakusanat: Poikkeamien tunnistaminen, aikasarjamallinnus, liiketoiminta metriikka

Digitalisoituvassa maailmassa siirretyn datan määrä ylittää ihmisen kyvyn analysoida sitä manuaalisesti. Liiketoimintametriikoiden tapauksessa datan volyymi ja metriikoiden määrä lisääntyy nopeasti. Massadatan aikakausi asettaa uusia vaatimuksia työkaluille ja menetelmille, joita käytetään asiantuntijoiden ilmiöalueosaamisen hyödyntämiseen päätöksenteossa. Poikkeavuuksien tunnistaminen on yksi yleisimmistä data-analyysi työkaluista, jota voidaan käyttää antamaan hyödyllisiä ja usein kriittisiä oivalluksia massadatasta.

Tämä tutkielma antaa selkeän kuvan moderneista menetelmistä liiketoimintametriikoiden poikkeavuuksien tunnistamisessa. Aluksi poikkeavuudet liiketoimintametriikoissa esitetään ilmiönä ja kirjallisuuskatsausta hyödynnetään löytämään metodeita poikkeavuuksien tunnistamiseen yksimuuttujaisissa aikasarjoissa. Kirjallisuuskatsausta näistä metodeista sekä tutkielmia aikaisemmista toteutuksista poikkeavuuksien tunnistamisessa suuressa mittakaavassa hyödynnettiin kehittämään työkalua, jota voidaan käyttää liiketoimintametriikoiden monitorointiin koko organisaation laajuudessa.

Kirjallisuuskatsauksen löydökset ja käytännön opit kehitetystä automaattisen poikkeavuuksien tunnistamisen työkalusta osoittavat, että poikkeavuuksien tunnistaminen voi antaa merkittävää lisäarvoa liiketoiminnalle, kun työkalua käytetään suuressa mittakaavassa. Päähaaste automaattisessa poikkeavuuksien tunnistamisen työkalussa on inhimilliset tekijät ratkaisun skaalautuvuudessa, jotka voidaan ratkaista intuitiivisella käyttöliittymällä ja oikealla valikoimalla poikkeavuuksien tunnistamis algoritmeja. Tämän työn lopputuloksia voidaan hyödyntää eri toimialoilla kaikissa yrityksissä, joissa liiketoimintametriikoiden määrä on ylittänyt analyytikoiden kyvyn monitoroida jokaista metriikkaa erikseen.

ACKNOWLEDGEMENTS

As tradition goes this the time and place to express my gratitude to the group of people who have supported me in writing this thesis.

First, I would like to thank Rovio Analytics lead Henri Heiskanen for granting me the opportunity to write this thesis and giving me free hands to construct it as I best see fit. I wish to also thank Asko Relas for supporting and giving feedback as well as Juho Autio for the technical guidance across analytics projects. I would also like to extend my thanks to all my co-workers at Rovio who have made writing this thesis possible, contributed to the project or helped me any other way along the way. Kudos!

Special thanks to my friends and family for the support they have given me through the process. A huge credit goes to the people I got the chance to study with for the inspiration of new ideas and peer learning. Lastly, I wish to thank LUT for competent education and my instructor for the feedback and guidance regarding this thesis.

Espoo, May 2021

Joni Kettunen

Table of Contents

List of figures

List of tables

ABBREVIATIONS

KPI	Key performance indicator
BI	Business Intelligence
ML	Machine learning
NN	Neural Network
UI	User interface

1 Introduction

The purpose of this chapter is to give the reader brief overview of the problem statement behind the thesis and formulate the issue into several research questions. The chapter starts with explaining the problem background, moves on to describe the problem & research questions in detail and explain how the problem is addressed in the following chapters.

1.1 Background

This thesis was written to get background information from academic literature which could be used to provide insights on one of the ongoing data analytics projects Rovio (company) is currently working on. The results and conclusions of the thesis may also contribute to academic research and be useful knowledge for organizations facing similar challenges.

Before progressing to stating the problem, it is necessary to briefly introduce Rovio as a company. Rovio Entertainment Corporation is a global mobile-first games company that creates, develops, and publishes mobile games, as well as licenses the Angry Birds brand for consumer products, movies, animations and other entertainment (Rovio, Rovio Investors, 2021). Rovio uses data analytics widely across corporate functions for causes such as improving player experience and game monetization. Rovio's data strategy in a nutshell is explained in the code of conduct: "*We marry data with creativity*" (Rovio, Rovio code of conduct, 2021).

Rovio had concepted idea about a tool that could be used to monitor data for abnormal behavior in business metrics. The tool should be tightly integrated with the existing BI platform and be able to connect to various data sources that exists in the company. Potential end users of the tool would work in the game teams, marketing or user acquisition departments. The tool is not intended to replace any of the existing monitoring that is built into many of the cloud-based services that are used, but instead as a tool to include abnormal behavior monitoring to any timeseries based variables in the analytics pipeline. Similar monitoring was already in place as a custom-made solution in many places, so having a single service that could be used to monitor for abnormal behavior (anomalies) in multiple solutions could provide the following benefits:

1. Easily available tool may lead to increased monitoring of anomalies since it is easier to implement.
2. Keeping track of anomalies in business metrics in a single place may make it easier to see correlations between anomalies
3. Lower the response time to possible incidents, if the incident is related to data that did not have earlier abnormal behavior monitoring enabled
4. Increased data quality. The same tool could be used to monitor for data quality-based metrics. An anomaly in business metric may be a real value or an issue in the data pipeline.

The users of the tool can work in analyst or project manager roles, so the tool should not be only a service with endpoints, but it should have an intuitive user interface that can be used to set up monitoring of anomalies for specific timeseries. Developer with more knowledge on timeseries modelling and the underlying data lineage could use the tool for more specific use cases. Figure 1 shows both use cases of anomaly detection as a black box and white box service.

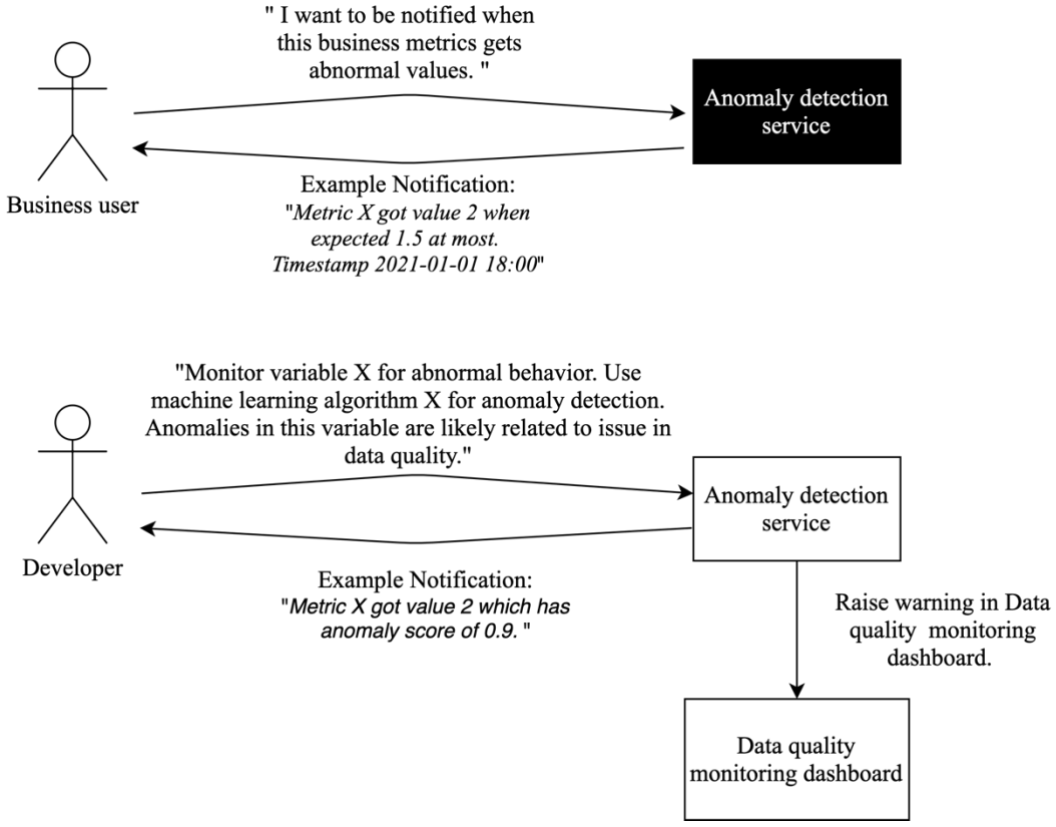


Figure 1. Use case scenarios of anomaly detection system

Anomaly detection

Anomaly detection has been an active research topic for a long time in the fields of statistics and machine learning. In digitalizing world, the amount of data transferred exceeds the human ability to study it manually. Therefore, automated data analysis has become a necessity. One of the most common data analysis tasks is the detection of abnormal behavior in the data which can be defined as anomaly detection. In statistical terms, anomalies are data points which deviate significantly from the normal distribution of the dataset and anomaly detection is the methodology to find them. (Mehrotra, Mohan, & Huang, 2017) The statistical definition for anomaly however is unable to capture some of the anomalies that may be present in a timeseries, more concrete definition for anomaly in timeseries is given in chapter 2.

The earliest studies in anomaly detection using statistical measures date back to the 19th century (Edgeworth, 1887). Over time more advanced methods are developed and in many cases the methods are specific to the type of the data analyzed. More advanced statistical anomaly detection methods have appeared and in recent years the increase in computational power has led to development and wide usage of machine learning based anomaly detection methods. In last decade various deep learning based methods have provided very successful results. (Munir, Siddiqui, Dengel, & And, 2019)

Anomaly detection can provide significant value in various industries. Detecting anomalies in data can translate to significant and often critical actionable insights in a wide variety of application domains. Anomaly detection can be performed for old data in a retrospective manner, or for new data that is currently being produced. This way anomaly detection can answer questions of “What should have been done” or “What should be done” depending on the type of actions that found anomalies cause. If the found anomaly is deemed significant and it happened in the past this might require larger set of actions. For example, anomaly detection can be used for intrusion detection in computer networks which requires finding anomalous patterns in the user activity history (Patcha & Park, 2007). In this case the access for the malicious user should be blocked, and the possibility for further security issues and data leaks should be studied. In many cases anomalies are monitored in real time, and the found anomalies require immediate response. For example, detecting cardiac (heart) abnormalities with

electrocardiogram (ECG) is a scenario where found anomaly requires immediate actions (Chuah & Fu, 2007). Digital adoption in various industries has resulted in data moving at increased speed and volume which has demanded improvements in anomaly detection practices. In many cases where anomaly detection was previously done in retrospective manner the requirements have changed and more anomaly detection applications require operational decision making in real time. (Anandakrishnan, Kumar, Statnikov, Faruquie, & Xu, 2017) For example, credit card transactions were monitored for anomalies which could indicate identity theft (Aleskerov, Freisleben, & Rao, 1997). In recent days, a suspicious transaction can be blocked and tracked in real time which has become possible with improvements in the credit card fraud detection techniques utilizing big data (Tran, et al., 2018). Table 1 contains several case studies where anomaly detection is used successfully to get actionable results.

Table 1. Example applications of time series anomaly detection

Anomaly detection application domain	Sources
Security	(Patcha & Park, 2007), (Lane & Brodley, 1997)
Healthcare	(Chuah & Fu, 2007), (Goldberger, Amaral, Glass, & Hausdorff, 2000)
Finance	(Aleskerov, Freisleben, & Rao, 1997), (Ahmed, Mahmood, & Islam, 2016), (Tran, et al., 2018)

1.2 Research objectives and scope

The objective of this study is to provide insights to the overall process of anomaly detection in business metric monitoring and how such monitoring system could be implemented. Therefore, this thesis does not go deeply into any single anomaly detection algorithm and instead the goal is to find several fitting anomaly detection algorithms that would be suitable for the solution. Literature review in existing anomaly detection algorithm comparison studies is used to find suitable algorithms, comparing anomaly detection method performances itself is not in the scope of this study. Suitable algorithms are described at high level and their usability is evaluated based on how well they would fit in the described business metric monitoring system.

Since most of the business metrics are single timeseries variables, this study is limited to only evaluate anomaly detection in univariate timeseries. Univariate timeseries is a sequence of single scalar value observations recorded over time. (Chandola, Banerjee, & Kumar, 2009). This definition will cover most of the cases in business metric monitoring and will rule out anomaly detection for multivariate timeseries and non-time series based symbolic sequences. These types of sequences require anomaly detection algorithms of a different kind and are not covered in the use cases of the anomaly detection tool described in chapter 1.1. With these limitations there are still plenty of viable anomaly detection techniques (Braei & Wagner, 2020) and therefore literature review is used to identify a limited set of anomaly detection methods that the described anomaly detection tool could include.

In the implemented anomaly detection tool majority of the development hours went into other tasks than the actual anomaly detection algorithm implementation part itself and since the anomaly detection was only a small part of the overall solution this thesis also focuses on what other parts are needed in a complete anomaly detection tool. As end result the overall value and usability of the solution are evaluated from business and organizational perspective. In order to gain value from the end results of the anomaly detection tool (the notifications of anomalies) the actions upon notifications are described at high level to provide a potential framework on how to act on different types of found anomalies in the business metrics.

With these objectives and limitations three research questions were formulated and can be found in Table 2. The first research question focuses on determining what anomalies in business metrics are and how companies can benefit from automatic anomaly detection in business metrics. The question is answered from managerial perspective to detect the value in monitoring of anomalies. The second research question focuses on the anomaly detection methods that can be used to identify anomalies in univariate timeseries. Furthermore, objective of the research question is to list several anomaly detection methods that are general enough to be able to detect anomalies in different kinds of business metrics. The limitations and use cases of each method are identified. The third research question is formulated to inspect the other required elements in an anomaly detection system and how the anomaly detection algorithm is a small modular part of the system.

Table 2. Research questions and objectives

Research questions	Objective
RQ 1. What is the current state of the art in business metric timeseries anomaly detection?	Define and categorize anomalies in business metrics and identify their different characteristics. Based on the most recent literature on anomaly detection systems identify industry practices in timeseries business metric monitoring.
RQ 2. How can anomalies be detected in business metric timeseries?	Identify several non-model based methods for anomaly detection as well as model based machine learning methods for the same purpose. Categorize which should be used on separate use cases.
RQ 3. What challenges are present in developing an automated end-to-end anomaly detection system?	Through case study and literature review identify and document the found challenges of end-to-end working anomaly detection system.

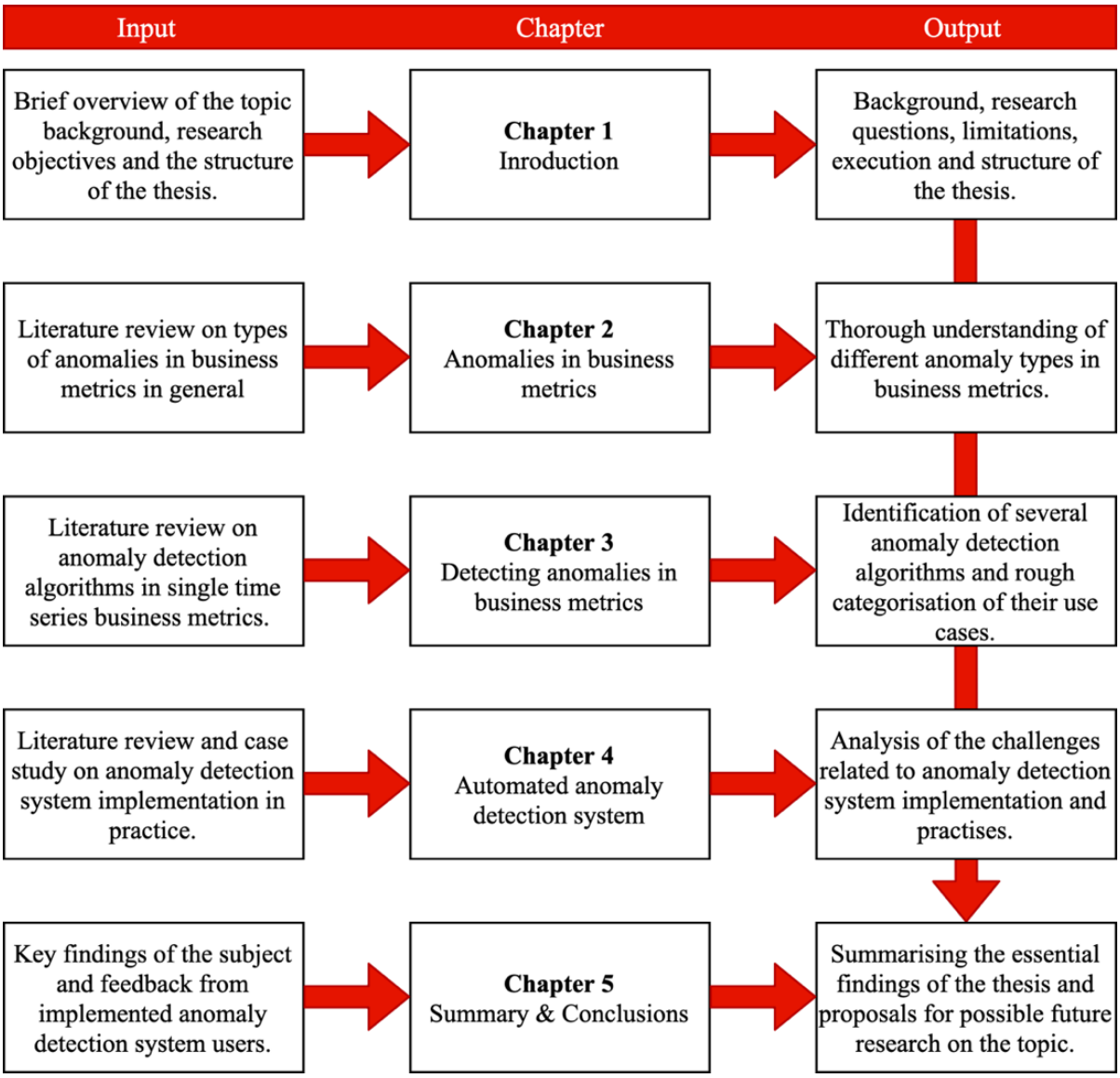
The findings of this thesis aim to provide understanding about anomaly detection practicalities in business metrics. Thus, the scope of the thesis is limited to detecting anomalies in single univariate timeseries which is significantly easier task than detecting anomalies in multivariate or other types of input data such as images. Since business metrics data is similar across industries, the findings are not only limited to software or game industry.

1.3 Structure of the report

This report consists of five main chapters with the first chapter being this introductory chapter which describes the background of the problem, states the research questions and how those questions are answered. Chapters two and three are the main literature review part of the thesis. Chapter two aims to provide the reader concrete description of anomalies in business metrics with going in details on topics such as the types of anomalies, significance of anomaly and the added value overall which can be achieved from identifying anomalies early. Chapter three consists of literature review of the algorithms that can be used to detect anomalies in business metrics, evaluates their overall performance, limitations and use cases. Chapter four focuses on

describing the other components present in an anomaly detection system and how the anomaly detection system works as end-to-end solution. The chapter also includes literature review of similar systems applied by other organizations earlier. Finally, in chapter five the findings of this study are summarized, and the main learnings are represented. The contribution of each chapter in this thesis are summarized in Figure 2.

Figure 2. Mapping the structure of the thesis



2 Anomalies in business metrics

In the literature, there are several similar definitions for anomalies. Often the term outlier is used to talk about the same subject. In statistical terms, an outlier can be identified as an unlikely event given the data distribution and an anomaly is a result that cannot be explained given the data base distribution (Salgado, Azevedo, Proenca, & Vieira, 2016). However, in context of business metrics these terms are interchangeable since unlike in many engineering fields, business metrics are not generally expected to have measurement errors. Therefore, each data value should be considered as a real value and labelled anomaly cannot be measurement error since the data is not gathered using sensors, instead the data is based on the captured data directly from the event that took place. Therefore, in this thesis outlier and anomaly are considered to represent the same phenomenon. However, even if business metric data is not expected to have measurement errors the data quality might still have other issues, which are described in depth at section 2.2.

One of the first definitions for anomalies was given by (Grubbs, Procedures for detecting outlying observations in samples, 1969) in the paper outliers were defined as: "*An outlying observation, or 'outlier' is one that appears to deviate remarkably from other members of the sample in which it occurs*". Later (Hawkins, 1980) described outliers as "*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by different mechanism*". According to (Braei & Wagner, 2020) anomalies have the two following characteristics:

1. Distribution of anomalies deviates significantly from the general distribution of the data
2. Majority of the datasets consists of normal data points. The anomalies form only a very minor part of the dataset.

In this study the term "business metric" means a quantifiable measure that can be used to track and monitor the status of a specific business process. In the scope of gaming industry this could mean for example the number of in-game purchases, the number of hours the player has played, or the number of ads shown to the player. Business metrics can contain several types of data such as categorical events or sequential non-series-based data, however as described in the

introduction chapter this thesis will focus solely on univariate timeseries based metrics due to use cases and the defined requirements of the anomaly detection system.

In order to go further in defining types of anomalies in univariate timeseries it is necessary to provide a definition for timeseries. A timeseries is a type of stochastic process. According to definition given by (Wei, 1989) "*A stochastic process is a family of indexed random variables Z(w, t) where w belongs to a sample space and t belongs to an index set.*" . A stochastic process can therefore be understood as a dictionary of key, value pairs. A time series is a sub-type of stochastic process where observations are taken with continuous measurements over time. A time series can be *multivariate* if the observations come from multiple sources or *univariate* meaning there is only one source. Also, timeseries can be *discrete* if all the observations are measured at specific point on time. (Braei & Wagner, 2020) The specified anomaly detection system described in chapter four is required to handle both equidistant and non-equidistant discrete timeseries since input data may have some missing observations or for some other reason not all datapoints have an equal time interval.

2.1 Types of anomalies in time series

A time series can be anomalous in comparison with other time series in a collection of time series or single time series can have anomalous values within the series (Mehrotra, Mohan, & Huang, 2017). For example, the data of players per hour for several different games can be a collection of time series and an anomaly in the set could be a timeseries that has significantly different pattern of players playing the game during the day compared to other games. This study will focus only on anomaly detection within timeseries due to specifications of the anomaly detection tool. An example of anomaly within a timeseries could be a significantly higher value of players playing the game compared with other hours of the day.

In the literature there are three widely mentioned types of anomalies that can exists within one time series. It is important to differentiate between these separate types of anomalies since different anomaly detection algorithms can be good at detecting one type of these anomalies and bad at other types (Braei & Wagner, 2020). If only one type of anomaly is known and expected to be present in a given timeseries, the simplest model that is able to detect the type

of anomaly should be chosen. The types of anomalies within timeseries can be categorized to *point anomalies*, *contextual anomalies* and *collective anomalies*.

