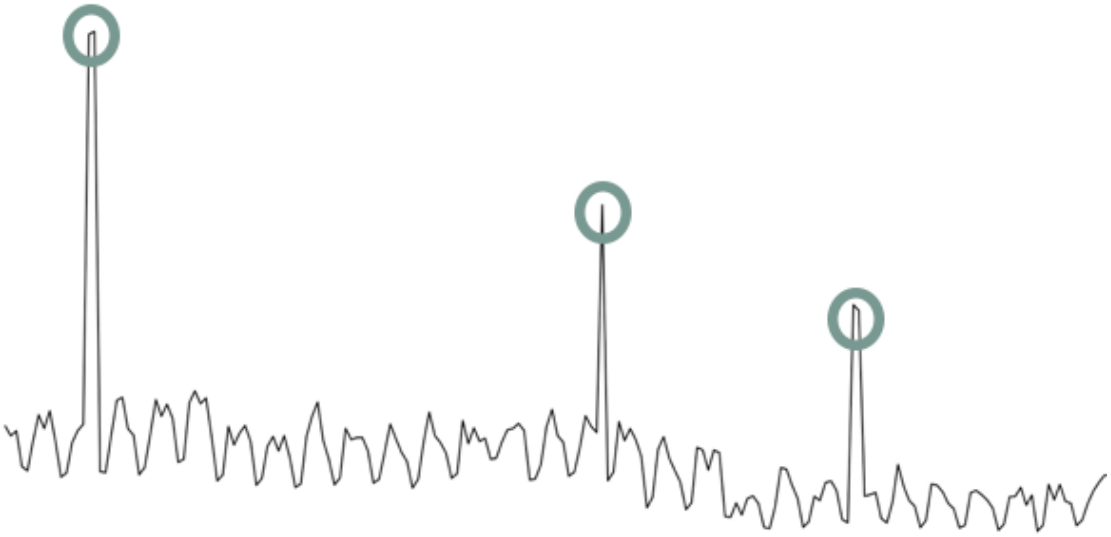

Anomaly Detection and Fault Localization

An Automated Process for Advertising Systems

Master's thesis in Computer Science - Algorithms, Language and Logic

Linnea Rudenius

Moa Persson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Anomaly Detection and Fault Localization

An Automated Process for Advertising Systems

LINNEA RUDENIUS

MOA PERSSON

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Anomaly Detection and Fault Localization
An Automated Process for Advertising Systems
LINNEA RUDENIUS
MOA PERSSON

Academic Supervisor: John Wiedenhoeft, Department of Computer Science and Engineering
Company Advisor: Gustav Munkby, Burt
Examiner: Morteza Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Time series with marked anomalous points.

Typeset in LaTeX
Gothenburg, Sweden 2018

Anomaly Detection and Fault Localization
An Automated Process for Advertising Systems
LINNEA RUDENIUS
MOA PERSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The aim of this thesis is to develop an automated process of identifying anomalies in time series and narrowing down the possible root causes. The thesis has been divided into three parts; forecasting, anomaly detection and fault localization. During the forecasting part, different time series models commonly used for forecasting were evaluated, and an exponential smoothing state space model was determined as the best fit for the data used in the project. For the anomaly detection part, an anomaly was defined as a significant deviation from a forecasted value, and different methods for determining a significant deviation were explored. For this part, a threshold learning algorithm was determined as the best method for identifying anomalies. The threshold learning algorithm uses input provided by operators and an updating rule for increasing or decreasing the current threshold. During the last part of this thesis, two different fault localization algorithms were implemented, and the results were compared in order to see which found the largest number of correct root causes. The best performing algorithm was a modified version of the Adtributor algorithm [3], where the modifications included making the algorithm recursive and adjusting the criteria used to determine root cause candidates.

The results of the forecasting- and anomaly detection part of this thesis were varied. We believe this is due to the limited amount of labelled data available and the different characteristics present in the time series used. The results from the fault localization were, however, very promising but need to be evaluated using a larger test set. Combining these three components, we believe that the automated process has great potential for discovering anomalies and narrowing down the root causes in a real application.

Keywords: Time Series, Forecasting, ARIMA, ETS, Anomaly Detection, Threshold Learning, Fault Localization

Acknowledgements

First of all, we would like to thank the company Burt for giving us the opportunity to conduct this project. In particular, we would like to thank our company supervisor Gustav Munkby for his guidance and advise throughout the entire project and for providing the tools necessary to complete this thesis. We would also like to thank Carl Leskinen and Carl Retzner for helping us understand all aspects of the problem at hand and always taking the time to answer our questions.

Secondly, we would like to thank our supervisor at Chalmers University of Technology, John Wiedenhoeft for continuously steering us in the right direction and making this work possible. Without our weekly meetings, we would not have gotten this far. We would also like to thank Morteza Chehreghani for being the examiner of this thesis.

During the fault localization part of this thesis, we stumbled upon some questions regarding a previous paper. However, due to the quick correspondence and help from two of the authors, the confusion was swiftly clarified. Thus, we would like to express our gratitude to Dan Pei and Yongqian Sun at Tsinghua University for providing us with throurough explanations and answers to our questions.

Finally, we would like to thank our partners Kristoffer Einarsson and Philip Dahlstedt for their love and support during the most intense parts of this thesis.

Moa Persson and Linnea Rudenius, Gothenburg, June 2018

Contents

List of Figures

List of Tables

List of Algorithms

1
Introduction

Data analysts at global media companies digest an enormous amount of data from disparate systems in order to provide the different business units with decision support. A recurring challenge for such analysts consists of analyzing time series of seemingly unrelated events, where a change in one business unit causes an undesired outcome in a different business unit. The longer a problem is allowed to linger without detection, the more money the company loses; either through expensive last-minute corrective actions or simply by lost revenue.

Burt is a data integration and analytics company whose solutions help publishers deal with the new media landscape. The company has developed a unique data integration platform to gather disparate data sources into one unified view, which puts the necessary prerequisites into place in order to address the aforementioned challenge of data analysis.

1.1 Background

Burt's customers are mainly publishers, each of whom typically own a couple of different sites. Every time a user visits one of these sites, the available ad spaces on the site are populated via different partners. The partners sell the ad spaces by different transaction types and record univariate time series of various metrics. These metrics could for instance relate to the sales and include numbers such as the number of *impressions* sold or the generated *revenue*. For each one of the publisher's sites, one can for example find a time series representing the total revenue generated by that site. This series can in turn be broken down into several time series representing numbers such as the revenue generated by each partner or the revenue generated by each transaction type. We refer to *site*, *partner* and *transaction type* in this example as *dimensions* and the number of possible values for each dimension as the *cardinality* of that dimension. When unexpected variations occur in the time series, these must be detected quickly since they could be indications of system failure for which corrective actions may need to be taken.

The work of an analyst employed by a publisher typically includes analyzing time series of various metrics recorded as a total for different sites in order to find whether any anomalies have occurred. If an anomaly is found, the analyst needs to determine *why* and *where* it occurred. A first step in this process would be to look at the different breakdowns of the time series to find out which dimensions the anomaly

was related to. The analyst would then continue by looking at the breakdowns for that dimension to narrow it down even further. However, not only is the process of identifying anomalies time consuming in itself, but in order to look at the different time series the analyst also has to pull reports from various different data sources. As the number of dimensions, or the cardinality of the dimensions, increases, the task quickly becomes overwhelming to perform manually. In combination with the added risk of human mistakes, the need for an automated process arises.

The data integration platform developed by Burt provides the possibility to access time series from all different data sources. With the ability to access all of the time series, we propose that it would be possible to automate the process of identifying anomalies and narrowing down the possible root causes. While numerous different methods exist for identifying anomalies, we argue that it would be beneficial to use a method where an anomaly is defined as a deviation from a predicted value. This kind of anomaly detection has been widely explored in different settings and was for instance used by Yaacob, Tan, Chien, and Tan [34] to detect potential attacks in a network. The reasoning behind why we believe this method to be appropriate is based on the fact that deviations from predicted values could not only be useful for detecting anomalies, but also for identifying possible root causes. That is, if a significant deviation has occurred in a time series, one might look for one or more deviations in the breakdowns for that time series that can explain at least some threshold of the sudden change. As the time series under investigation in this project may have different characteristics, the model that should be used when predicting future values needs to be flexible. Also, it is important that the model can handle seasonality, as most time series have a clear weekly pattern. By comparing existing models that meet these requirements, we will select the one most suited to our data. With identified deviations from expected behaviour, we then suggest that some algorithm for fault localization can be used to efficiently find a potential set of root causes. This set could then be presented to an analyst enabling them to get a coherent picture of where the anomaly occurred and thus narrowing down the scope of possible root causes to investigate.

1.2 Aim

The aim of this thesis was to develop an automated process of identifying anomalies and narrowing down the possible root causes. A suitable model was implemented to find deviations from expected behaviour, and these deviations used as input to a fault localization algorithm in order to find a set of the most likely root causes.

1.3 Limitations

There are several different existing metrics that could be interesting to analyze, however, this project will only consider the *revenue* metric.

During the first phase of this project, we will conduct an evaluation over different time series models commonly used for forecasting. As the different time series used in the project may inhibit different characteristics, it may be the case that the series require different models. However, in this thesis we will not consider using different models for different time series. We will thus select the best performing model over all time series and use this during the anomaly detection and fault localization. Furthermore, no automated process for determining the characteristics of the time series will be implemented, and thus all time series will be treated the same.

The anomaly detection part of this thesis will only be applied to time series representing total revenue for a specific customer or total revenue for a specific business unit. The time series can be broken into several other time series representing different dimensions, but these will not be investigated until the fault localization step. The reason for this is that although an anomaly could have occurred in such a breakdown series, it can be considered negligible if it does not impact the total revenue.

Although Burt has access to large amounts of data, there is no labelled data available for the purposes of this thesis. This means that anomalies and their root causes will have to be manually labelled in order to evaluate the performance of each step of the thesis. Since this is a time consuming process, the labelling will be performed only using visual means, and thus the series may not be correctly labelled. It also means that we will have access to a rather small amount of labelled data, and thus the results may be somewhat skewed.

1.4 Thesis Disposition

The rest of this paper is organized as follows. In Chapter 2, we present a description of the structure of the data that was used throughout the thesis. We then present all relevant theory needed in order to understand the various aspects of the thesis in Chapter 3. In Chapter 4, we evaluate different existing models for forecasting points in the time series and select the best model. Using this model, we then move on to identifying anomalies and present a threshold learning algorithm in Chapter 5. In Chapter 6, we present our implementations of two different fault localization algorithms that determine the root causes of the anomalies detected in the previous chapter. In each of the Chapters 4 - 6, we present the methods used in each step, describe how the evaluations were performed, present the results and end with a discussion of said results. Finally, in Chapter 7, we present our conclusions of the thesis along with possible future work.

2
Data

The data used in this thesis consists of different time series representing daily *revenue* generated for a period of one year. Each time series can possess multiple *dimensional attributes* describing things such as what *business unit* the revenue was generated for, which *partner*, what *transaction type* or what *device type*. Those time series can then be combined into a multi-dimensional array which, when summed up, is equal to the time series of total revenue. If we consider the case when three different dimensions exist; partner, business unit and transaction type (TT), the total revenue can be illustrated as the cube shown in Figure 2.1. In the cube, each cell contains one time series and corresponds to revenue generated by a specific partner, business unit *and* transaction type.

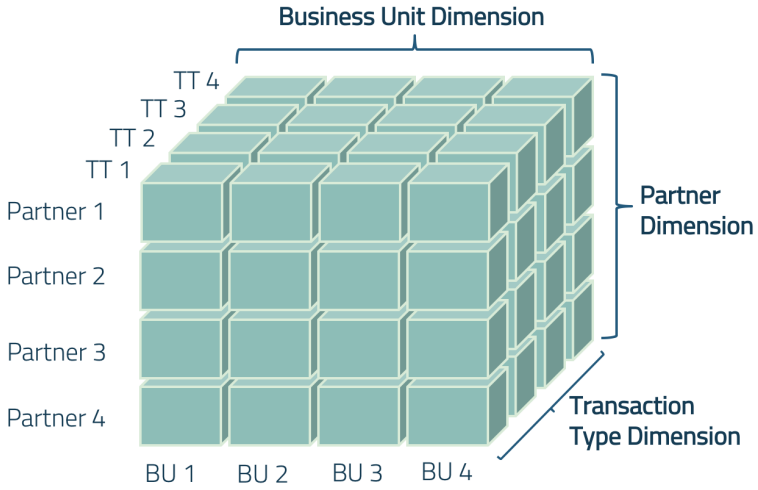

Figure 2.1: An illustration of the total revenue as a cube of three different dimensions; partner, business unit and transaction type (TT)

By slicing the cube shown in Figure 2.1 in different ways we can construct subcubes representing the total revenue for a subset of dimensional attributes. An example of this is shown Figure 2.2. At the bottom, a subcube has been constructed by slicing the total cube along the y-axis, which represents the partner dimension. The subcube, which contains the blue cells, sums up to the total revenue generated by Partner 1. Similarly, at the top, a subcube has been constructed by slicing the total cube along both the y-axis *and* the z-axis, which represents the transaction type dimension. This subcube, which contains the light green cells, sums up to the total revenue generated by Partner 4 *and* Transaction Type 1.

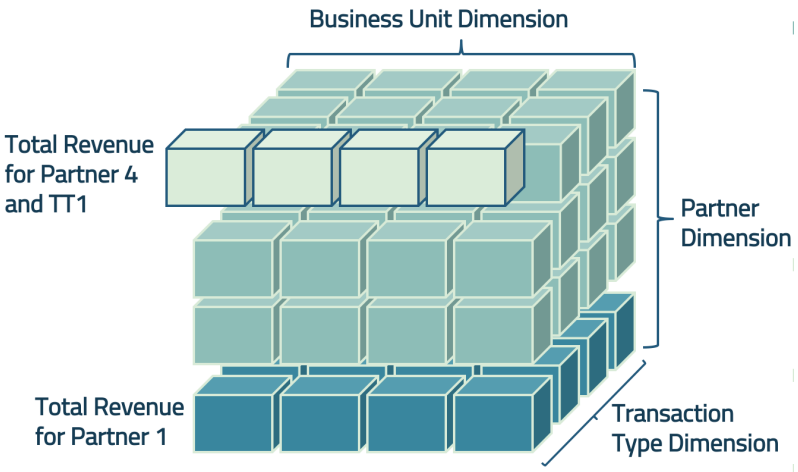

Figure 2.2: An illustration of how subcubes can be constructed by slicing the total cube along different axes.

When slicing a cube along one or more axes, several different subcubes can be constructed depending on *where* the total cube is sliced. We refer to the set of all different subcubes that can be constructed by slicing along one or more axes as the *partition set* for the dimensions that the axes represent. An example of partitio sets for all possible combinations of dimensions can bee seen in Figure 2.3. The last partitioned set in the figure corresponds to slicing the cube along all dimensions. When doing this, the subcube created contains only one cell. Thus, this cell will also be equal to the total revenue for the subcube. The set of all such subcubes is referred to as the LEAF set. And a subcube containing only one cell is referred to as a LEAF cube.

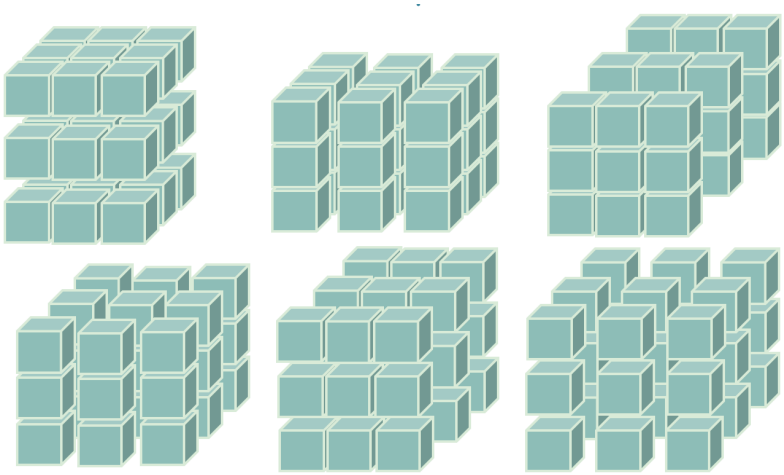

Figure 2.3: An illustration of partitioned sets for different combinations of dimensions.

In Figure 2.4, an example of two different time series used in the thesis is presented. From the figure one can see that although both series exhibit a weekly seasonal pattern, they have different characteristics. The time series at the top has an almost constant variance and values fluctuating around 500000, while the time series at the bottom is much noisier and has values of a much smaller scale.

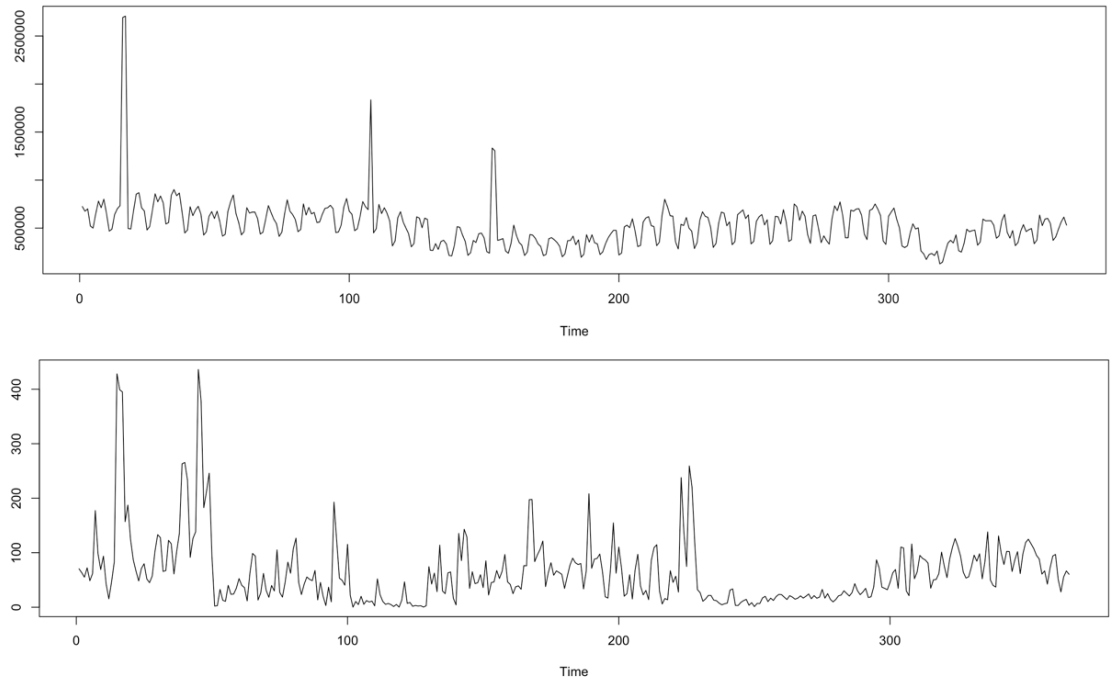

Figure 2.4: An example of two time series used in the thesis.

3
Theory

3.1 Time Series Analysis

A time series is a sequence of values, measured over an evenly spaced period of time, for instance monthly or weekly data. Although time itself is a continuous variable, the measurements are made at specific points in time and so appear as discrete observations [6]. An underlying feature of a time series is that, typically, adjacent observations are dependent, and techniques for the analysis of this dependence is known as *time series analysis*. A common procedure when dealing with time series is the identification of a statistical model that can represent the time series as a stochastic process. These models can then be applied in order to predict future values of a time series.

3.1.1 Characteristics

One central part of time series analysis is the understanding and identification of the characteristics of the series. If certain patterns can be identified, this will be of help when choosing a statistical model to represent the series. It is common to decompose a time series into its three components; the trend, the seasonality and the irregular component [30].

Trend: When there exists a long-term increase or decrease in the data, which is without calender related or irregular effects, this is called a trend [11].

Seasonality: When a series is influenced by seasonal factors such as day of the week or month of the year, this is called a seasonal pattern. When a time series exhibits seasonality, it is always of a fixed time period [11]. If the seasonality is constant, the time series is additive, but if the seasonality increases over time, it is a multiplicative series [30].

Cyclic Patterns: Cyclic patterns are often confused with seasonal patterns, however, cyclic patterns are not of a fixed time period and the duration is usually of at least two years [11].

Time Series Decomposition: When analyzing a time series, it is often useful to split it into several components, each representing one of the underlying characteristics. For an additive model we could, as explained by Hyndman and Athanasopoulos

[11], represent a time series as

$$y_t = S_t + T_t + E_t$$

where y_t is the value at time t, S_t is the seasonal component, T_t is the trend component and E_t is the error component. Similarly for a multiplicative model, Hyndman and Athanasopoulos [11] provides the following representation.

$$y_t = S_t \times T_t \times E_t$$

.

Heteroscedasticity: Heteroscedasticity is a term used to describe data that has error terms with differing variance over time [33]. If the variance is constant, this is instead known as *homoscedasticity*.

The Backshift Operator: A useful notational device when working with time series lags is the *backshift operator* which is defined as

$$By_t = y_{t-1}$$

This means that B operating on y_t has the effect of shifting the data back one period [11].