Point anomaly

A point anomaly is a value in time series that is characterized by a substantial variation in the value from the preceding datapoints (Mehrotra, Mohan, & Huang, 2017). These types of anomalies are also mentioned in the literature as event anomalies or global anomalies (Cohen, 2020). This is the simplest and most common type of anomaly. Majority of anomaly detection research focus on the identification of point anomalies. For example, consider the topic of credit fraud detection. The total amount spent can be considered as a feature that has strong significance in classification of credit frauds. If a transaction has very large amount of money spent compared to normal range of expenditure for the individual, that transaction will be a point anomaly. (Chandola, Banerjee, & Kumar, 2009) Figure 3 contains timeseries with an example point anomaly at x-axis value 17.

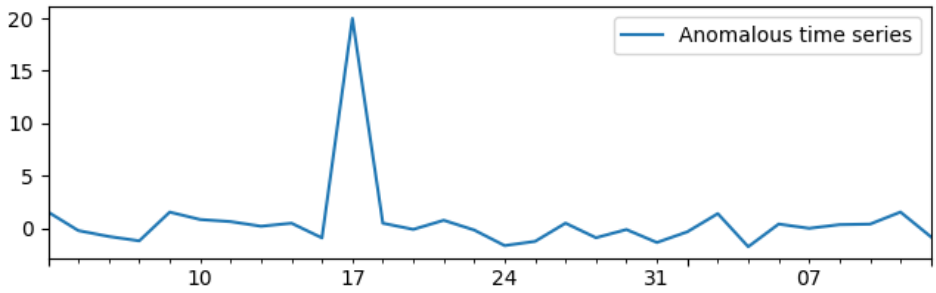


Figure 3. Example point anomaly

Contextual anomaly

Context based anomalies consists of data points that might seem normal at first glance but are considered anomalies in the context of the datapoints (Alla & Suman, 2019). For example, a sudden surge in sales during black Friday or a sudden decrease in sales during Superbowl can be classified as normal behavior because of the context. On the other hand, if sales remain relatively stable during the weekend that could be a contextual anomaly, since the sales were expected to change given the weekend context.

The notion of a context is formed by the structure of the underlying dataset and has to be specified as part of the anomaly detection problem formulation. Each datapoint in timeseries is defined using the two following attributes (Chandola, Banerjee, & Kumar, 2009):

1. *Contextual attributes*. The contextual attributes determine the context of the data instance. In time-series data time is the contextual attribute that determines the position of the observation in the timeseries.
2. *Behavioral attributes.* The behavioral attribute determines the non-contextual characteristics of a data point. For example, in time series there could be a base constant value and daily cyclical variation. In this case the base constant value would be a behavioral attribute and cyclical variation a contextual attribute. Another possible behavioral attribute could be the average rate of increase in the value.

Contextual anomaly is determined by using the behavioral attribute values within a specific context. An observation can be considered as contextual anomaly in one context and exactly same value (in terms of behavioral attributes) could be a normal value in different context. (Chandola, Banerjee, & Kumar, 2009) In order to determine contextual anomalies, the algorithm should be able to differentiate between the contextual and behavioral attributes which in case of time series data mean the algorithm should be able to detect the datapoint context which can usually be seen as periodical patterns.

The Figure 4 contains an example of a contextual anomaly in timeseries data. Here the daily temperature at t_2 would be considered as normal behavior during the winter months, but an

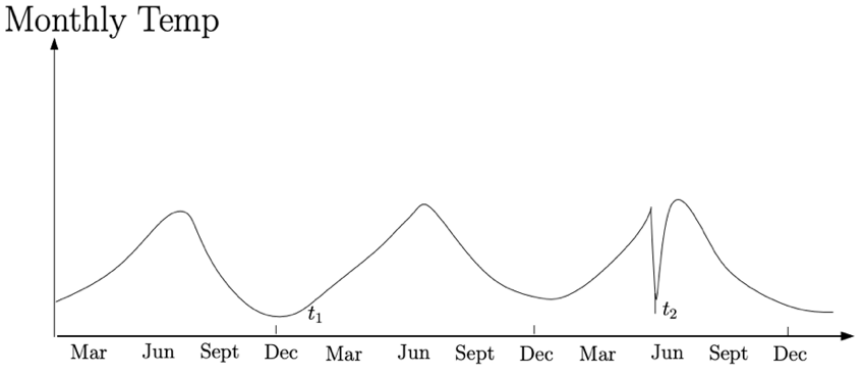


Figure 4. Contextual anomaly

anomaly during summer months. Here the behavioral attribute would be the base temperature and contextual attribute the pattern of temperature change across year. (Capretz & Hayes, 2015)

Detecting contextual anomalies can be achieved with the simplest statistical anomaly detection methods assuming that the context of the value are the other values in the near history of the timeseries. However, if the context is a weekly, monthly or even yearly seasonality in the variable standard anomaly detection algorithms may not be able to detect the pattern and give incorrect results (Toledano, Cohen, Yonatan, & Tadeski, 2017).

Collective anomaly

Collective anomaly or pattern-based anomaly are patterns and trends in the data that deviate from their historical counterparts. In collective anomaly a number of time series observations forms a pattern that is anomalous in the timeseries. (Alla & Suman, 2019) The individual data points in collective anomaly are not themselves considered anomalous in either contextual or point-anomaly (global) sense. In time series data collective anomalies can be described as normal peaks and valleys occurring outside of a timeframe when that seasonal sequence is normal or as a combination of time series that is in an outlier state as a group. (Cohen, 2020)

The Figure 5 represents collective anomaly in electrocardiogram output. The highlighted part implies an anomaly since the same value is present for abnormally long time. (Goldberger, Amaral, Glass, & Hausdorff, 2000) . The individual values do not deviate from the global datapoints and the datapoints within the context do not vary. Therefore, the individual observations do not fall under point anomaly or contextual anomaly categories. This however depends on how the context of the datapoint is defined and with different context definition–n single observations could fall under the contextual anomaly detection case. Key difference in this case is that even if the single observations would fall under the contextual anomaly category, the surrounding datapoints would be considered as normal which is not the case. A collective anomaly is a sequence of observations that are anomalous together.

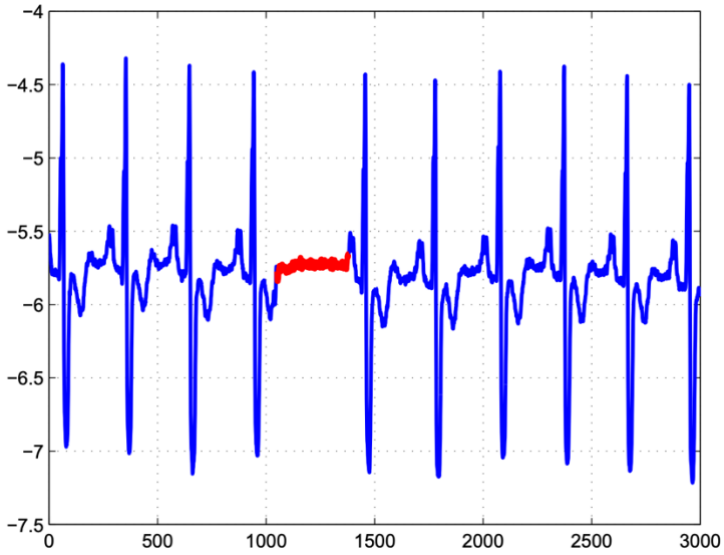


Figure 5. Collective anomaly in electrocardiogram output (Goldberger, Amaral, Glass, & Hausdorff, 2000)

While point and contextual anomalies can be identified with simple statistical methods collective anomaly detection often requires more sophisticated methods and algorithms. These methods are very different from the methods for point and contextual anomaly detection methods or are a combination of different models. (Chandola, Banerjee, & Kumar, 2009)

2.2 Anomaly detection for data quality monitoring

Anomaly in business metric can either be a real anomaly based on real events or an issue in data quality. After an anomaly has been found the cause should be studied, and if the cause is an issue in data quality steps to fix the underlying data can be taken such as data reprocessing or removing the incorrect values. Anomaly detection can be used as a tool to monitor several data quality dimensions and often configuring monitoring in the business metric itself can be used to raise alerts on data quality.

Data quality itself is a wide topic and there exists various methods and frameworks to ensure data quality in large scale. One of the often-mentioned methods is anomaly detection which can be also utilized on single univariate timeseries level. The issue of data quality is often split to data quality dimensions. In the literature commonly mentioned data quality dimensions include *Accuracy*, *Completeness*, *Consistency, Validity* and *Timeliness* (Karr, Sanil, & Banks, 2005).

Out of these dimensions' anomaly detection can be used to find issues in accuracy (Kasunic, 2011) and completeness (Stanley, When Data Disappears, 2020) dimensions. In order to use anomaly detection to monitor for issues in data quality in the other dimensions more details from the "normal" behavior of the metric would be required. In order to monitor for data consistency, it would be required to know how the metric is allowed to change and validity would require knowing if the metric accurately measures what it is intended to measure. Therefore, anomaly detection for data quality monitoring is difficult to generalize for other dimensions, and in the anomaly detection tool described in chapter 1 use cases of anomaly detection in data quality monitoring include only accuracy and completeness dimension monitoring.

Accuracy refers to the literal notion if a data record is precise or not. In many cases it is obvious if the observation is accurate or not. For example, a person reporting their age correctly is an accurate record. In exogenous data, the definition of accurate data is not as straightforward. For example, if there is a middleman in filling the person age to the data it cannot be known if that data was correct in the first place. (Karr, Sanil, & Banks, 2005) Anomaly detection is widely used to detect issues in data accuracy among various domains (Karkouch, Mousannif, Moatassime, & Noel, 2016). In case of business metrics, the data is not expected to be noisy so defining an accepted systematic error would not yield benefits. However, if the timeseries contains inaccurate values for example due to human or networking issues the anomaly detection algorithm could be able to spot it in business metrics. In time series an accurate datapoint should have the correct timestamp. In some cases, the analytics event timestamp can be defined when the event is received and not when the event is sent. In this case the data received would receive an inaccurate timestamp when there is for example delay in processing for the incoming events. Such delays could cause anomalies in a metric that keeps track of events per time units, such as purchases per hour.

Completeness dimension refers to tracking if a record has missing values or not. Defining dataset completeness is however not that straightforward since the relation of missing and legitimate values can be confounding meaning that it is not known if the missing value had an effect on the following legitimate values. (Huang, Lee, & Wang, 1999) In terms of univariate timeseries in business metrics, the missing values can be easy to spot if the timeseries should

be discrete with equidistant observations. However not all business metrics are discrete time series and therefore detecting missing values becomes more difficult. The completeness dimension can however be monitored with anomaly detection methods when using an aggregated timeseries based on for example the total event count per hour. If there are significant number of missing values the event count per hour gets abnormal values.

Stanley (Stanley, Airbnb quality data for all, 2020) described in the article how anomaly detection or time series forecasting is used to improve data quality at Airbnb. Anomaly detection in the single timeseries row count is used to keep track if the underlying data has been updated or if the most recent date has row count in the expected range. Business metrics are monitored as well for suspicious changes which could mean issues is data completeness and accuracy. Stanley also mentioned that at Airbnb anomaly detection and other data quality checks are a required common practice in the new pipelines that are built, and this has successfully prevented issues in the new pipelines (Stanley, When Data Disappears, 2020). Using anomaly detection in large scale when building analytics pipelines seems to have clear benefits in data engineering perspective and the use cases of anomaly detection are not limited to only business intelligence and data science fields.

3 Detecting anomalies in business metrics

Anomaly detection in time-series is linked to time-series modelling and in practice anomaly detection is a two-class classification problem. When handling anomaly detection in timeseries with time-series modelling methods, the model is fitted to available training data and the estimated values are calculated with the adjacent past values. The number of previous values used in the detection can be defined as a sliding window. Sliding window is the subsequence of the time-series which is used as input to calculate the following timestamp, or in other words the sliding window contains all the datapoints the anomaly detection model uses to calculate the next value. Formally let w be the width of the sliding window, the timeseries is x_i and the model used for forecasting is M. To forecast x_i the following function can be used (Braei & Wagner, 2020):

$$M : \mathbb{R}^w \rightarrow \mathbb{R}$$
$$\hat{x}_i = M\big((x_{i-w}, \dots, x_{i-1})\big)$$

The anomaly score e_i can be computed for example using Euclidean distance d from the predicted value $\hat{x}$ and the real value x scaled with the real value:

$$e_i = \frac{d(x_i, \hat{x}_i)}{|x_i|}$$

The anomaly score represents the degree to which the data instance is considered anomalous. Therefore, the output of the anomaly detection methods that provide an anomaly score is a ranked list of anomalies. Using the anomaly scores the cut-off threshold of flagging the anomaly can be changed. (Chandola, Banerjee, & Kumar, 2009) The cut-off threshold can be adjusted to modify the sensitivity of the algorithm used and since anomaly score is the usual output of each anomaly detection algorithm the sensitivity setting for detecting anomalies in the timeseries is not algorithm specific.

After the anomaly scores are calculated and the cut-off threshold for flagging the anomaly is specified there are four different outcomes for the results (Mehrotra, Mohan, & Huang, 2017):

1. Correct detection: (True positive or True negative): The Observation is detected correctly as normal value or correctly as an anomaly. Correct detection in terms of business metric is often entirely up to human interpretation, the observation is either considered to vary enough from normal to be considered anomalous or the opposite.
2. False positive: The observation continues to be normal, but the data value is labeled as anomalous.
3. False Negative: The business metric value is abnormal, but the observation is labeled incorrectly as normal value.

These outcomes form the Confusion matrix which is a commonly known classification task performance evaluation tool. From the confusion matrix, the anomaly detection performance can be evaluated by calculating metrics such as classification accuracy which is the number of correctly flagged values divided by all values in the timeseries. (Alla & Suman, 2019) Figure 6 contains a confusion matrix.

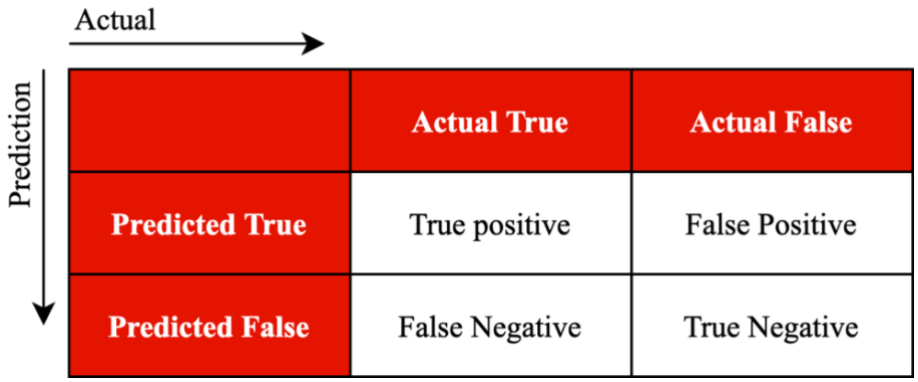


Figure 6. Confusion matrix

One popular and often used metric derived from the confusion matrix is the *receiver operating characteristics curve* or ROC-Curve and the associated metric *area under the curve* (AUC) which represents the area under the ROC-Curve. AUC is widely used metric in anomaly detection performance evaluation. The ROC-Curve plots the values of the *true positive rate*

(TPR) and the *false positive rate* (FPR) based on different threshold values of classifying the value as an anomaly according to anomaly score. Formally TPR and FPR are:

$$TPR = \frac{Number\ of\ true\ positives\ (TP)}{Number\ of\ predicted\ positives\ (P)}$$

$$FPR = \frac{Number\ of\ false\ positives\ (FP)}{Number\ of\ true\ negatives\ (N)}$$

In order to compute ROC-curve different thresholds for anomaly score are iterated through in order to get the TPR and FPR for each threshold. These values are then plotted showing a curve starting at the origin and ending in the point (1,1). The AUC value is the area under the curve with values ranging from zero to one with one meaning all values were classified correctly. (Braei & Wagner, 2020)

3.1 Anomaly detection algorithm types

Anomaly detection algorithms in univariate timeseries are commonly divided into two categories, the statistical anomaly detection methods and the machine learning based anomaly detection methods. Machine learning methods can be further divided into classical machine learning methods and algorithms that leverage neural networks in any way. Another way to categorize an anomaly detection method is according to the level of supervision required when using the algorithm: supervised, unsupervised and semi supervised. (Braei & Wagner, 2020) (Munir, Siddiqui, Dengel, & And, 2019) These categories are briefly introduced before moving on to literature study on algorithm performance and describing individual algorithms.

Statistical and machine learning methods

The difference between statistical and machine learning model is vague. In high level statistical approaches assume that data is generated by a given stochastic data model while machine learning based (algorithmic) models treat the data mechanism as unknown. In a statistical model, the model parameters are calculated from the data whereas in machine learning modelling the high-level approach is to find a function $f(x)$ that operates on x to predict the

responses y. Figure 7 represents the differences of both approaches to modelling. (Breiman, 2001) In other words, the machine learning methods are based on the implicit assumption that it is not relevant from the model perspective how the underlying data generation process is done, and the trained machine learning model should still be able to produce accurate predictions (Papacharalampous, Tyralis, & Koutsoyiannis, 2019).

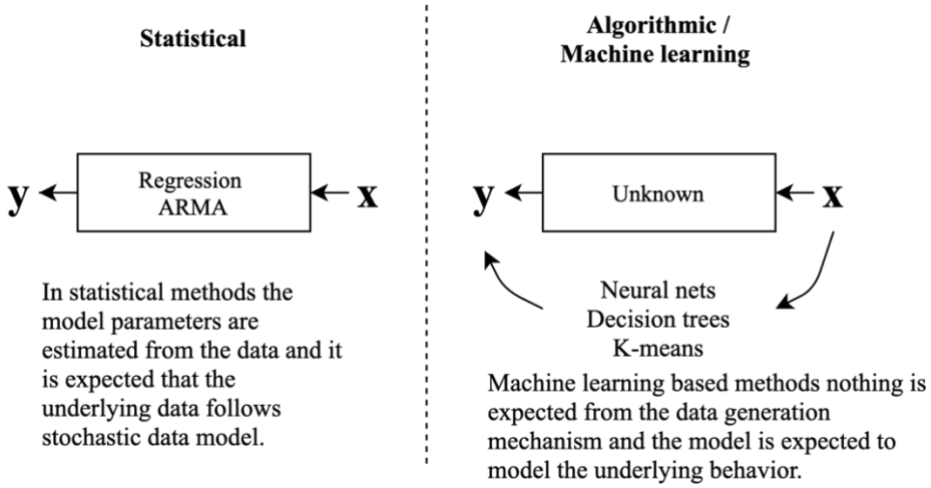


Figure 7. Statistical and algorithmic modelling

On top of statistical and ML based methods there can be purely rule based methods which can be defined as anomaly detection. Simple rule-based methods such as defining a limit for the business metric value or comparing the percentage change of the business metric can be classified as anomaly detection since they aim at the same goal, flagging anomalous values. There are obvious dilemmas in rule-based methods for example in accuracy of the labeling but in some use cases the simplest and easiest to interpret method to flag anomalous values has benefits over model-based methods.

Supervised anomaly detection

Supervised anomaly detection is a technique where the training data has labels for both known anomalies and normal data points. This way anomaly detection model can be trained on known anomalies and the model is able to learn anomaly characteristics based on the known anomalies. Supervised anomaly detection is therefore a binary classification task and any classification

model can be used for the task. (Alla & Suman, 2019) However, there are major issues which mostly prevent using supervised anomaly detection to detect anomalies in business metrics in practice. In a normal anomaly detection scenario, there are far fewer anomalous datapoints compared to normal observations in the training data. Imbalanced class distribution brings issues to model training and proper anomaly detection. The required training data set would have to be very large since anomalous values exists sparsely. Another issue is the flagging of known anomalies for training data which would require human interpretation. If the anomalous values are very uncommon, flagging the known anomalies by hand would take a lot of time. (Chandola, Banerjee, & Kumar, 2009) Supervised anomaly detection however has a place in the anomaly detection toolset and can be used as a "second model" on top of unsupervised anomaly detection when the unsupervised anomaly detection system been running for a while and the user has provided feedback on if the anomaly recognized by the unsupervised method is a real anomaly or not. (Laptev, Amizadeh, & Flint, 2015)

Semi-Supervised anomaly detection

Semi-supervised anomaly detection techniques assume that training data has labeled instances only for the normal class. These types of anomaly detection methods are more widely applicable than supervised techniques since they do not require labels for the anomaly class. (Chandola, Banerjee, & Kumar, 2009) Ideally, a semi-supervised model will learn what normal data points look like, so that the model can flag anomalous data points since they differ from normal data points. (Alla & Suman, 2019) In the literature there seems to be very few use cases for semi-supervised anomaly detections for univariate time-series, so these types of algorithms are not included in the thesis.

Unsupervised anomaly detection

Unsupervised anomaly detection techniques do not require labels in the training data and are therefore the most widely applicable algorithms to the anomaly detection problem. The unsupervised techniques make the implicit assumption that normal data instances are much more frequent than anomalies on the test data. Based on this assumption the normal behavior is defined and anomalies are categorized based on the data metrics, for example in case of time

series the scaled distance to fitted value can be used. (Chandola, Banerjee, & Kumar, 2009) In this study, most of the methods are unsupervised since labeled data is not often available for the business metrics.

Unsupervised anomaly detection algorithms have the following characteristics (Mehrotra, Mohan, & Huang, 2017):

1. Normal behavior of the data is dynamically defined. Prior training data set for normal behavior is not required
2. Outliers must be detected effectively even if the distribution of the data is not known
3. The algorithm should be adaptable to different application domains. The algorithm should be able to provide good results without requiring substantial domain knowledge or major modifications.

3.2 Time-series data pre-processing for anomaly detection purposes

Timeseries based data can have specific features that can have major effect on the anomaly detection accuracy if actions on *data preprocessing* are not taken. Exceptionally statistical anomaly detection methods can suffer if *data stationarity* is not achieved or present in original data. (Makridakis, Spiliotis, & Assimakopoilos, 2018) Machine learning methods can be more effective on modelling any type of input data and can be therefore applied directly to input data in some cases (Gorr, 1994). However, there are studies which show that using data pre-processing methods such as deseasonalization and detrending have significant improvement on time series anomaly detection performance on neural network based models. This would mean that neural networks are not always able to capture seasonal or trend variations effectively in raw unprocessed data or that the detrending or deseasonalization done in training the neural network can lead to forecasting errors. (Zhang & Qi, 2005) Since anomaly detection is effectively a time series forecasting task similar data pre-processing should be also considered for machine learning utilizing anomaly detection methods in single univariate timeseries.