3.1.1.1 Stationarity

Stationarity is an important concept when modelling time series, and refers to the existence of time-invariant behaviour, that is, it exhibits no trend or seasonality and the variance is constant over time. Time series that are not stationary may have to be transformed before certain models can be fitted to the data. A time series can be either *weakly* or *strictly* stationary. Fan and Yao [8] provide the following two definitions of weak and strict stationarity.

Weak Stationarity: *A time series* $\{X_t, t = 0, \pm1, \pm2, ...\}$ *is stationary if* $E(X_t^2) < \infty$ *for each t, and*

(i) $E(X_t)$ *is a constant, independent of t, and*
(ii) $Cov(X_t, X_{t+k})$ *is independent of t for each k.*

Strict Stationarity: *A time series* $\{X_T, t = 0, \pm1, \pm2, ...\}$ *is strictly stationary if* $(X_1, ..., X_N)$ *and* $(X_{1+k}, ..., X_{n+k})$ *have the same joint distributions for any integer* $n \geq 1$ *and any integer k.*

3.1.1.2 Unit Roots and Differencing

Although stationarity is a common assumption, it is rarely the case when modeling a time series [6]. As previously mentioned, it is common for time series to exhibit some trend or seasonality, or both. In order to perform a time series analysis it is therefore common to start by performing some form of *unit root test* to determine whether or not the data is stationary [11]. A non-stationary time series will need *differencing* before an appropriate model can be fitted.

3.1.1.3 Unit Roots

The characteristic equation for a stochastic process can be written as

$$y_t = \alpha y_{t-1} + u_t, \qquad (t = 2, ..., T)$$

where u_t is a white noise process [5]. When the root of this equation is 1 or -1, i.e. if $\alpha = \pm 1$, then y_t has a *unit root*. If there is a unit root present in the seasonal component of a time series, then this is called a *seasonal unit root*. When a unit root is present, i.e. when $|\alpha| = 1$, we can write y_t as

$$y_t = \sum_{i=1}^{t} u_i + y_0$$

and when $|\alpha| < 1$, we can write

$$y_t = \sum_{i=0}^{\infty} \alpha^i u_{t-i}$$

The stochastic properties of the two representations of y_t above differ significantly and have been summarized by Engle and Granger [7]. One example is that the data will become more and more variable when $\alpha = 1$, but move within a fixed range when $|\alpha| < 1$. For the purposes of this paper, it is, however, only important to understand that the presence of a unit root means that the process is not stationary [11].

3.1.1.4 Unit Root Tests

Unit root tests are statistical hypothesis tests of stationarity that have been designed to determine whether or not *differencing* of the data is required [11]. Below are presented two such tests that are used by two of the times series models presented in the following section.

The first one is the Kwiatkowski–Phillips-Schmidt–Shin (KPSS) test. The null hypothesis here is that a unit root is absent and therefore the data is stationary [20]. The Osborn, Chui, Smith and Birchenhall (OCSB) test is the second such test, proposed by Osborn, Chui, Smith, and Birchenhall [23]. However, according to Rodrigues and Osborn [26], the OCSB test is defined in such a way that a *first differencing* is assumed, and therefore, unlike the previously mentioned test, OCSB testing does not allow for explicit testing against stationarity. Instead, the null hypothesis of this test is that a seasonal unit root is present and therefore *seasonal differencing* is required.

3.1.1.5 Differencing

As previously mentioned, a non-stationary time series may need differencing before certain models can be fitted to the data. Two such models were evaluated in this

thesis and will be described in Section 3.2.

Differencing computes the differences between consecutive observations and is a common way to make a time series stationary [11]. By removing the changes in the level of a time series, differencing stabilizes the mean and can thus eliminate both trend and seasonality. The differenced series, known as the *first differenced series*, is then the series of the changes between consecutive values of the original series and can be written as

$$y'_t = y_t - y_{t-1}$$

where y'_t is the differenced value at time t and y_t is the original value. The backshift operator is particularly useful for describing the differencing process and using this, the first difference can instead be written as

$$y'_t = y_t - y_{t-1} = y_t - By_t = (1 - B)y_t$$

Since it is not possible to calculate a difference for the first value of the series, the differenced series will have one value less than the original time series.

There are cases when the differenced data does not appear stationary and therefore it may be necessary to difference the data a second time, this is known as *second order differencing*. A second order differencing is written as

$$y''_t = y'_t - y'_{t-1}$$

or, using the backshift operator, as

$$y''_t = (1 - B)^2 y_t$$

where y''_t is the second order differenced value at time t.

3.1.1.6 Seasonal Differencing

If the data has a strong seasonal pattern, i.e. when seasonal unit roots are present, it is common to apply a type of differencing called *seasonal differencing* [5]. In some cases, it may be necessary to apply both seasonal differencing as well as first order differencing in order to achieve a stationary series. The seasonal difference is the change between an observation and its corresponding observation from the previous season and can be written as

$$y'_t = y_t - y_{t-m} = (1 - B^m)y_t$$

where m is the number of seasons [11]. For instance, if the data exhibits monthly seasonality m would be set to 12.

3.1.2 Autocorrelation

A useful way of detecting characteristics in a time series is through inspection of the *autocorrelation* for that series. The autocorrelation for a time series measures the

linear relationship between *lagged* values of the same series. Thus, there are several different autocorrelation coefficients corresponding to different lags. With a lag k, the autocorrelation coefficient r_k measures the relationship between y_t and y_{t-k} in the time series, and can be defined as follows [12], where $\bar{y}$ is the mean value.

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}$$

Of course, this formula only makes sense for stationary series where the mean and variance are constant throughout the series. When plotting the different autocorrelation coefficients for different lags, this is commonly known to show the *autocorrelation function* (ACF). One of the main reasons for analyzing the autocorrelation of a time series is that it represents samples made in space and time [6]. As such, there may be some element of spatial and/or temporal association between the data items. Most statistical techniques require the samples to be truly independent, and so the ACF will show whether or not this requirement is met.

So, the autocorrelations measure the relationship between y_t and y_{t-k} for different values of k. If y_t and y_{t-1} are correlated, then y_{t-1} and y_{t-2} must also be correlated. This means that y_t and y_{t-2} might also be correlated, simply due to the fact that they are both connected to y_{t-1} [11]. *Partial autocorrelations* (PACF) are used to overcome this problem. The partial autocorrelation instead measures the relationship between y_t and y_{t-k} after removing the effects of other time lags $1, 2, ..., k-1$. This means that the first partial autocorrelation is the same as the first autocorrelation, since there is nothing between them to be removed.

3.2 Time Series Models

With knowledge of the characteristics of a time series, an appropriate model describing the series can be selected and fitted. There are numerous different models representing different stochastic processes. A few of these models are relevant for the purpose of this thesis and are therefore presented below.

3.2.1 Autoregressive Models

An autoregressive (AR) model, as described by Box, Jenkins, Reinsel, and Ljung [4], is a stochastic model that expresses a current value as a finite, linear aggregate of previous values of the process and a random shock, or error term. That is, the estimated value at a given time t is determined by the immediately previous values multiplied by their autocorrelation, plus a residual error term [6].

An AR model of order p, AR(p), can be written as

$$x_t = \alpha_1 x_{t-1} + ... + \alpha_p x_{t-p} + z_t$$

where x_t is the predicted value at time t, α represents the correlation coefficients at lags $1, 2, ..., p$ and z_t is the residual error term [6].

The order of an AR model is determined by the number of immediately preceding values in the process that need to be used for predicting the value at the present time t. In order to identify the order of an AR model, one commonly investigates the PACF of the time series. The partial autocorrelations that are significantly larger or smaller than zero, indicate the lagged terms of x that are useful predictors of x_t [32].

3.2.2 Moving Average Models

A moving average (MA) model is another type of model that can be used for forecasting time series values. A *simple moving average* of order q expresses the estimated value at time t as the average of the data values over an interval of q observations [6]. That is to say, the estimated value at time t is the simple average of the observed value at time t and the preceding $q-1$ time steps. With this approach, all observations are equally weighted with a weight of $1/q$. However, one can add a smoothing parameter, $0 < \alpha < 1$, in order to put more emphasis on recent values. This is called an *exponentially weighted moving average*, and the formula can be expressed as follows.

$$\hat{x}_t = \sum_{t=t-q+1}^{t} \alpha_t x_t, 2 \leq q \leq n, \texttt{ subject to } \sum_{t=t-q+1}^{t} \alpha_t = 1$$

When using the exponentially weighted moving average, it is necessary that the weights sum to 1, and that they reduce in size geometrically [6]. Although *simple moving averages* and *exponentially weighted moving averages* can be used for forecasting, they do not, according to De Smith [6], have a direct link to a statistical model. Instead, a moving average model of order q combines the averaging process with a random process, and can be expressed as follows. Here, $\{z_t\}$ is a set of independent and identically distributed random variables with zero mean and known fixed variance, and β_i is the weight applied to prior values in the process.

$$x_t = \beta_0 z_t + \beta_1 z_{t-1} + ... + \beta_q z_{t-q}$$

Just like the AR model, the order of the MA model is determined by the number of immediately preceding values used to estimate the current value. However, in the case of the MA model, one typically investigates the ACF to determine the order. For an MA(q) model, one can generally say that there are no non-zero autocorrelations for the first q lags, and all autocorrelations beyond q lags are zero [30].

3.2.3 Autoregressive Moving Average Models

By combining the AR models with MA models we can produce a family of models that can be applied to an even wider range of situations [6]. These models are called *autoregressive moving average (ARMA) models*. The AR and MA models are

combined by simply adding them together as a model of order (p, q), where we have p AR terms and q MA terms as follows:

$$x_t = \alpha_1 x_{t-1} + ... + \alpha_p x_{t-p} + z_t + \beta_1 z_{t-1} + ... + \beta_q z_{t-q}$$

The estimated value at time t is then expressed as the sum of p AR terms that compute the current value as the weighted sum of the p most recent values, plus the sum of q MA terms representing the average variation of random variation over the q previous periods. The combined ARMA models can be used to model time series using fewer terms overall than either an AR or an MA model by themselves. However, ARMA models assume that the time series is stationary and therefore differencing may be necessary before fitting an ARMA model.

3.2.4 Autoregressive Integrated Moving Average Models

When differencing is included in the ARMA model, it is known as an autoregressive *integrated* moving average (ARIMA) model [6]. The integrated part of the model refers to the fact that the time series has been initially differenced. Then when the modeling is complete the results are integrated to produce the final estimations. ARIMA modeling is also commonly known as Box-Jenkins modeling, named after the authors who were central to the development of these models.

The first step in the ARIMA modeling process is to difference the time series until it is stationary. The order of differencing is denoted by d, thus making the order of the ARIMA models (p, d, q) and the model can be written as

$$y'_t = c + \phi_1 y'_{t-1} + ... + \phi_p y'_{t-p} + \theta_1 e_{t-1} + ... + \theta_q e_{t-q} + e_t$$

where y'_t is the differenced series, which may have been differenced more than once. The predictors on the right hand side include both lagged values of y_t and lagged errors.

Using the backshift operator, this can also be written as

$$(1 - \phi_1 B - ... - \phi_p B^p)(1 - B)^d y_t = c + (1 + \theta_1 B + ... + \theta_q B^q) e_t$$

$$\begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \text{AR}(p) & d \text{ differences} & \text{MA}(q) \end{array}$$

3.2.4.1 Seasonal ARIMA Models

By including additional seasonal terms in the ARIMA models, they also become capable of modeling a wide range of seasonal data. The *seasonal* ARIMA models, known as SARIMA models, have the order $(p, d, q)(P, D, Q)_m$ where the uppercase orders represent the seasonal components and m is the number of periods per season [11].

The seasonal components consist of similar terms to those of the non-seasonal components, the difference being that they involve backshifts of the seasonal period. For instance, an ARIMA$(1,1,1)(1,1,1)_4$ model, for quarterly data, can be written as

$$(1-\phi_1 B)(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1+\theta_1 B)(1+\Theta_1 B^4)e_t$$

where the seasonal terms are simply multiplied with the non-seasonal terms.

3.2.5 Exponential Smoothing

The previously explained *exponentially weighted moving average* is also called *simple exponential smoothing*. A forecast using simple exponential smoothing with the smoothing parameter α can be expressed as follows.

$$\begin{aligned} &\textbf{Forecast Equation:} && \hat{y}_{t+h} = \ell_t \\ &\textbf{Level Equation:} && \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1} \end{aligned}$$

The method is, according to Hyndman and Athanasopoulos [11], suitable for forecasting data with no trend or seasonal pattern. With Holt's linear trend model, the simple exponential smoothing was extended for data containing a trend. The equations are now divided into one representing the level and one representing the trend.

$$\begin{aligned} &\textbf{Forecast Equation:} && \hat{y}_{t+h} = \ell_t + hb_t \\ &\textbf{Level Equation:} && \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ &\textbf{Trend Equation:} && b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \end{aligned}$$

Here, α is as previously the smoothing parameter for the level ($0 \leq \alpha \leq 1$) and β is the smoothing parameter for the trend with the same restrictions. However, this approach does, as the name suggests, only consider linear trends. With the following variation from Holt's linear trend, the level and slope are multiplied in order to allow for an exponential trend.

$$\begin{aligned} &\textbf{Forecast Equation:} && \hat{y}_{t+h} = \ell_t b_t^h \\ &\textbf{Level Equation:} && \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} b_{t-1}) \\ &\textbf{Trend Equation:} && b_t = \beta \frac{\ell_t}{\ell_{t-1}} + (1-\beta)b_{t-1} \end{aligned}$$

Hyndman and Athanasopoulos [11] do, however, suggest that forecast models using Holt's linear trend, as well as models using exponential trends, tend to over-forecast. In order to avoid this problem, a damping parameter ϕ can be used so that the trend approaches a constant some time in the future.

In order to address seasonality in time series, the Holt-Winters seasonal method introduces yet another equation [11]. There are two variations of the method; the additive and the multiplicative. The additive method is, according to Hyndman and Athanasopoulos [11], preferred when the seasonal variations are constant through the series, and the multiplicative method when the seasonal variations are changing with the level of the series. The additive version is defined as follows, where

$h_m^+ = \lfloor (h-1) \bmod m \rfloor + 1$.

Forecast Equation:	$\hat{y}_{t+h} = \ell + hb_t + s_{t-m+h_m^+}$
Level Equation:	$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$
Trend Equation:	$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}$
Seasonal Equation:	$s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$

And the multiplicative version.

Forecast Equation:	$\hat{y}_{t+h} = (\ell + hb_t)s_{t-m+h_m^+}$
Level Equation:	$\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})$
Trend Equation:	$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}$
Trend Equation:	$s_t = \gamma\frac{y_t}{(\ell_{t-1}+b_{t-1})} + (1-\gamma)s_{t-m}$

The trend and seasonal components explained can also be combined to form other variations of exponential smoothing. Hyndman, Koehler, Snyder, and Grose [16] provides table 3.1 for classification of the different methods. Here, NN represents simple exponential smoothing, AN represents Holt's linear trend method, AA is the Holt Winters' additive method and AM is the Holt Winters' multiplicative method.

Table 3.1: Classification of exponential smoothing methods

Trend Component	**Seasonal Component**		
	N (none)	A (additive)	M (multiplicative)
N (none)	NN	NA	NM
A (additive)	AN	AA	AM
M (multiplicative)	MN	MA	MM
D (damped)	DN	NA	DM

3.2.6 ETS - State Space Models for Exponential Smoothing

A state space model for exponential smoothing consists of a measurement equation describing the observed data and some transition equations describing how the unobserved components change over time [11]. The models produce the same point forecasts as their respective exponential smoothing method, but can in addition generate prediction intervals.

Hyndman, Koehler, Snyder, and Grose [16] follow a framework described by Ord, Koehler, and Snyder [22], which includes a state vector $\vec{x}_t$ and the following state space equations.

$$Y_t = h(\vec{x}_{t-1}) + k(\vec{x}_{t-1})\varepsilon_t$$

$$\vec{x}_t = f(\vec{x}_{t-1}) + g(\vec{x}_{t-1})\varepsilon_t$$

Here $\{\varepsilon_t\}$ is a Gaussian white noise process with mean zero and variance σ^2. Hyndman, Koehler, Snyder, and Grose [16] defines $\vec{x}_t = (\ell_t, b_t, s_t, s_{t-1}, ..., s_{t-(m-1)})$, $e_t = k(\vec{x}_{t-1})\varepsilon_t$, $\mu_t = h(\vec{x}_{t-1})$ and thus $Y_t = \mu_t + e_t$, where μ_t is the one-step forecast at time $t-1$ and e_t is the one-step error forecast. For each exponential smoothing method there exist two models; one with additive errors and one with multiplicative errors. For models with additive errors, the one-step forecast errors are assumed to, just as $\{\varepsilon_t\}$, be a Gaussian white noise process with mean zero and variance σ^2. Thus, in case of additive errors, $k(\vec{x}_t) = 1$ and $Y_t = \mu_t + \varepsilon_t$. The model with multiplicative errors is, according to Hyndman, Koehler, Snyder, and Grose [16], instead written as $Y_t = \mu_t(1 + \varepsilon_t)$. Both the models with additive errors and the one with the multiplicative errors generate the same forecast, but provide different prediction intervals.

To separate the models with additive errors from the ones with multiplicative ones, an additional letter is added to the classifications in table 3.1. The state space models can now be referred to as ETS(*,*,*), where the stars are replaced by type of error (E), type of trend (T) and type of seasonality (S).

3.2.7 Parameter Estimation

For any of the time series models mentioned above, it is important to choose appropriate parameters in order to produce a model of good fit. There are several different ways to compare sets of parameters and the following methods are used during the forecasting section of this thesis.

3.2.7.1 Maximum Likelihood Estimation

Given a density distribution $p_z(z, \theta)$ and sample data $D_N = z_1, z_2, ...z_N$ i.i.d. drawn from this distribution, the joint probability density of the sample data can be written as

$$p_{D_N}(D_N, \theta) = \prod_{i=1}^{N} p_z(z_i, \theta) = L_N(\theta)$$

where for a fixed D_N, $L_N(\cdot)$ is a function of θ and is called the empirical likelihood of θ given D_N. The likelihood function quantifies the relative abilities of the various parameter values to explain the observed data. The maximum likelihood estimate $\hat{\theta}$ is the value for which the empirical likelihood, $L_N(\theta)$, has a maximum [11]. In other words, $L_N(\theta)$ is the probability of observing the given data as a function of θ, and $\hat{\theta}$ is the value that makes the observed data the most probable [9].

3.2.7.2 Conditional Sum of Squares

The conditional sum of squares (CSS) estimator is obtained by minimizing the sum of squared residuals. The estimator is derived from the Gaussian likelihood conditional on initial values and is often denoted the conditional maximum likelihood estimator. For example, in the AR(k) model we set aside k observations as initial values, and conditioning on these implies that Gaussian maximum likelihood estimation is equivalent to CSS estimation [17].

Box, Jenkins, Reinsel, and Ljung [4] defines the CSS function, $S_*(\boldsymbol{\phi}, \boldsymbol{\theta})$, as follows

$$S_*(\boldsymbol{\phi}, \boldsymbol{\theta}) = \sum_{t=1}^{n} a_t^2(\boldsymbol{\phi}, \boldsymbol{\theta} | \mathbf{w}_*, \mathbf{a}_*, \mathbf{w})$$

3.2.7.3 Information Criteria

The Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) are two common measures that are used to evaluate how effectively two different models describe a dataset [6]. They can also be used to compare the same model with different parameter settings in order to find the best fit. The best model is then considered to be the one that minimizes the AIC or BIC value. Complex models, using a large number of parameters, often achieve an improved fit. However, it can be argued that a model with too many parameters provides a much lower level of information about the data, and are often referred to as *over-fitted*. Because of this, AIC and BIC use a penalty factor for the number of parameters.

The AIC is computed as

$$\mathrm{AIC} = n \ln\left(\frac{\mathrm{RSS}}{n}\right) + 2k$$

where RSS is the sum of squared residuals that measures how well the model fits the data, n is the number of observations and k is the number of parameters. The AIC is more generally written as

$$\mathrm{AIC} = -2L + qk$$

where L is the maximized log likelihood function and q, which is usually set to 2, works as a penalty factor on the number of parameters used.

The BIC is similar but it penalizes the additional parameters to a greater degree than the AIC, resulting in the selection of models of a lower complexity. When the data has a normal distribution, it is written as

$$\mathrm{BIC} = \frac{\mathrm{RSS}}{\sigma_e^2} + k \ln(n)$$

where σ_e^2 is the error variance. It can also be written more generally as

$$\mathrm{BIC} = -2L + k \ln(n)$$

.

3.3 Evaluation of Forecasting Models

It may not always be obvious which time series model is the most suitable for a given dataset. Therefore, several measures have been proposed that are commonly used in the literature when comparing the forecast results of different models. However, according to Bergmeir and Benítez [2], all commonly used error measures have shortcomings, and no commonly accepted measure exists that is robust, scale-independent, easy to compute, use and interpret. In this section, we present the measures that are most relevant to this thesis, along with their respective advantages and disadvantages.

3.3.1 Mean Absolute Error

The mean absolute error (MAE) is a measure of the forecast error and is defined as

$$\text{MAE} = \frac{\sum_{i=1}^{n} |A_t - F_t|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

where F_t are the forecasted values, A_t are the actual values and thus e_i are the error values [15].

The MAE measure is among the most intuitive and easiest to interpret, but has the disadvantage that it is a *scale-dependent* measure. This means that it is on the same scale as the data. Therefore, it is not appropriate to use when making comparisons between different time series of different scales [15]. However, it is a popular measure to use when comparing forecast methods on a single time series [11].

3.3.2 Mean Square Error

The mean square error (MSE) is similar to the MAE but is defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(A_t - F_t)^2 = \frac{\sum_{i=1}^{n} e_i^2}{n}$$

where F_t are the forecasted values, A_t are the actual values and e_i are the error values [15]. The MSE is, just as the MAE, a scale-dependent measure.

3.3.3 Mean Absolute Percentage Error

The mean absolute percentage error (MAPE) is a commonly used measure when comparing between different time series since it has the advantage of being scale-independent and is easy to understand [11]. It is defined by Kim and Kim [18] as follows

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{A_t - F_t}{A_t}\right|$$

where N is the number of data points, A_t are the actual values and F_t are the forecasted values.

Although the MAPE measure is scale independent, it has the disadvantage of being infinite or undefined if A_t equals zero. Also, the percentage errors at small values of A_t are high meaning that the measure shows an extremely skewed distribution [15, 2]. A third disadvantage of the MAPE measure is that it puts a heavier penalty on positive errors than negative errors.

3.3.4 Mean Absolute Scaled Error

The mean absolute scaled error (MASE) is yet another measure used for forecast accuracy. It was proposed by Hyndman and Koehler [15] and uses a so called *scaled error* which they define as follows.

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}$$

where e_t is the forecast error at time t, Y_t is the observed value and n is the number of observations.

For seasonal data, the error is similarly defined as

$$q_t = \frac{e_t}{\frac{1}{n-m}\sum_{i=m+1}^{n}|Y_i - Y_{i-m}|}$$

where m is the seasonal period [11].

With this notation, q_t is the absolute error divided by the in-sample MAE from a naïve forecasting method, in this case the random walk method. The MASE is then simply the average of the scaled error q_t.

$$\text{MASE} = \frac{1}{n}\sum_{t=1}^{n} q_t$$

When the value of MASE is smaller than 1, the proposed method gives, on average, smaller errors than the one-step errors from the naïve method. The authors proposing the MASE measure argue that MAE may still be preferred in situations where all series are on the same scale, and that MAPE may be preferred when all data are positive and much greater than 0. However, when the time series are of different scales and include zero, or close to zero, values, the MASE measure should be the preferred one.

3.4 Binary Classification

Binary classification refers to the task of classifying elements in a set into two groups based on some classification rule. In this thesis binary classification will be used both in the anomaly detection, where an element can be either anomalous or not, and in

the fault localization, where an an element is either a root cause or it is not. Several commonly used measures can be applied in order to evaluate the performance of binary classification tests. *Sensitivity and specificity* are two such common measures that together define the ability of a test to detect the presence or absence of a specific condition, in other words a likelihood ratio [1]. Another common measure is *precision*, which can be used when the number of true negatives of a test set is unknown or much larger than the actual number of relevant elements.

Sensitivity: The sensitivity, also known as the true positive rate (TPR) or the *recall*, measures the proportion of positives that are correctly identified as positive [31]. The sensitivity is defined as

$$\text{TPR} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Specificity: The specificity, also known as the true negative rate (TNR), measures the proportion of negatives that are correctly identified as negative [31]. It is defined as

$$\text{TNR} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

Precision: The precision, also known as the positive predictive value (PPV), measures the probability that elements classified as positive are actually positive [24]. It is defined as

$$\text{PPV} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

F Score: The F Score is defined as the harmonic mean of the precision and recall and ranges from zero to one [27]. It is calculated according to the following equation

$$\text{F Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Gain in Certainty: The expected *gain in certainty* is a measure used to compare models and is simply defined by Koepsell and Connell [19] as

$$\text{Gain} = \text{Sensitivity} + \text{Specificity}$$

A good test will maximize the gain and so high values of sensitivity and specificity are desired. A gain equal to one indicates that the test does not provide any information and is thus equivalent to guessing.

4
Forecasting

The first part of this thesis is to forecast values for future points in the time series. This chapter describes how the models that were determined relevant for this thesis were tested and evaluated. The results are then presented and discussed at the end of this section.

4.1 Method

The different models have been evaluated on real customer data and the combined results have been used to determine which model is the most appropriate for the forecasting step.

4.1.1 Evaluation Data

In order to evaluate both the forecasting, and the anomaly detection described in the next chapter, there was a need for labelling the provided time series such that known anomalous points could be used for accuracy calculations. Labelling the time series is, however, a time-consuming process. In order to find whether a suspicious point in a time series is anomalous, rigorous analysis is needed of related time series, systems and other circumstances which may have had an effect at that point in time. Since the purpose of the anomaly detection is to identify suspicious points which may need further investigation, we argue that a rigorous analysis is not necessary for the purpose of the evaluation, and that labelling of suspicious points is sufficient. A problem with suspicious points is, however, that different analysts may find different points suspicious, and that there is no definition of what such a point should look like. In order to, to some extent, avoid a biased labelling, we let three different analysts at the company have a look at the time series and label the points that they considered as suspicious. The analysts labelled 45 different time series and for each one, the results were compared and the labelling selected such that a point was determined to be suspicious if at least two of the analysts agreed with the statement. The 45 labelled time series were then used when performing the evaluation of the forecasting and the anomaly detection.

4.1.2 Model Selection

The data used in this thesis consists of time series of revenues collected for various dimensions. Depending on which dimension the time series represents, different

characteristics are present in the time series. While some series have values fluctuating around 500 000, others will vary around only a few hundred. As the scale of the series decreases, patterns tend to become less apparent and the series noisier. Looking at some of the time series, they also show signs of heteroscedasticity. With all the different kinds of characteristics present in the time series, it is reasonable to assume that the different series require different models. Also, noisy series are naturally hard to predict and may not be possible to model at all due to their randomness. Although the different time series have different characteristics, developing an automated process for choosing the correct type of model depending on these characteristics is beyond the scope of this thesis. Thus, we will try to find the *one* type of model that can most properly represent all of the time series. This is, however, a challenging task, and it may be the case that the best model still produces bad predictions for some of the series. Even though the time series exhibit different characteristics, we can make some assumptions about universal ones due to the specific domain. Since there are numerous different ways to model time series, we have used these assumptions to find models we believe to be appropriate.

As the revenue in the time series is based on ads loaded onto a web-page, it is also based on the number of people visiting that page. As web-page views tend to vary depending on the day of the week, it is expected that the time series shows a weekly seasonal pattern. Although trends can occur as the attention for a site grows or diminishes, we do not expect to find any long-term trends in the series. Due to seasonal patterns and possible short-term trends, the time series can not be expected to be stationary. An initial guess for modelling the time series is therefore to use either a SARIMA model or an ETS model, as both these are commonly used when handling seasonality and non-stationarity. However, the assumptions about the time series might not be correct and because of this, a regular ARIMA model was evaluated as well.

4.1.2.1 ARIMA and SARIMA models

The ARIMA and SARIMA models used in the evaluation were determined automatically for each time series using the `auto.arima` method in the `forecast` package (version 8.3) for `R` (version 3.4.3) [13]. The method first determines the order of differencing by the use of unit-root tests [14]. For non-seasonal data, d is selected by successive KPSS unit-root tests, where a significant result indicates a need for further differencing. For seasonal data, D is set to either 0 or 1 depending on an OCSB test [10]. When D has been selected, d is determined using KPSS unit-root tests. When the order of differencing has been selected, the orders of p, q, P and Q are, as described by Hyndman and Khandakar [14], determined as follows.

First, the following four possible models are evaluated and the one with the smallest AIC value selected as the current model. If $d + D \leq 1$, the models are fitted with $c \neq 0$, and otherwise $c = 0$. Here, m is the seasonal frequency, which in our case was set to 7 for the SARIMA model and 1 for the ARIMA model.

- ARIMA(2, d, 2) if $m = 1$ and ARIMA(2, d, 2)(1, D, 1) if $m > 1$

- ARIMA(0, d, 0) if $m = 1$ and ARIMA(0, d, 0)(0, D, 0) if $m > 1$
- ARIMA(1, d, 0) if $m = 1$ and ARIMA(1, d, 0)(1, D, 0) if $m > 1$
- ARIMA(0, d, 1) if $m = 1$ and ARIMA(0, d, 1)(0, D, 1) if $m > 1$

Secondly, up to thirteen variations of the current model are considered. Whenever a model with lower AIC is found, it becomes the new current model and the process is repeated. This goes on until none of the model variations have lower AIC than the current one. The variations that are considered in the process are models where..

- .. one of p, q, P and D is allowed to vary by $\pm$ 1 from the current model.
- .. p and q both vary by $\pm$ 1 from the current model.
- .. P and Q both vary by $\pm$ 1 from the current model.
- .. the constant c is included if the current model has $c = 1$ or excluded if the current model has $c \neq 0$.

When estimating the parameters of the models during the selection process, the `auto.arima` uses conditional sum of squares and an approximation of the information criteria in order to avoid excessive computation times. The final model is, however, computed using maximum likelihood estimation [10].

The number of seasonal differences are, according to the documentation, sometimes poorly chosen when using the `auto.arima` method [10] and the recommendation is to set $D = 1$ for time series that show a strong seasonal pattern. Since one assumption regarding the time series was that they have a weekly seasonal pattern, an additional SARIMA model was evaluated where $m = 7$ and $D = 1$.

4.1.2.2 ETS models

The ETS models used in the evaluation were determined automatically for each time series using the `ets` method in the `forecast` package (version 8.3) for `R` (version 3.4.3) [13]. The algorithm that the `ets` method uses for choosing the best model is, as described by Hyndman, Koehler, Snyder, and Grose [16], divided into two steps. First, all appropriate ETS models are applied to the series and the parameters optimized. Then, the best model is selected as the model with lowest AIC value. When determining which ETS models are appropriate, the algorithm takes into consideration the existence of zeros and negative values as well as the seasonality of the time series. If there are zeros or negative values in the series, neither multiplicative error models nor additive error models with multiplicative trend- or seasonality are considered. Also, if there is no seasonal period in the data, none of the seasonal methods are considered. The parameters of the ETS models considered in the algorithm can be optimized in different ways. In this thesis, five different optimization criteria have been used during the evaluation; log-likelihood, MAE, MSE, the average MSE of the last 3 points and the standard deviation of the residuals.

The parameters $\vec{\theta} = (\alpha, \beta, \gamma, \phi)$ and initial states $\vec{X}_0 = (\ell_0, b_0, s_0, s_{-1}, ..., s_{-m+1})$ are estimated by minimizing the chosen optimization criteria. The estimation is constrained by restricting the values of α, β, γ and ϕ as follows.

$$0.1 \leq \alpha \leq 0.9$$
$$0.1 \leq \beta \leq 0.9$$
$$0.1 \leq \gamma \leq 0.9$$
$$0.1 \leq \phi \leq 1$$

Hyndman, Koehler, Snyder, and Grose [16] argue that α, β, and γ are usually restricted to values between 0 and 1, but that a smaller range can help avoid instabilities occurring. In addition, the initial states X_0 are restricted so that the seasonal indices add to zero for additive seasonality, and to m for multiplicative seasonality.

4.1.3 Preprocessing the Data

When producing forecasts with ARIMA- or ETS models, previous values in the data are used for the computation. Thus, any outlier in the previous values may heavily impact the accuracy of the forecast. In order to avoid this problem, one would ideally want to remove the outliers from the data that are used in the calculations of the forecast. This approach does, however, come with additional problems. First, in order to remove the outliers, we first need to know where the actual outliers are. Thus, the success of the approach is determined by the amount of labels provided by the operators as well as the accuracy of those labels. The other problem is that once an outlier has been found, it needs to be replaced by an appropriate value, where this appropriate value needs to be defined.

In this thesis, we have evaluated a simpler approach to the outlier removal procedure by making the assumption that any value below the 5th percentile, and any value above the 95th percentile, can be considered an outlier. These outliers are then trimmed to an appropriate value using winsorization. Winsorization is a technique commonly used to handle outliers in a distribution of data where their presence can exert disproportionate influence when conducting statistical analyses [25]. The process of winsorizing is to convert unusually high values to the value of the highest data point that is not considered to be an outlier. In other words, an extremely high value is reduced in magnitude to a value that is still at the high end of the distribution. In the same way, unusually low values are converted to the lowest value that is not considered an outlier. This process preserves the information that a certain data point had among the highest, or lowest, values in a distribution, while protecting against the harmful effects of outliers.

The models were, during the evaluation, fitted to both the raw- and the winsorized data.

4.1.4 Evaluation

When evaluating forecasting methods, the most common approach for data partitioning is, according to Bergmeir and Benítez [2], the use of last block evaluation. With this type of evaluation, a part at the end of each series is reserved and not used

when determining the models. One advantage of this method is that the model can be trained and used as it would in a real application situation. Another advantage is that with the assumption that future values of the series depend on past ones, the natural dependencies of the series are respected [2]. The last block evaluation does, however, have the disadvantage that only one forecast per series can be calculated and also that it does not make full use of the data.

In this thesis, the chosen model should be used for forecasting one point in each series every day, and the error of that forecast should be used to determine whether an anomaly has occurred or not. As the characteristics of the time series may change as time passes, the model should be refitted every time a new forecast is made. The refitting of models may be computationally expensive, but we argue that this can be overlooked as the forecasting will only be performed once each day. In order to properly mimic the application situation when evaluating the models, the use of a *rolling-origin-recalibration* evaluation was determined to be the best fit.

The *rolling-origin-recalibration* evaluation is a type of last block evaluation, but where the forecasts are performed by sequentially moving values from the test set to the training set [2]. Starting with some definitions, the final point of time in the training set is often referred to as the *forecast origin*, and the number of time periods being forecasted called the *forecast horizon*. With the rolling-origin-recalibration evaluation, the forecast origin ranges from t_0 to $T - h$, where T is the last point of time in the series and h is the forecast horizon. For each forecast origin, the model is refitted using all available data in the training set and an h-point forecast generated. Tashman [29] argues that this reoptimization of the model is preferred over using the same parameters for all forecasts, as it desensitizes error measures to events unique to the original fit period. In this thesis, the rolling-origin-recalibration evaluation was performed, and 365 forecasts generated, for each time series with the forecast horizon set to 1 in order to make the evaluation as similar to the real situation as possible.

4.1.4.1 Accuracy Measures

Once the rolling-origin-recalibration evaluation has been performed for a model and time series, we are left with a series of forecasting errors. Since it is inconvenient and time consuming to analyze all these errors individually, we used some common measures to evaluate the accuracy. The first measure used was the MAE. Since the evaluated time series were of different scales, the MAE measure was deemed not fit to compare across all series. Instead, MAE was used to determine which model performed best for all time series individually.

In order to compare the accuracy of the different models across time series, a scale-independent measure was needed. The MAPE measure was not considered appropriate since some of the series contain observations of value 0. Also, since this thesis deals with revenue, a negative error can be regarded as more important than a positive error, but the MASE measure puts a heavier penalty on the latter. To avoid

these problems, the MASE measure was used as a measure for forecast accuracy across all series in the thesis.

Although we want the forecasts to be accurate, we do not want the models to accurately predict anomalous values. That is, the model should not adjust to outliers. Since the time series used for evaluation contain some anomalous values, it is expected that there will be large forecasting errors at these points. Both the MAE and MASE measure makes use of mean values in some sense. However, it is well known that the mean is not a robust measure. Thus, in order for the results not to be disproportionately influenced by outliers, points in the series that had been marked as suspicious by the analysts were removed when calculating the measures.

Since we in this thesis used data for a period of one year, forecasts could be produced for 365 points. However, when producing forecasts for the first points, there is hardly any data in the training set. Thus, the accuracy of the forecasts for these points may not properly represent the overall performance of the model. Because of this, only the last 200 non-anomalous points were used when calculating the MAE and MASE.

4.2 Results

The results obtained from the forecasting are presented below. The models used for the evaluation are given IDs, and table 4.1 shows which model corresponds to each ID.

Table 4.1: IDs used for the different models

ID	Model
ARIMA	ARIMA
SARIMA	SARIMA
SARIMA(D)	SARIMA with $D = 1$
ETS Log	ETS with log-likelihood criteria
ETS MSE	ETS with MSE criteria
ETS MAE	ETS with MAE criteria
ETS AMSE	ETS with average MSE criteria
ETS Stdev	ETS with standard deviation criteria
ARIMA(w)	ARIMA applied to winsorized data
SARIMA(w)	SARIMA applied to winsorized data
SARIMA(D,w)	SARIMA with $D = 1$ applied to winsorized data
ETS Log(w)	ETS with log-likelihood criteria applied to winsorized data
ETS MSE(w)	ETS with MSE criteria applied to winsorized data
ETS MAE(w)	ETS with MAE criteria applied to winsorized data
ETS AMSE(w)	ETS with average MSE criteria applied to winsorized data
ETS Stdev(w)	ETS with standard deviation criteria applied to winsorized data

The MASE was calculated over the 200 forecasted points for each time series and

each model. The box plot in Figure 4.1 shows the distribution of the obtained values, and the statistics of the box plot are shown in table 4.2. From the box plot, one can see that the ARIMA model when applied to both raw and winsorized data performed much worse than any of the other models. However, the statistics of the other models are similar, and it's hard to tell which performed better just by looking at the plot. To investigate the performance, Figure 4.2 shows a heatmap of MASE values for the different time series and models.

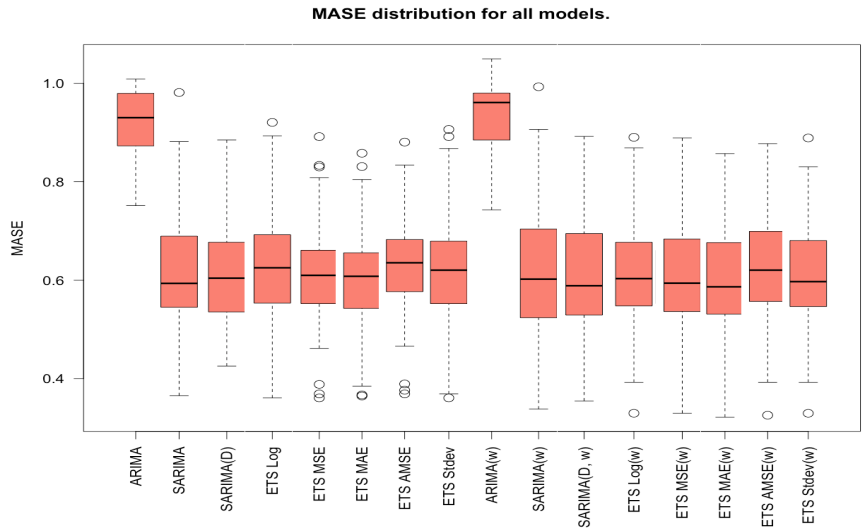

Figure 4.1: Distribution of MASE for different models

Table 4.2: Statistics from the MASE box plot for the different models

Model	Lower Whisker	Lower Hinge	Median	Upper Hinge	Upper Whisker
ARIMA	0.751	0.873	0.930	0.980	1.089
SARIMA	0.365	0.545	0.593	0.690	0.882
SARIMA(D)	0.425	0.535	0.604	0.677	0.885
ETS Log	0.361	0.553	0.625	0.692	0.893
ETS MSE	0.461	0.552	0.610	0.661	0.808
ETS MAE	0.384	0.542	0.608	0.656	0.804
ETS AMSE	0.466	0.577	0.635	0.683	0.834
ETS Stdev	0.369	0.552	0.620	0.679	0.868
ARIMA(w)	0.743	0.885	0.961	0.980	1.050
SARIMA(w)	0.338	0.524	0.602	0.704	0.906
SARIMA(D,w)	0.354	0.529	0.589	0.695	0.892
ETS Log(w)	0.392	0.548	0.603	0.677	0.869
ETS MSE(w)	0.329	0.536	0.594	0.684	0.889
ETS MAE(w)	0.321	0.531	0.586	0.676	0.857
ETS AMSE(w)	0.392	0.557	0.620	0.699	0.878
ETS Stdev(w)	0.392	0.546	0.597	0.680	0.830

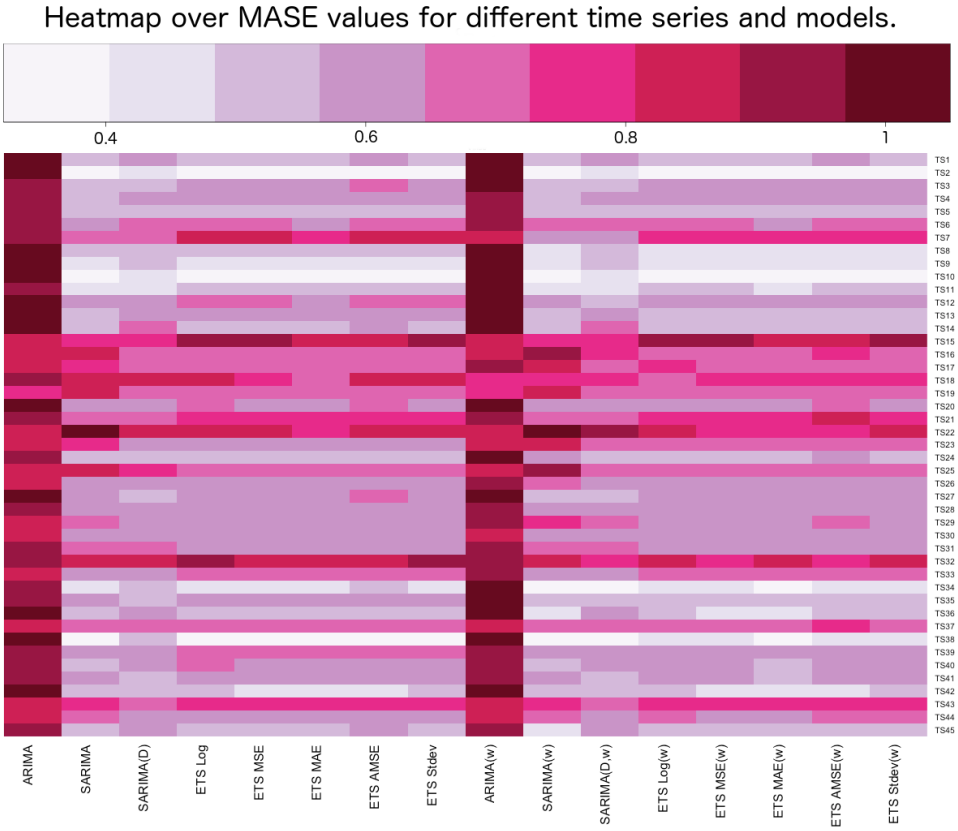

Figure 4.2: Heatmap over MASE values for different time series and models.