Data pre-processing for timeseries is often done to achieve *stationarity*. A strictly stationary process is one where for any $t_1,\ t_2, \dots, t_T\ \in\ Z$ any $k \in Z$ and $T = 1, 2, \dots$:

$$Fy_{t_1}, y_{t_2}, \dots y_{t_T}(y_1, \dots, y_T) = Fy_{t_1+k}, y_{t_2+k}, \dots y_{t_T+k}(y_1, \dots, y_T)$$

Where F denotes the joint distribution function of the set of random variables. It can also be stated that the probability measure for the sequence y_t is the same as that for $\{y_{t+k}\}\forall k$ (Tong, 1990). This means that a series is a strictly stationary if the distribution of its values remains the same as time progresses, meaning that the probability that y falls within a particular interval is the as at any time in the past or the future (Brooks, 2014, ss. 251-252). Easier to understand definition for stationarity is that *weak stationary* series can be defined as one with *constant mean*, *constant variance* and *constant autocovariances* for each given lag. (Brooks, 2014, s. 353). The difference between strict and weak stationarity is that in strict stationarity the distribution of the timeseries is exactly same through time which means that in strict stationary no assumptions on the data distribution are made.

Achieving constant mean can be achieved by *detrending* the data. Trending data can be easy to visually identify, however a statistical test may be necessary in context of the anomaly detection tool since the decision of whether to perform detrending should not be left to users, since not performing detrending can have negative impact on the anomaly detection accuracy. One test that can be used to detect upwards or downwards trend is the Cox-Stuart test (Cox & Stuart, 1955). Explaining how the test works is not in the scope of this thesis, but as output it can provide confidence value on trends being present in the data. If the timeseries had a trend, a simple method for detrending is differencing. A differenced series is the change between consecutive observations in the original series which can be written as $y'_t = y_t - y_{t-1}$. This would mean that the series is differenced once which can be written as $y_t \sim I(1)$. This means that by differencing once the series has become stationary in the mean. There can be cases where differencing once is not enough, Brooks mentioned examples such as nominal consumer prices or nominal wages. In that case the Cox-Stuart test could be done a second time to check for stationarity in the mean and if there is still an observable trend the differencing process can be done once again. (Brooks, 2014) Figure 8 contains an example of time series differencing.

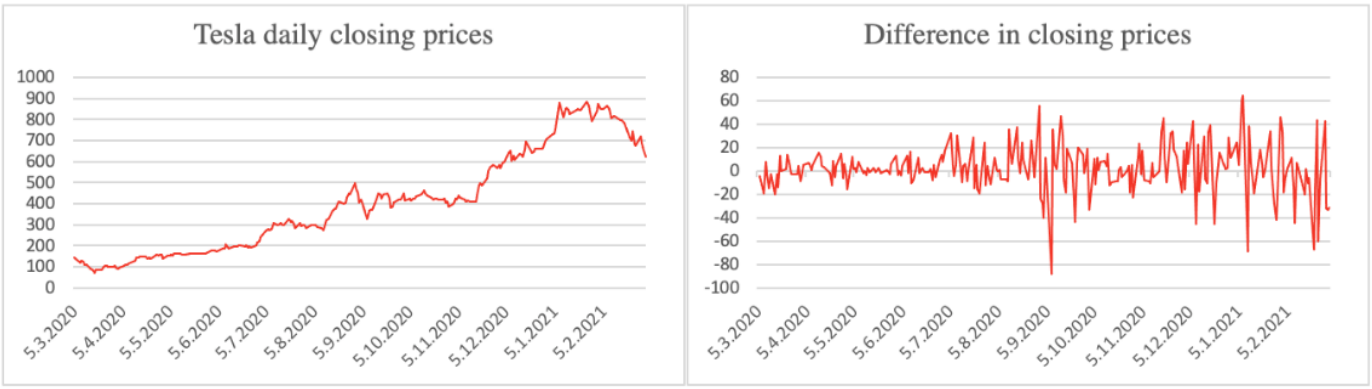


Figure 8. Differencing example

Achieving constant variance can be achieved by transforming the data dependent variables to normal shape meaning that the histogram of the values should follow roughly normal distribution. One common method to remove variance in the data is using the box-cox transformation method (Box & Cox, 1964). In box-cox transformation, the transformed data is generated using equation:

$$y(\lambda) = \begin{cases} \dfrac{(y+\lambda_2)^{\lambda_1} - 1}{\lambda_1}, if\ \lambda_1 \neq 0 \\ \log(y+\lambda_2) \quad , if\ \lambda_1 = 0 \end{cases}$$

Here $\lambda_1$represents the transformation exponent which can be optimized based on how close to normal distribution the results follow and λ_2 provides an offset to the data to make all values positive and can therefore be the negative value of the minimum value in the series. If the series contains only positive values, λ_2 can be defined as zero. Figure 9 contains an example of box-cox transformation using randomly generated data from Beta distribution. From the left figure it can be clearly seen that the histogram of the values is clearly skewed right meaning that randomly generated series does not have constant variance.

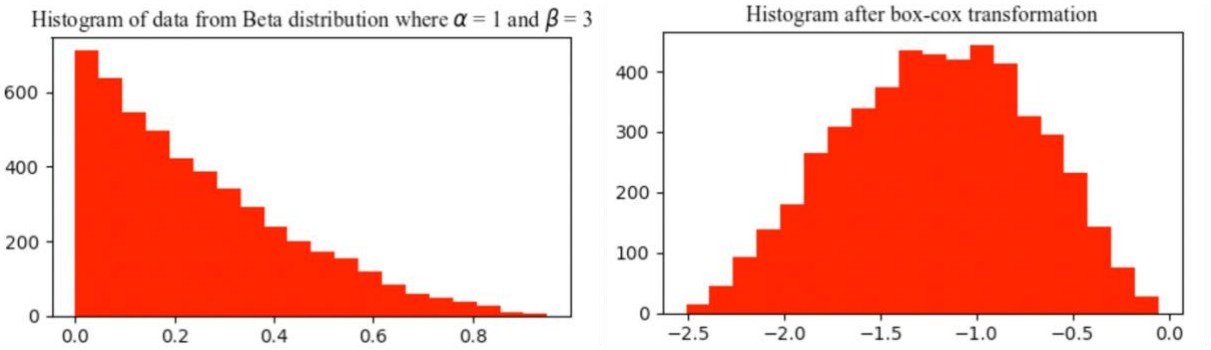


Figure 9. Box-Cox transformation

After transforming the data to achieve the stationarity in variance the anomaly detection results get more difficult to interpret and it may not be clear why statistical and machine learning based methods get improved accuracy after the transformation. However, many statistical models require stationary data to perform well and the accuracy of several machine learning method can be increased by transforming data to constant variance (Makridakis, Spiliotis, & Assimakopoilos, 2018). Taking logarithm or doing the higher lambda value box-cox transformation is a common practice in financial data modelling as well (Brooks, 2014).

Autocovariances in series represent how y is related to its previous values. If a series is stationary in terms of constant autocovariance, the covariance between y_t and y_{t-1} is the same as the covariance between y_{t-5} and y_{t-6} etc. This can be expressed as the autocovariance function (Brooks, 2014):

$$E\big(y_r - E(y_t)\big)\big(y_{t-s} - E(y_{t-s})\big) = y_{s,}, s = 0, 1, 2, \dots$$

When $s = 0$ the autocovariance at lag zero is obtained, which actually returns the variance of the y itself. The covariances of y_s between the previous values are known as autocovariances since they measure the relationship of y and its previous values. For example, a purely random process that has no discernible structure would get zero autocovariance since the values are not connected with the previous values in any way. This type of process is also known as white noise. (Brooks, 2014) In essence every forecasting model in univariate timeseries which only relies on the data from the previous datapoints tries to identify the autocovariances of the series. If the series has been preprocessed to have constant mean and constant variance, the only two

things left to model are the autocovariances and possible *seasonality* in the data. However, there are also ways to *deseasonalize* the data.

In chapter 2 the contextual anomaly was defined as having an anomalous value in the current context of the datapoint but the datapoint would not have been an anomaly earlier due to seasonality of the series which was seen from figure 4. Seasonality of a timeseries can often be easily noticed with visual observation, however there exists several statistical methods to determine if the timeseries has seasonal behavior. Autocorrelation function represented earlier can be used to identify seasonality in the data. If the timeseries exhibits seasonal patterns, it will show repetitive pattern in the autocorrelation plot. If the autocorrelation function contains peaks between similar intervals there is a probability that the timeseries series contains seasonality (Freeman, Merriman, Beaver, & Mueen, 2019). Another common way to check for seasonality is fitting regression model with dummy variables to the timeseries. In case of quarterly data, the regression equation would be $y_t = \beta_1 + y_1 * D_1 + y_2 * D_2 + y_3 * D_3 + y_4 * D_4 + u_t$. Here dummy variable would receive value 1 if the timeseries datapoint exists in the corresponding quarter. In practice the dummy variables would work by changing the intercept point, so the average value of the dependent variable given all explanatory variables is permitted to change across the quarters. During first quarter, the intercept would be $\hat{\beta}_1 + \hat{y}_1$ since $D_1 = 1$ and $D_2 = 0$, $D_3 = 0$ $D_4 = 0$ for all quarter 1 observations. In case of daily or hourly seasonality the number of dummy variables and regressors would be simply increased. After the regression model is fitted to the data the significance of the coefficient estimates can be analyzed for each quarter, and if the significances are high a conclusion with high confidence can be made that the particular time series does contain seasonality. (Brooks, 2014) If according to the coefficient estimates the timeseries contains seasonality the deseasonalized values can be achieved by subtracting the predicted values from the original observations. Another more common way to perform deseasonalization are additive and multiplicative decomposition approaches (Wheelwright & Makridakis, 1998).

Time series decomposition is a common tool in time series analysis. In practice univariate time series can be modelled to be part of four separate components which are the actual level, trend, seasonality and noise. The goal of time series decomposition is to extract each of these components out of the timeseries. The decomposition model can be additive such as:

$$y(t) = Level + Trend + Seasonality + Noise$$

Or the model can be multiplicative:

$$y(t) = Level * Trend * Seasonality * Noise$$

An additive model is a linear model where changes over time are made by the same amount. Multiplicative model on the other hand is a non-linear model and can model timeseries which are for example quadratic or exponential. The key difference is that the changes in the model increase or decrease over time. (Wheelwright & Makridakis, 1998) After the deseasonalized data is achieved the training of ML weights or the optimization and fitting of the statistical model can be done with the seasonally adjusted data. The forecasted values can be obtained using deseasonalization to calculate the final values, which can be done easily by adding the seasonalized component of the decomposition results to the predicted values obtained from the level. Some anomaly detection methods such as ARIMA have versions that include seasonal element built into the model itself so in these cases deseasonalization should not be done. Also, some machine learning and neural network models can learn the seasonality so there is no need to perform deseasonalization. However, deseasonalization can improve forecasting performance in some cases. (Makridakis, Spiliotis, & Assimakopoilos, 2018)

Explaining the time series decomposition process is not in the scope of this thesis, however example decomposition results explain the end results well in general. Figure 10 contains an example of timeseries decomposition done for hourly analytics event count data for one of Rovio's games. From the raw timeline it can be clearly seen that there is daily seasonality in the data, and the decomposition method was able to identify the daily pattern.

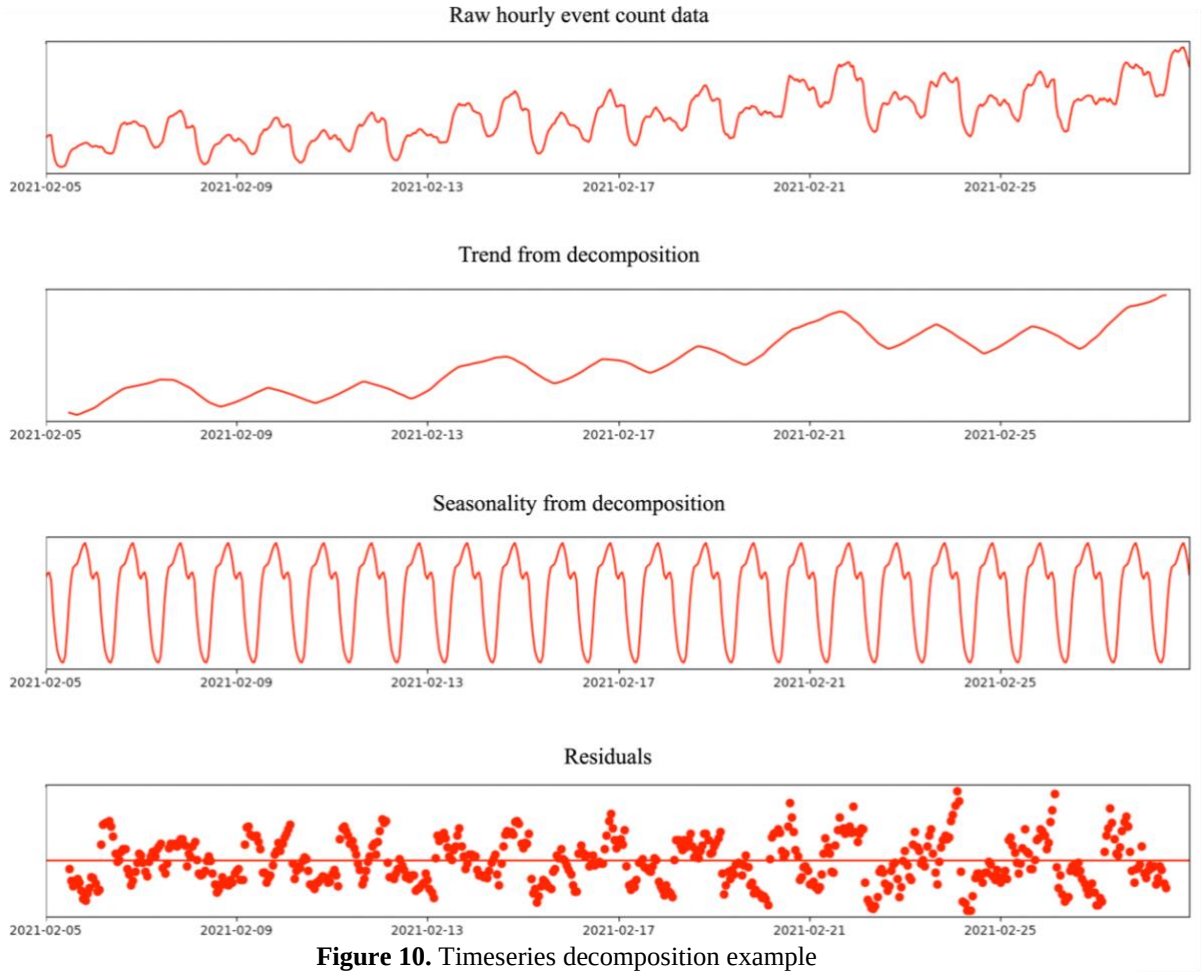


Figure 10. Timeseries decomposition example

In this section three data preprocessing steps applicable to univariate timeseries were described. In practice, it is however unclear what preprocessing steps should be taken since the required reprocessing is dependent on the used anomaly detection algorithm and the structure of the timeseries. In an automated anomaly detection tool, the question raises should the decision of which preprocessing steps to take be left to the user or should there be defined rules on how the required preprocessing steps are automatically done. From business user perspective, the anomaly detection tool should be as easy as possible to use and data preprocessing should be handled automatically but in some cases the anomaly detection accuracy can be improved by deciding what if any data preprocessing should be made. This also comes down to what anomaly detection algorithms should be available in the tool, since some methods do not require any data preprocessing perhaps preprocessing is not needed at all. There are also multiple anomaly detection libraries that can handle both the data preprocessing and anomaly detection,

and in this case data preprocessing would not need to be separate step in the perspective of developing an anomaly detection system.

Different pre-processing alternatives also make it challenging to compare anomaly detection algorithms together since the pre-processing can have a major impact on the anomaly detection accuracy. This is especially the case when comparing a statistical anomaly detection method with a method that utilizes neural networks to flag the anomalies, since the neural network based model can perform better if some preprocessing is taken, but the algorithms can also learn the seasonality and therefore deseasonalization is not needed (Braei & Wagner, 2020).

In many cases the time series that should be inspected for anomalies is not discrete. The definition for discrete timeseries was defined in chapter 2. Non-discrete time-series modelling has several issues, for example the algorithm is not able to detect the seasonality and many models expect the timeseries to be discrete in order to train the model properly. In case of handling non-discrete timeseries the reason for being non-discrete is important to classify. The time series might be supposed to be discrete with equally spaced observations, but it is missing values and therefore has become non-equidistant. Other possibility is that the timeseries is a collection of *temporal data* and the observations are not taken between equal timeframes. Rule based anomaly detection algorithms presented in chapter 3.4.1 do not require discrete data but some of the presented more advanced algorithms require discrete data in order to give accurate results.

Missing values can be filled in with technique called *imputation* (Moritz, Sarda, Bratz-Beielstein, Zaefferer, & Stork, 2015). There are several types of missing values which should be taken into account when choosing the imputation method. The type of missing value also has an effect on the imputation accuracy meaning that some types of missing values are more difficult to fill in. The different types of missing values in timeseries are tied to the "*missingness mechanism"* (Pratama, Permanasari, Ardiyanto, & Indrayani, 2016):

1. *Missing Completely At Random* (MCAR) type of missingness means that values are missing completely at random which means that the probability of missingness is the same for all timeseries points.

2. *Missing At Random* (MAR) type of missingness means that the probability that variable is missing is only related to available information, not the variable value itself. In terms of timeseries this means that there could be missing values on equal time intervals
3. *Not Missing At Random* (NMAR) missingness means that the probability that the variable is missing is dependent on the absolute value variable itself.

Commonly, the missing data handling methods are divided into conventional and imputation-based methods. Conventional methods include ignoring the missed value and mean, median or mode replacement of the missing value. (Moritz, Sarda, Bratz-Beielstein, Zaefferer, & Stork, 2015) In timeseries based modelling where the goal of missing value handling is getting discrete timeseries ignoring or deleting the missing value is not an option. The simple mean, median or mode replacement methods mean that the missing value is calculated as aggregate of the nearby values or the whole dataset. This kind of missing value handling is able to effectively handle MCAR and in some cases MAR type of missingness, but it should not be used for NMAR type missingness (Moritz, Sarda, Bratz-Beielstein, Zaefferer, & Stork, 2015). Obvious methodology for filling the NMAR type of missing values is training a predictive timeseries model from the data part that has no missing values, but in some cases there is not enough complete continuous training data available since missing value frequency is too high. The conventional methods in some cases have a negative impact on the further modelling and may lead to biased results which in terms of anomaly detection could mean poorly trained anomaly detection model. Therefore, more advanced imputation methods which leverage machine learning and time series modelling have been introduced (Moritz, Sarda, Bratz-Beielstein, Zaefferer, & Stork, 2015) (Pratama, Permanasari, Ardiyanto, & Indrayani, 2016). Describing these methods is not in the scope of this study, but at high level the anomaly detection tool should check each timeseries for missing values and replace the missing values using an imputation method.

In many cases, the timeseries is continuous and contains *temporal data* which means that individual observations represent the variable state in time but the timeframe between values is completely indeterministic (Chaudhari, Rana, Mehta, Mistry, & Raghuwanshi, 2014). The process of transforming this type of series to discrete timeseries is called *discretizing*. Figure 11 contains a simple discretization example of a data that does not contain noise. Discretizing temporal data is a study field on its own and there are many complex methodologies for it. After

temporal data is discretized the number of values in the timeseries is reduced which means reduced memory usage. Discretizing temporal data also makes the data easier to be represented and most importantly easier to use in modelling techniques. (Chaudhari, Rana, Mehta, Mistry, & Raghuwanshi, 2014) In the scope of the described anomaly detection system in chapter 1 discretizing temporal data would make it possible to leverage the anomaly detection tool to wider range of timeseries available. However, since discretization of temporal values is so wide topic the actual methodologies are not in the scope of this thesis. In some cases, there is already discretized version of the temporal data available since the same data was used for other modelling purpose and therefore discretization can be ruled out from the anomaly detection tool requirements. Also, in some cases it might be possible to aggregate the underlying timeseries for example using mean value over certain time unit. This might obviously make it difficult to detect point anomalies in the raw dataset but could be sufficient in terms of anomaly detection accuracy.

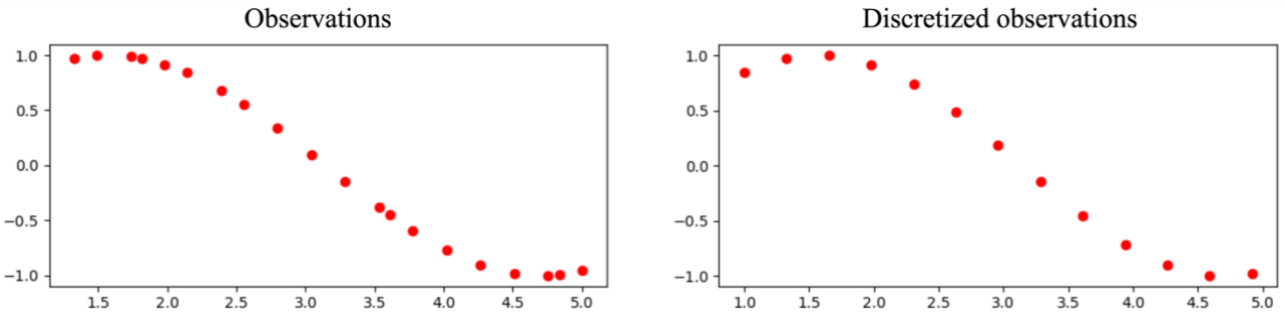


Figure 11. Example discretization

3.3 Previous studies of anomaly detection in univariate time series

In this section, the goal is to do literature review in previous studies which compare univariate anomaly detection algorithm performance between datasets. The studies which do not present any new algorithm should give unbiased overview of the algorithm performances. All studies mentioned in this section are using freely available benchmark datasets which can be seen from Table 3.

Table 3 Datasets used in the anomaly detection benchmark studies

Dataset	Description	Used in studies
Yahoo Services network traffic (Amizaded, Laptev, & Billawala, 2015)	Univariate time-series dataset containing traffic data to Yahoo services. The anomalies are labeled by human. The whole dataset includes also synthetic datasets to benchmark point-anomalies, seasonality and changepoint anomalies (sudden change in timeseries behavioral attributes.	(Braei & Wagner, 2020) (Munir, Siddiqui, Dengel, & And, 2019) (Xu, et al., 2018)
M3 Competition dataset (Hibon & Makridakis, 2000)	M3-competition was an academic time series forecasting competition. The data contained 3003 univariate time series from several industry sources.	Subset of 1045 monthly time series used by: (Makridakis, Spiliotis, & Assimakopoilos, 2018)
Numenta anomaly benchmark (NAB) (Lavin & Ahmad, 2015)	Numenta anomaly detection benchmark is a testing framework for benchmarking several anomaly detection algorithms. The open-source repository contains framework for adding own algorithms as well as test data from various industries.	(Freeman, Merriman, Beaver, & Mueen, 2019) Subset of the data used by in: (Braei & Wagner, 2020) Whole dataset: (Lavin & Ahmad, 2015) (Amarbayasgalan, Pham, Theera-Umpom, & Ryu, 2020) (Munir, Siddiqui, Dengel, & And, 2019)
Ibm-common-stock-closing-prices dataset (Hyndman & Khandakar, 2008)	Consists of daily stock closing prices for IBM from 1962 to 1965.	(Freeman, Merriman, Beaver, & Mueen, 2019)

In this study the algorithms for anomaly detection are chosen based on the following factors:

1. The algorithm is mentioned in multiple studies and can perform anomaly detection in variety of application domains effectively.
2. There is an existing well documented open-source python library that includes the algorithm, and the library does not have an overwhelming number of dependencies. Or

the algorithm is mathematically simple enough to be written as a tailor-made solution. This limitation is placed due to implementation time limitations.

3. The algorithm should not require a lot of configuration to perform labeling, meaning business user could use the method through the anomaly detection tool described in chapter 1.

Different studies use different metrics on comparing the anomaly detection performance. Some studies use Mean Absolute Percentage Error (MAPE) to perform comparison on the model performance which is calculated from the differences of the predicted and the actual values. While MAPE can be used to compare the modelling and forecasting performance of different algorithms it is distinct from the anomaly detection performance, since the goal of anomaly detection is to separate the anomalies from the data rather than achieving best fitting model on the data. In order to compare the anomaly detection performance, the measure should be calculated from the binary anomaly classification results. One popular measure is the F score which can be calculated as (Alla & Suman, 2019):

$$F = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

Where *tp* is the number correctly flagged anomalies, *fp* is the number of normal points flagged as anomalies and *fn* is the number of anomalies flagged as normal points.

There are plenty of anomaly detection algorithms available for univariate timeseries data. Braei and Wagner studied the performance of twenty univariate anomaly detection methods with 368 univariate timeseries in order to study the performance of each anomaly detection method. The datasets were from the commonly used Yahoo and Numenta datasets that have been used for benchmarking also in other studies. The end results showed that statistical approaches tend to perform consistently better on univariate time-series for detecting point and collective anomalies than the compared machine learning based methods. Statistical methods were also significantly less computationally intensive and required less parameter tuning to achieve the results. However, in the study they stated as well that if the univariate dataset consisted of mainly contextual anomalies machine learning methods outperformed statistical methods since

they were able to learn the seasonal patterns. Interesting finding in the study was that in these contextual anomaly datasets the classical machine learning algorithms outperformed even the more recently introduced algorithms that leverage deep learning. In a dataset that contained mostly contextual anomalies k-means algorithm provided the best AUC value. Figure 12 contains the results of the study. Each algorithm is plotted against the computational time and AUC value. (Braei & Wagner, 2020) From the figure it can be seen that the autoregressive *AR* moving average *MA* models had the best average AUC value among all datasets and those were also computationally much less intensive than most of the machine learning based methods. Exponential smoothing *ES* and simple exponential smoothing *SES* methods also performed relatively well.

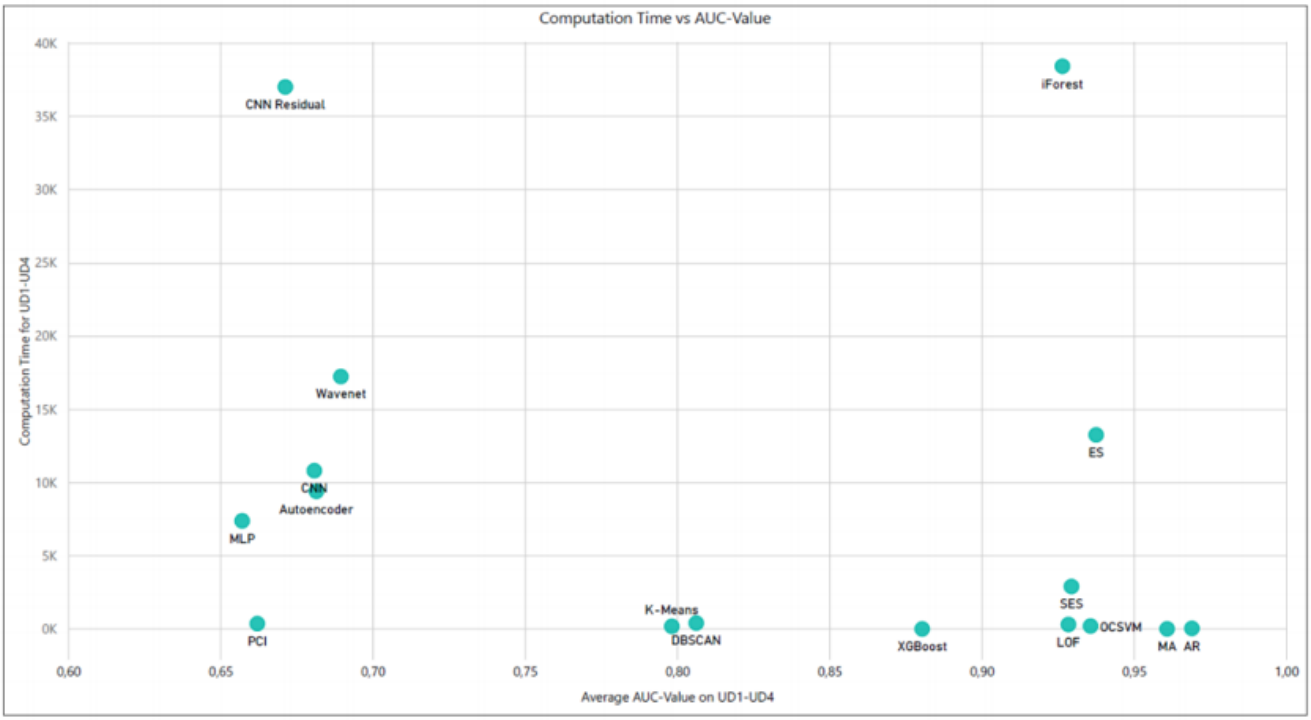


Figure 12. Average AUC-value vs computation time (Braei & Wagner, 2020)

Makridakis et al. studied the forecasting performance with a large dataset using several ML and statistical methods. They evaluated the forecasting performance of eight statistical and eight machine learning based methods and the results showed that the statistical methods tend to outperform the machine learning models in terms of accuracy and computational requirements. In the results, all statistical methods outperformed the machine learning methods in forecasting one value ahead, which would be the most common scenario in time series anomaly detection.

In the study, the results from the best to the worst statistical methods were: ETS (variation of exponential smoothing), ARIMA, Damped, Holt and Damped Comb, Theta model and simple exponential smoothing. They also pointed out in the issue that hyperparameter tuning and avoiding over-fitting in the machine learning models is not a straightforward process. In statistical models such as ARIMA over-fitting can be directly controlled by using information criteria methods. Therefore, achieving the optimal parameters for statistical model is an unambiguous process while hyperparameter tuning and over-fitting may lead to varying performance on machine learning models. In the study, the taken data pre-processing steps were documented well and the study included review on what kind of impact data preprocessing steps taken have on machine learning algorithms. From the study results it can be seen that data pre-processing improved the one step ahead prediction accuracy on all of the machine learning algorithms on average in all the timeseries used in comparison. (Makridakis, Spiliotis, & Assimakopoilos, 2018) While this study was about forecasting performance in univariate timeseries the results are comparable since anomaly detection is in essence a forecasting one datapoint ahead.

In recent years many "branded" algorithms or anomaly detection frameworks have been introduced. These algorithms include Facebook Prophet (Taylor & Letham, 2018), Facebook Neural Prophet (Lewinson, 2020), Twitter S-H-ESD (Hochenbaum, Vallis, & Kejariwal, 2017), Etsy skyline (Etsy, 2015) and Numenta HTM (Numenta, 2015). These algorithms are very complex and often lack comprehensive third-party reviews and benchmarking. Recent developments in the field of machine learning and deep learning have led to introducing algorithms such as Donut (Xu, et al., 2018) and DeepAnT (Munir, Siddiqui, Dengel, & And, 2019). There is a lack of studies that compare the more recently introduced algorithms and in the papers the algorithms are introduced there is a lack of comparison to the traditional statistical methods that are commonly used for univariate timeseries. This might be due to inherent differences that exists between the fields of statistical modelling and algorithmic modelling mentioned by Breiman (Breiman, 2001).

Freeman et al. (Freeman, Merriman, Beaver, & Mueen, 2019) studied the performance of some of the more recently introduced timeseries anomaly detection methods using the Numenta benchmark datasets as test data. They also included SARIMA algorithm in the comparison

which is a version of ARIMA algorithm which has seasonality element incorporated to the model. In the study, they proposed a framework on how to intelligently choose an anomaly detection method based on the characteristics that the time series displays. These characteristics of the time-series are *seasonality*, *trends* and *concept drift*. If the timeseries has concept drifts, there is a chance that the anomaly detection algorithm identifies false positive anomalies. Concept drifts in time series mean that the normal behavior changes suddenly over time (Saurav, et al., 2018). Identifying concept drifts in time series automatically is a difficult task especially if the number of concept drifts in the dataset is unknown. One possible way to detect concept drifts in the dataset is using t-distributions where each new concept referred to as *run*. In order to detect the drifts posterior probability $\left(P(r_t|x_{1:t})\right)$ of the current run r_t's length at each timestep $x_i \; for \; i = 1, 2 \ldots t$ can be displayed. (Freeman, Merriman, Beaver, & Mueen, 2019) Example of this process can be seen from Figure 13.

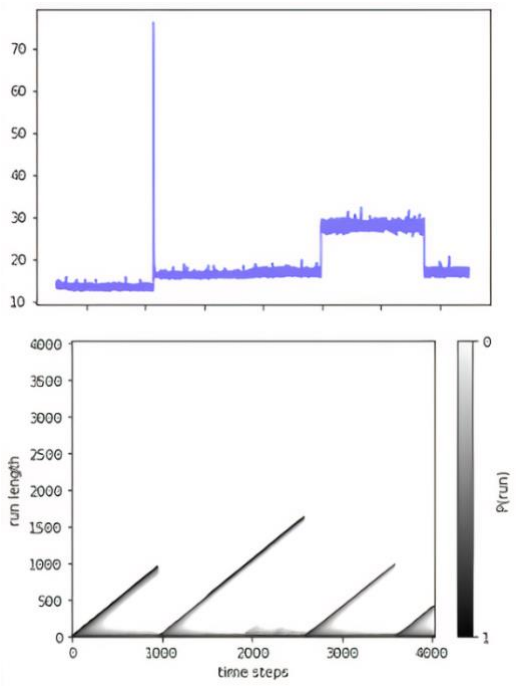


Figure 13. Time series concept drift example (Freeman, Merriman, Beaver, & Mueen, 2019)

In terms of handling the concept drifts in the timeseries there could be data preprocessing techniques to make the data continuous, but this could have an impact on the interpretability of the anomaly detection results. Also, concept drifts could be only removed on the past data, which would mean that only the training data could be cleaned out of them. However, many if

not all statistical and machine methods put more weight on the more recent past values in the forecasting, therefore the anomaly detection algorithms should be able to adapt to concept drifts fairly quickly.

The other two characteristics Freeman et al. proposed to be used on deciding which anomaly detection algorithm to use were trends and seasonality in the timeseries. In many studies these are removed from the timeseries as part of pre-processing steps (Makridakis, Spiliotis, & Assimakopoilos, 2018) (Zhang & Qi, 2005) while in others deseasonalization or detrending is not performed. The trend was documented to be removed in the comparison when using SARIMA. Since all the algorithms used in the study were able to handle seasonality, deseasonalization was not needed. From the study results it can be seen that some of the more recently introduced algorithms such as *Facebook Prophet* and *Donut* algorithms were able to perform better on timeseries that contained seasonality and concept drifts. (Freeman, Merriman, Beaver, & Mueen, 2019) Facebook prophet algorithm has built in mechanism to handle concept drifts in the timeseries data and can adapt to such changes (Taylor & Letham, 2018). The more complex and recently introduced algorithms seem to have better accuracy in some cases however this study had rather limited set of timeseries in comparison. (Freeman, Merriman, Beaver, & Mueen, 2019) Interesting finding in the study of Freeman et. al. is that using the characteristic of the timeseries could be used in the decision of which algorithm to choose for anomaly detection.

Numenta anomaly detection benchmark is an open-source repository that includes a framework for benchmarking anomaly detection algorithms using a set of currently 58 selected timeseries. The best performing algorithm used in the study is Numenta HTM which is an algorithm based on the computational theory of the neocortex. (Lavin & Ahmad, 2015) The benchmark mainly consists of the more recently introduced anomaly detection algorithms and lacks all statistical methods and many of the traditional machine learning algorithms that are commonly mentioned in literature. This makes it difficult to interpret the results in due to lack of algorithms included in the benchmark and also due to limited set of timeseries used in the study. Wu et. Al. compared the same Numenta HTM algorithm against ARIMA and in the study they found that Numenta HTM had significantly better modelling performance in terms of lower mean absolute percentage error compared to ARIMA (Wu, Zeng, Chen, & Tang, 2016). Lower mean absolute

error is however not directly connected to the anomaly detection accuracy since it only describes how well the created model fits into the data, the study does not include statistics on the actual anomaly detection classification accuracy.

Amarbayasgalan et. al. studied the performance of several machine learning based anomaly detection algorithms and also introduced RE-ADTS neural network based anomaly detection algorithm. The introduced RE-ADTS algorithm is an autoregressive based deep autoencoder model that utilizes time-series reconstruction. Time series reconstruction in the algorithm means that the RE-ADTS algorithm divides the timeseries into sub-sequences and uses these sub-sequences to determine if the value is an anomaly based on the previous values in the same sub-sequence. After dividing the algorithm into the sub-sequences the algorithm uses deep autoencoder model to learn the data distribution which is afterwards used to reconstruct time-series close to normal. If the value is an anomaly, the reconstruction error is higher than that of the normal data. The study used the same Numenta dataset and also included many of the algorithms that were included in the original Numenta benchmark. In this study, the evaluation metric was slightly different. Amarbayasgalan et. al. used the common metrics such as F-score and AUC score whereas Numenta anomaly benchmark uses its own algorithm score metric called normalized NAB score. From study results it can be seen that the RE-ADTS algorithm introduced in the study received the best performance in terms of F-score and AUC score. (Amarbayasgalan, Pham, Theera-Umpom, & Ryu, 2020) Interesting to note in the study Numenta HTM algorithm performed worse than Twitter ADVec algorithm while in the Numenta study this was other way around, might be due to a different version of the algorithm used or differences in hyperparameter optimization since the benchmark dataset should be the same. Since the RE-ADTS algorithm uses the sub-sequences of the timeseries to learn about the data distribution it can be expected that the algorithm is able to detect contextual anomalies well since the model training is only done with the context the datapoint exists, which means the previous *t* values in the timeseries.

Similar comparison on the more recently introduced algorithms was done by (Munir, Siddiqui, Dengel, & And, 2019). In this study, they also introduced a convolutional neural network (CNN) based algorithm, called Deep-learning based Anomaly detection approach (DeepAnT), to forecast time-series and detect anomalies based on the error of the prediction. Usually,

convolutional neural networks are used in more advanced data analysis tasks such as computer vision for object detection, classification and segmentation (Albawi, Mohammed, & Al-Zawi, 2017). While in classification of images with CNN the input data is two dimensional in case of time-series the CNN uses one dimensional input. This means that the convolution layers become one dimensional as well. (Munir, Siddiqui, Dengel, & And, 2019) Figure 14 contains the architecture of the introduced DeepAnT algorithm in the paper.

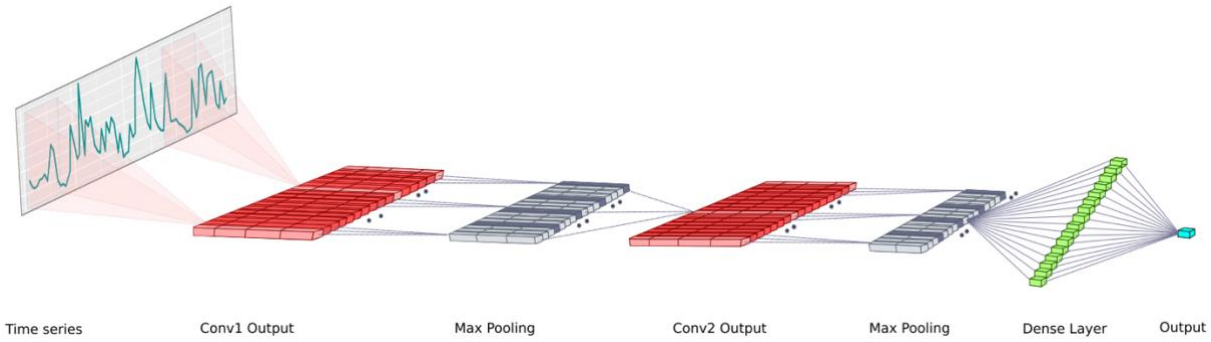


Figure 14. DeepAnT CNN architecture for time-series prediction (Munir, Siddiqui, Dengel, & And, 2019)

In the study Munir et. al. used three separate datasets and two of them were the same were the common Yahoo dataset and dataset from the Numenta anomaly detection benchmark. From the results it can be seen that DeepAnT was able to outperform in majority of the tested timeseries. The study used a subset of 20 timeseries from the Numenta anomaly detection benchmark and the DeepAnT algorithm got the best F-score in all the tests outperforming the Numenta, Twitter ADVec and other algorithms.

The DeepAnT type CNN algorithm was also compared in the studies from (Braei & Wagner, 2020) and (Amarbayasgalan, Pham, Theera-Umpom, & Ryu, 2020). In the study of Amarbayasgalan et. al. the DeepAnT algorithm was the second-best performing algorithm after the RE-ADTS neural network algorithm introduced in the study. In the study form Braei and Wagner DeepAnT type CNN was among the worst performing algorithm out of the 20 compared in the study. Table 4 contains summary of the different algorithms used in each study mentioned in this section.

Anomaly detection method	Method type	(Braei & Wagner, 2020)	(Makridakis, Spiliotis, & Assimakopoilos, 2018)	(Freeman, Merriman, Beaver, & Mueen, 2019)	(Lavin & Ahmad, 2015)	(Amarbayasgalan, Pham, Theera-Umpom, & Ryu, 2020)	(Munir, Siddiqui, Dengel, & And, 2019)
AR / MA / ARIMA Any autoregressive moving average model	Statistical	X	X	X			
Exponential smoothing	Statistical	X	X				
PCI Prediction confidence interval	Statistical	X					
Naïve 2 (Random walk model)	Statistical		X				
Theta	Statistical		X				
Prophet Facebook prophet	Statistical			X			
Twitter S-H-ESD Generalized ESD test for detecting anomalies	Statistical			X	X	X	X
Yahoo EDAGS (several variants)	Statistical					X	
Relative Entropy	Statistical					X	X
K-means	ML	X					
DBSCAN Density based spatial clustering	ML	X					
LOF Local Outlier Factor	ML	X					X
Random forest (or random forest variations)	ML	X			X		
SVM One class support vector machines	ML	X	X				
XGBoost Extreme gradient boosting	ML	X					
CART regression trees	ML						
GP Gaussian Process	ML		X		X	X	X
Bayes ChangePoint	ML					X	X
MLP Multi-layer perceptron	NN	X	X				
CNN Convolutional neural networks (e.g., DeepAnt)	NN	X				X	X
ResNet Residual neural networks	NN	X					
Wavenet	NN	X					
LSTM Long short term memory network	NN	X	X				
GRU Gated recurrent unit	NN	X					
Autoencoder	NN	X					
BNN Bayesian Neural Network	NN		X				
GRNN Generalized Regression Neural Networks	NN		X				
RBF Radial Basis Functions (RBF)	NN		X				
RNN Recurrent Neural Network (RNN)	NN		X	X			
Donut	NN			X			
Numenta HTM	NN				X	X	
RE-ADTS (several variants)	NN					X	
Etsy Skyline	Combination of several diff methods				X	X	X

Table 4 Anomaly detection methodologies used in the studies

The number of existing algorithms for anomaly detection in univariate timeseries is over exhausting and the algorithms listed in table 2 only scratched the surface of all the algorithms available for the purpose. Many of the presented algorithms have variations that may be able to perform better in some cases. Also, the results from the studies mentioned have conflicting results. In the studies from (Munir, Siddiqui, Dengel, & And, 2019), (Amarbayasgalan, Pham, Theera-Umpom, & Ryu, 2020) and (Lavin & Ahmad, 2015) there was a new algorithm introduced and in each of these studies the introduced algorithm had the best performance in labeling anomalies, which might be indicator of potential research bias. These algorithms also used data from the same benchmark dataset, although some of these studies used a subset of the total 58 timeseries in the Numenta benchmark dataset. All these three studies mention the Twitter S-H-ESD algorithm and the Numenta-HTM algorithm but in each study the ordering of the algorithm performance is different. The studies that introduced a new machine learning based method to anomaly detection also lacked the comparison to traditional statistical methods, which could have given a great baseline for the results since the performance of the machine learning based approaches tends to vary due to potential overlearning and ambiguous settings of optimal hyper parameters of each algorithm.

The studies from (Braei & Wagner, 2020) (Makridakis, Spiliotis, & Assimakopoilos, 2018) and (Freeman, Merriman, Beaver, & Mueen, 2019) included also statistical methods in the comparison. In the study from Freeman et. al. SARIMA was compared with several of the recently introduced methods and the performance of the SARIMA was very close to that of the other methods included the comparison although this study included a limited amount of timeseries in the comparison. The two most comprehensive studies in terms of size of the included benchmark datasets and included algorithms are the studies from (Braei & Wagner, 2020) and (Makridakis, Spiliotis, & Assimakopoilos, 2018) which showed results that the statistical methods were consistently outperforming the machine learning based methods in anomaly detection and forecasting.

Since the results of the mentioned studies give conflicting results, it is difficult to choose algorithms for the anomaly detection system described in chapter 1. A recently published paper from Wu and Keogh describes some of the issues in the current time-series anomaly detection benchmark studies by listing flaws in the commonly used benchmark datasets. The listed flaws

in the article are *triviality, unrealistic anomaly density, mislabeled ground truth* and *run-to-failure bias*. Due to these flaws Wu and Keogh claim in the paper that the many published comparisons of timeseries anomaly detection algorithms may be unreliable and more importantly much of the apparent progress in univariate anomaly detection algorithms may be illusionary. They also state that most of the recent research and interest in anomaly detection in timeseries comes from transferring the domain knowledge from the applications where deep learning has shown considerable success. Anomaly detection in univariate timeseries is however inherently much simpler problem and leveraging deep learning may not provide improvements in performance. Wu and Koegh state that they have not seen results that would indicate deep learning methods outperforming statistical methods consistently in any study. (Wu & Keogh, 2020)

The Occam's Razor rule that states back to 13th century states that "*Entities should not be multiplied without necessity*" which means that simplest explanation is usually the right one. (Thorburn, 1915) In the field of machine learning this would mean that one should prefer the model that gives similar results and is less complex. While statistical models can be complex, they are usually far less complex than machine learning and deep learning methods. In the study from Braei and Wagner the compute time of all statistical methods mentioned were far less that the compute time of the machine learning methods (Braei & Wagner, 2020).