In Figure 4.2, the models when applied to the raw data can be seen to the left and the models when applied to the winsorized data can be seen to the right. Just as in the boxplot, we notice here that the ARIMA models had the worst performance for almost all time series, showing a darker shade than all other models in the heatmap. One can also notice that there are series for which the ETS models perform better than the SARIMA models, and other series for which the SARIMA models perform better than the ETS models. For instance, the SARIMA models show a better performance for TS7, TS15 and TS33 while the ETS models show a better performance for TS25 and TS31. There are also many series for which the performance of the regular SARIMA and the SARIMA with $D = 1$ differs, such as TS1, TS16 and TS34. However, which one performed better out of the two varies from series to series. Looking only at the ETS models, they show a similar performance on many of the series. Comparing the ETS models when applied to the raw data, we note that the ETS model with MAE as an optimization criterion never performed worse than any of the other ETS models. When applying the models to the winsorized data, the ETS model with MAE as an optimization criteria performed was outperformed only twice.

In order to see how the performance of the different models vary from series to series, the following plots show some of the time series used for the evaluation. Note that only the last 200 non-anomalous points were used to calculate the forecast measures, and that all previous points were used to calibrate the model for each forecast pro-

duced.

In Figure 4.3, TS39 is plotted first with forecasts produced by the SARIMA model, and then with forecasts produced by the ETS MAE model. For this time series, the SARIMA model produced a better result. One explanation for this can be found in the marked area in the plot, where forecasts produced by the ETS MAE model look similar to that of naïve forecasts, that is the forecast at each point is almost exactly the same value as the previous point. The lines showing the forecasted values thus looks much like a lagged version of the line showing the actual values. The SARIMA model for the same area does, however, produce more appropriate forecasts.

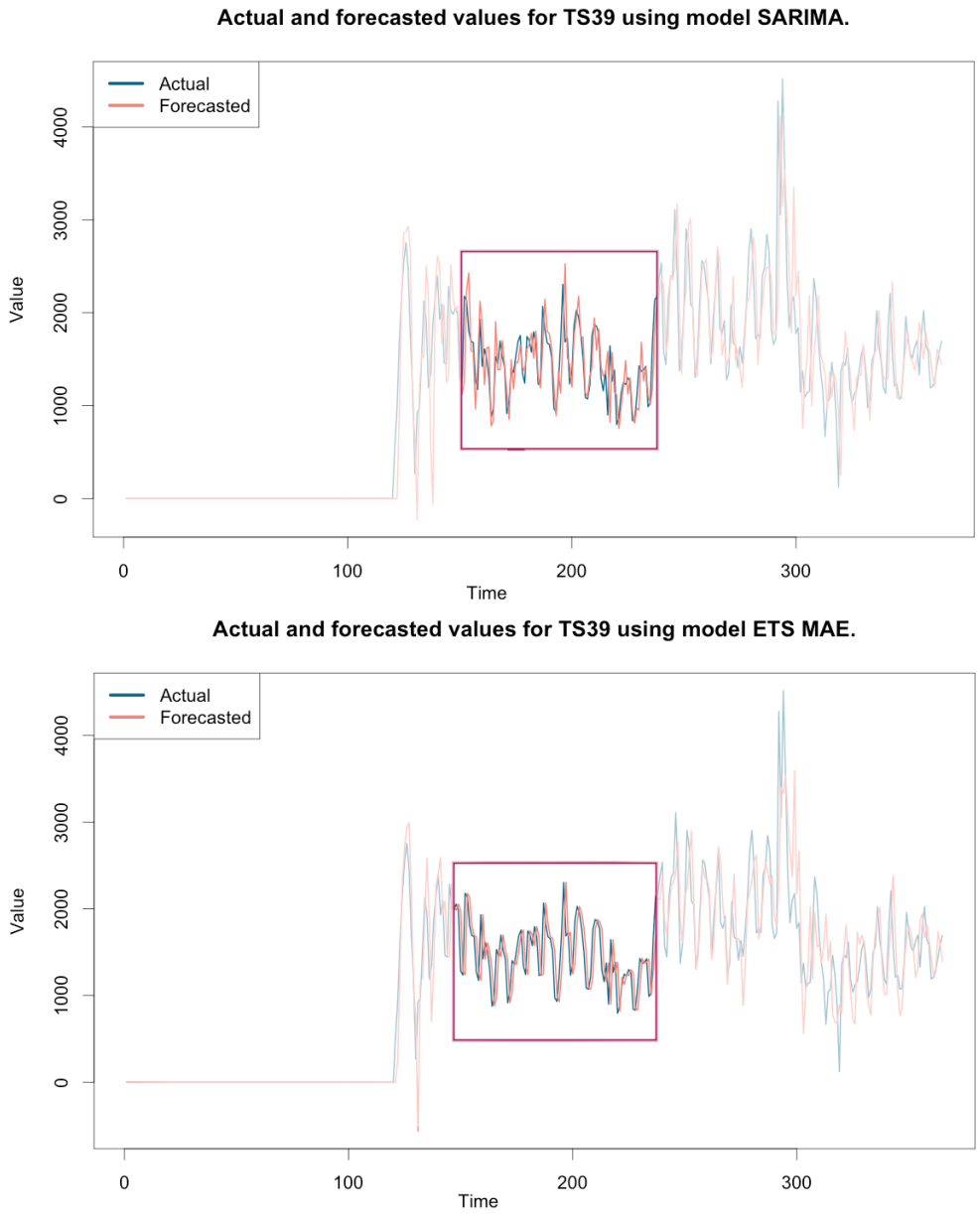

Figure 4.3: Actual values and forecasts produced by SARIMA and ETS MAE for TS39. The marked area shows an area where forecasts are similar to naïve ones for the ETS model, while they are more appropriate for the SARIMA model.

In Figure 4.4, TS17 is plotted with forecasts produced by the SARIMA model and the ETS MAE model. For this series, the ETS MAE model produced better forecasts. One reason for this is shown in the marked area, where the SARIMA model, in contrast to ETS MAE, does not appropriately forecast the drops of the weakly pattern.

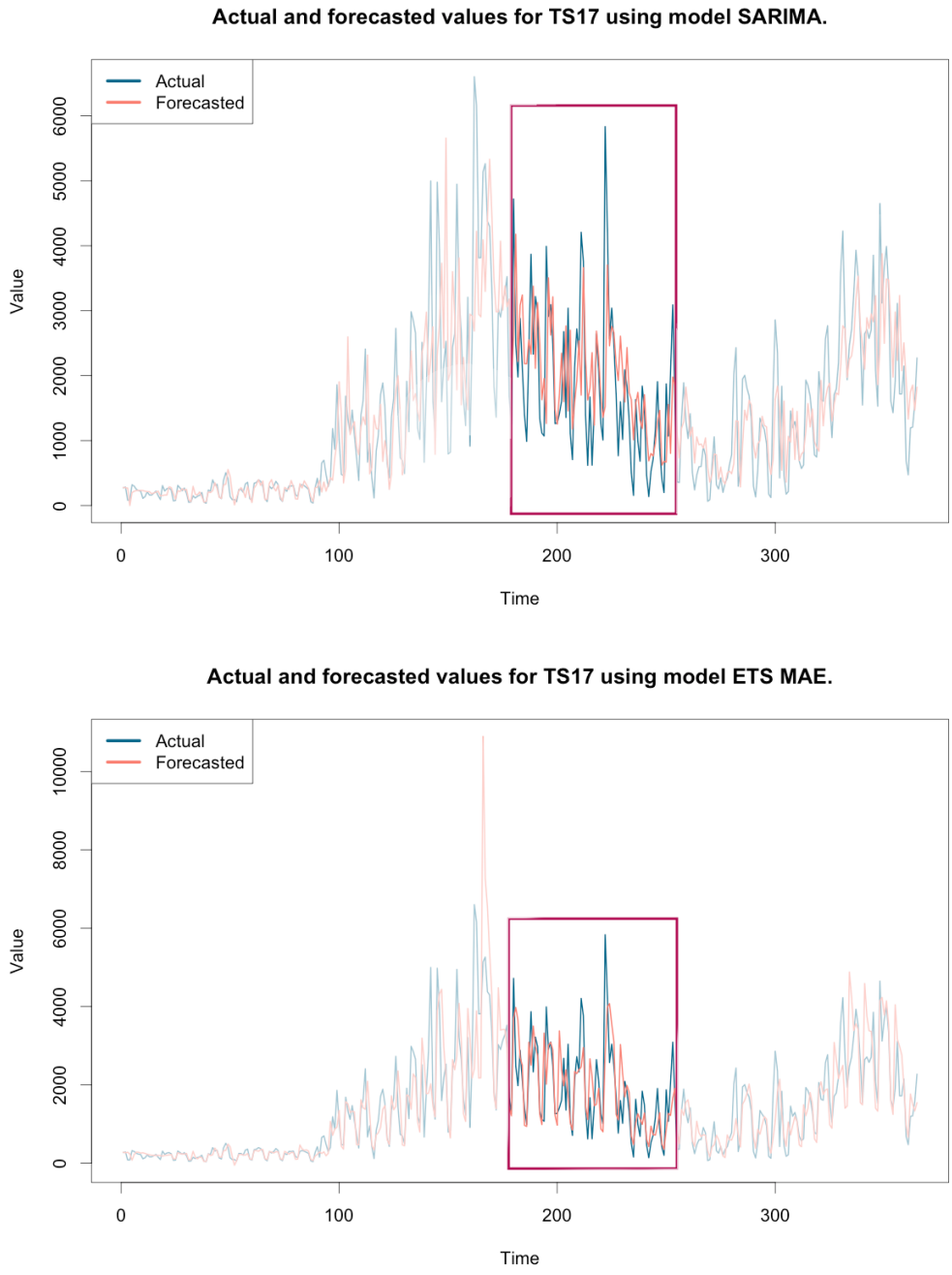

Figure 4.4: Actual values and forecasts produced by SARIMA and ETS MAE for TS17. The marked area shows an area where the SARIMA model does not appropriate forecast the drops of the weakly pattern, while the ETS MAE model do.

Comparing the results produced by the models when applied to the raw data to the results when applied to the winsorized data, one can, by looking at the heatmap in Figure **??**, see that the winsorization sometimes leads to better forecasts (as for ETS MAE and TS11) and sometimes to worse (as for ETS AMSE and TS29). In Figure 4.5 we see forecasts produced by the ETS MAE model when applied to raw and winsorized data respectively for TS16. Looking at the MASE values in Figure 4.2 for this time series, the winsorization appeared to have little effect, but the marked area in the figure shows how the forecasts were affected.

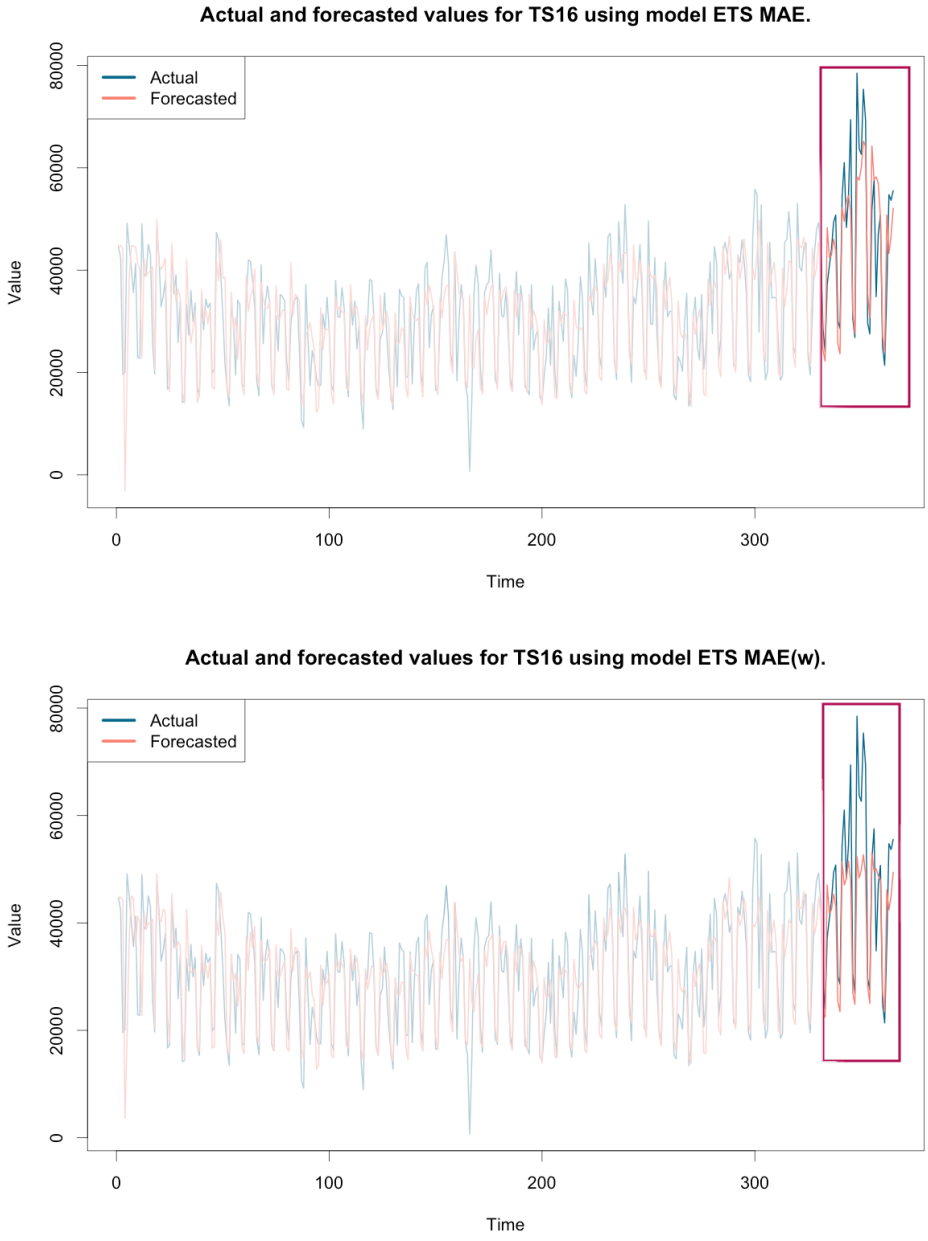

Figure 4.5: Actual values and forecasts produced by ETS MAE and ETS MAE(w) for TS16. The marked area shows an area where the forecasts produced differs due to winsorization.

One further thing that can be noticed is that the different models have different ways of handling consecutive observations of value 0. This is shown in Figure 4.6, where the forecasts produced by the ARIMA, SARIMA and ETS MAE model are plotted.

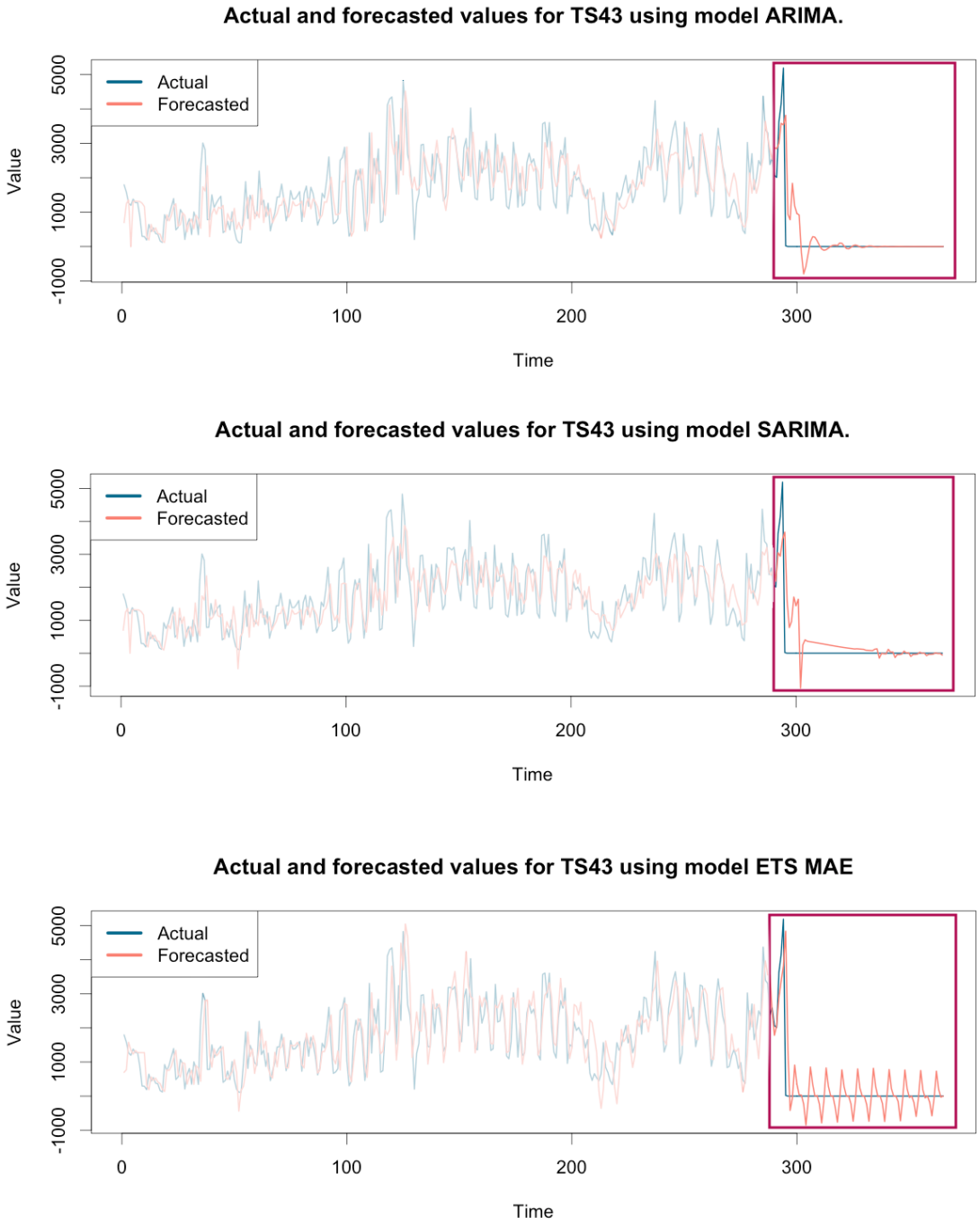

Figure 4.6: Actual values and forecasts produced by ARIMA, SARIMA and ETS MAE for TS43. The marked area shows how the different models handle consecutive observations of value 0 differently.

To determine which model performed better for each time series individually, the MAE measure was used. The results were then summed up to show how many times each model had the best performance. A bar chart of this can be seen in Figure 4.7. As seen in this chart, the ETS MAE model had the best performance the most number of times.

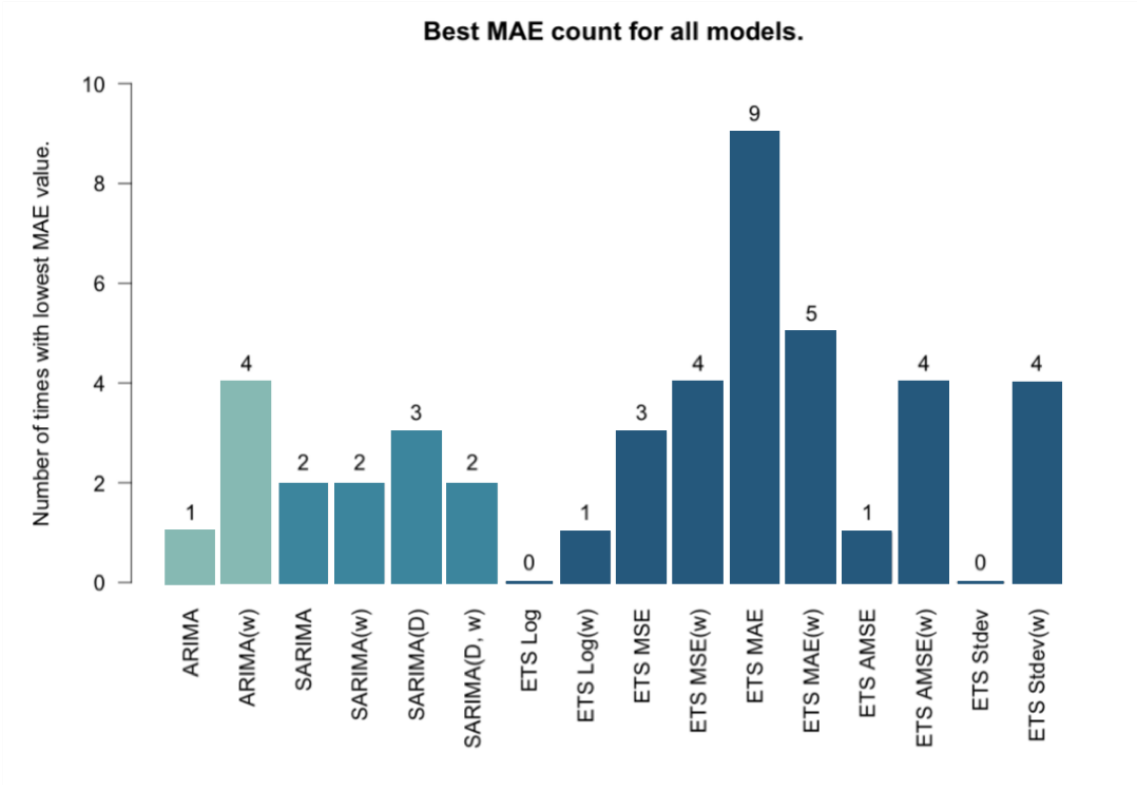

Figure 4.7: Bar chart showing how many times each model had the better forecasting performance.

4.3 Discussion

The first forecasting results consist of the distribution of MASE for the different models shown in Figure 4.1. First of all, we note that the two versions of the ARIMA models performed much worse than the other models. The median of the MASE for these models was approximately 0.93 and 0.96 while for all other models the value was close to 0.6. These two models were also the only two producing MASE values larger than 1, which can be interpreted as a worse performance than that of the naïve prediction. All other models consistently performed better than the naïve prediction, although only slightly better for some time series. The worse performance of the ARIMA models can also be seen in Figure 4.2, and is an expected result and a confirmation of the assumption that the series have seasonal variations.

From Figure 4.1, we can see that most models had an accuracy varying between approximately $0.3 - 0.4$ and $0.8 - 0.9$ for the different time series. While 0.3 or 0.4 can be regarded as a good performance, 0.8 or 0.9 means the model did not perform

much better than the naïve prediction. These variations in forecasting accuracy may be derived from the fact that the different time series have different characteristics and that one model simply may not be able to model all these different characteristics. The fact that different models are appropriate for different time series is also shown in the heatmap in Figure 4.2, where for some time series the SARIMA models perform better and for others the ETS models perform better. However, from the bar chart in Figure 4.7, we see that the ARIMA and SARIMA models together only performed better in 14 cases out of 45, that is in approximately 30% of the cases. The ETS models thus had the better performance for about 70% of the different time series. One reason for this better performance might be the existence of heteroscedasticity in the time series, as such time series may, according to Hyndman and Khandakar [14] be better modelled using exponential smoothing state space models. This can be seen in Figure 4.4, where the variance of the time series is non-constant, and the ETS MAE model outperforms the SARIMA model. However, the SARIMA models seem to outperform the ETS models when there is missing data in the beginning of the time series, as shown in 4.3.

Another reason for the variations in accuracy may be that some of the time series are noisy and very hard to predict. As seen in the heatmap in Figure 4.2, some entire rows have a very light shade and others have a darker one. These rows clearly corresponds to time series of different difficulties. Noisy and hard to predict time series are, however, usually found when the values of the time series are regarded as low. We argue that the accuracy of the time series with low values are not regarded as important as the accuracy of the time series with higher values, as the importance of a revenue time series is directly related to the amount of revenue the time series represents.

As the ETS models performed better than the SARIMA models in 70% of the cases, it was decided that an ETS model should be the preferred one. However, the different ETS models had similar performance which made it difficult to determine the most appropriate one. However, since the ETS model using MAE as an optimization criteria performed better the most number of times according to Figure 4.7, and since it never performed worse than any other model according to Figure 4.2, it was determined to be the preferred model.

Another problem found when evaluating the results, and which also affects the measured accuracy of the models, is the existence of consecutive points with value 0. Looking at the time series shown in Figure 4.6, the three models ARIMA, SARIMA and ETS MAE all handle the last values of the series differently. The first time the series reaches the value 0, this is an obvious anomaly and we do not expect the model to capture the point. However, as the number of consecutive zeros increases, so does the probability that the next value will also be 0. Thus, after a certain amount of days we should expect the 0 value to be the new normal state and the forecast should therefore also be equal to 0. This behaviour seems to have been captured most accurately by the ARIMA model, and the behaviour also seems to have been captured by the SARIMA model. The ETS models did, however, all

show similar behaviour to the one shown in Figure 4.6. That is, the forecasts do not plane out to 0, but keep a seasonal pattern around the 0 value. This behaviour is not optimal, and something that should be considered when using the ETS model in a real application. Another issue shown in the three mentioned figures is that the models all predict values that are less than 0. Since the time series represents revenue, the value can in the real series never fall below zero, and thus the negative predictions need to be handled for example by setting all negative predictions to 0.

In Figure 4.5, we can compare the forecasts generated by the ETS model with MAE as an optimization criteria when applied the raw- and the winsorized data. At the end of the series, the values are somewhat larger than previously as shown in the marked area. These points were marked as anomalies by the experts, and thus we should not expect the models to capture the behaviour. Looking at the figure, we can see that the ETS MAE model, when fitted to the raw data, does seem to capture the behaviour since the forecasts start increasing as the values of the series increase. The forecasts produced with the ETS model fitted to the winsorized data however, does not increase in value and thus seem to capture the normal behaviour of the series more accurately. Therefore, it seems as though the model fitted to the winsorized data is more robust and might perform better during the anomaly detection. Thus, it was decided that the ETS model using MAE as an optimization criteria should be evaluated both applied to the raw- and the winsorized data during the anomaly detection part of the thesis.

5
Anomaly Detection

Once a model has been found that can appropriately forecast future values, the next step of the thesis is to determine whether the *actual* values are anomalous or not. This section describes how different techniques were tested and evaluated and finally, the results are presented and discussed.

5.1 Method

Anomalies are, in this thesis, defined as significant deviations from an expected value, where the expected value is calculated by the forecasting method. The challenge with this approach is to define a suitable threshold to use as a definition for a significant deviation.

5.1.1 Dynamic Thresholds

It is reasonable to assume that, since the time series may exhibit different characteristics and are of different scale, a single threshold value will not be suitable for all time series. Also, the characteristics of the time series may change as time progresses, meaning that a threshold value appropriate today may not be appropriate in the future. The process of anomaly detection should, however, not require operators to manually determine threshold values for the different time series.

One approach to a dynamic threshold is to use the prediction intervals produced by the model. A new prediction interval is determined every time a new forecast is generated, and so the intervals will not have static values. Using this method, any value falling outside of the prediction interval would be considered anomalous. This approach does, however require a specification of what level of confidence the prediction interval should have.

Another approach regarding the threshold values is based on basic idea of the threshold learning algorithm presented by Liu, Zhao, Sui, Cheng, et al. [21]. Here, a threshold τ is used and any observation where $A_t > F_t + \tau$ or $A_t < F_t - \tau$, is considered anomalous. The idea is that the threshold value can be learned and continuously updated from labels provided by the operators. This approach does not require the operators to have any knowledge about the measure used and provides a convenient way to tune the threshold to the appropriate value. Starting with an initial threshold value τ, if a false positive label is provided by an operator at time t, τ is updated

so that the absolute error at time t would not be recognized as an anomaly. On the other hand, if a false negative label is provided, τ is updated so that the absolute error at time t *would* be recognized as an anomaly. The problem with this approach is that an initial value of the threshold must be chosen without any prior knowledge.

Since we in this thesis are dealing with revenue, it is expected that deviations where the actual value is lower than the forecasted value should be recognized for smaller magnitudes than deviations where the actual value is higher than the forecasted value. Thus, we have used two different thresholds when performing the threshold learning algorithm; one for lower deviations and one for higher. The idea is that, starting with two initial threshold values τ_{lower} and τ_{upper}, the procedure is carried out as described above, but only one of the thresholds is updated depending on whether the actual value was smaller or larger than the forecasted value.

5.1.2 Evaluation

In this thesis, both described threshold techniques were evaluated. For the prediction intervals, confidence levels of 50, 55, 60, 70, 80 and 95% were tried in order to see which produced the better results. For the threshold learning algorithm, the initial threshold was set to the 50% prediction interval. The number of true positives, false positive, false negatives and true negatives were calculated for each threshold technique and each time series. For comparison, the values were also calculated using a static threshold. The static threshold was selected individually for each time series as the one producing the highest gain of certainty. The sensitivities and specificities for the different techniques and time series were then calculated, and the distribution of the gain of certainty for the different threshold techniques was compared. The sensitivity and specificity were then calculated for each technique once again, but this time over all time series, and the results were plotted in order to visually compare the values. The values were compared to the use of one static threshold for all series, where the static threshold was selected as the one producing the highest gain of certainty across all series.

During the evaluation, it was noticed that the threshold learning algorithm consistently used too large values for the two thresholds. Since the values of the thresholds are only increased when a false positive is found, one theory for the large threshold values was that large deviations for points not marked as suspicious increase the threshold values by a disproportionate amount. Even though the points may be correctly labelled, we argue that such large non-suspicious deviations should not affect the threshold values unless they are repeated several times. The reason for this is that a few additional false positives are negligible provided that the true positive rate increases. In order to achieve this trade-off, we changed the threshold learning algorithm to only increase the threshold value after five false positives had been found, in which case the new threshold was set to the median of these values. With this approach, one large deviation producing a false positive will not have an impact on the new threshold value, but if the large deviation is repeated five times

and still not considered a true positive, the threshold will be raised to this level.

Evaluating all the different models with all different threshold techniques would give $16 \cdot 8 = 128$ different combinations to compare. In order to reduce the number of combinations, it was determined that only the best model from the forecasting phase should be evaluated. However, in order to evaluate the effect of the winsorization in the anomaly detection as well, it was determined that the best model should be used both with and without winsorization during this evaluation.

5.2 Results

The results obtained from the anomaly detection are presented below. The different threshold techniques have been given IDs, and Table 5.1 shows which ID corresponds to each technique.

Table 5.1: IDs used for the different threshold techniques

ID	Technique
PI(50)	Prediction Interval with 50% Confidence Level
PI(55)	Prediction Interval with 55% Confidence Level
PI(60)	Prediction Interval with 60% Confidence Level
PI(70)	Prediction Interval with 70% Confidence Level
PI(80)	Prediction Interval with 80% Confidence Level
PI(95)	Prediction Interval with 95% Confidence Level
TLA	Threshold Learning Algorithm
TLA(5)	Threshold Learning Algorithm with 5-times updating rule
Static	Static threshold

The distribution of gain in certainty for the different techniques when calculated for each time series individually are displayed in Figure 5.1 and Figure 5.2. Figure 5.1 corresponds to model ETS MAE and Figure 5.2 to model ETS MAE(w). Here we can see that the static threshold has the best gain in certainty for both the raw and winsorized data. The remaining techniques are obtaining more similar values. However, for the raw data, TLA(5) is performing slightly better. The statistics are also shown in table 5.2 and table 5.3.

In Figure 5.3 the sensitivity and specificity for the different techniques when calculated over all time series are plotted. The values, when calculated for the prediction intervals and the static thresholds, are displayed as lines, where each point on the line represents a different confidence level or static threshold value. For the threshold learning algorithms, the threshold is dynamically updated with no static value set and so each technique is only plotted by one point. This shows that the static threshold is no longer the best. Instead TLA and TLA(5), are now the superior techniques with the highest gains in certainty. The sensitivity and specificity values

are shown in table 5.4 where, for the static threshold, only the values that produced the best gain in certainty are shown.

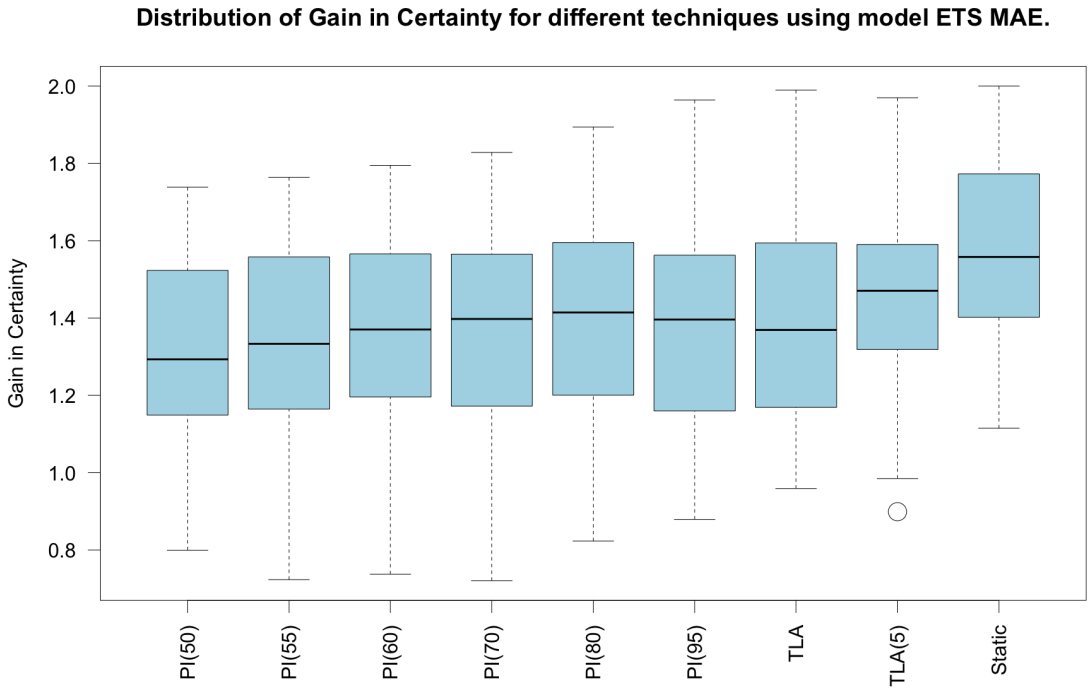

Figure 5.1: Distribution of gain in certainty for different threshold techniques when using model ETS MAE

Table 5.2: Statistics from the gain of certainty box plot for model ETS MAE

Technique	Lower Whisker	Lower Hinge	Median	Upper Hinge	Upper Whisker
PI(50)	0.799	1.149	1.293	1.523	1.738
PI(55)	0.723	1.164	1.333	1.558	1.764
PI(60)	0.738	1.196	1.370	1.566	1.795
PI(70)	0.721	1.172	1.398	1.565	1.828
PI(80)	0.823	1.200	1.414	1.595	1.894
PI(95)	0.879	1.160	1.396	1.563	1.964
TLA	0.959	1.169	1.369	1.594	1.990
TLA(5)	0.985	1.319	1.470	1.590	1.970
Static	1.115	1.402	1.558	1.773	2.000

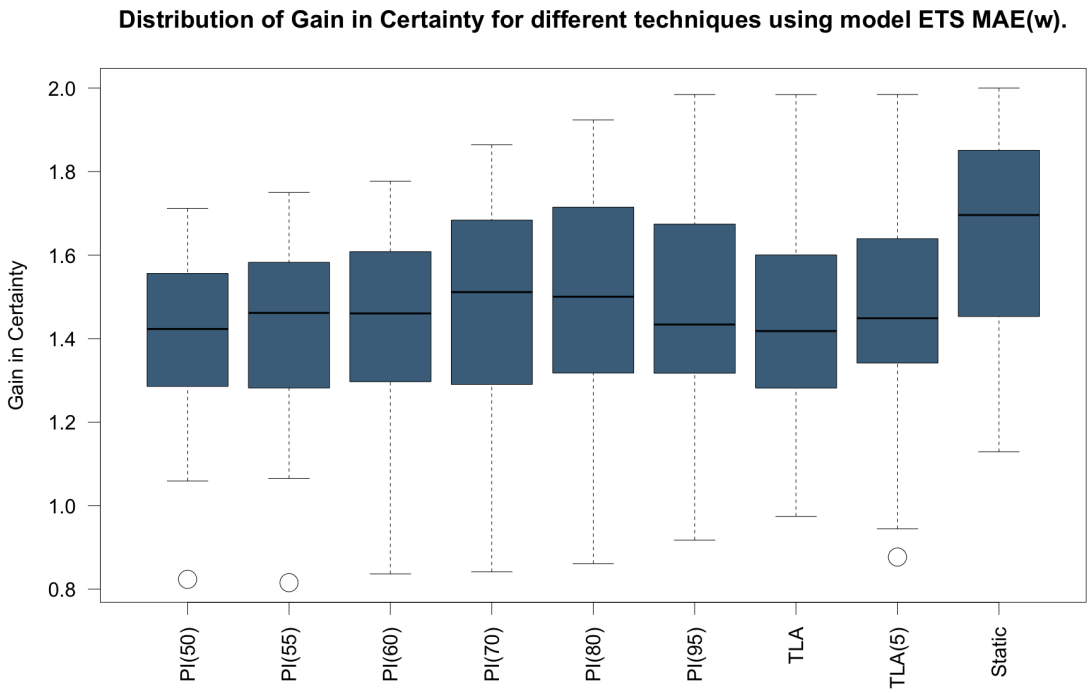

Figure 5.2: Distribution of gain in certainty for different threshold techniques when using model ETS MAE(w)

Table 5.3: Statistics from the gain of certainty box plot for model ETS MAE(w)

Technique	Lower Whisker	Lower Hinge	Median	Upper Hinge	Upper Whisker
PI(50)	1.059	1.286	1.423	1.556	1.712
PI(55)	1.065	1.282	1.462	1.583	1.750
PI(60)	0.836	1.297	1.460	1.608	1.777
PI(70)	0.841	1.290	1.511	1.684	1.864
PI(80)	0.861	1.318	1.500	1.715	1.924
PI(95)	0.918	1.317	1.434	1.674	1.985
TLA	0.974	1.282	1.418	1.601	1.985
TLA(5)	0.944	1.341	1.449	1.639	1.985
Static	1.129	1.453	1.696	1.851	2.000

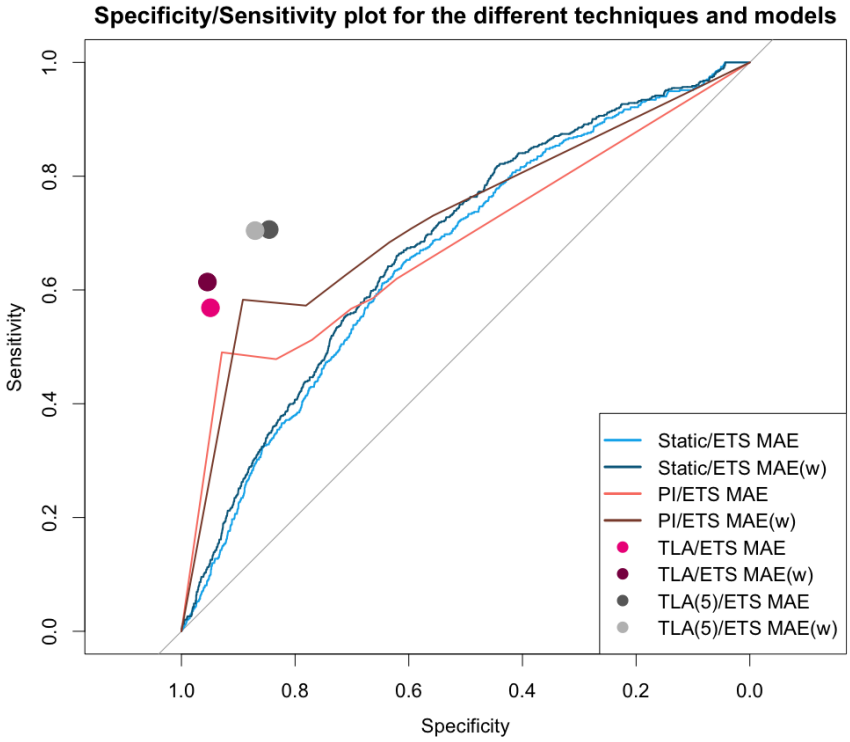

Figure 5.3: Specificity/Sensitivity plot for different threshold techniques and models when calculated over all time series

Table 5.4: Gain in certainty measured over all time series

Technique	Model	Sensitivity	Specificity	Gain in Certainty
PI(50)	ETS MAE	0.620	0.621	1.241
	ETS MAE(w)	0.731	0.557	1.288
PI(55)	ETS MAE	0.586	0.663	1.248
	ETS MAE(w)	0.708	0.594	1.303
PI(60)	ETS MAE	0.567	0.702	1.269
	ETS MAE(w)	0.684	0.634	1.318
PI(70)	ETS MAE	0.512	0.770	1.283
	ETS MAE(w)	0.625	0.712	1.337
PI(80)	ETS MAE	0.478	0.833	1.312
	ETS MAE(w)	0.573	0.781	1.353
PI(95)	ETS MAE	0.490	0.929	1.419
	ETS MAE(w)	0.583	0.892	1.475
TLA	ETS MAE	0.569	0.949	1.518
	ETS MAE(w)	0.614	0.954	1.568
TLA(5)	ETS MAE	0.706	0.846	1.552
	ETS MAE(w)	0.704	0.870	1.575
Static	ETS MAE	0.634	0.624	1.258
	ETS MAE(w)	0.634	0.621	1.255

5.3 Discussion

Looking at the results in Figures 5.1 and 5.2 of the anomaly detection, we can see that the static threshold has the best gain in certainty for both the raw data and for the winsorized data. It is also clear from Tables 5.2 and 5.3 that not only does it have the highest values, it is also the only threshold technique that does not result in values below 1. This means that it is the only technique that never performs worse than a random guess. We can also see that the values for the model fitted to the winsorized data are almost always higher than for the model fitted to the raw data, although the difference is not always significant. For model ETS MAE fitted to the raw data, the second best threshold technique is TLA(5), the threshold learning with the 5-times updating rule. For this technique the lower whisker, lower hinge and the median are all higher than the remaining techniques and the upper hinge and upper whisker are only marginally smaller than for the highest values. For model ETS MAE(w), however, it is much more difficult to determine which technique is second to the static threshold since the highest resulting values are spread among several techniques.