In perspective of detecting anomalies in business metrics using a complex black box model would not necessarily give wanted results even if the accuracy of the model exceeds the accuracy of the statistical or simple rule-based methods. Interpretability of how the model ended up flagging the datapoint as an anomaly plays a big role in understanding the cause of the anomaly and what actions should be taken. In the whitepaper of Numenta anomaly detection system they even compared the complex neural network algorithm Numenta HTM to simple thresholds that are a commonly used in keeping track of anomalous values (Numenta, 2015). From the end user perspective, a simple threshold value may be more useful to inform about anomalous values in the business metric than any model-based method if the user knows that the metric should not get any values above that limit when the metric behaves normally. A simple threshold is the simplest version of an expert based anomaly detection system. There are however other methods how expert opinion can be used as part of an anomaly detection

framework. Expert opinion can be utilized by leveraging semi-supervised or supervised methods and one way is to add a supervised method on top of an unsupervised method based on the true positive flagged anomalies. Such method is described in section 3.5.

The classical statistical methods ARIMA, exponential smoothing and the variants of these two algorithms gave consistently great results compared with even the most complex recently introduced algorithms according to conducted literature review. Therefore, these are described in more detail in the following chapter. Since the Twitter S-H-ESD model and the Facebook prophet models were widely mentioned in the studies and these methods have easy to use and well documented libraries these methods are also described in high level.

3.4 Selected anomaly detection methods

In the previous section literature review was used to identify several models adequate for the anomaly detection tool. In this chapter these methods are described in high level. While the last section covered only model-based anomaly detection methods some simple rule-based methods are mentioned, since in the implemented anomaly detection tool rule-based methods were deemed to have most use cases.

3.4.1 Rule based anomaly detection methods

Rule based method means that there is no statistical or machine learning model to train and the categorization of normal and anomalies are done by comparing against past values. Rule based methods have the advantage of being the easiest to interpret from the end user perspective and do not require any knowledge of the timeseries modelling techniques. This however comes at the cost of often significantly worse anomaly detection accuracy. The following rule-based anomaly detection methods are shortly described: *Simple threshold, percentage change* and *Bollinger bands*.

Simple threshold

Defining a threshold for the anomalous values is the simplest possible way to perform anomaly detection. It is up to debate if simple threshold counts as an anomaly detection technique but in practice simple thresholds are widely used in the industry. In many cases, a simple threshold is enough to monitor a timeseries for anomalous values, say for example cases where there is a need to monitor timeseries since there is an absolute limit the timeseries values should not reach. From data engineering perspective, a useful simple threshold could be to monitor hourly data volume in a case where extreme variability in the data volume is expected, but hourly data volume should not drop under a certain level. Another case example where a simple threshold is useful would come from monitoring the player level pass rate in puzzle mobile games. In this case simple threshold could be used to monitor if a pass rate of single ingame puzzle decreases under a specified threshold which could have negative impact on the overall player experience.

For many cases where a simple threshold is in place a timeseries modelling approach would be a better option. A simple threshold is very likely to give false positive alarms after a while, this is due to the contextual attributes changing over time. The simple threshold rule set by the area expert only covers the current state and the changing contextual attributes require updates to the threshold value. If there is a need to constantly update the threshold limit, the cost of maintenance quickly justifies the usage of a modelling-based approach for the anomaly detection. Simple threshold is also completely unable to detect collective anomalies or other pattern type anomalies, which may also be present.

Obvious advantage for a simple threshold method is that it does not require discrete timeseries or any type of timeseries pre-processing. In some cases, business metric KPI may be stored as a single value and the aggregated historical time series values for the time series values are not available. For example, the KPI can be a result of a complex query that is updated daily and the historical query results are not stored in time series format. The simple threshold method could be still used to notify or alert the user on abnormal KPI values.

Percentage change

Comparing the percentage change of the value in the timeseries against the past value at defined timeframe is a simple method that is very intuitive from end user perspective. Formally, the anomalous values can be retrieved using equation:

$$e(y_t) = \begin{cases} 1\ , if\ \left|\frac{y_t - y_{t-\lambda}}{y_{t-\lambda}}\right| \geq\ \alpha\ , y_{t-\lambda} \neq 0 \\ 0\ , if\ \left|\frac{y_t - y_{t-\lambda}}{y_{t-\lambda}}\right| <\ \alpha\ , y_{t-\lambda} \neq 0 \end{cases}$$

Where $y_{t-\lambda}$ represents the past value the value y_t is compared against and λ represents the number of values between the two values, assuming the timeseries is discrete. Timeseries have different incoming data intervals, so λ needs to be defined separately. The percentage change threshold α can be used to set the sensitivity of flagging the anomalous values. Obvious benefit of a simple percentage change algorithm is the ability to easily take into account fixed length seasonality.

Obvious flaws for the percentage change method include inability to take into account different length seasonalities or trends. If the timeseries would be pre-processed to remove all seasonality or trend in the timeseries the interpretability of the method would greatly suffer, which is the main reason why it is favorable. Detrending the timeseries with differencing the values would make the percentage change essentially to compare the changes in variability, which would make the results difficult to interpret as well. In case of percentage change algorithm, the detrending could be done via fitting a linear trend model to the data and subtracting the fitted values or via standardizing the data by subtracting the mean of the data and dividing with the standard deviation. This way taking differences can be avoided and the scaled absolute level can be used in the equation. Percentage change is also only comparing the relative change, which means that the absolute change in the timeseries value might not be significant, but the relative change is high, this can be the case especially if timeseries values are close to zero.

Bollinger Bands

Bollinger bands or as commonly called trading bands is a commonly used method in finance to get insight on the price structure. The method was popularized by John Bollinger of Bollinger Capital Management in the 1980s. Bollinger Bands is one of the most popular methods used in technical analysis in the field of finance for a long time. In practice the formula for Bollinger bands is very simple. (Bollinger, 1992)

$$\sigma = \sqrt{\frac{\sum_{j=1}^{N}\left(X_j - \bar{\bar{X}}\right)^2}{N}}$$

$$\bar{\bar{X}} = \frac{\sum_{j=1}^{N} X_j}{N}$$

$$Upper\ band = \bar{\bar{X}} + \alpha\sigma$$

$$Middle\ band = \bar{\bar{X}}$$

$$Lower\ band = \bar{\bar{X}} - \alpha\sigma$$

Where X_j represents the timeseries values $\bar{\bar{X}}$ is the moving average and σ is the standard deviation over *N* periods. By default, Bollinger bands are plotted two standard deviations above or below a simple moving average, this would mean the α value would get the value of 2. (Bollinger, 1992) For anomaly detection purposes the sensitivity of the model can be altered by adjusting the bands width by changing the parameter α. Another parameter in Bollinger bands is the timeframe *N* which is the number of observations included in the Bollinger bands moving average calculation. Figure 15 contains an example of Bollinger bands fitted on Rovio stock from 2019 June to 2021 March timeframe.

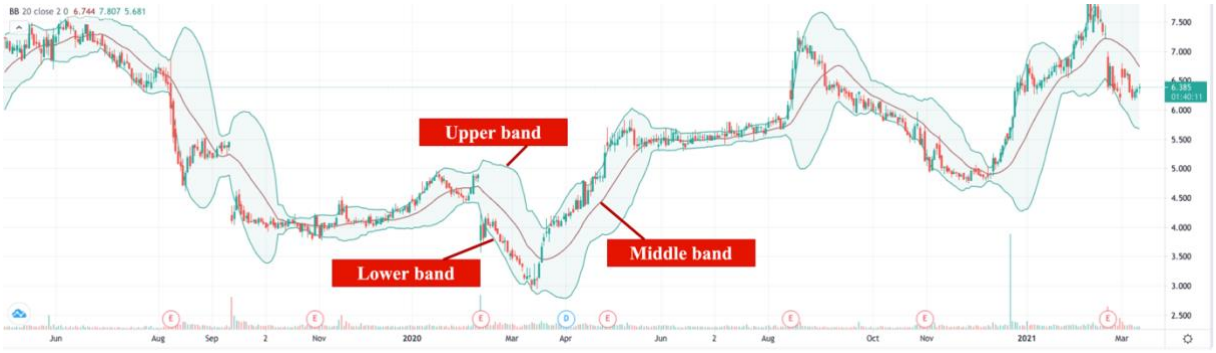


Figure 15. Bollinger bands example

While the most common use case for Bollinger bands is in the technical analysis of financial instruments it is also a viable method to be used in anomaly detection. For example, Patti et. al. studied the usage of Bollinger bands in detecting anomalous patterns in communication data in order to predict terrorist cell activity (Patti, Belov, Craven, & Thayer, 2009). Likewise, with the other rule-based algorithms Bollinger bands require very little computational power which means detecting anomalies in a massive scale does not require large computing clusters and the fitted Bollinger bands can be retrieved in very little time. In practice Bollinger bands can be used for anomaly detection by simply flagging the values outside the bands as anomalies. Since Bollinger bands is based on simple moving average and standard deviation it cannot take into account any seasonality in the data or long-term trends. Even though these drawbacks Bollinger bands can still be used to detect "breakouts" as anomalies in the timeseries, it has been widely used for the same purpose in technical trading.

While the presented three rule-based anomaly detection methods have disadvantages in anomaly detection accuracy they are presented and implemented in the anomaly detection tool mainly in order to provide the anomaly detection tool end users intuitive and easy to interpret way of monitoring business metric behavior. Providing anomaly detection methods which are easy to understand and interpret might increase the usage of the anomaly detection among business user's corporation wide. After the anomaly detection tool end user has received false positive anomalies, the user might have the time and interest to take a look at the more advanced model-based anomaly detection methods that provide better accuracy. While setting up monitoring using either rule-based method or model-based method takes about the same time, the interpretation of anomaly detection results become more difficult.

3.4.2 Model based anomaly detection methods

In model-based anomaly detection, a statistical or algorithmic model is trained from the training dataset and the trained model is used to predict proceeding values in the timeseries. In the model-based approach the question arises when to do the training or is training done each time the scheduled anomaly detection process takes place? Or if anomaly detection is a real time process should the model be retrained every 10 or every thousand datapoints? In some cases, the behavioral attributes of the timeseries vary quicker and in other cases the same trained model can provide effective results after a long period of time. Model retraining becomes an issue when anomaly detection is done at large scale. Large scale can mean thousands to millions of timeseries to check for anomalies. The model retraining should be done between specified retraining intervals *w*. The retraining interval can be dynamic based on for example the scaled difference in the absolute value, which would trigger the model training process.

In terms of computational resources needed, decreasing retraining interval is expected to increase the computational requirements in exponential manner. For example, if training a model is expected to require a hundred abstract computational units and the discrete equidistant time-series gets data every minute the obvious relation from retraining interval to computational intensity per hour can be seen in Figure 16.

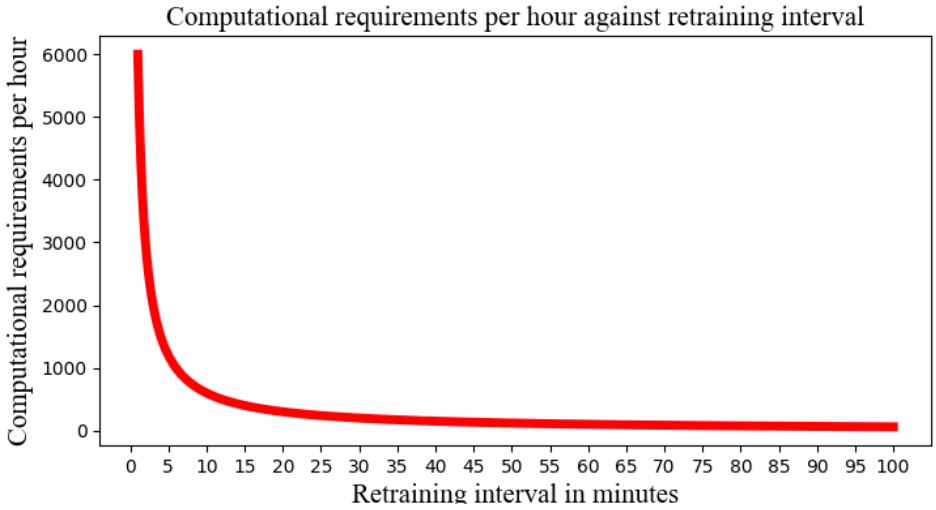


Figure 16. Computational intensity of anomaly detection

This does not take into account that there are possible ways of feeding more training data to the already trained model, however such techniques are not possible with the chosen algorithms. The chosen model-based algorithms are based on statistical timeseries modelling and according to literature study the computational requirements of the statistical methods are way lower than the requirements of the algorithmic (machine learning based) models. If the anomaly detection is done on an hourly scheduled basis, it is expected that retraining the models every hour does not require excessive computational requirements.

For every model-based method, the anomaly detection process is similar on high level. Only difference is if the model-based method is scheduled with a retraining interval or not. The high-level algorithms of model-based methods without a retraining interval can be seen from Figure 17 and the algorithm with a static retraining interval is presented at Figure 18. In these examples' anomalies are looked on new arriving values for clarity, the process is similar if anomaly detection is performed in retrospective manner.

Algorithm 1: Statistical anomaly detection method

Given : Time series $x_i \in X$, Testing start point t'
Result: Array of anomaly scores a
if *New data has arrived in* X **then**
  Predict $x_i(t')..., x_i(n)$; thus obtain $\hat{x}_i(t')..., \hat{x}_i(n)$;
  for $\hat{x}$ *in* $\hat{x}_i(t')..., \hat{x}_i(n)$ **do**
    Calculate anomaly score a_i for $\hat{x}_i$
  end
end

Figure 17. High level statistical anomaly detection method algorithm

Algorithm 2: Model based anomaly detection method

Given : Time series $x_i \in X$, Retraining Interval w, Training time period t, Testing start point t', Anomaly detection model M
Result: Array of anomaly scores a
if *New data has arrived in* X **then**
  if *Retraining interval* w *has passed since latest model training* **then**
    Update the Model M on the first t values of the time series X
  end
  Apply M to predict $x_i(t')..., x_i(n)$; thus obtain $\hat{x}_i(t')..., \hat{x}_i(n)$;
  for $\hat{x}$ *in* $\hat{x}_i(t')..., \hat{x}_i(n)$ **do**
    Calculate anomaly score a_i for $\hat{x}_i$
  end
end

Figure 18. High level model-based anomaly detection algorithm with training interval

Autoregressive moving average methods

Autoregressive moving average (ARMA) model is a statistical time series modelling method originally described by Peter Whittle back in 1951. ARMA models provide a method to describe a stationary stochastic process in terms of two polynomials, the autoregression (AR) and moving average (MA) (Whitle, 1951). While the method is old according to the conducted literature review ARMA model still performs well in univariate timeseries modelling against the more recently introduced algorithms given that seasonality is taken account via data pre-processing or by modifying the classical model by incorporating seasonality to the model (SARMA).

The basic formula for ARMA model can be written as (Brooks, 2014):

$$(1 - \sum_{i=1}^{p} \phi_i y_{t-i}) y_t = \mu + (1 + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}) + \varepsilon_t$$

Or as:

$$\phi_p(L) y_t = \mu + \theta_q(L) + \varepsilon_t$$

Where L is the lag operator which causes the observation to be shifted *n* times backwards. Formally lag operator can be expressed as:

$$L^n y_t = y_{t-n}$$

In the formula $\phi_p(L)$ is the autoregressive polynomial and $\theta_q(L)$ is the moving average polynomial which can be formally expressed as (Brooks, 2014):

$$\phi_p(L) = 1 - \phi_1 L - \phi_2 L^2 - \cdots - \phi_p L^p$$
$$\theta_q(L) = 1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q$$

In ARMA(*p, q) p* is the number of autoregressive lags included in the polynomial and respectively *q* is the number of moving average lags included. MA part of the equation can be seen as a linear combination of the last *q* moving average prediction errors $\{\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}\}$. The autoregressive part is a model where the y_t is dependent on the values that the variable took in previous periods plus an error term. (Brooks, 2014)

Training the autoregressive part can be done simply using linear equations with least-squared regression. Moving average process can be trained by calculating the q length moving average values and optimizing sequentially each value coefficient θ_t starting from the first lag and moving to backwards p number of lags. While training the coefficient is an unambiguous process deciding correct number of lags is not, and there exists several methods that can be used for optimization. (Braei & Wagner, 2020) The number of lags can be defined using a visual approach with autocorrelation function (ACF) and partial autocorrelation function (PACF) plots but in order to determine the number of lags computationally information criteria can be used. There are several information criteria that can be used, three of these are Akaike's (AIC) Schwarz's (SBIC) and Hannan-Quinn (HQIC) information criterions. (Brooks, 2014, ss. 273-276)

$$AIC = \ln(\hat{\sigma}^2) + \frac{2k}{T}$$

$$SBIC = \ln(\hat{\sigma}^2) + \frac{k}{T}\ln T$$

$$HQIC = \ln(\hat{\sigma}^2) + \frac{2k}{T}\ln\ln T$$

Here $\hat{\sigma}^2$ is the residual variance of the fitted model, k is the total number of parameters estimated ($p + q + 1$) and T is the number of observations in the training timeseries. Different number of lags in AR and MA polynomials can be tested through iterative process with the objective to minimize the value of the chosen information criteria. After the number of lags is specified for the timeseries it is unlikely that different number of lags would provide lower information criteria results when the model is retrained as part of the scheduled anomaly detection process. Therefore, iterating through different lag combinations is not required in the retraining process and only the coefficients ϕ_p and θ_q can be updated, however if the underlying data generation process changes over time and the anomaly detection gives incorrect results tuning the number of lags in the model can increase the accuracy.

Autoregressive model requires stationary timeseries, therefore it is important to pre-process the data. Section 3.2 contained the most common data-preprocessing methods for univariate

timeseries. There are however ARMA method variations ARIMA and SARIMA which incorporate detrending and seasonality removal into the model. Removing seasonality from the timeseries is not required in order to achieve stationary timeseries, but handling seasonality can drastically increase the anomaly detection accuracy.

In Autoregressive Integrated Moving Average (ARIMA) model the differencing technique described earlier is incorporated to the model. Differencing can be incorporated to the model by adding differentiation factor *d* to the model ARIMA(*p, d, q)*. In most of the cases differencing once is enough to achieve stationarity. Formally, ARIMA model can be achieved by adding $(1-L)^d$ term to the plain ARMA formula (Brooks, 2014):

$$\phi_p(L)(1-L)^d y_t = \mu + \theta_q(L)\varepsilon_t$$

Seasonal Autoregressive Integrated Moving Average (SARIMA) models can be used effectively to describe time series that exhibit non-stationary behaviors both within seasons and across seasons. (Wang, Feng, & Liu, 2013) Formally SARIMA model can be expressed as:

$$\phi_p(L)\Phi_P(L^s)^d(1-L^s)^D y_t = \mu + \theta_q(L)\Theta_Q(L^s)\varepsilon_t$$

Where the seasonal polynomials $\Phi_P(L)$ and $\Theta_Q(L) =$ can be written as

$$\Phi_P(L) = 1 - \Phi_1 L - \Phi_2 L^2 - \cdots - \Phi_P L^P$$
$$\Theta_Q(L) = 1 + \Theta_1 L + \Theta_2 L^2 + \cdots + \Theta_Q L^Q$$

Where *P* is the order of seasonal autoregression, *D* is the number of seasonal differences, *Q* is the order of seasonal moving average and *S* is the length of the season. In terms of determining the seasonal part lags *P* and *D* the same trial-and-error approach via comparing information criterion can be used. Length of the season can usually be interpret by visual observation of the raw timeseries and given as hyperparameter to the model. Including the seasonal element requires the timeseries to be discrete since the previous values since the seasonality polynomials include *S* number of past observations to the polynomial.

Overall, according to conducted literature review ARIMA models have great performance in anomaly detection tasks and the method is able to detect different types of anomalies in various application domains. While being very effective at anomaly detection, ARIMA requires very little computing power which makes it a viable option on detecting anomalies on large scale. The studies from (Braei & Wagner, 2020) and (Freeman, Merriman, Beaver, & Mueen, 2019) showed that ARIMA is unable to detect contextual anomalies in some cases, and if collective anomalies are expected another more sophisticated algorithm can be chosen. ARIMA algorithm is provided as part of various popular python machine learning and data analytics libraries such as Statsmodels (Statsmodels, 2021).

Exponential smoothing based methods

Exponential smoothing is another classical timeseries modelling method that was able to detect anomalies in various datasets with great accuracy according to studies made by (Braei & Wagner, 2020) (Makridakis, Spiliotis, & Assimakopoilos, 2018). Exponential smoothing is believed to be first introduced by Robert Brown in fifties when he was researching methods for Navy inventory systems. Formally the simplest form of exponential smoothing can be presented as (Brown R. , 1957):

$$\hat{y}_{t+1} = \alpha \sum_{k=0}^{t-1} (1-\alpha)^k y_{t-k} + (1-\alpha)^t y_0 , \qquad \text{where } 0 \leq \alpha \leq 1$$

Or with recursive formula as:

$$\hat{y}_{t+1} = \alpha y_t + (1-\alpha) y_{t-1}$$
$$\hat{y}_{t+1} = \alpha y_t + \alpha(1-\alpha) y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \ldots + \alpha(1-\alpha)^N y_{t-N}$$

Where α is the *smoothing factor* which means that larger values of α reduce the amount of smoothing and $\alpha = 1$ would simply mean the next value is the latter observation. Low α values have greater smoothing effect and make the $\hat{y}_{t+1}$ less responsive to more recent changes. From the formula it can be seen that the weight of the past value becomes exponentially lower higher the *N* is, hence the name exponential smoothing. Optimization of the smoothing factor can be done by minimizing the sum of squared residuals. The optimization is however non-linear

minimization problem which can make the computational requirements of exponential smoothing algorithm optimization more demanding than ARIMA models. (Brown R. , 1963)

The simplest exponential smoothing model results is easy interpret and according to studies by (Braei & Wagner, 2020) and (Makridakis, Spiliotis, & Assimakopoilos, 2018) the methodology gives accurate predictions on univariate timeseries compared to more complicated models. Simple exponential smoothing however works only well for stationary data, which means that trend and seasonality do not exist in the timeseries or data pre-processing has been done to capture those elements. Likewise with ARIMA model, there are variations of exponential smoothing model that take into account trend and seasonality. These variations are called double and triple exponential smoothing or often Holt's linear trend method and Holt-Winter's seasonal method which can capture both trend and seasonality from the data. (Gardner, 1985)

The simple exponential smoothing model can be extended to incorporate trend element. Methodology to achieve this is believed to be first introduced by Charles Holt in 1957. Incorporating trend in the model is easy to understand when the model is split into the simple exponential smoothing (level equation) and trend parts (Holt, 1957). Formally the exponential smoothing with additive trend element can be expressed as (Gardner, 1985):

$$
\begin{aligned}
Forecast\ equation \quad & \hat{y}_{t+h} = \mathrm{I}_t + hb_t \\
Level\ equation(SES) \quad & \mathrm{I}_t = \alpha y_t + (1-\alpha)(\mathrm{I}_{t-1} + b_{t-1}) \\
Trend\ equation \quad & b_t = \beta(\mathrm{I}_t - \mathrm{I}_{t-1}) + (1-\beta)\, b_{t-1}
\end{aligned}
$$

Here I_t denotes the level of the series at time *t* when trend has been taken into account and b_t denotes the estimate of the trend (slope) of the timeseries at time *t*. Parameter *h* is the number of steps ahead forecasted value is predicted, meaning h-steps ahead forecast is equal to the last estimated level plus *h* times the last estimated trend value. Parameter β denotes the smoothing factor for the trend. Likewise with the simple exponential smoothing the I_t is simply a weighted average of the observations y_t and the one step ahead training forecast given by $\mathrm{I}_{t-1} + b_{t-1}$. The trend equation is weighted average of the estimated trend based on $\mathrm{I}_t - \mathrm{I}_{t-1}$ and b_{t-1}, which denotes the previous estimate of the trend. (Holt, 1957)

The double exponential smoothing method that adapts to trend was later developed further by incorporating seasonality in the algorithm. The exponential smoothing variation that has both trend and seasonal elements incorporated is often called triple exponential smoothing or Holt-Winter's since the method is believed to be first introduced by Peter Winters in 1960 (Winters, 1960).