These results indicate that the static threshold technique is the best choice when selecting separate thresholds for each time series. However, this is precisely what we wish to avoid and the goal is to find a dynamic technique that performs well for all time series. Therefore we need to look at the resulting gain in certainty when calculated over all time series. Looking at Figure 5.3 and Table 5.4 we can see that, as expected, the static threshold technique no longer produces the best results. In fact, the static threshold now produces some of the lowest gains in certainty and is thus not suitable for all time series. The techniques that perform best are now the threshold learning techniques, in particular the technique TLA(5) as it has the highest gain for both the raw and winsorized data. This is particularly clear when looking at Figure 5.3, where the curves representing the static thresholds are close to the diagonal whereas the values resulting from the TLA(5) technique are closer to the upper left corner where a perfect result would be. The original threshold learning algorithm, technique TLA, is a close second for all time series. However, this technique has a higher specificity and a lower sensitivity than the revised threshold learning technique resulting in fewer false positives and a larger number of false negatives. We argue that it is better to have a slightly larger amount of false positives to investigate than to have a large amount of false negatives that may be missed, and thus that the revised threshold learning algorithm is the preferred technique. Comparing the model fitted to the raw and the winsorized data, the best overall gain in certainty was found when the learning threshold with the five-times updating rule was applied to the winsorized data. However, in order to avoid assumptions about the anomalies, we decided to use technique TLA(5) applied to the raw data since these values are only marginally smaller than that of the winsorized data.

Although the results produced aren't perfect, we believe that a specificity value of 0.846 is acceptable since, as previously mentioned, it is better to find a few false positives than to miss too many anomalies. However, since the sensitivity is only

0.706, this means that we are in fact missing 30% of the true positives, which is not a great result. There are several reasons that could explain this, the first being the fact that the amount of available labelled data was very limited. Another reason is that the labelled data accessible to us is in fact not necessarily accurate since *points of interest* were manually labelled by several people without any further investigation.

As previously discussed, using the same model for several different time series is a challenging task since they have variations in characteristics, seasonality and variance. The more difficult it is to forecast the values, the harder it will be to find appropriate thresholds to capture the anomalies. Thus, the low sensitivity values may not only be a consequence of poorly labelled data, but also stem from the fact that the performance of the model varies depending on the characteristics of the time series. However, as the noisy and hard to predict series typically correspond to series with low values, false negatives in those series are considered less important than false negatives in series of high values. We have also seen that the forecasting performs badly when the actual values are zero, which in some cases results in a large amount of false positives which affects the results negatively. It may be more appropriate to handle such time series separately, however this would involve additional analysis to determine which time series warrant this separate treatment and thus falls outside the scope of this thesis.

Regarding the threshold learning algorithm, the efficiency largely depends on the amount of provided labels. When performing the evaluation, the algorithm was fed with one label for each point in time. It is, however, unreasonable to assume that this amount of labels would be produced in a real application. Thus, a remaining question is how well the algorithm will perform with only a limited amount of labels. Another question is *how* these labels should be collected. Though false positive labels can somewhat easily be retrieved by an operator labelling the notifications received by the anomaly detection, false negative labels requires more manual effort to collect. One alternative could be to save a certain number of forecasted values with a low confidence level and return them to the user perhaps once a month or week for manual labelling.

6
Fault Localization

Once the ETS MAE model has been applied to produce forecasts and anomalies have been discovered using the TLA(5) threshold technique, the next step of this thesis is to determine *where* the anomalies come from. In other words, the next step is to find the root causes of the anomalies. We propose a way of doing this by looking at the dimensions of the cube representing the total time series in which the anomaly was originally found. This section first describes some related works that have served as inspiration for the fault localization methods presented in this thesis. This is followed by a description of the methods used, the evaluation results and finally a discussion of the results obtained in this step.

6.1 Related Work

There have been a few attempts at fault localization algorithms for time series in recent years. The two most relevant works for the purposes of this thesis are, to our knowledge, the algorithms *Adtributor* [3] and *HotSpot* [28].

6.1.1 Adtributor

Adtributor was proposed by Bhagwan, Kumar, Ramjee, Varghese, et al. [3] and is a tool used for revenue debugging in advertising systems. The authors use an ARMA model to detect anomalies in the time series and then present a root cause algorithm that uses the concepts of *explanatory power*, *succinctness* and *surprise* to identify the subcubes that are most likely to be the cause of a given anomaly.

The explanatory power of a subcube is defined by the authors as the percentage of the change in the total revenue that can be explained by the change in the given subcube's total revenue. Let t represent the cube for which the total anomaly was found, and let c_i represent a subcube created by slicing t along one axis. The explanatory power is then defined as follows, where $F(c)$ is the forecasted total value of cube c and $A(c)$ is the actual total value of cube c.

$$\mathrm{EP}(c_i) = \frac{A(c_i) - F(c_i)}{A(t) - F(t)}$$

The concept of surprise is to quantify the change in distribution of measure values across each dimension in order to isolate anomalous dimensions. It is explained by

the authors that a dimension that has a large change in its distribution is more likely to be a root cause than one which does not exhibit such a change. More specifically, the authors denote the prior probability value of a subcube c_i as $p(c_i) = F(c_i)/F(t)$ and the posterior probability value of the subcube as $d(c_i) = A(c_i)/A(t)$. Observations are considered surprising if the prior probability distribution is significantly different from the posterior probability distribution. The surprise of a subcube c_i is then defined as follows.

$$S(c_i, p(c_i), q(c_i)) = 0.5 \cdot \left(p(c_i) \cdot \log\left(\frac{2 \cdot p(c_i)}{p(c_i) + q(c_i)} \right) + q(c_i) \cdot \log\left(\frac{2 \cdot q(c_i)}{p(c_i) + q(c_i)} \right) \right)$$

The final concept, succinctness, is based on Occam's razor and suggests that the most succinct set, as long as it explains the change within a certain margin of error, is the best explanation for the anomaly.

The algorithm is triggered when an anomaly is detected and works by searching through each partition set for all dimensions individually. For each subcube in the partition set, the explanatory power and surprise are calculated. The subcubes are then sorted in descending order according to their surprise, and traversed once more. Now, if a subcube has an explanatory power larger than a predefined threshold T_{EEP}, it is added to a *candidate set*. The traversal continues until the explanatory power of the candidate set exceeds another predefined threshold T_{EP}. This candidate set is then added to the *explanatory set*. Once the search is complete, the algorithm is left with an explanatory set containing one candidate set for each dimension, and the three most surprising candidate sets are then selected as the most likely root causes.

In Figure 6.1, an illustrating example is shown of subcubes considered by the Adtributor algorithm. The total cube in the example has three dimensions and thus three partition sets will be traversed. The blue subcubes indicate which cubes have been added to a candidate set for each dimension separately.

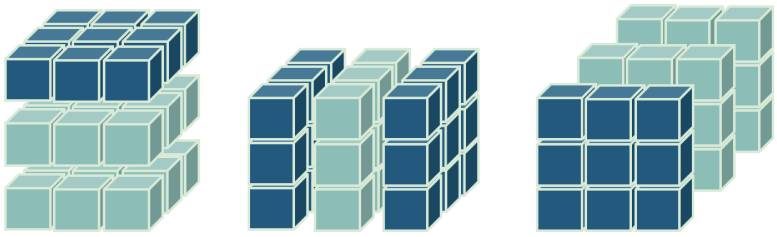

Figure 6.1: An illustration of subcubes considered by the Adtributor algorithm.

The obvious limitation of the Adtributor algorithm is that it will only search for root causes for each dimension individually. While this undoubtedly reduces the amount of time that an analyst would need to spend performing the task manually, they would still have to perform additional searches through the remaining combinations of dimensions to find the actual root cause.

6.1.2 HotSpot

Unlike Adtributor, HotSpot is an algorithm that searches through all different combinations of dimensions before determining which set of subcubes is most likely to be the root cause. It was proposed by Sun, Zhao, Su, Liu, et al. [28], and uses Monte Carlo Tree Search (MCTS) in combination with hierarchical pruning to reduce the search space. To find anomalies, the authors use the mean, which is considered the forecasted value, and the standard deviation to calculate an upper and a lower threshold. If the actual value is beyond the thresholds, then that point is considered anomalous.

Instead of using explanatory power and surprise, HotSpot uses the concept of *ripple effect* to calculate a *potential score*. This score is then used globally to compare the potential of different subcube sets to be the root cause. The ripple effect is based on the intuition that when the value at a root cause subcube suddenly increases or drops, all cells within that cube will also change accordingly. Thus, the ripple effect refers to how an anomaly is distributed through the cells of a cube. Assuming that a certain cube $c \notin \text{LEAF}$ is the root cause of an anomaly, then the authors suggest that all subcubes $l \in \text{LEAF}$ where $l \subset c$ will get their proportional share of the change in c's total value according to the following equation.

$$V(l) = F(l) - (F(c) - A(c)) \cdot \frac{F(l)}{F(c)}, F(c) \neq 0$$

If c is the root cause, then the subcubes $l \in \text{LEAF}$, where $l \not\subset c$, should not be affected by the anomaly and thus their actual value should be close to the forecasted one. Using this notion, one can deduce new values for all $l \in \text{LEAF}$ and compare them to the actual values in order to determine the potential of c to be the root cause. The deduced value of a subcube $l_1 \in \text{LEAF}$, where $l_1 \subset c$, is then set to $V(l_1)$ while the deduced value of another subcube $l_2 \in \text{LEAF}$, where $l \not\subset c$, is set to $F(l_2)$. For a set S, the property can be reused for all subcubes $c \in S$. The distance between the deduced values and the actual values is calculated using the Euclidean distance for vectors. The distance is then divided by the distance between the forecasted values and the actual values of the same subcubes. The smaller the quotient is, the more potential a subcube has to be the root cause. The potential score is defined as follows.

$$\text{PS} = \max\left(1 - \frac{d(\vec{a}, \vec{v})}{d(\vec{a}, \vec{f})}, 0\right)$$

Here, $\vec{a}$ is the vector containing the actual values of LEAF subcubes, $\vec{f}$ are the forecasted values and $\vec{v}$ are the deduced values. The score ranges from 0 to 1, where a higher score indicates a higher potential and is used as the value function in MCTS. Like the Adtributor algorithm, the succinctness principle is applied when determining the most likely root cause set among several sets with the same potential score.

The algorithm can be summarized as follows. Starting with each dimension separately, the potential scores of all the subcubes in the partition set of that dimension

are calculated. These scores are then used as value functions in MCTS in order to find the *best set* of subcubes for each partition set. This is done by constructing sets of different combinations of subcubes and calculating the potential scores of the new sets. A best set is found when a set with a potential score that exceeds a predefined threshold PT is found, or MCTS reaches a certain number of iterations. In the latter case, the set with highest potential score is returned as the best set. After considering each dimension separately, HotSpot considers combinations of two dimensions. However, before moving on, hierarchical pruning is applied by pruning any subcubes with their corresponding cells, that are not in the returned best set. HotSpot continues adding dimensions and finding best sets until all combinations have been considered or the threshold PT has been reached. In the end, if the threshold has not been reached, the best set with the highest potential score out of all different combinations of dimensions will be returned as the root cause set.

In Figure 6.2, an illustrating example is shown of subcubes considered by the HotSpot algorithm. The total cube in the example has three dimensions and thus six partition sets will be traversed. The blue subcubes indicate which cubes have been added to the best set for each combination of dimensions.

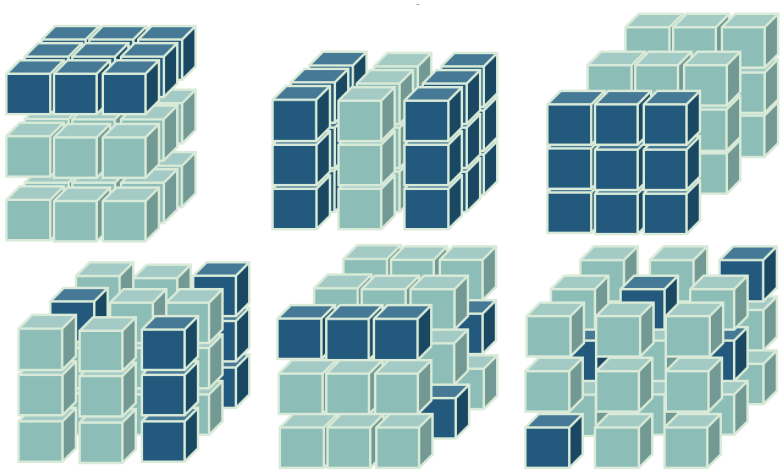

Figure 6.2: An illustration of subcubes considered by the HotSpot algorithm.

We demonstrate how HotSpot finds the root cause set with an example used in the paper presenting the algorithm. The names of the dimensions are, however, changed to match the domain of this thesis. In the example, the cube representing the total revenue has two dimensions; Partner and Ad Unit, and can thus be visualized as shown in Table 6.1. In the table cells, the first value shown represents the forecasted value for that cell and the second represents the actual value. The total, shown to the bottom and to the right, represents the forecasted- and actual values of total revenue for the different subcubes. The correct root cause for this example is the set {Partner 1, Partner 2}.

Table 6.1: An example of data used to calculate potential scores. The values in this example are copied from the paper presenting the HotSpot algorithm [28]. The first values in each cell represent the forecasted value and the second ones represent the actual values.

	Partner 1	**Partner 2**	**Partner 3**	**Total**
Ad Unit 1	$20 \rightarrow 14$	$15 \rightarrow 9$	$10 \rightarrow 10$	$45 \rightarrow 33$
Ad Unit 2	$10 \rightarrow 7$	$25 \rightarrow 15$	$20 \rightarrow 20$	$55 \rightarrow 42$
Total	$30 \rightarrow 21$	$40 \rightarrow 24$	$30 \rightarrow 30$	$100 \rightarrow 75$

The best set for each partition set is found first. We can slice the total cube along the x-axis, representing the partner dimension, the y-axis, representing the ad unit dimension, or along both of the axes. Slicing the cube along both axes will result in a subcube belonging to the LEAF set. The vector of forecasts for the LEAF set, $\vec{f}$, is defined as $(20, 15, 10, 10, 25, 20)$ and the vector of actual values, $\vec{a}$, is defined as $(14, 9, 10, 7, 15, 20)$. The partition set of the first dimension, partner, contains three subcubes, each representing the total revenue for Partner 1, Partner 2 and Partner 3 respectively. These subcubes can be combined into six different sets. For the set containing only Partner 1, the deduced values are, following the previously explained notion of the ripple effect, calculated to $(14, 15, 10, 7, 25, 29)$ and the potential score of the set is 0.13. The best set for this dimension is {Partner 1, Partner 2} with a potential score of 1.

For the second dimension, ad unit, the best set is {Ad Unit 1, Ad Unit 2} with a potential score of 0.47. When combining both the parter- and ad unit dimension, we get a partition set consisting of LEAF subcubes only. Thus, the ripple effect cannot be applied to an assumed root cause set S since the previous definition of deduced values for a subcube c assumes that $c \notin$ LEAF. In personal communication with the authors of the paper, they stated that the deduced value for a LEAF subcube l when $S \subseteq$ LEAF, is simply $A(l)$ if $l \in S$ and $F(l)$ otherwise. Following this notion, there are two sets with a potential score of 1 in the example, the first one being the set containing all LEAF subcubes and the second one being {(Partner 1, Ad Unit 1), (Partner 1, Ad Unit 2), (Partner 2, Ad Unit 1), (Partner 2, Ad Unit 2)}. The second one is chosen as the best set for the cube since it contains fewer subcubes. Out of all different best sets, {Partner 1, Partner 2} is chosen as the root cause by the same criteria.

Although the HotSpot algorithm correctly identifies the root cause set for the above example, the perfectly accurate forecasts are not very realistic. Thus, we present another example where the same values are used but small inaccuracies are added to the forecasts. The example is presented in Table 6.2.

Table 6.2: An example of data used to calculate potential scores. The first values in each cell represents the forecasted values and the second ones represent the actual values. The values in this example are first copied from the paper presenting the HotSpot algorithm [28], and then inaccuracies have been added to the forecasts.

	Partner 1	**Partner 2**	**Partner 3**	**Total**
Ad Unit 1	$20 \rightarrow 14$	$15 \rightarrow 9$	$9.9 \rightarrow 10$	$44.9 \rightarrow 33$
Ad Unit 2	$10 \rightarrow 7$	$25 \rightarrow 15$	$19.9 \rightarrow 20$	$54.9 \rightarrow 42$
Total	$30 \rightarrow 21$	$40 \rightarrow 24$	$29.8 \rightarrow 30$	$89.8 \rightarrow 75$

In the first example, {Partner 1, Partner 2} was the best set for the partner dimension. This time the same set is chosen, however, the potential score is now 0.99. Similarly, for the ad unit dimension, the same set is chosen as the best set but this time with a potential score of 0.3. Looking at the combination of the two dimensions, the set containing all subcubes is now the only set with a potential score of 1. Since this is the largest potential score found over all partition sets, this set will also be returned as the root cause. Note here that the change in the two LEAF subcubes (Partner 3, Ad Unit 1) and (Partner 3, Ad Unit 2) have a change in the opposite direction of the total change. This example demonstrates that the results from the algorithm are affected even by very small forecast inaccuracies.

The reason for the results being affected can be derived from the calculations of potential scores for LEAF subcubes. When the assumed root cause set S consists of LEAF subcubes, the deduced value for a subcube $l \in S$ is equal to $A(l)$. However, this implies that the distance between the actual value and the deduced value will be 0. Thus, when $S = \text{LEAF}$, $\vec{a} = \vec{v}$ and $d(\vec{a}, \vec{v}) = 0$. Inserting this into the equation for potential scores, we get a potential score of 1 regardless of the value of $d(\vec{a}, \vec{f})$. This means that the potential score of a set containing all LEAF subcubes will always be equal to 1. With perfectly accurate forecasts, this does not matter since the real root cause set will also have a potential score of 1 and be chosen based on succinctness as shown in the first example. However, in reality forecasts are generally not perfectly accurate, and the real root cause set will rarely have a potential score of 1. Therefore, the best set out of all combinations in all partition sets will always be the set containing all LEAF subcubes. Moreover, adding a subcube $l \in \text{LEAF}$ to the set $S \subseteq \text{LEAF}$ will always increase the potential score of S, since the distance between the actual value and the deduced value for l will always be 0. This also holds for the case where the change in l is in the opposite direction of the total change, as demonstrated in the second example.

The calculation of the potential scores of LEAF subcubes in HotSpot is clearly a limitation of the algorithm. However, the authors set a threshold for the potential score and the algorithm is terminated whenever said threshold is reached. Thus, if the threshold is exceeded before reaching the LEAF subcubes, this limitation will have no affect on the results.

6.2 Method

Due to the limitations of both HotSpot and Adtributor, it was determined that neither algorithm should be used as is for this thesis. Instead, two other approaches were examined. In the first approach, the HotSpot algorithm was modified to calculate more appropriate potential scores for LEAF subcubes, and the concept of explanatory power was incorporated in the algorithm. In the second approach, the Adtributor algorithm was altered to search through more combinations of dimensions and one concept from the HotSpot algorithm was incorporated in an attempt to further improve the results.

6.2.1 Revised HotSpot

In order to motivate the modifications made do the HotSpot algorithm we present yet another simple example in Table 6.3. This example is based on real data, although simplified for clarity and values were altered due to confidentiality. In the example, we have two partners and three ad units. One of the ad units exists for both partners, while the other two exist only for the first partner. In the table, the first value in each cell represents the forecasted value and the second value represent the actual value.

Table 6.3: An example of data used to calculate potential scores. The first value in each cell represents the forecasted value and the second represents the actual.