The Holt-Winter's method has two separate alternatives to incorporate seasonality in the model, the additive and multiplicative methods. The additive method is suitable when the seasonal variations are roughly constant throughout the timeseries and multiplicative methods is preferred when seasonal variations are changing proportionally to the level of the series. (Gardner, 2006) In other words, if the seasonal effect is relative to the absolute value of the timeseries the multiplicative method should be used. In additive method the seasonal component is expressed in absolute terms in the scale of the timeseries, which means that in the equation the series is seasonally adjusted by subtracting the seasonality component. Therefore, within each year, the seasonal component in additive method will add up to approximately zero. In the case of the multiplicative model, the seasonal component is expressed in relative terms and the timeseries seasonality is adjusted by dividing with the seasonal component.

Formally the additive Holt-Winter's method can be expressed as (Gardner, 2006):

$$
\begin{aligned}
Forecast\ equation \quad & \hat{y}_{t+h} = \mathrm{I}_t + hb_t + s_{t+h-m+1} \\
Level\ equation(SES) \quad & \mathrm{I}_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\mathrm{I}_{t-1} + b_{t-1}) \\
Trend\ equation \quad & b_t = \beta(\mathrm{I}_t - \mathrm{I}_{t-1}) + (1-\beta)\, b_{t-1} \\
Seasonality\ equation \quad & s_t = \gamma(y_t - \mathrm{I}_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}
\end{aligned}
$$

Here s_t represents the seasonal component and γ is the exponential smoothing factor for seasonality. The parameter *m* represents the frequency of the seasonality in the timeseries. For example, if the timeseries had quarterly seasonality and the timeseries has observation for each day, *m* would receive the value of 90. The seasonal component is a weighted average between the current seasonal index and the seasonal index of the same season *m* time periods ago.

In practice most of the time the seasonal effect is multiplicative. For example, if the total in app purchases are 20% higher during Saturday and Sunday than weekdays the seasonality is multiplicative in nature. Formally the multiplicative Holt-Winter's method can be expressed as (Gardner, 2006):

$$\begin{aligned} Forecast\ equation \quad & \hat{y}_{t+h} = (\mathrm{I}_t + hb_t)s_{t+h-m+1} \\ Level\ equation(SES) \quad & \mathrm{I}_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(\mathrm{I}_{t-1} + b_{t-1}) \\ Trend\ equation \quad & b_t = \beta(\mathrm{I}_t - \mathrm{I}_{t-1}) + (1-\beta)\, b_{t-1} \\ Seasonality\ equation \quad & s_t = \gamma \frac{y_t}{\mathrm{I}_{t-1} + b_{t-1}} + (1-\gamma)s_{t-m} \end{aligned}$$

The simple exponential smoothing, and its variants are widely used in a variety of application domains mainly because of the simplicity and adaptability to different datasets. The algorithm is included in ML libraries such as TensorFlow and Statsmodels (TensorFlow, 2021) (Statsmodels, 2021)

3.4.3 Anomaly detection frameworks

In the previous chapters 3.4.1 and 3.4.2 several algorithms for anomaly detection were presented. When using these methods, it is up to the user to figure out what kind of data re-processing methods should be used and what algorithm is suitable for each dataset. It is also up to the user to decide how to implement the algorithm in practice, there are various libraries that include the algorithms. Instead of hand-picking an algorithm and determining the data pre-processing steps one can use more complete forecasting frameworks that can be used to get timeseries more predictions easily. Since anomaly detection is in essence a timeseries prediction task, these frameworks can be used in anomaly detection as well.

In this study, a forecasting framework means that the methodology contains common data-preprocessing steps as well as algorithm that would be suitable for predicting values for various types of timeseries. In section 3.3 literature review was conducted in the existing studies that compared the performance of some of the more complete prediction frameworks. Many of these studies included Twitter's anomaly detection library which was performing well against the

counterparts. Facebook prophet is another forecasting library that has gained popularity, however there are fewer academic papers that compare its performance. While using these frameworks is easy it is worthwhile to understand how the timeseries predictions are actually done, therefore in this section these frameworks are described briefly.

Twitter's anomaly detection

Twitter developed and open sourced a methodology for anomaly detection to cope with the anomaly detection problem in the context of social networking data. Since the volume of social networking data is enormous and the data inherently contains seasonal and trend components there was a need to develop a statistical anomaly detection algorithm that can be used to detect anomalies in scalable cloud infrastructure. The existing methodologies were deemed to be not applicable for this purpose. In the paper Twitter introduces two algorithms, the Seasonal ESD (S-ESD) and Seasonal Hybrid ESD (S-H-ESD) algorithms. (Hochenbaum, Vallis, & Kejariwal, 2017)

In high level both Twitter algorithms are fairly simple. The algorithm decomposes the timeseries into seasonal, residual and trend and then applies technique called Extreme Studentized Deviate (ESD) to detect anomalies. Figure 19 contains S-ESD algorithm description on high level. (Hochenbaum, Vallis, & Kejariwal, 2017)

Algorithm 3: Twitter Seasonal Extreme Studentized Deviate S-ESD

Required: $x_i \in X$ = Time series
n = Number of observations in X
k = max anomalies (Number of iterations in ESD)
$k <= (n * 0.49)$

Result: X_a Vector where each element is a tuple (timestamp, anomaly score)

Steps

1. Perform STL decomposition variant for the timeseries and extract seasonal S_x component
2. Compute median $\widetilde{X}$
3. Compute residual component
   $R_x = X - S_x - \widetilde{X}$
4. Detect anomalies in the residuals component using ESD
   $X_a = ESD(R_x, k)$

return X_a

Figure 19. Twitter S-ESD Algorithm

In practice the variable *k* max anomalies should be defined to be as low as needed to get only the true positive anomalies, this is however completely dependent on the context of the timeseries meaning that in some use cases the algorithm should be more sensitive in flagging datapoint as an anomaly. The first step of S-ESD algorithm is STL decomposition which is acronym for "Seasonal and Trend decomposition using Loess". Loess is a methodology for estimating nonlinear relationships. The STL decomposition method was first introduced back in 1990 by Robert Cleveland, William Cleveland, McRae and Terpenning (Cleveland, Cleveland, McRae, & Terpenning, 1990) .

STL is a robust approach to time series decomposition that can handle any type of seasonality in the data since the seasonal component is allowed to change over time. STL is robust to outliers in the data which makes it a great decomposition method for timeseries that contain anomalous values since the anomalies will not affect the estimates of the trend and seasonal components. If used correctly, the anomalous behavior in the timeseries is completely captured with the residual R_x component. Formally residual is estimated as: (Cleveland, Cleveland, McRae, & Terpenning, 1990)

$$R_x = X - T_x - S_x$$

The STL algorithm consists of an inner loop that derives the trend T_x, seasonal S_x and residual R_x components and an outer loop that increases the robustness of the algorithm with respect to anomalies. The inner loop operates on sub-cycles of the timeseries, which comprise of the values at each position of the seasonal cycle. For example, if a timeseries has yearly seasonality for each month, the first sub-cycle in the series would contain January values, the second February, and so forth. In the inner loop the trend component is identified using a simple moving average filter that smooths– the sub-cycle series in order to further derive the seasonal component from it. The seasonal component is estimated using LOESS (locally estimated scatterplot smoothing) that allows the decomposition to fit more complex functions than the simpler decomposition additive or multiplicative approaches. The STL decomposition iteratively converges on the decomposition until the difference between iterations is lower than a specified threshold. (Hochenbaum, Vallis, & Kejariwal, 2017)

The twitter algorithm uses STL variant where instead of calculating trend component T_x for each timeseries sub-cycle a simple median $\tilde{X}$ is used for the whole timeseries to derive the residual component. Formally the STL variant can be expressed as (Hochenbaum, Vallis, & Kejariwal, 2017):

$$R_x = X - \tilde{X} - S_x$$

Using the median was found to be more efficient in some cases where regular STL decomposition yielded spurious anomalies in the residual component which were not present in the original timeseries, meaning false positives. (Hochenbaum, Vallis, & Kejariwal, 2017) This technique might be effective when there is no significant long-term trend in the dataset, however according to the conducted literature review in chapter 3.3 the twitter algorithm was performing well with a variety of different timeseries.

After the residual component has been extracted, the twitter uses Extreme Studentized Deviate (ESD) test to detect the anomalies. The ESD test is a well-known test that can be used to detect on or more outliers in univariate dataset that follows approximately normal distribution. Given the upper bound *k* for the number of anomalies in the dataset the ESD test has the following hypothesis (Rosner, 1975):

H_0: The are no outliers in the dataset
H_a: There are up to *k* anomalies in the dataset

The following test statistics is used for the *k* most extreme values in the data set:

$$C_k = \frac{\max_k |x_k - \bar{x}|}{s}$$

In the formula $\bar{x}$ denotes the sample mean and *s* is the sample standard deviation. After the test statistics for the values are retrieved these values are compared with critical values to determine if the datapoint is anomalous $C_k > \lambda_k$. The critical values are calculated with formula:

$$\lambda_k = \frac{(n-k)t_{p,n-k-1}}{\sqrt{(n-k-1+t^2_{p,n-k-1})(n-k+1)}}$$

Here *n* is the number of observations and $t_{p,v}$ represents t-distribution with *v* degrees of freedom and $p = 1 - \frac{\alpha}{2(n-1+1)}$. The anomaly flagging methodology with ESD is iterative, when a datapoint is marked as an anomalous the threshold is computed again after the datapoint with highest C_k is removed from the dataset. This process is repeated *k* times until the test statistics value is lower than the critical value for all datapoints. (Rosner, 1975)

The ESD method is able to detect different types of anomalies well but this methodology does not perform as well in datasets which contain a higher percentage of anomalous values. For this reason, the second twitter algorithm variant Seasonal Hybrid ESD (S-H-ESD) uses median and Median Absolute Deviation (MAD) in the test statistics formula. Using median and MAD puts less weight on the anomalous values in the test statistics and this can lead to higher model accuracy. (Hochenbaum, Vallis, & Kejariwal, 2017)

Since the twitter branded anomaly detection method is available as open-source library and it is based on well-known and widely used STL decomposition and ESD test, it is a viable alternative to be used to do anomaly detection in large scale. The methodology is also mentioned in several academic studies and according to the studies mentioned in chapter 3.3 the algorithm performs well against other statistical and ML based methods.

It is worthwhile to mention that the S-ESD and S-H-ESD algorithms differ from other algorithms in this study since they are not based on anomaly score which would be calculated from the distance between the predicted and actual value, instead the statistical ESD test is used to differentiate the anomalous values from the rest of the values. The S-ESD algorithm cannot be used for time series predictions. For the other algorithms confidence interval based on anomaly score and the forecasted values can be plotted with the actual values, which provides an intuitive approach for the business user to see anomaly detection results. This is not possible to do with the twitter algorithms, since only the flagged anomalous values can be visualized. From anomaly detection tool end user perspective this might be confusing since it might be

more difficult to visually answer to the question: "*How close to anomaly was this specific datapoint?*". The difference between test statistics and the corresponding critical value can be represented, but this does not give the same information to end user.

Facebook Prophet model

Facebook prophet is an open-source forecasting library published by the Facebook corporation. A distinguishing feature for the Prophet model is the "analyst-in-the-loop" approach to modelling, meaning that the algorithm is designed to have the ability to incorporate expert opinion into the timeseries modelling problem in order to achieve higher accuracy forecasts. While parameter tuning is also possible in the other algorithms mentioned previously with regard to for example the seasonality length, Facebook Prophet model makes incorporating expert opinion easy in the library level. According to Facebook the analyst-in-the-loop design is effective in taking into account the deep domain expertise of the business metric users who necessarily do not have a lot of experience in time series modelling, see Figure 20. (Taylor & Letham, 2018)

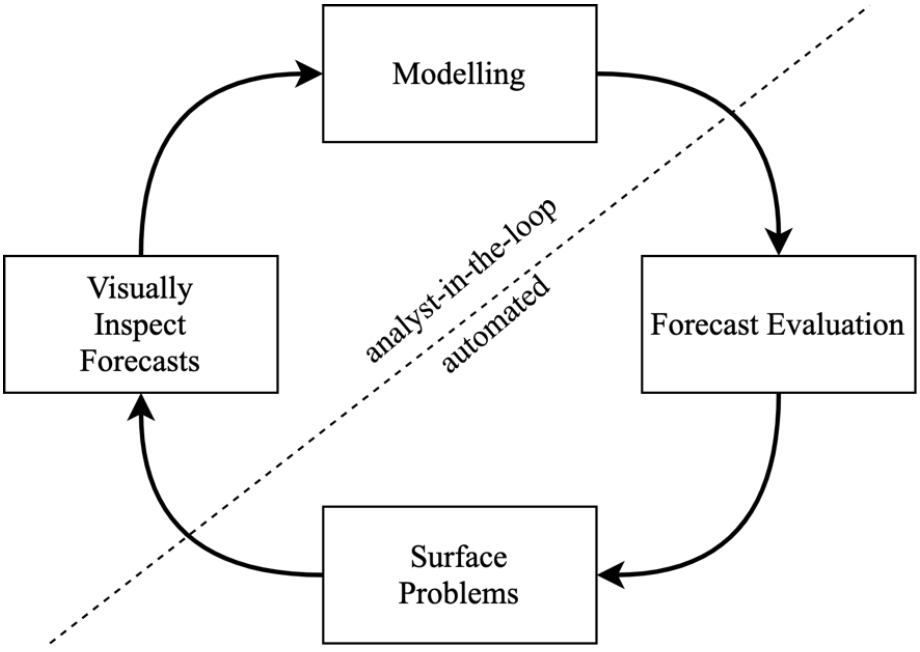


Figure 20. Analyst-in-the-loop modelling

In the Prophet model analyst can apply their expertise in the following parameters when optimizing the model (Taylor & Letham, 2018):

1. *Capacities*: Analyst can specify carrying capacity for the saturating growth model. For example, analyst can have external data for the total available market size
2. *Changepoints*: The Prophet model can take timeseries changepoints as input and take those into account in modelling. A changepoint in timeseries can be any expected change in the behavioral or contextual attributes of the timeseries. For example, major change in user acquisition investments or a campaign in mobile game can be seen as change points.
3. *Holidays and seasonality*: The Prophet model can take into account known holidays. For example, there can be expected to be yearly seasonality on New Year's Eve.
4. *Smoothing parameters*: The Prophet algorithm contains three smoothing parameters which can be used to smooth seasonality effect, holiday effect and trend.

Similarly to Twitter S-ESD algorithm, Prophet also uses decomposable time series modelling method. In Prophet algorithm the timeseries contains three main components: trend, seasonality and holidays. Formally this can be expressed as:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

In the formula *g(t)* represents the trend function that models the non-periodic changes in the values of the timeseries, *s(t)* represents the seasonal component and *h(t)* represents the effects of holidays which might occur on an irregular schedule over one or more days. The error term ϵ_t represents any changes which are not accommodated by the model, the values of the error term are normally distributed. (Taylor & Letham, 2018)

The Prophet model is inherently different from the ARIMA and exponential smoothing methods introduced earlier since the Prophet model does not explicitly take in to account the temporal dependence structure in the data and instead frames the forecasting problem as a curve-fitting exercise. Since the temporal structure is not used in Prophet algorithm this means that the algorithm does not require the timeseries data to have measurements regularly spaced,

which also means that interpolation for missing values is not required. This means that Prophet model can be used in a variety of timeseries without any preprocessing needed. This also makes the Prophet algorithm easy to compute, meaning that analyst can tweak the model parameters and get the fitted values and forecast results in very short time. (Taylor & Letham, 2018)

The trend component in Prophet algorithm is calculated using a logistic growth model which is tweaked to incorporate changepoints and time-varying capacity to the equation. Logistic growth model is a classical modelling technique which can be used to model population growth in natural ecosystems, for example it was used to model population growth in United States back in 1920 (Pearl & Reed, 1920). Formally the piecewise logistic growth model used in Prophet algorithm is (Taylor & Letham, 2018):

$$g(t) = \frac{C(t)}{1 + e^{-(k+a_j(t)^T\delta)(t-(m+a(t)^T\gamma_j))}}$$

where

$$a_j(t) = \begin{cases} 1, if\ t \geq s_j \\ 0, otherwise \end{cases}$$

$$\gamma_j = (s_j - m - \sum_{l<j} \gamma_l)(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \leq j} \delta_l})$$

Where *C(t)* is the time-varying capacity, *k* is the growth rate and *m* an offset parameter. Unlike in the classical logistical growth model the carrying capacity is not fixed. For example, in context of mobile games the total market size of mobile games is growing and in modelling the carrying capacity can be expressed to be time-varying capacity *C(t)* instead of constant *C*.

The logistical growth component in the Prophet trend component is effectively taking into account concept drift using changepoints in the dataset. See Figure 13 in chapter 3.3 for an example of concept drift in timeseries. By explicitly incorporating changepoints in the logistic growth equation, the growth rate is allowed to change. The changepoints can be defined by the analyst using Prophet algorithm for increased modelling accuracy or the Prophet algorithm can detect the changepoints on its own. The parameter s_j is a vector of changepoints in the model and δ_j is the change in rate that happens at time s_j. Therefore, the growth change rate at any

time t is the base change rate k plus all the change rate adjustments up to that point: $k + \sum_{j:t>s_j} \delta_j$. (Taylor & Letham, 2018)

The Problem library also contains linear trend with changepoints option instead of logistic growth which can be used if the forecasting problem does not exhibit saturating growth. Formally the linear model can be expressed as (Taylor & Letham, 2018):

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma_j)$$
$$\gamma_j = -s_j\delta_j$$

Where k is the base growth rate as in the logistic grow model, δ has the rate adjustments and m is the offset parameter. Since the Prophet is an open-source project, community has also proposed versions where different growth models are used.

Modelling seasonality in the Prophet algorithm is done using Fourier series. Fourier series are used in large spectrum of fields in sciences and engineering to represent how series can be decomposed into oscillatory components. The process of modelling an oscillatory process with Fourier series is called Fourier analysis. Fourier series is a periodic function composed of harmonically related sine and cosine functions combined by weighted summation. With appropriate weights one cycle of the summation can be made to approximate an arbitrary function in the corresponding interval. (Yitzhak, 1976) Prophet algorithm uses the standard Fourier series without the intercept term since the intercept is modelled with the trend term. Formally the Fourier series can be expressed as:

$$s(t) = \sum_{n=1}^{N} (a_n \cos(\frac{2\pi nt}{P}) + b_n \sin(\frac{2\pi nt}{P}))$$

Where P is the regular period the time series is expected to have, for example if there is yearly seasonality P = 365.25 or for weekly seasonality P = 7. Parameter N is the number of observations in the season and parameters a_n and b_n represent the weights of the oscillatory behavior. The weights can be represented with a single vector $\beta = [a_1, b_1, \ldots, a_N, b_N]$ which

contains weights for sinusoid functions. Estimating these parameters is done by constructing a matrix of seasonality vectors for each value of *t* in the historical and future data with different lengths of the seasonality. The length of the season or seasons can be automatically detected via optimizing using Akaike's information criteria or the analyst-in-the-loop methodology can be used to define the season length by hand. (Taylor & Letham, 2018) In business metrics, the seasonality type is often obvious and therefore model fitting errors can be prevented by defining the seasonality type by hand.

The last component used in the Prophet model is the holiday *h(t)* component. Holidays and other yearly events provide large somewhat predictable shocks to timeseries that do not necessarily follow a periodic pattern, which means that their effects would not be well modelled with a smooth yearly seasonal cycle. The impact of a particular holiday is usually similar year over year therefore incorporating the holiday component into the forecasting model can yield better forecast accuracy. The Prophet forecasting library includes a list of common holidays which are usually country specific, so the analyst can choose which holidays to incorporate in the model. In order to incorporate the holiday component in the model an indicator function representing whether time *t* is during holiday should be added. Formally this can be expressed as (Taylor & Letham, 2018):

$$h(t) = Z(t)\kappa_L$$
$$Z(t) = [1(t \in D_1), \ldots, 1(t \in D_L)]$$

Where *Z(t)* is the indicator function returning 1 if the day is part of a particular holiday and κ_L is the weight of the corresponding change in the forecast.

Prophet library can return the confidence intervals for the predictions which makes the algorithm easy to use for anomaly detection purposes since calculating separate anomaly score is not needed. As a statistical model, the computational requirements of the Prophet are low and can be made even lower by storing the parameter values and not fitting the model each time anomaly detection for new arrived values is scheduled to be made. In the Prophet paper the issue of making predictions in scale were not limited to computational requirements but also to

the human factor, and the Prophet algorithm was developed specifically to overcome these limitations.

The analyst-in-the-loop approach combined with an easy-to-use model that is capable of forecasting several types of timeseries with high accuracy makes the Prophet forecasting library a great alternative for anomaly detection applications. While there are not that many academic papers written that would compare the performance of the Prophet algorithm in anomaly detection applications the Prophet algorithm is currently widely used in the industry for forecasting tasks, which can be noted from the popularity of the Prophet repository and the various articles written about it. Briefly, the Facebook Prophet forecasting library provides a well-documented, easily tunable and well performing algorithm that can be used in various application domains to forecast univariate timeseries.