	Partner 1	**Partner 2**	**Total**
Ad Unit 1	0.1 → 50 000	N/A	0.1 → 50 000
Ad Unit 2	0.2 → 100 000	N/A	0.2 → 100 000
Ad Unit 3	5 000 → 5 050	1 000 → 1 040	6 000 → 6 090
Total	5 000.3 → 155 050	1 000 → 1 040	6 000.3 → 156 090

In this example, a significant deviation in the total value has been detected, as it has increased from 6000.3 to 156090. Looking at the example, it is obvious that the root cause for this anomaly can be found in Ad Unit 1 and Ad Unit 2. The values of the third ad unit have also increased, but by such a small margin that it can be considered an expected fluctuation. Following the idea behind the HotSpot algorithm, we calculate the potential scores for all combinations of subcubes in all partition sets. For example, in the partner dimension partition set we have three different combinations to consider; {Partner 1}, {Partner 2} and {Partner 1, Partner 2}. Taking the first of these, we get the deduced values $\vec{v} = (3.1, 6.2, 155040.7, 1000)$ for the LEAF subcubes. With $\vec{a} = (50000, 100000, 5050, 1040)$ and $\vec{f} = (0.1, 0.2, 5000, 1000)$ we then get a potential score of 0. In fact, for the partner dimension partition set, Partner 2 has the highest potential score with a value of $6.4 \cdot 10^{-8}$. For the ad unit partition set, the highest potential score is found for the set {Ad Unit 1, Ad Unit 2, Ad Unit 3} with a value of 0.9997. However, the highest potential score of all sets is found in the partition set containing the LEAF subcubes, where, if all subcubes in the set are included, the potential score is equal to 1. This has to do with the

previously mentioned drawbacks of the potential score calculations for LEAF subcubes. That is, we set $v(l) = a(l)$ and thus $\vec{a} = \vec{v}$ and $d(\vec{a}, \vec{v}) = 0$.

The example here does not just show the drawbacks of using the actual value as the deduced value when calculating the potential score for LEAF subcubes, but also shows other disadvantages of the method. First, Partner 2 has the highest potential score in the partner dimension partition set. Thus, if hierarchical pruning was used, then all cells in the Partner 1 subcube would be pruned even though the these cells are clearly more likely to be root causes. Due to this shortcoming and the fact that the algorithm in our domain would only be executed once each day, we argue that the hierarchical pruning can be left out of our implementation.

One further issue that can be noted is the fact that {Ad Unit 1, Ad Unit 2, Ad Unit 3} is the set with highest potential score for the ad unit partition set. This set is closely followed by {Ad Unit 1, Ad Unit 2} which has a potential score of 0.9994. The potential score increases when adding the third ad unit to the set since the deduced values for LEAF subcubes within the Ad Unit 3 subcube will in this example be closer to the actual values than what the forecasted values would be. However, we argue that adding the third ad unit to the set should *not* increase the potential score, as it does not contribute with a substantial amount to the total deviation. In order to avoid this problem, we have in the revised version incorporated the concept of explanatory power in the algorithm. We argue that a subcube with an explanatory power < 0.01 is very unlikely to be a root cause, and thus such subcubes will be pruned in each partition set separately before conducting the search over combinations.

Going back to the problem of deduced values for LEAF subcubes, we propose a new solution. If a single subcube in LEAF is the root cause for the anomaly, then with perfectly accurate forecasts, this subcube would account for the entire deviation in the total value. However, if the root cause is a set of multiple LEAF subcubes we can not deduce values as easily. The ripple effect can, as explained previously, not help us and since the deviations are unexpected, we can not calculate the proportion of the total anomaly that the different LEAF subcubes should account for. Instead, we propose a simplified measure to use for the assumed root cause set $S \subseteq \text{LEAF}$, where the total anomaly is distributed equally among all subcubes $l \in S$. Thus, if $N(S)$ is the number of subcubes in S, then the deduced value for a subcube $l \in S$ is calculated as follows, where t is the cube in which the total anomaly was found.

$$V(l) = F(l) - \frac{F(t) - A(t)}{N(S)}$$

We now apply the new concepts to the same example as previously described. For the partner dimension partition set, {Partner 2} has an explanatory power of only 0.0003 and will thus not be considered as a root cause subcube. The only possible set is now {Partner 1} which has a potential score of 0. In the ad unit partition set, {Ad Unit 3} will also be pruned. Instead, {Ad Unit 1, Ad Unit 2} now has the highest potential score of 0.9994. As for the LEAF subcubes, the highest potential

score, with a value of 0.68 can be found for the set {(Partner 1, Ad Unit 1), (Partner 1, Ad Unit 2)}. Overall, the highest potential score is found in the ad unit partition set. The modified HotSpot algorithm can be found in Algorithm 1.

Algorithm 1 The modified HotSpot algorithm.

```
function ModifiedHotSpot(PT: threshold, M: max iterations)
    potentialSets = []
    for each partition set P do
        for each subcube c ∈ P do
            c.ps = potential score for c
            c.ep = explanatory power for c
            reject c if c.ep < 0.01
        end for
        for 1..M do
            select: a node n is selected from current state tree S
            expand: a new node n' = n ∪ x is added to S, x = arg max_{c∈(P−n)} c.ps
            evaluate: n'.ps = potential score for n'
            backup: update values on path to n'
            if n'.ps ≥ PT then
                bestSet = n'
                break
            end if
        end for
        bestSet = arg max_{n∈S} n.ps
        potentialSets.add(bestSet)
    end for
    potentialSets.sortDescend(bestSet.ps)
    return potentialSets.take(3)
end function
```

The original HotSpot algorithm only takes the one set with highest potential score to present as the root cause. That is, the algorithm assumes that the root causes are confined within one partition set only. The authors of the original algorithm justify this by pointing out that simultaneous root causes in multiple partition sets are extremely rare in reality [28]. However, some of the test cases used for evaluation actually *do* contain potential root causes in multiple partition sets. This may be due to the fact that the data of our domain is different from that used for developing the original algorithm, or it may be the case that the root causes in our data are labelled incorrectly. To overcome this problem, we display the top three sets instead of just one.

6.2.2 Recursive Adtributor

As previously mentioned, the Adtributor algorithm looks at each dimension individually and determines the candidate sets using surprise and explanatory power. In order for this algorithm to work for multiple combinations of dimensions, we apply

the algorithm recursively to each one of the subcubes in the candidate sets. Every time the algorithm is applied, the new candidate sets are saved as an attribute of the subcube, which in the end means the candidate sets can be found in a tree like structure. The original Adtributor algorithm as explained by Bhagwan, Kumar, Ramjee, Varghese, et al. [3] is presented in Algorithm 2 and the recursive version in Algorithm 3

Algorithm 2 The original Adtributor algorithm.

```
function ADTRIBUTOR(t: total cube)
    for each dimension d do
        for each subcube c ∈ d.partitionSet do
            p = F(c)/F(t), q = A(c)/A(t)
            c.surprise = S(c, p, q)
        end for
    end for
    explanatorySet = []
    for each dimension d do
        sortedC = d.partitionSet.sortDescend(c.surprise)
        candidateSet = {}, explains = 0, surprise = 0
        for each c ∈ sortedC do
            ep = (A(c) − F(c))/(A(t) − F(t))
            if ep > T_EEP then
                candidateSet.add(c)
                surprise += c.surprise, explains += ep
            end if
            if explains > T_EP then
                candidateSet.surprise = surprise
                explanatorySet.add(candidateSet)
                break
            end if
        end for
    end for
    explanatorySet.sortDescend(candidateSet.surprise)
    return explanatorySet.take(3)
end function
```

Algorithm 3 The recursive Adtributor algorithm.

```
function RecursiveAdtributor(t: total cube)
    explanatorySet = Adtributor(t)
    for each candidateSet in explanatorySet do
        for each subcube c in candidateSet do
            if c ∉ LEAF then
                c.explanatorySet = RecursiveAtributor(c)
            end if
        end for
    end for
    return explanatorySet
end function
```

In Figure 6.3, one step of the recursive Adtributor is illustrated. First a subcube of a partition set is added to the candidate set. The Adtributor is then applied once again with this subcube as the total cube, thus considering the partition sets for each individual dimension of that subcube.

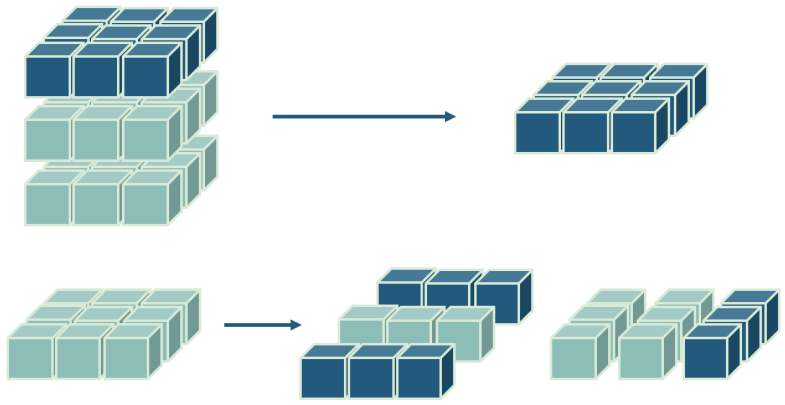

Figure 6.3: An illustration of subcubes considered by the Recursive Adtributor algorithm.

6.2.3 Revised Recursive Adtributor

During the evaluation of the recursive Adtributor algorithm we noticed a few things. First, in our case the T_{EEP} threshold needs to be set very low, as root causes sometimes can be found where the explanatory power is only one or two percent and we don't want to miss these subcubes. Secondly, the success of the algorithm is largely dependent on the value of the T_{EP} threshold. Setting it too high will lead to the inclusion of subcubes that are not relevant, and setting it too low will exclude relevant subcubes. Thirdly, the algorithm will keep looking for root causes until a LEAF subcube is found, although the root cause may in fact be located in a partition set with a smaller number of dimensions.

In an attempt to address the first two of the aforementioned problems, we suggest the following; remove the T_{EP} threshold and use some other, more intuitive, criterion in conjunction with the T_{EEP} threshold. By removing the T_{EP} threshold, the algorithm will look through all subcubes and thus not risk missing any relevant ones. However, in order not to include too many subcubes we need a better criterion to determine which subcubes are relevant. Since the T_{EEP} threshold needs to be set very low in our case, this criterion alone is not sufficient to determine whether a subcube could have caused an anomaly in others. However, it can still serve as a minimum requirement and make sure that subcubes are only included in the candidate set if their change was in the same direction as the total anomaly. As an additional requirement, we suggest the use of prediction intervals. The reasoning behind this suggestion is that in order for a subcube to be considered as the root cause, it should not only contribute to a proportion of the change, but also have some unexpected variation.

One of the core ideas of HotSpot that was added to our revised Adtributor algorithm is the *ripple effect*. This implies that if a subcube c is the root cause of an anomaly, then the values of all the cells within that subcube would be distributed as expected. However, this should also apply to the different partition sets that can be constructed for the subcube. That is, if c is a root cause, all subcubes in any of the partition sets should have gotten their proportional share of the change. Thus, if all subcubes in a partition set are part of the candidate set, then the entire set has been affected and it would make little sense to look at further breakdowns. Thus, in an attempt to solve the third problem previously mentioned, we decided that the revised recursive Adtributor should stop the search for candidates if we find that the candidate set contains all subcubes of the currently examined partition set. The revised recursive Adtributor can be found in Algorithm 4

Algorithm 4 The revised recursive Adtributor algorithm.

```
function RevisedAdtributor(t: total cube)
    for each dimension d do
        for each subcube c ∈ d.partitionSet do
            p = F(c)/F(t)
            q = A(c)/A(t)
            c.surprise = S(c, p, q)
        end for
    end for
    explanatorySet = []
    for each dimension d do
        candidateSet = {}
        for each subcube c ∈ d.partitionSet do
            ep = (A(c) − F(c))/(A(t) − F(t))
            outside = true if A(c) is outside prediction interval of c
            if ep > T_EEP and outside then
                candidateSet.add(c)
                surprise += c.surprise
            end if
        end for
        if candidateSet.length < d.partitionSet.length then
            explanatorySet.add(candidateSet)
        end if
    end for
    explanatorySet.sortDescend(candidateSet.surprise)
    return explanatorySet.take(3)
end function

function RevisedRecursiveAdtributor(t: total cube)
    explanatorySet = Adtributor(t)
    for each candidateSet in explanatorySet do
        for each subcube c in candidateSet do
            if c ∉ LEAF then
                c.explanatorySet = RecursiveAtributor(c)
            end if
        end for
    end for
    return explanatorySet
end function
```

6.2.4 Evaluation

For evaluation of the fault localization algorithms, eleven different cases of anomalies were used with labels of cubes that should be recognized as root causes. The fault localization algorithms work by a top-down approach, that is they detect anomalies in a total series and try to find the root cause by looking at the dimensions of that series. Even though you could technically apply the algorithm to any time series, it would make little sense to use it on time series not having any dimensions or time series of small magnitude. Thus, the anomalies used as test cases were picked from time series representing either the total revenue for a certain business unit, or the total revenue for a certain partner within a business unit. The eleven anomalies used in the evaluation of the fault localization algorithm could be traced to a total of 71 root causes found in the breakdowns, and the aim would thus be to find as many of these root causes as possible.

The eleven test cases were used to compare the results of the two different recursive Adtributors and the original- and modified HotSpot algorithm. For the original recursive Adtributor, we ran the algorithm on the test cases with two different values of T_{EP}; 80% and 95%. The value of the T_{EEP} threshold was set to 1% for both the original and the revised algorithm. The same threshold for explanatory power was also used in the modified HotSpot algorithm for the pruning. Furthermore, the threshold for potential score was set to 0.99. As we experienced no problems with run time, the maximum number of iterations for HotSpot was set to 400. The precision and recall of the different algorithms were then used to compare the results.

Following is an example of the test cases used during the evaluation. Figure 6.4 shows an obvious anomaly found in the total revenue for a specific business unit. This anomaly can then be traced to a specific partner within that business unit which is shown in Figure 6.5. Finally, the anomaly can be traced to two specific ad units for this partner and this can be seen in Figure 6.6

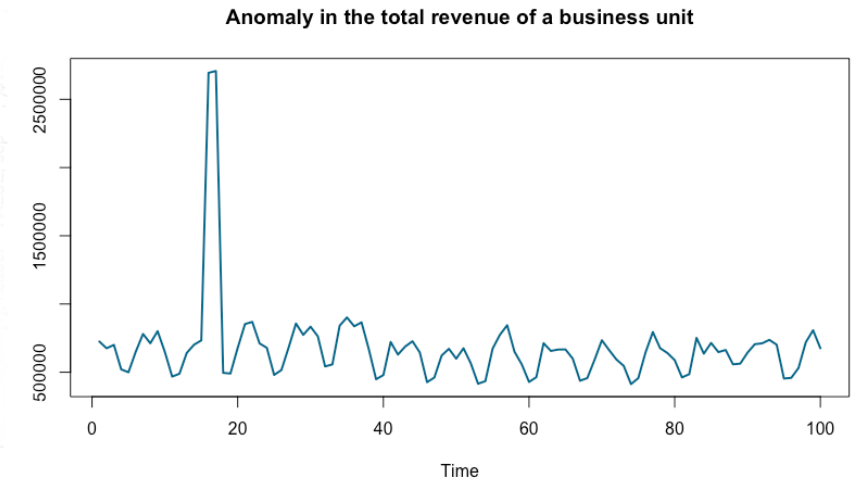

Figure 6.4: Anomaly detected in the total revenue for a specific business unit.

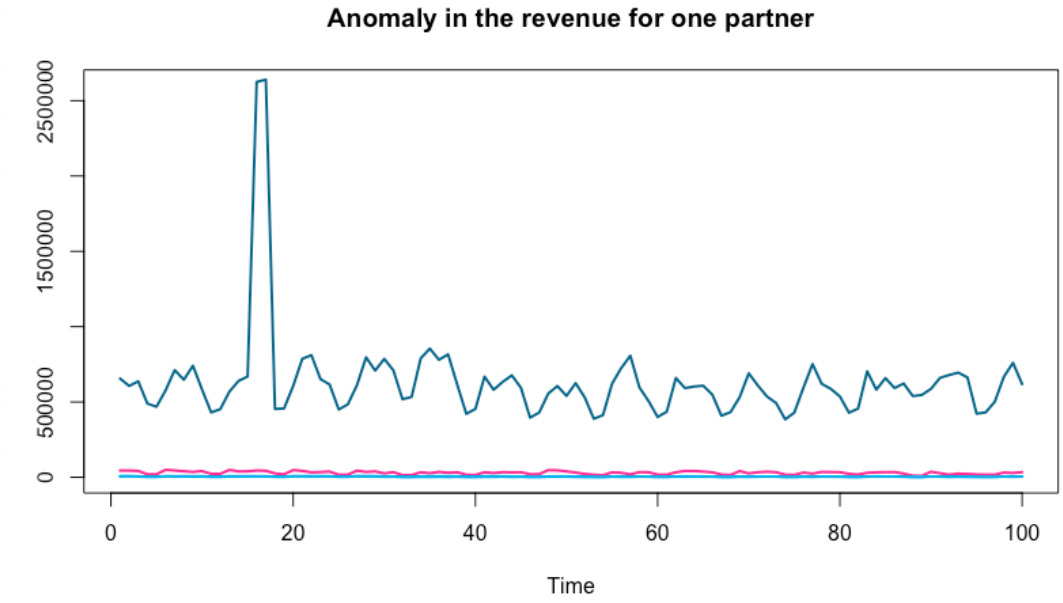

Figure 6.5: The partners within an anomalous business unit where one partner alone contributes to the total anomaly.

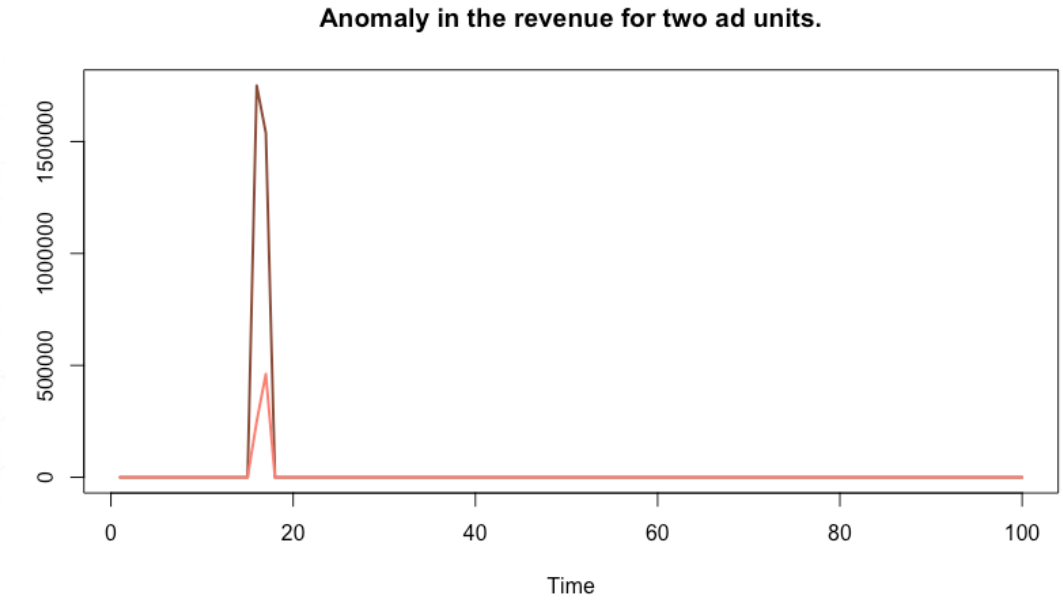

Figure 6.6: The ad units for a specific partner within an anomalous business unit where two ad units contribute to the total anomaly.

6.3 Results

The different algorithms were applied to eleven test cases, the resulting precision and recall for each case are shown in Table 6.4 and the F-scores are shown in Table 6.5. In both tables, RA stands for Recursive Adtributor. The values from Table 6.4 are also plotted in Figure 6.7 where some jitter has been added to the overlapping points for clearer display. What is not displayed in the tables is that the original HotSpot algorithm, in multiple test cases, included subcubes as root causes whose changes were in the opposite direction of the total anomaly. This was the only algorithm that produced such results.

Table 6.4: The resulting precision and recall values from running the different algorithms on the eleven test cases. Here, RA stands for Recursive Adtributor.

Case	Nr of Root Causes	Measure	RA 95%	RA 80%	Revised RA	HotSpot	Modified HotSpot
1	2	Precision	1	1	1	0.33	1
		Recall	1	1	1	1	1
2	11	Precision	1	1	1	1	1
		Recall	0.91	0.45	1	0.55	0.91
3	8	Precision	0.7	0.6	1	0.63	0.5
		Recall	0.86	0.38	1	0.63	0.25
4	2	Precision	1	1	1	0.29	1
		Recall	1	1	1	1	1
5	2	Precision	1	1	1	1	1
		Recall	1	1	1	1	1
6	3	Precision	1	1	1	1	1
		Recall	0.67	0.67	0.67	0.67	0.67
7	4	Precision	0.75	0.5	1	0.33	0.6
		Recall	0.75	0.25	1	0.5	0.75
8	25	Precision	0.76	0.82	1	0.8	1
		Recall	0.76	0.72	0.8	0.16	0.48
9	4	Precision	1	1	1	0.23	0.8
		Recall	0.75	0.25	1	0.75	1
10	2	Precision	1	1	1	0.13	1
		Recall	1	1	1	1	1
11	8	Precision	0.67	0.67	0.89	0.75	0.7
		Recall	1	0.75	1	0.75	0.88
All	71	Precision	0.81	0.81	0.98	0.48	0.84
		Recall	0.85	0.62	0.92	0.51	0.68

Table 6.5: The resulting F-scores from running the different algorithms on the eleven test cases. Here, RA stands for Recursive Adtributor.

Case	Nr of Root Causes	RA 95%	RA 80%	Revised RA	HotSpot	Modified HotSpot
1	2	1	1	1	0.5	1
2	11	0.95	0.62	1	0.71	0.95
3	8	0.77	0.47	1	0.63	0.33
4	2	1	1	1	0.45	1
5	2	1	1	1	1	1
6	3	0.8	0.8	0.8	0.8	0.8
7	4	0.75	0.33	1	0.4	0.67
8	25	0.76	0.77	0.89	0.27	0.65
9	4	0.86	0.4	1	0.35	0.89
10	2	1	1	1	0.23	1
11	8	0.8	0.671	0.94	0.75	0.78
All	71	0.83	0.71	0.95	0.49	0.75

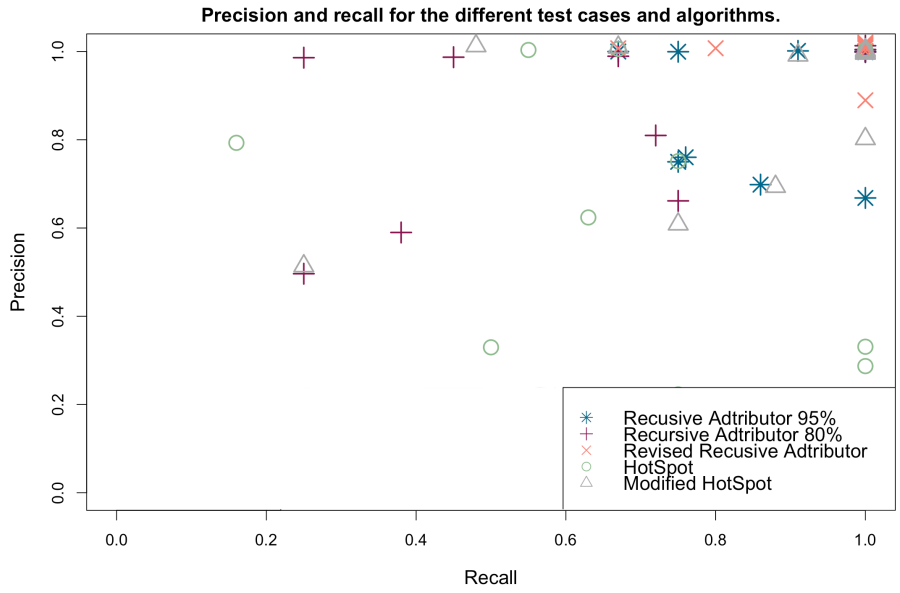

Figure 6.7: Plot of the precision and recall for all the different recursive Adtributor- and HotSpot algorithms when run on the eleven test cases.

In Figure 6.7, values located at the top right corner are considered good, while values closer to the bottom left indicates a worse performance. From the plot, we can see that some of the algorithms have a larger variation in their results. More specifically, both HotSpot algorithms and the recursive Adtributor with an 80% T_{EP} threshold show a larger spread in the values than the other two algorithms.

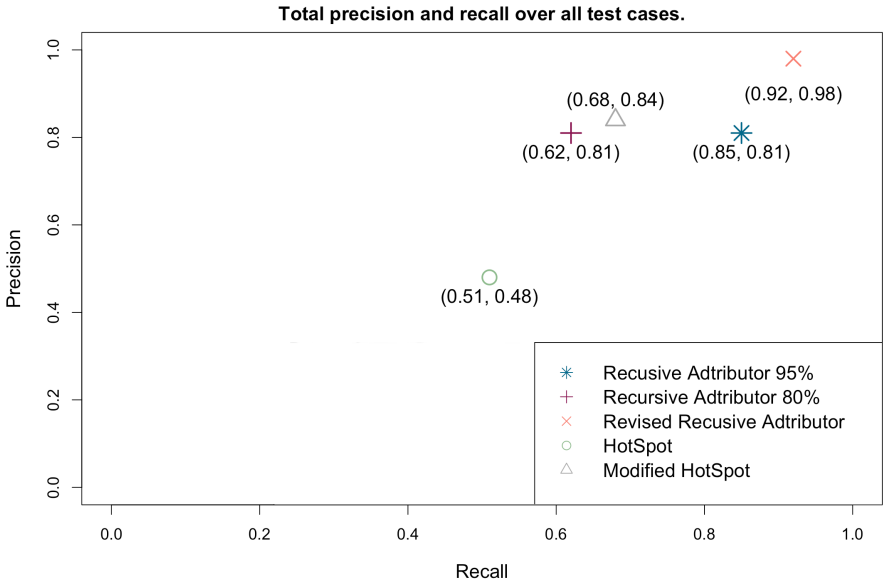

Figure 6.8: Plot of the total precision and recall for all the different recursive Adtributor- and HotSpot algorithms over all eleven test cases.

Figure 6.8 shows the total performance of the algorithms when calculated over all test cases. The values correspond to the ones that can be found in the total row of Table 6.4. As expected, the best performing algorithms in the top right corner are also the algorithms with the smallest spread in Figure 6.7.

6.4 Discussion

Looking at the overall results from the fault localization it is clear that the original HotSpot algorithm is the worst performing one as it has the lowest F-score. Moreover, the algorithm was the only one without perfect performance for case 1, 4 and 10. This can be derived from the fact that all of these were cases where the root causes could be found in LEAF subcubes, and the original HotSpot algorithm does not have an efficient way of calculating potential scores for such subcubes. The original HotSpot was also the only algorithm producing results where the proposed root cause set contained subcubes with a change in the opposite direction of the total anomaly, which should never be the case.

Comparing the original HotSpot algorithm to the modified one, the modified version only had a worse performance in one of the test cases, more specifically test case 3. This case consisted of 8 root causes, all located in LEAF subcubes. The modified version of the HotSpot algorithm uses a simplified measure where the total anomaly is distributed equally among LEAF subcubes to calculate the potential score, which may affect its ability to locate several root causes in LEAF. Comparing the overall performance of the original HotSpot algorithm and the modified version, it is clear that the modified version is significantly better.

When comparing the two original recursive Adtributor algorithms, we can see that the 80% algorithm had a much worse overall recall. This means that it has found a smaller number of actual root causes, or in other words, has a larger number of false negatives. This result is not surprising since the algorithm will terminate sooner than the 95% Adtributor, and therefore fewer subcubes will be evaluated before returning the root cause set. The precision of the two algorithms are quite similar in the different test cases and are in fact equal over all cases. This can be seen in both Table 6.4 and Figure 6.8. The fact that the two algorithms have the same precision, but different recall, tells us that the 95% Adtributor has a larger number of false positives than the 80% Adtributor. That is to say, even though the 80% Adtributor has more false negatives, it also returns fewer incorrect root causes when compared to the 95% algorithm. This result is also unsurprising for the same reason as before, that is that the 80% algorithm will evaluate and thus return fewer subcubes than the 95% version. From these results it is reasonable to assume that raising the threshold to more than 95% will result in a higher number of true positives but also a higher number of false positives. Likewise, lowering the threshold to below 80% will result in a higher number of false negatives but also fewer false positives. In order to choose the correct thresholds for the recursive Adtributor, one would have to test several different thresholds and select the one that results in the desired balance between false positives and false negatives. It was, however, decided that this test would not be performed since the results of the revised recursive Adtributor were so superior to both the recursive Adtributor algorithms.

Looking only at the revised recursive Adtributor, both the recall and the precision over all test cases are very close to 1, and it had an F-score of 0.95. In Table 6.5 it can be seen that the F-score was less than 1 in only three of the cases. These results are not only better than that of all the other algorithms, but they are also very close to being perfect results. The root causes returned by this algorithm all have an explanatory power of at least 1% and are also outside the prediction intervals. Therefore it is reasonable to assume that one reason for false negatives and false positives in the results is the less than perfect forecasting from the first step of the thesis.

From the results, it is clear that the revised recursive Adtributor was the best in comparison to the original Adtributor versions and that the modified HotSpot algorithm performed better than the original one. If we move to compare the results from the revised Adtributor and the modified HotSpot algorithm, the measures show that the Adtributor had the better performance. Both the overall precision and recall are greater for the Adtributor algorithm, and it outperformed the HotSpot in all test cases where the results were not equal. As mentioned previously, the HotSpot algorithm assumes that all root causes for an anomaly are confined within one partition set. Although we do present three different potential sets, the different combinations of partition sets are not considered and thus the accuracy may suffer.

Although the modified HotSpot algorithm performed worse than the revised Adtributor, there are still some advantages of using HotSpot. The Adtributor algorithm adds all subcubes to a candidate set if they meet certain criteria, while the HotSpot

algorithm considers all possible combinations in a partition set as long as the explanatory power of the subcube exceeds 1%. Thus, HotSpot might be able to find less obvious combinations of root cause subcubes. Furthermore, the revised Adtributor is heavily dependent on the prediction interval, as no subcube with a forecast within the interval will be added to the candidate set. An issue with the test cases used is that the anomalies are quite obvious, with very large spikes or drops, which means they will obviously also fall outside the prediction interval. However, there might exist cases where the root cause subcubes actually do have forecasts within the prediction interval. The revised Adtributor will not consider such subcubes, while the HotSpot algorithm will.

Due to the limited amount of test data available it is reasonable to question the superior results of the revised Adtributor algorithm. Since the test cases were found manually through a visual search of available time series, it is unlikely that they can represent the entire data set. Moreover, this limited number of test cases may have resulted in over-fitting the variables which could mean that the algorithm will perform much worse for other sets of data. The visual manner in which the test cases were selected may also mean that the root causes we expect the algorithm to find are in fact not correct. For instance, there could also be more root causes that are more difficult to detect visually but that an analyst would want to be informed of. However, without any labelled data it is difficult to apply the different algorithms to less obvious anomalies as there would be no way of evaluating the results. Another issue is that the anomalies, as previously mentioned, are quite obvious with very large spikes or drops, which may not be representative of all the anomalies that would be deemed relevant by an analyst. The implications of these issues means that these test cases should perhaps be considered as a sanity check rather than an evaluation.

7
Conclusion

The purpose of this thesis was to develop an automated process of identifying anomalies and narrowing down the possible root causes. In order to achieve this, we have evaluated existing time series models for forecasting and different methods for anomaly detection. Using the best performing methods we have then implemented a fault localization algorithm for finding root causes in a dataset.

For deciding which time series model should be used for forecasting, we compared the results of different variations of ARIMA, SARIMA and ETS models. The results from this evaluation showed that the ETS model with MAE as the optimization criterion had the best performance the most number of times and was therefore selected and used during the remaining parts of the thesis. The quality of the forecasts produced by this model was varying. We believe that one reason for this is the fact that one model may not properly be able to capture all the different sets of characteristics present in the time series. Another reason for the varying quality may also be that some of the series are naturally hard to predict, as they have noisy weekly patterns and a non-constant variance. Lastly, the presence of consecutive observations of value 0 was not always handled properly, and may thus need to be handled separately.

During the anomaly detection part of this thesis, anomalies were defined as significant deviations from forecasted values and different notions of what constitutes a significant deviation were evaluated. We compared using a static threshold, using different prediction intervals generated by the forecasting method, and using two different threshold learning algorithms. The threshold learning algorithm with a revised updating rule was the technique that had the best performance over all time series. The quality of the results produced by this method were, however, not perfect as it missed one third of all anomalies in the series. We believe the main reasons for this inaccuracy are the reliability and limited amount of labelled data as well as the difficulty of predicting values in noisy time series.

The last part of this thesis consisted of implementing an algorithm for fault localization. To our knowledge, two such algorithms exist that can be used in our domain; HotSpot [28] and Adtributor [3]. Due to limitations of both algorithms, we have in this thesis implemented modified versions of the algorithms. The modifications of the Adtributor algorithm included making it recursive and adjusting the criteria used for putting subcubes in a candidate set. Similarly, the HotSpot algorithm was modified by adjusting calculations for the potential of a set to be the root cause.

The results from the evaluation of the different algorithms clearly show that the modified HotSpot algorithm outperformed the original one. However, the revised recursive Adtributor also outperformed the modified HotSpot with an almost perfect result. As the main challenge of this thesis has been the limited amount of labelled data, these high scores may be the result of overfitting and the performance of the different methods might be skewed. Another challenge has been the differing characteristics of the time series, since they do not only affect the quality of the forecasting, but also the anomaly detection and fault localization.

Overall, we believe that a combination of the three components in this thesis into an automated process has great potential for discovering anomalies and narrowing down the root causes in a real application.

7.1 Future Work

Although we are pleased with the results of this thesis, there are several improvements that could be made. First of all, when it comes to the forecasting step of this thesis, one model was used for all the different time series. In order to obtain more accurate forecasts, a future improvement could be to implement an automated process that selects the appropriate model based on the identified characteristics of each series. This way each time series would be modelled separately and thus handled according to its unique set of characteristics. Also, one might want to treat time series with consecutive observations of value 0 separately.

For the threshold learning algorithm determined to be the best performing, labels need to be provided by an operator. In this thesis, we have used one label for each point in time. However, it is reasonable to assume that this will not be the case when used in a real application. Thus, one might want to further evaluate to which extent the amount of provided labels affects the performance of the anomaly detection. Another question that needs to be answered is how the labels should be collected. Though false positive labels can somewhat easily be retrieved by an operator labelling the notifications received by the anomaly detection, false negative labels require more manual effort to collect. Since labels are crucial for the algorithm to work, a convenient way of collecting these are also an important factor in the success of the anomaly detection.

In order to properly evaluate the performance of our fault localization algorithms, they would need to be applied to much larger amounts of data. Since the test set used in this thesis was small, and the cases were obvious, the results produced by the algorithms may not be representative of the entire dataset, or the algorithms may be overfitted. Therefore more test data is necessary in order to adequately evaluate the algorithm.

After applying the forecasting, anomaly detection and fault localization, a next step would be to generate an intelligent recommendation to the operator about

possible corrective actions. However, in order to provide such a recommendation, a process is needed where more background knowledge of the system is incorporated. For instance, there are several other metrics that would be interesting to analyze, such as the number of impressions or clicks, since they are often closely related to the revenue. By applying fault localization over all such metrics and comparing the results, more information could be obtained. It may also be appropriate to incorporate more information about the various dimensions for each business unit and the relationships that exist between them.

Bibliography

[1] J.C. Arezzo, S. Seto, and H. H. Schaumburg. "Chapter 16 - Sensory-motor assessment in clinical research trials". In: *Peripheral Nerve Disorders*. Ed. by Gérard Said and Christian Krarup. Vol. 115. Elsevier, 2013, pp. 265 –278. DOI: https://doi.org/10.1016/B978-0-444-52902-2.00016-3.

[2] C. Bergmeir and J. M. Benítez. "On the use of cross-validation for time series predictor evaluation". In: *Information Sciences* 191 (2012), pp. 192 –213. DOI: https://doi.org/10.1016/j.ins.2011.12.028.

[3] R. Bhagwan, R. Kumar, R. Ramjee, G. Varghese, S. Mohapatra, H. Manoharan, and P. Shah. "Adtributor: Revenue Debugging in Advertising Systems". In: *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. Seattle, WA: USENIX Association, 2014, pp. 43–55.

[4] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015.

[5] I. Choi. *Almost all about unit roots: Foundations, developments, and applications*. Cambridge University Press, Jan. 2015.

[6] M.J. De Smith. *Statistical Analysis Handbook: A Comprehensive Handbook of Statistical Concepts, Techniques and Software Tools*. Winchelsea, UK: The Winchelsea Press, 2013.

[7] R. F. Engle and C. W. J. Granger. "Co-Integration and Error Correction: Representation, Estimation, and Testing". In: *Econometrica* 55.2 (1987), pp. 251–276.

[8] J. Fan and Q. Yao. "Characteristics of Time Series". In: *Nonlinear Time Series: Nonparametric and Parametric Methods*. New York, NY: Springer New York, 2003, pp. 29–88. DOI: 10.1007/978-0-387-69395-8_2.

[9] S. Holmes. *Lecture 11: Maximum Likelihood Estimation*. 2001. URL: http://economics.ozier.com/econ626/lec/econ626lecture11_slides.pdf (visited on 05/30/2018).

[10] R. J. Hyndman. *auto.arima: Fit Best ARIMA Model To Univariate Time Series*. URL: https://www.rdocumentation.org/packages/forecast/versions/8.3/topics/auto.arima (visited on 03/12/2018).

[11] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. Otexts: Melbourne, Australia, 2013.

[12] R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. 2nd edition. Otexts: Melbourne, Australia, 2018.

[13] R. J. Hyndman, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen. *forecast: Forecasting func-*

tions for time series and linear models. R package version 8.3. 2018. URL: http://pkg.robjhyndman.com/forecast.

[14] R. J. Hyndman and Y. Khandakar. "Automatic Time Series Forecasting: the forecast Package for R". In: *Journal of Statistical Software* 27.3 (2008). DOI: 10.18637/jss.v000.i00.

[15] R. J. Hyndman and A. B. Koehler. "Another look at measures of forecast accuracy". In: *International Journal of Forecasting* 22.4 (2006), pp. 679 –688. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001.

[16] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose. "A state space framework for automatic forecasting using exponential smoothing methods". In: *International Journal of Forecasting* 18.3 (2002), pp. 439 –454. DOI: https://doi.org/10.1016/S0169-2070(01)00110-8.

[17] S. Johansen and M. Ø. Nielsen. "The Role of Initital Values in Conditional Sum-of-Squares Estimation of Nonstationary Fractional Time Series Models". In: *Econometric Theory* 32.5 (Oct. 2016), pp. 1095–1139.

[18] S. Kim and H. Kim. "A new metric of absolute percentage error for intermittent demand forecasts". In: *International Journal of Forecasting* 32.3 (2016), pp. 669 –679. DOI: https://doi.org/10.1016/j.ijforecast.2015.12.003.

[19] T.D. Koepsell and F. A. Connell. "Measures of gain in certainty from a diagnostic test". In: *American Journal of Epidemiology* 121.5 (1985), pp. 744–753. DOI: 10.1093/aje/121.5.744.

[20] D. Kwiatkowski, P.CB. Phillips, P. Schmidt, and Y. Shin. "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?" In: *Journal of econometrics* 54.1-3 (1992), pp. 159–178.

[21] D. Liu, Y. Zhao, K. Sui, S. Cheng, D. Pei, C. Quan, J. Luo, X. Jing, and M. Feng. "Learning thresholds for PV change detection from operators' labels". In: *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. 2015, pp. 1–9. DOI: 10.1109/PCCC.2015.7410322.

[22] J. K. Ord, A. B. Koehler, and R. D. Snyder. "Estimation and Prediction for a Class of Dynamic Nonlinear Statistical Models". In: *Journal of the American Statistical Association* 92.440 (1997), pp. 1621–1629.

[23] D.R. Osborn, A.PL. Chui, J.P. Smith, and C.R. Birchenhall. "Seasonality and the order of integration for consumption". In: *Oxford Bulletin of Economics and Statistics* 50.4 (1988), pp. 361–377.

[24] Boston University School of Public Health. *Positive and Negative Predictive Value*. 2016. URL: http://sphweb.bumc.bu.edu/otlt/MPH-Modules/EP/EP713_Screening/EP713_Screening5.html (visited on 05/30/2018).

[25] A. Reifman and K. Keyton. "Winsorize". In: *Encyclopedia of research design* (2010), pp. 1636–1638. DOI: http://dx.doi.org/10.4135/9781412961288.n502.

[26] P.M.M. Rodrigues and D.R. Osborn. "Performance of seasonal unit root tests for monthly data". In: *Journal of Applied Statistics* 26.8 (1999), pp. 985–1004. DOI: 10.1080/02664769921981.

[27] Y. Sasaki. "The truth of the F-measure". In: *Teach Tutor mater* 1.5 (2007).

[28] Y. Sun, Y. Zhao, Y. Su, D. Liu, X. Nie, Y. Meng, S. Cheng, D. Pei, S. Zhang, X. Qu, and X. Guo. "HotSpot: Anomaly Localization for Additive KPIs with Multi-Dimensional Attributes". In: *IEEE Access* (2018). DOI: 10.1109/ACCESS.2018.2804764.

[29] L.J. Tashman. "Out-of-sample tests of forecasting accuracy: an analysis and review". In: *International Journal of Forecasting* 16.4 (2000), pp. 437 –450. DOI: https://doi.org/10.1016/S0169-2070(00)00065-0.

[30] The Pennsylvania State University. *Applied Time Series Analysis*. 2018. URL: https://onlinecourses.science.psu.edu/stat510/ (visited on 05/30/2018).

[31] The Pennsylvania State University. *Epidemiological Research Methods*. 2018. URL: https://onlinecourses.science.psu.edu/stat501 (visited on 05/30/2018).

[32] The Pennsylvania State University. *Regression Methods*. 2018. URL: https://onlinecourses.science.psu.edu/stat501 (visited on 05/30/2018).

[33] R. Williams. *Heteroskedasticity*. 2015. URL: https://www3.nd.edu/~rwilliam/stats2/l25.pdf (visited on 05/30/2018).

[34] A. H. Yaacob, I. K. T. Tan, S. F. Chien, and H. K. Tan. "ARIMA Based Network Anomaly Detection". In: *2010 Second International Conference on Communication Software and Networks*. 2010, pp. 205–209. DOI: 10.1109/ICCSN.2010.55.