3.5 Significance of the anomaly

Not all anomalies should be considered equal. Some are more significant than others and the outcome or reaction that the anomaly causes depend on how significant the anomaly is. Defining the significance of the anomaly is not an easy task since the consequences of the anomaly are often unknown. While there are many studies done on anomaly detection in timeseries and business metrics there is very little studies made on evaluating the significance of the anomalies found or on how the significance of the anomaly should be defined. In high level there can be two ways to measure the significance of a single anomaly. The features of the single anomaly can be evaluated or the consequences of the anomaly in a system can be forecasted in order to evaluate the anomaly significance.

Tolenado et al. proposed the usage of abnormal behavior learning to be used in the process of evaluating the significance of the found anomaly in time series based business metric. For the learning process, the anomaly scores were stored with the number of consequential anomalies to measure how long the timeseries contained abnormal behavior. On top of these features' anomaly volatility and absolute peak value of subsequent anomalies were identified as anomaly significance measures. When these features were scaled on the level of all existing anomalies in the timeseries, an anomaly significance score could be calculated using multivariate gaussian

distribution. (Toledano, Cohen, Yonatan, & Tadeski, 2017) This way the anomalies could be ranked within the single timeseries, or within a group of timeseries that have similar data generation process underneath, meaning that the single timeseries would have similar contextual and behavioral attributes. For example, monitoring the daily ad revenue for a game in each country would generate several timeseries which have a similar data generation process underneath. This could be also expanded to differentiate between operating systems or the even between different games. Since anomalies are events that occur rarely, a single timeseries would not likely contain enough anomalies in order to rank anomaly significance compared to other anomalies. Therefore, the anomalies should be ranked among all the anomalies that have a similar data generation system.

After data about the anomaly features is collected, this makes it possible to add a supervised layer to the anomaly detection process. After the anomaly is identified, as part of the anomaly detection system a notification about the anomaly can be sent to the responsible person. If this person has the ability to flag the anomaly as true or false positive, this makes it possible to train a supervised classification model to filter the found anomalies based on the features calculated from the anomaly. (Laptev, Amizadeh, & Flint, 2015) This type of supervised anomaly filtering layer can significantly increase the accuracy of the anomaly detection system which leads to overall higher business value. Also, the accuracy of the anomaly detection system would increase over time, as the supervised anomaly filtering algorithm would have more data on the correctly and incorrectly labeled anomalous values by the unsupervised algorithm. Tagging the found anomalies as true or false positives can in some cases take significant amount of work time, but in some cases having a very sophisticated anomaly detection system in place for the most important key business metrics could yield significant benefits. On the other hand, a supervised filtering layer could only make the process of flagging anomalous values too complex to be interpreted, which can be the case especially if the cost of having false positive flagged anomalies is low.

Instead of estimating the anomaly significance from the anomaly features behavioral topology learning based methods can be used to evaluate the consequences of the particular anomaly at system level. The behavior topology learning relates to learning the actual differences between the relationship among different metrics in the system. An anomaly in single business metric

may have an impact on a set of other timeseries metrics, modelling these connections is possible with complex models. (Toledano, Cohen, Yonatan, & Tadeski, 2017) Behavior topology learning based methods would require complex monitoring system to be in place in order to develop a model that could learn the relations between metrics and multiple KPI. However often analyst intuition can be used to evaluate what kind of effect would an anomaly in particular metric have at the system level. The analyst can evaluate what kind of impact an anomaly would have in easily measurable values such as total revenue or more abstract unit such as user experience. This way the significance of anomalies between separate metrics can be differentiated.

From data engineering perspective an easy way to evaluate the anomaly significance is to check where in the data lineage the metric value containing anomaly lies. Data lineage refers to the programmed rules of what happens to data and where it moves over time. Certain types of categorizations of the state where data exists can be done, Databricks company divides the data into bronze, silver and golden states (Heitz & Lee, 2019). The Bronze state data contains raw ingested data from the data generating process, silver tables contain filtered augmented data with a single source of truth and golden data contains ready to use business level aggregate data. A metric that is calculated earlier in the data lineage can be deemed to have higher anomaly significance potential since the metric is used in a higher number of aggregations later on in the data lineage. For example, an anomaly in the raw volume of bronze state data would have an impact in every other business metric later on in the data lineage. Therefore, the anomaly would have large impact on the system level, but according to the features captured from the anomaly itself the anomaly might be deemed as insignificant. An anomaly earlier in the data lineage does not necessarily mean that the business impact of the anomaly would be higher, but data lineage is still an important thing to consider when setting up anomaly detection monitoring. Figure 21 represents the high-level data lineage in an analytics pipeline.

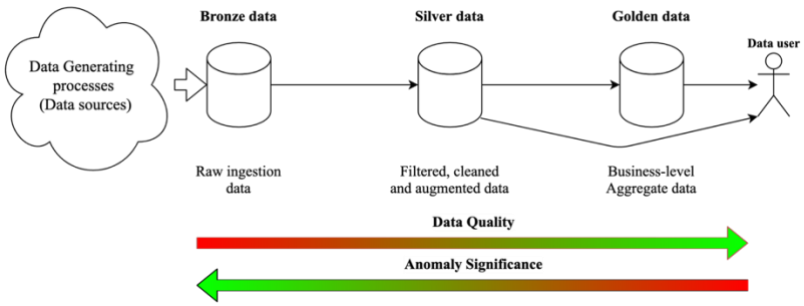


Figure 21. High level data lineage

4 Automated anomaly detection system

While recognizing anomalies in business metrics using the algorithms mentioned in this study as separate proof of concept scripts is easy, developing a system that can perform anomaly detection automatically at scale is much more difficult. The real business value from anomaly detection in business metrics comes from solving the relatively simple problem of anomaly detection in univariate time series in large scale. Since anomalies are rarely occurring events, monitoring only several business metrics is likely to give very little to no reward for the organization, even if those metrics are the key business metrics included in the management dashboards.

In cloud computing era many if not all of the used services in a cloud-based data hosting infrastructure have monitoring in place to keep track of availability and performance of the used services. However, the data that these services process often do not have abnormal behavior monitoring in place since:

1. Developers do not understand the business metric and what should be considered anomalous behavior
2. Business users who have deep domain knowledge in the business do not have the time, skillset or tools available to configure monitoring for large amount of business metrics in their field.

Monitoring the availability and performance of the data processing infrastructure are requirements for the data-based decision making, but the business value of the data platform comes from the insights derived from the data. Therefore, there is an obvious need to bridge the gap between developers and business users and make anomalous behavior monitoring of business metrics available organization wide. In recent years the job market for business intelligence developers and business analysts who have the necessary skillset to understand the business and also develop data driven solutions has been growing (Kauflin, 2017). While these talented people can bridge the gap in many data-driven decision-making use cases, monitoring business metrics for anomalies is a relatively simple problem in the fields of data science and data analysis. Therefore, an anomaly detection system with a simple user interface could be

used by people working not only in business intelligence related roles. This system can bridge the gap, overcome the technical debt of business users and make business metrics monitoring available organization wide.

4.1 Previous implementations and design considerations

In chapter 3 various algorithms and libraries to do anomaly detection were described in detail, however there are also various companies that offer anomaly detection tools that integrate with the existing data infrastructure as a tool or as a service. All the three major cloud providers have easily available services that can be used for building anomaly detection monitoring in their ecosystems (Amazon, 2021) (Microsoft, 2021) (Google, 2020). There are also various productized solutions for anomaly detection offered, for example the ones from Anomalo and Anodot (Anomalo, 2021) (Anodot, 2021). Since there are various solutions offered, from managerial perspective there is the option to outsource majority of the development via using a productized solution. However, if the data platform is seen as a key factor in business, a business-metrics anomaly detection system can be built in-house to assure the system is compatible with the existing BI solutions. Developing in house solution can also have other benefits such as extra features which can deliver competitive advantage against competitors. In this section several papers made on this topic are studied in order to understand the requirements and challenges of a business metric anomaly detection system. The implementations are reflected against the implemented anomaly detection system which was developed in house.

According to Tolenado et.al. the following design considerations should be taken into account when building an automated anomaly detection system (Toledano, Cohen, Yonatan, & Tadeski, 2017):

1. Timeliness: How quickly are answers needed to determine if datapoint is an anomaly or not? Does the timeseries need to be monitored in real time, hourly or every day?
2. Scale: Does the system need to process hundreds of metrics or millions? How large will the datasets be?
3. Rate of Change: Does the data tend to change rapidly or is the system being measured relatively static?

4. Robustness without supervision: Can the system rely on human feedback in some of the modelling choices (supervised or semi supervised algorithms) or should the system be able to recognize anomalies completely unsupervised?
5. Conciseness: If many different metrics are being monitored, should the system aggregate an answer that tells the whole picture or is it enough to detect anomalies at each metric?

All of these design considerations should be taken into account in the architecture of the system and the models used in anomaly detection. In many cases the granularity of the business metric timeseries can be hourly, daily or even longer. Business metrics are often aggregations of data, and therefore scheduling anomaly detection for business metric can be done with longer intervals. If the same anomaly detection system is used for datasets that have almost real time data, which is often the case when going closer to data ingestion layer in the data lineage, there might be a need to have a tighter schedule for the anomaly detection process. Having a real time anomaly detection system would require anomaly detection service that is running all the time, this might also require changes in the data ingestion layer to the anomaly detection system itself.

In the papers describing Yahoo and Microsoft anomaly detection system there is a data ingestion pipeline that makes the real-time timeseries available for the anomaly detection system (Ren, et al., 2019) (Laptev, Amizadeh, & Flint, 2015). If real time anomaly detection is not identified as hard requirement, there is no need to build continuous data ingestion pipelines to the anomaly detection system which simplifies the architecture. A major advantage in the cloud-based database systems is the advantage to separate compute and storage requirements (Storm, 2019). This way in a batch-based anomaly detection system the timeseries based data can remain in the source, and the data is queried for the anomaly detection purposes depending on how often the anomaly detection is scheduled. The found anomalies are stored, and after all the configured timeseries to monitor are inspected for anomalies, compute is no longer required and the compute cluster can be closed. Compared with a real-time anomaly detection system, compute should be available all the time.

Scalability of the anomaly detection system hosted in cloud is not restricted by the amount of compute and storage available, but because of the human factor. For every monitored business

metric there should be a person or a group of people responsible to monitor the anomalous events, even if the anomalous events are only used as input to other automated tasks. The first challenge in the scalability is to get people in organization to see the potential business benefits of monitoring business metrics for anomalies and the second challenge is providing an easy to use tool for people with different set of domain specific skills. Answer to the first challenge is obvious, raising awareness in the organization. Answer to the second challenge requires an intuitive user interface to the anomaly detection algorithms and a selection of algorithms that are usable for various application domains. The Facebook Prophet framework mentioned earlier in this study introduced the analyst-in-the-loop algorithm that can make it easier to configure forecasting or anomaly detection for a timeseries (Taylor & Letham, 2018). Also, the non-model-based approaches to anomaly detection introduced in chapter 3.4.1 provide users a starting point to anomaly detection that does not require any prior knowledge of timeseries modelling. Another topic related to the scalability of the anomaly detection system is the selection of different data sources available. In order to have usable anomaly detection system organization wide the system should be able to connect to majority of the possible data sources. As described in the previous section a batch-processing based anomaly detection system can call the configured database service endpoint and only keep the data for the duration of the anomaly detection process in memory. This way the service can securely connect to various data sources and provide the found anomalies.

Another consideration for the system scalability is the model training interval. More complex anomaly detection models may take considerable amount of processing power as was discussed in chapter 3. Storing the model hyperparameters and updating the model between specified intervals is an obvious answer to the training problem, another solution is to train a model and use the same model for similar timeseries. The Yahoo EDAGS anomaly detection system was configured to use the same model in the context of a large web application serving environment, since the applications were broken horizontally into tiers of similar servers the system metrics were monitored with same model on separate servers (Laptev, Amizadeh, & Flint, 2015). This way the required memory, batch workload amount and I/O against model database were reduced. While the Yahoo application was for monitoring anomalies in web hosting metrics, a similar approach can be used to monitor for anomalies in business metrics. For example, the data generation process of a single game can be considered similar across countries where the

game is played, and if say for example the number of in-app-purchases were monitored for each country the same model could be used to flag the anomalies.

The rate-of-change and robustness without supervision in the anomaly detection system place requirements to the set of available anomaly detection algorithms. For many metrics a simple non-model-based algorithm to detect anomalies is enough, if the metric is relatively stable and the contextual and behavioral attributes of the metric do not have much variation. However, timeseries may have seasonality on different intervals, varying trendiness over time or contextual changes. In order for the anomaly detection system to have great performance for the more complex timeseries modelling problems there should be also algorithms that can take these features into account. According to the literature review in chapter 3 the algorithm selection can influence only limited amount in the accuracy of the best performing algorithms, and in order to increase the accuracy of the anomaly detection system in cases where the metric contains very high rate-of-change a supervised layer can be added to the anomaly detection system as described in chapter 3.5. The anomaly detection systems described Anodot and Yahoo have the supervised anomaly detection layer in place (Toledano, Cohen, Yonatan, & Tadeski, 2017) (Laptev, Amizadeh, & Flint, 2015). In both papers the supervised layer was described to significantly increase the anomaly detection accuracy over the best performing unsupervised algorithms used.

The conciseness of the anomaly detection system can be limited to detect anomalies in single univariate business metrics. The use cases where a multivariate system is monitored for abnormal behavior are common, but difficult to generalize meaning that each multivariate anomaly detection system requires system specific configurations. Also, multivariate anomaly detection requires deep domain knowledge and expertise in the methodologies. Therefore, in this study the anomaly detection tool can handle only anomalies in univariate timeseries. This does not however mean that the abnormal behavior could not be monitored on organization or system level. If the anomalies are stored in one place and many business functions in the organization have configured anomaly detection monitoring in place it is possible to develop dashboards from the current status of the system with different visualizations of the recent anomalies. This way it might be possible to see the overall status of the system and interpret different activities in the business functions.

4.2 Implemented anomaly detection system

In this section, the implemented anomaly detection system is described in high level. Likewise with the other mentioned anomaly detection systems in the last section the main components of the system are: User interface for configuring monitoring for a business metric, the database where configurations for metric monitoring are stored, the process for detecting the actual anomalies and notification client to communicate users the anomaly detection results. Figure 22 contains the high-level process flow of the anomaly detection system.

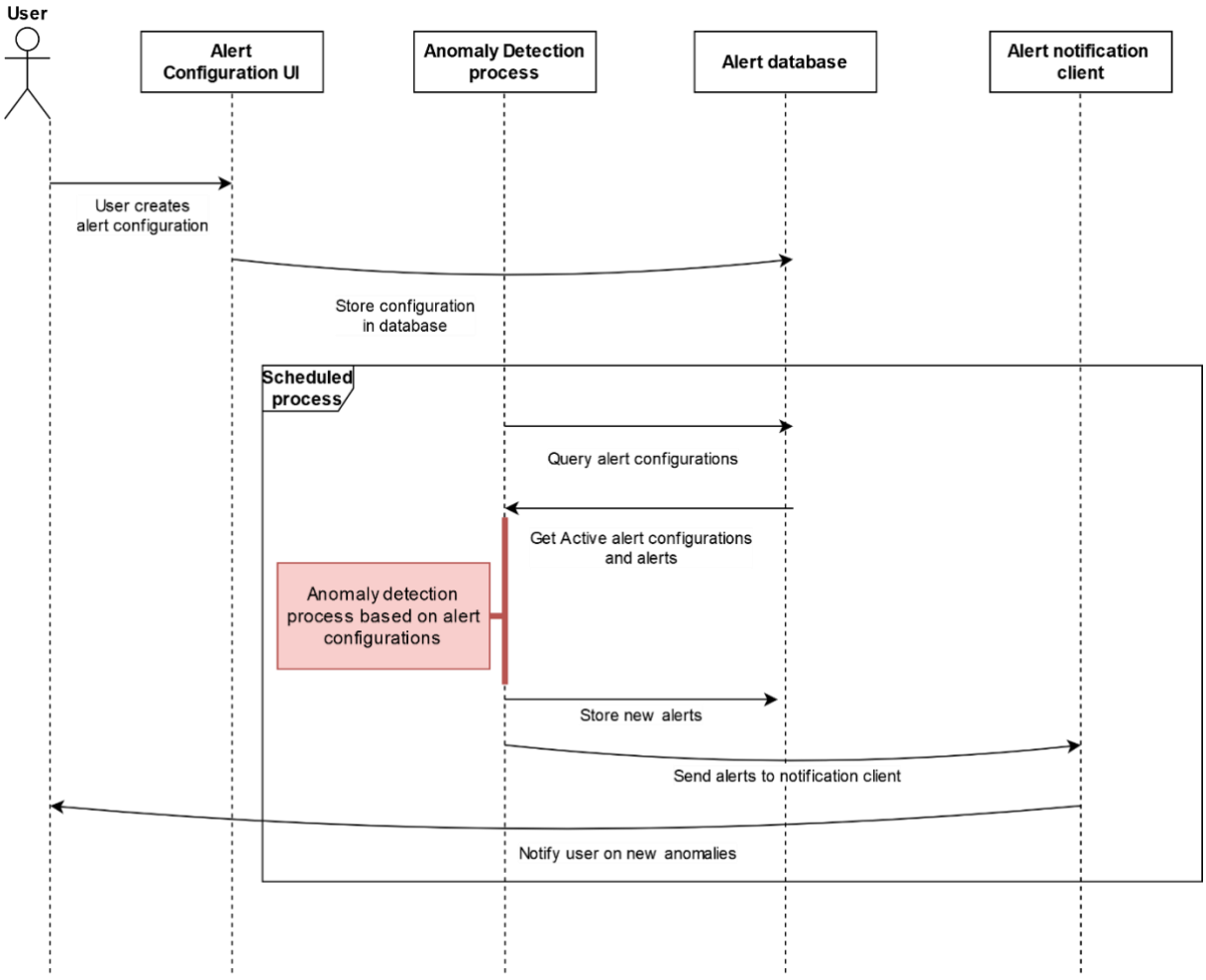


Figure 22. High level anomaly detection system

The actual anomaly detection part of the system took only a minor share of the development effort since in essence anomaly detection in univariate timeseries is a relatively simple task. The process of flagging anomalies with the algorithms described in chapter 3 is a

straightforward process as a proof-of-concept work but developing a productized version that utilizes these algorithms to do anomaly detection in scale is much more difficult task. The most important design consideration of anomaly detection system, the human factor in anomaly detection scalability, is solved in the anomaly configuration UI and the backend database which stores the alert configurations. With an intuitive user interface business user can configure anomaly detection with little effort for high number of business metrics. The alert database combined with microservices layer provides APIs for handling the anomaly detection configurations and keeping track which anomalies are present. Figure 23 shows a high-level description of the anomaly detection process.

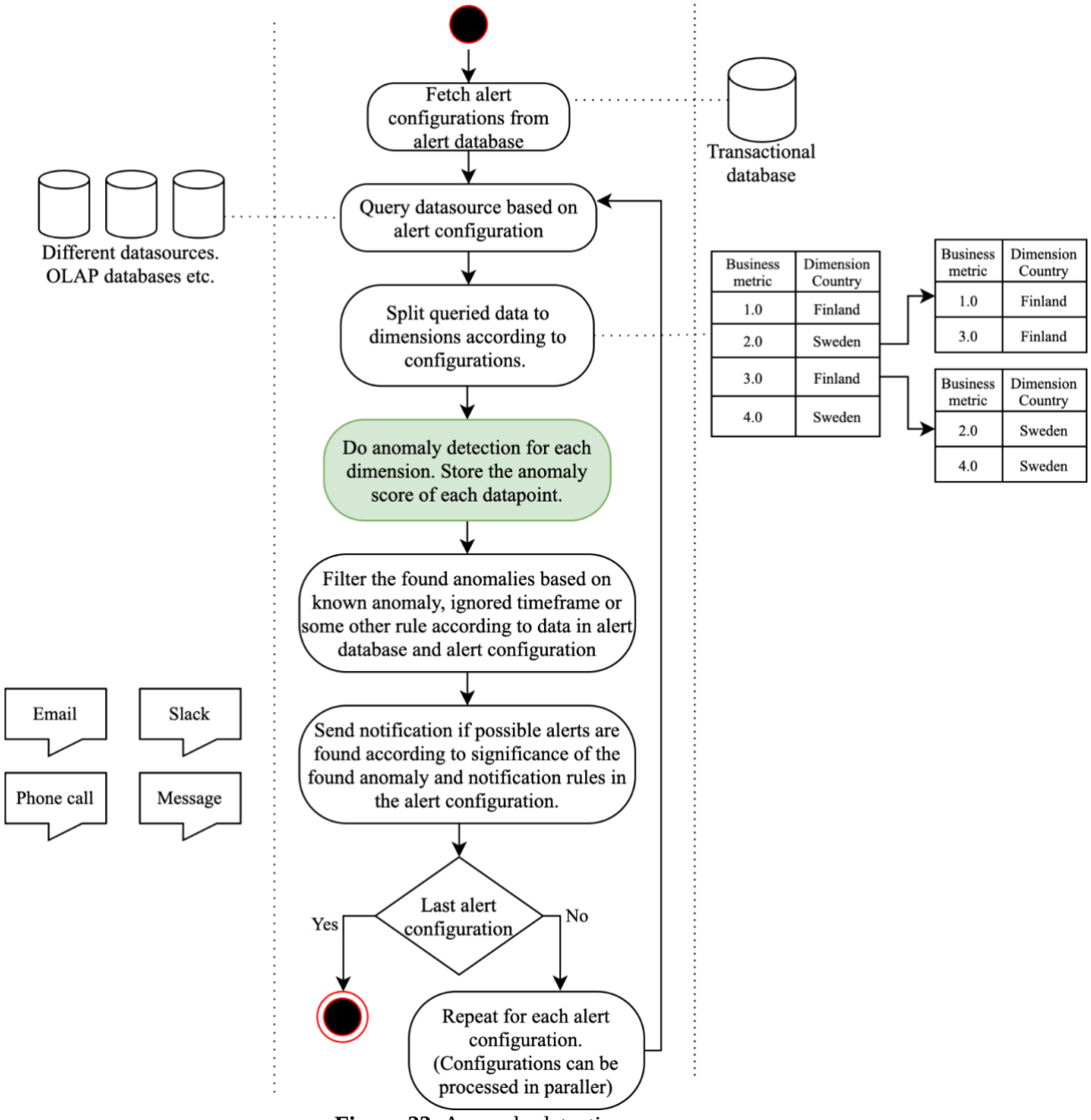


Figure 23. Anomaly detection process

The anomaly detection process can be configured for vast majority of the available business metric datasets in the organization, this is achieved by fetching the observed timeseries through query engines such as Presto or Apache Hive. The alert configuration contains information of the SQL query that is located in another service. The SQL query can be the same query that is used on existing business metric dashboard, which means that users can easily setup anomaly monitoring for the metrics they observe from the business intelligence dashboards in a daily basis. This is a key design to overcome the issue in anomaly detection monitoring scalability, the system can monitor for any business metric that can be queried with the existing query engines. Also, the anomaly detection system keeps the timeseries metric in memory only for the duration of flagging the anomalies in the metric which simplifies the process.

Second key design in terms of scalability is the ability to divide the monitored business metric into dimensions. This way with a little amount of work monitoring can be setup for a high number of business metric timeseries. For example, the ad revenue for the game can be inspected on the high level, but dividing the ad revenue into countries, operating system or versions of the game can quickly provide a high number of timeseries to monitor. Monitoring the business metric in high level may lead to un-noticed abnormal behavior but split into dimensions the algorithm can notice the anomalous behavior on the single timeseries.

After anomaly detection for business metric is done, the notification of found anomalies is sent according to the configurations. In order to gain any business value from the notifications, actions upon notifications should be taken. In many cases, the only required outcome is to make the responsible person or persons aware of the current state of the business metric and no further actions are required. However, in many cases the value from the anomaly detection comes from the user actions which are based on the anomalous value notification. Here are some possible example cases or scenarios where anomaly detection could provide business value:

- Junior developer made changes to data processing pipeline which caused error in particular step in the data lineage and caused major delays. Monitoring in data timeliness metric raised alarm.
- New feature added in a game could cause a sudden decrease in monetization, using anomaly detection the response time to reacting could be lowered.

- Different game variants could be tested in A/B testing manner, and one variant could be noted to have significantly decreased monetization metrics. This could be noticed quickly with anomaly detection before human inspection.

In high level the anomaly can be considered as an error in data quality or real anomaly in business metrics, this was discussed in detail in chapter 2.2. Another way to categorize the found anomaly is the anomaly significance which described in detail in chapter 3.5. Based on these anomaly categorizations a high-level categorization on the actions needed on the anomaly notification can be done, which can be seen from Figure 24.

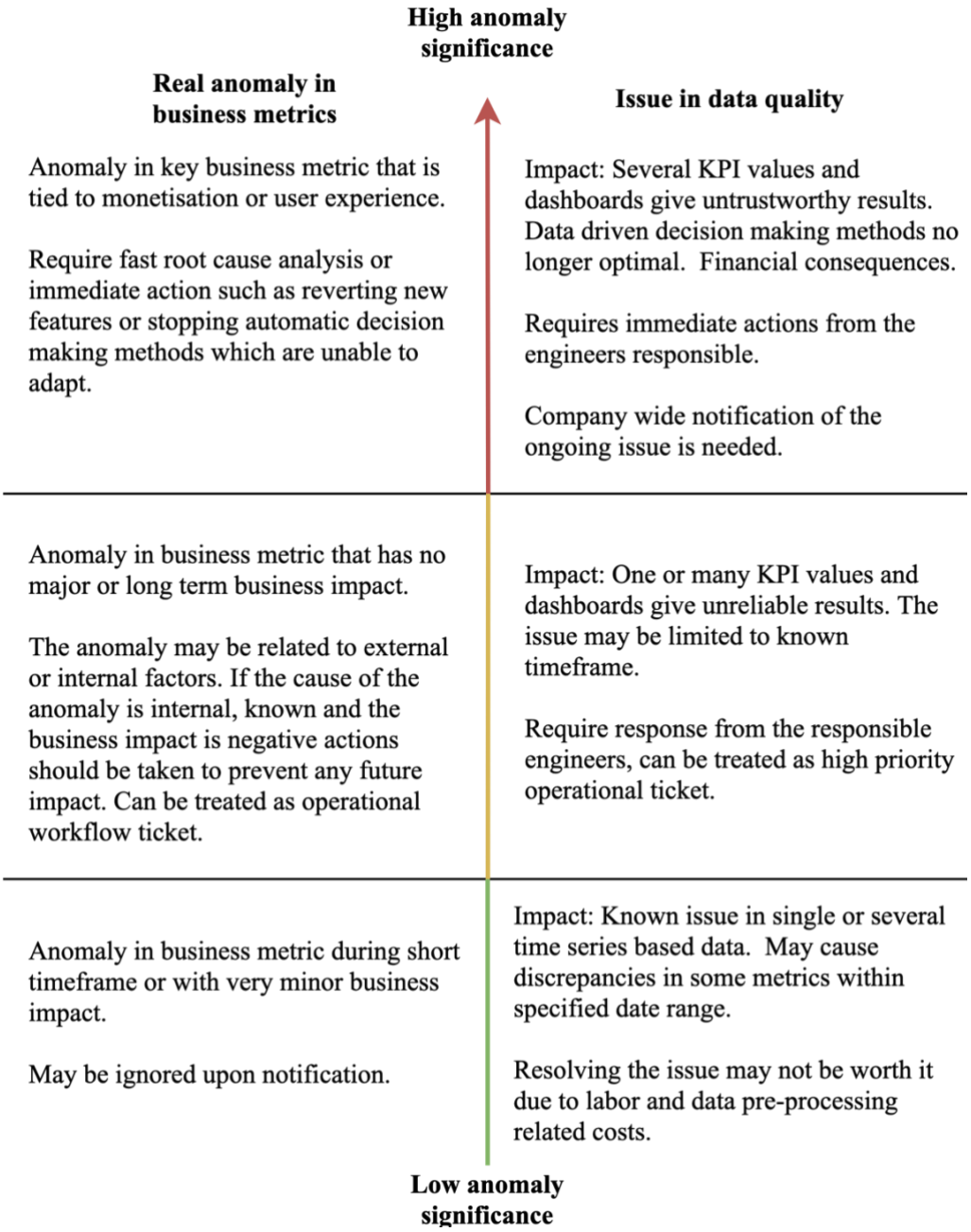


Figure 24. Anomaly significance categorization

5 Summary & Conclusions

The primary objective of this study was to provide insights from the scientific literature and previous solutions to the implemented anomaly detection system. In order to do that a comprehensive study was made on business metrics as a phenomenon, univariate anomaly detection algorithms and existing anomaly detection system implementations. This study explored the objective according to the requirements and as a result answered all the research questions. The first research question was:

What is the current state of the art in business metric timeseries anomaly detection?

Detection of abnormal behavior in timeseries metrics was identified to be one of the most common data analytics tasks used in various industries. Anomaly detection in purely engineering fields has been industry practice for a long time, and in the recent years due to increased activity in the field of business intelligence anomaly detection in business metrics has become more common. Likewise with engineering fields, understanding and reacting to anomalies in business metrics requires domain specific expertise from the business function. In this study, the gap between timeseries modelling in identifying anomalies in business metric and the domain knowledge in business functions was identified. This gap can be filled with tools that simplify the process of monitoring the key business metrics. The demand for these kinds of tools is increasing and many companies already have business metric anomaly monitoring as a standard practice. The second sub-question was written as:

How can anomalies be detected in business metrics?

This study conducted broad literature review on the commonly used algorithms used in univariate timeseries to detect anomalies. Several studies of anomaly detection algorithm performance comparison were evaluated and cross-referenced. Based on the findings it seems that the classical statistical timeseries modelling methods have comparable accuracy to more recently introduced machine learning and deep learning-based approaches to univariate timeseries anomaly detection. Major benefit of using statistical models is the interpretability of the results and the required compute, which make statistical models more applicable to anomaly

detection system where scalability in terms of easiness of use is directly connected to model result interpretability and algorithm complexity. The third sub-question was written as:

What challenges are present in developing an automated end-to-end anomaly detection system?

The primary challenge in an automated anomaly detection system was identified to be the human factor in scalability. Univariate timeseries modelling with simple algorithms is in essence a very simple problem but leveraging these techniques at scale in organization always requires human involvement in identifying and setting monitoring for the business metrics. This challenge can be overcome with a user-friendly interface for setting up monitoring for the business metrics that are already in use and with a selection of anomaly detection algorithms that are easy to use from the analyst or business user perspective. Another major challenge is fetching the data for anomaly detection purposes, the differences between architecture requirements with respect to anomaly detection timeliness were identified and a scheduled batch processing-based approach was found to be sufficient for vast majority of the business metrics.

As a future research topic around the implemented anomaly detection system, it would be interesting to add the feature to the alert notifications to confirm whether the found anomaly was true or false positive. This information could be later on used to implement a supervised layer on top of the unsupervised anomaly detection algorithm and the labeled anomaly timeseries could also be used as benchmark datasets in choosing the most appropriate unsupervised anomaly detection algorithm. The labeling of anomalies would however take considerable amount of time and would at first hand require actual active usage of the anomaly detection tool. On this front there is still work to be done on getting the user experience right, monitoring business metrics should be made as easy as possible.

6 References

Ahmed, M., Mahmood, A. N., & Islam, R. (2016). A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems 55*, 278-288.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *International Conference on Engineering and Technology (ICET)*, (pp. 1-6). Analya, Turkey.

Aleskerov, E., Freisleben, B., & Rao, B. (1997). Cardwatch: A neural network based database mining system for credit card fraud detection. *Proceedings of IEEE Computational Intelligence for Financial Engineering*, (pp. 220-226).

Alla, S., & Suman, K. A. (2019). *Beginning Anomally Detection Using Python-Based Deep Learning*. New jersey: Apress.

Amarbayasgalan, T., Pham, V. H., Theera-Umpom, N., & Ryu, K. H. (2020). Unsupervised Anomaly Detection Approach for Time-Series in Multi-Domains Using Deep Reconstruction Error. *Symmetry 12(8)*, 1251-1272.

Amazon. (2021). *Amazon CloudWatch.* Retrieved from Using CloudWatch Anomally Detection: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch_Anomaly_Detection.html

Amizaded, S., Laptev, N., & Billawala, Y. (2015, March 25). *Announcing A Benchmark Dataset for Time Series Anomaly Detection*. (Yahoo) Retrieved March 2021, from Yahoo Research: https://research.yahoo.com/news/announcing-benchmark-dataset-time-series-anomaly-detection

Anodot. (2021). *Anodot homepage*. Retrieved April 17, 2021, from https://www.anodot.com/

Anomalo. (2021). Retrieved April 17, 2021, from https://www.anomalo.com/

Assimakopoilos, V., & Nikolopoilos, K. (2000). The theta model: a decomposition approach to forecasting. *International Journal of Forecasting 16*, 521-530.

Bollinger, J. (1992). Using Bollinger Bands. *Stocks & Commodities, 10*(2), 47-51.

Box, G. E., & Cox, D. R. (1964). An Analysis of Transformations. *Journal of the Royal Statistical Society, 26*(2), 211-252.

Box, G. E., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control.* San Fransisco: Holden-Day.

Braei, M., & Wagner, S. (2020). *ANOMALY DETECTION IN UNIVARIATE TIME-SERIES: A SURVEY ON THE STATE-OF-THE-ART*. Department of Computer Science Technische Universität Darmstadt.

Breiman, L. (2001). Statistical Modeling: The Two Cultures. *Statistical Science 16(3)*, 199-231.

Brooks, C. (2014). *Inroductory Econometrics for Finance* (Vol. 3). Cambridge: Cambridge University Press.

Brown, R. (1957). Exponential smoothing for predicting demand. *Operations Research, 5*(1).

Brown, R. (1963). *Smoothing, forecasting and prediction of discrete time series.* Englewood Ciffs, N.J., Prentice-Hall.

Capretz, M., & Hayes, M. (2015). Contextual anomaly detection framework for big sensor data. *Journal of Big Data 2*.

Chalapathy, R., & Chawla, S. (2019). *DEEP LEARNING FOR ANOMALY DETECTION: A SURVEY*. University of Sydney, Qatar Computing Research Instirute.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection : A Survey. *ACM Computing Surveys*, 1-72.

Chaudhari, P., Rana, D. P., Mehta, R. G., Mistry, N. J., & Raghuwanshi, M. M. (2014). Discretization of temporal data: a survey. *arXiv preprint arXiv:1402:4283*.

Chuah, M. C., & Fu, F. (2007). *ECG Anomaly Detection via Time Series Analysis.* Lehigh: Department of Computer Science & Engineering Lehigh University.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statisticcs, 6*(1), 3-73.

Cohen, I. (2020). *A Quick Guide to the Different Types of Outliers*. Retrieved March 1, 2021, from https://www.anodot.com/blog/quick-guide-different-types-outliers/

Cox, D. R., & Stuart, A. (1955). Some Quick Sign Tests For Trend in Location and Dispersion. *Biometrika, 42*(1/2), 80-95.

Edgeworth, F. Y. (1887). On discordant observations. *Philosophical Magazine*, 364-375.

Etsy. (2015, October 14). *Etsy Skyline Algorithm*. Retrieved March 3, 2021, from https://github.com/etsy/skyline

Freeman, C., Merriman, J., Beaver, I., & Mueen, A. (2019). Experimental Comparison of Online Anomaly Detection Algorithms. *The Thirty-Second International Flairs Conference* (pp. 364-369). New Mexico: University of New Mexico.

Gardner, E. S. (1985). Exponential Smoothing: The State of the Art. *Journal of Forecasting, 4*, 1-28.

Gardner, E. S. (2006). Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting, 22*(4), 637-666.

Goldberger, A. L., Amaral, L. A., Glass, L., & Hausdorff, J. M. (2000). Components of a new research resource for complex physiologic signals. *Circulation 101,* 2015-2020.

Google. (2020, November 18). *Use real-time anomaly detection reference patterns to combat fraud.* Retrieved April 17, 2021, from https://cloud.google.com/blog/products/data-analytics/using-automated-ml-streaming-architecture-to-find-anomalies

Gorr, W. (1994). Research prospective on neural network forecasting. *Internationall Journal of Forecasting 10(1)*, 1-4.

Grubbs, F. E. (1950). Sample criteria for testing outlying observations. *Annals of Mathematical Statistiics, 21*(1), 27-58.

Grubbs, F. E. (1969). Procedures for detecting outlying observations in samples. *Technometrics 11(1)*, 1-21.

Hawkins, D. M. (1980). *Identification of Outliers.* London: Chapman And Hall.

Heitz, B., & Lee, D. (2019, August 14). *Productionizing Machine Learning with Delta Lake.* Retrieved April 2021, from Databricks blog: https://databricks.com/blog/2019/08/14/productionizing-machine-learning-with-delta-lake.html

Hibon, M., & Makridakis, S. (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting 16*, 451-476.

Hochenbaum, J., Vallis, O. S., & Kejariwal, A. (2017). *Automatic Anomaly Detection in the Cloud via Statistical Learning.* Twitter Inc.

Holt, C. (1957). Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages. *ONR Memorandum, 52*.

Huang, K. T., Lee, Y. W., & Wang, R. Y. (1999). *Quality Information and Knowlledge Management.* Prentice Hall, Upper Saddle River, NJ.

Hyndman, R., & Khandakar, Y. (2008). Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software 26(3)*.

Karkouch, A., Mousannif, H., Moatassime, H. A., & Noel, T. (2016). Data Qualilty in Internet of Things: A state-of-the-art survey. *Journal of Network and Computer Applications*.

Karr, A. F., Sanil, A. P., & Banks, D. L. (2005). *Data Quality: A Statistical Perspective.* Duke: Nationall Institute of Statistical Sciences.

Kauflin, J. (2017, July 20). *The Five Most In-Demand Skills For Data Analysis Jobs.* Retrieved April 17, 2021, from Forbes news: https://www.forbes.com/sites/jeffkauflin/2017/07/20/the-five-most-in-demand-skills-for-data-analysis-jobs/?sh=7bd186bd2c7c

Lane, T., & Brodley, C. E. (1997). *Sequence Matching and Learning in Anomaly Detection for Computer Security.* School of Electrical and Computer Engineering, Purdue University.

Laptev, N., Amizadeh, S., & Flint, I. (2015). *Generic and Scalable Framework for Automated Time-series Anomaly Detection.* Sunnyvale, CA, USA: Yahoo Labs.

Lavin, A., & Ahmad, S. (2015). Evaluating Real-time Anomaly Detection Algorithms – the Numenta Anomaly Benchmark. *14th International Conference on Machine Learning and Applications.* Miami.

Lewinson, E. (2020, December 9). *Facebook's Prophet + Deep Learning = NeuralProphet*. Retrieved March 8, 2021, from https://towardsdatascience.com/facebooks-prophet-deep-learning-neuralprophet-76796aed1d86

Makridakis, S., Spiliotis, E., & Assimakopoilos, V. (2018). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *P-LoS ONE 13(3)*.

Mehrotra, K. G., Mohan, C. K., & Huang, H. (2017). *Anomaly Detection Principles and Algorithms.* New York: Department of Electrical Engineering and Computer Science, Syracuse University.

Microsoft. (2021). *Azure Anomaly Detector*. Retrieved from https://azure.microsoft.com/en-us/services/cognitive-services/anomaly-detector/#overview

Moritz, S., Sarda, A., Bratz-Beielstein, T., Zaefferer, M., & Stork, J. (2015). *Comparison of different Methods for Univariate Time Series Imputation in R.* Cologne University of Applied Sciences.

Munir, M., Siddiqui, S. A., Dengel, A., & And, S. A. (2019). *DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series.* Kaiserslautern: Technische Universität Kaiserslautern, German Research Center for Artificial Intelligence.

Numenta. (2015). *The Science of Anomaly Detection. How HTM Enables Anomaly Detection in Streaming Data.* Numenta.

Papacharalampous, G., Tyralis, H., & Koutsoyiannis, D. (2019). Comparison of stochastic and machine learning methods for multip-step ahead forecasting of hydrological processes. *Stochastic Environmental Research and Risk Assesment 33(2)*, 481-514.

Patcha, A., & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks 51*, 3448-3470.

Patti, J., Belov, N., Craven, P., & Thayer, T. (2009). Applying Adaptive Anomaly Detection to Human Networks. *Human Behavior Computational Modelling*.

Pearl, R., & Reed, L. J. (1920). On the rate of growth of the population of the United states since 1790 and its mathematical representation. *Proceedings of the National Academy of Sciences of the United States of America. 6*, pp. 275-288. Johns Hopkins Univercity.

Pratama, I., Permanasari, A. E., Ardiyanto, I., & Indrayani, R. (2016). A Review of Missing Values Handling Methods on Time-Series Data. *International Conference on Information Technology Systems and Innovation (ICITSI)* (pp. 1-6). Yogyakarta, Indonesia: Universitas Gadjah Mada.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., . . . Zhang, Q. (2019). Time-Series Anomaly Detection Service at Microsoft. *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (pp. 3009-3017). New York: Microsoft.

Rosner, B. (1975). On the detection of many outliers. *Technometrics, 17*(2), 221-227.

Rovio. (2021). *Rovio code of conduct*. Retrieved February 24, 2021, from https://www.rovio.com/code-of-conduct/

Rovio. (2021). *Rovio Investors*. Retrieved February 24, 2021, from https://investors.rovio.com/en/about-us/factsheet

Salgado, C. M., Azevedo, C., Proenca, H., & Vieira, S. M. (2016). *Noise Versus Outliers.* Cham: Springer International Publishing.

Saurav, S., Malhotra, P., Gugulothu, N., Vig, L., Agarwal, P., & Shroff, G. (2018). Online anomaly detection with concept drift adaptation using recurrent neural networks. *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data* (pp. 78-87). ACM.

Stanley, J. (2020, November 10). *When Data Disappears.* Retrieved 2 15, 2021, from Anomalo: https://medium.com/anomalo-hq/when-data-disappears-d97f9deecf54

Stanley, J. (2020, September 2). *Airbnb quality data for all.* Retrieved 2 15, 2021, from Anomalo: https://medium.com/anomalo-hq/airbnb-quality-data-for-all-a4ca6b4c97e6

Statsmodels. (2021, February 2). *Statsmodels.* Retrieved 3 2021, from Exponential smoothing: https://www.statsmodels.org/stable/examples/notebooks/generated/exponential_smoothing.html

Statsmodels. (2021). *Statsmodels ARIMA*. Retrieved March 12, 2021, from https://www.statsmodels.org/devel/generated/statsmodels.tsa.arima_model.ARIMA.html

Storm, A. (2019, January 14). *Seperating compute and storage.* Retrieved from Medium: https://ajstorm.medium.com/separating-compute-and-storage-59def4f27d64

Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. *The American Statistician, 72*(1), 37-45.

TensorFlow. (2021, Janurary 22). *Exponential Moving Average*. Retrieved April 2021, from https://www.tensorflow.org/api_docs/python/tf/train/ExponentialMovingAverage

Thorburn, W. M. (1915). OCCAM'S RAZOR. *Mind, XXIV*(2), 287-288.

Toledano, M., Cohen, I., Yonatan, B.-S., & Tadeski, I. (2017). Real-time anomaly detection system for time series at scale. *Proceedings of Machine Learning Research KDD: Workshop on Anomaly Detection in Finance* (pp. 56-65). Ra'anana, Israel: Anodot Ltd.

Tong, H. (1990). *Nonlinear Time Series: a Dynamical Systems Approach.* Oxford: Oxford University Press.

Wang, S., Feng, J., & Liu, G. (2013). Application of seasonal time series model in the precipitation forecast. *Mathematical and Computer Modelling, 58*(3-4), 677-683.

Wei, W. (1989). *Time series Analysis: Univariate and Multivariate Methods.* Pearson Addison Wesley.

Wheelwright, S. C., & Makridakis, S. (1998). *Forecasting: Methods and Applications.* John Wiley & Sons.

Whitle, P. (1951). *Hypothesis testing in time series analysis.* New York: Hafner Publishing Company.

Winters, P. (1960). Forecasting Sales by Expoenentially Weighted Moving Averages. *Management Science, 6*(3), 324-342.

Wu, J., Zeng, W., Chen, Z., & Tang, X.-f. (2016). Hierarchical Temprary Memory Method for Time-series-based Anomaly Detection. *Neurocomputing*, 535-546.

Wu, R., & Keogh, E. J. (2020). *Current Time Series Anomaly Detection Benchmarks are Flawed and are Creation the Illusion of Progress.*

Xu, H., Chen, W., Zhao, N., Li, Z., Bu, J., Zi, Z., . . . Qiao, H. (2018). Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications. *Proceedings of the 2018 World Wide Web Conference* (pp. 187-196). Tsinghua University & Alibaba Group.

Yahoo. (2015, March 25). *Announcing A Benchmark Dataset for Time Series Anomaly Dtection*. Retrieved March 3, 2021, from https://research.yahoo.com/news/announcing-benchmark-dataset-time-series-anomaly-detection

Yitzhak, K. (1976). *An introduction to harmonic analysis* (Vol. 2). New York: Dover Publications.

Zhang, G. P., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research 160*, 501-514.