YAKAMOZ AND SUNSHINE:
TWO NOVEL TIME SERIES ANOMALY DETECTION METHODS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

METİN BOĞAZLIYAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
STATISTICS

JUNE 2021

Approval of the thesis:

YAKAMOZ AND SUNSHINE:
TWO NOVEL TIME SERIES ANOMALY DETECTION METHODS

submitted by **METİN BOĞAZLIYAN** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Statistics Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Özlem İlk Dağ
Head of Department, **Statistics** _____

Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil
Supervisor, **Statistics, METU** _____

Examining Committee Members:

Prof. Dr. Seher Nur Sülkü
Econometrics, Ankara Hacı Bayram Veli University _____

Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil
Statistics, METU _____

Prof. Dr. Ömür Uğur
Institute of Applied Mathematics, METU _____

Prof. Dr. Mehmet Yılmaz
Statistics, Ankara University _____

Assist. Prof. Dr. Fulya Gökalp Yavuz
Statistics, METU _____

Date: 22.06.2021

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Metin Boğazlıyan

Signature :

ABSTRACT

YAKAMOZ AND SUNSHINE:
TWO NOVEL TIME SERIES ANOMALY DETECTION METHODS

Boğazlıyan, Metin
Ph.D., Department of Statistics
Supervisor: Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil

June 2021, 285 pages

In modern society, availability and reliability of data have become crucial. Hence, one important task for a variety of fields is to detect abnormal instances in data. One of the objectives of identification of anomalies is to detect outlying data points, leading to new discoveries. In this case, anomalies themselves are of primary interest. The other objective is to detect anomalies in the preprocessing step in statistical analysis that may otherwise lead to model misspecification, biased parameter estimation and incorrect results. Obtaining forecasts using time series is a subject that has never lost its popularity, and it is becoming ever more important with the development of technology. The presence of even a few abnormal observations in the series, on the other hand, has a negative effect on these forecasts. In view of the problems, it is understood that there is a need for widely applicable and effective anomaly detection methods.

Our main goal is to provide model-free, unsupervised, efficient and accurate detection of anomalous observations in time series data. For this reason, two novel time series anomaly detection methods have been proposed. These methods do not require anomalous samples in their training datasets. This skill has considerable practical value since the collection of labelled data can be difficult. In addition, the location of the anomaly does not affect the

performance of the methods. Success of the proposed methods in finding anomaly/anomalies is investigated using real and simulated time series datasets. From the performance comparison results, it is seen that the proposed methods are very effective in finding anomaly/anomalies in these time series regardless of the variation in the data.

Keywords: Anomaly Detection, Machine Learning

ÖZ

YAKAMOZ VE SUNSHINE:
İKİ YENİ ZAMAN SERİSİ ANOMALİ TESPİT YÖNTEMİ

Boğazlıyan, Metin
Doktora, İstatistik Bölümü
Tez Yöneticisi: Doç. Dr. Ceylan Talu Yozgatlıgil

Haziran 2021, 285 pages

Modern toplumda, verilerin kullanılabilirliği ve güvenilirliği çok önemli hale geldi. Bu nedenle, çeşitli alanlar için önemli bir görev verilerdeki anormal durumları tespit etmektir. Anormalliklerin belirlenmesinin amaçlarından biri yeni keşiflere yol açacak şekilde aykırı veri noktalarını tespit etmektir. Bu durumda, aykırılıkların kendileri birincil ilgi konusudur. Diğer amaç, istatistiksel analizde ön işleme adımında aykırılıkları tespit etmektir, aksi durumda aykırılıklar model yanlış belirlemeye, yanlı parametre tahminine ve yanlış sonuçlara yol açabilir. Zaman serilerini kullanarak tahminler elde etmek popülerliğini hiç kaybetmeyen bir konudur ve teknolojinin gelişmesiyle daha da önem kazanmaktadır. Zaman serisinde birkaç aykırı gözlemin bile olması bu tahminler üzerinde olumsuz etkiye sahiptir. Sorunlar göz önüne alındığında, yaygın olarak uygulanabilir ve etkili aykırılık tespit yöntemlerine ihtiyaç olduğu anlaşılmaktadır.

Bu çalışmada ana hedefimiz, zaman serisi verilerindeki aykırı gözlemlerin modelsiz, denetimsiz, verimli ve doğru tespitini sağlamaktır. Bu nedenle, iki yeni zaman serisi aykırılık tespit yöntemi önerilmiştir. Bu yöntemler, eğitim veri setlerinde aykırı örnekler gerektirmez. Etiketli verilerin toplanması zor olabileceğinden, bu beceri önemli ölçüde

pratik değere sahiptir. Ayrıca, aykırılığın yeri yöntemlerin performansını etkilemez. Aykırılık/aykırılıkları bulmada önerilen yöntemlerin başarısı, gerçek ve simüle edilmiş zaman serisi veri setleri kullanılarak araştırılmıştır. Performans karşılaştırma sonuçlarından, önerilen yöntemlerin verilerdeki varyasyona bakılmaksızın zaman serilerindeki aykırılığı/aykırılıkları bulmada oldukça etkili olduğu görülmektedir.

Anahtar Kelimeler: Anomali Tespiti, Makine Öğrenimi

To the beautiful people of my beautiful country

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Assoc. Prof. Dr. Ceylan Talu Yozgatlıgil, for her constant support and guidance.

I would like to thank the members of the examining committee, Prof. Dr. Seher Nur Sülkü, Prof. Dr. Ömür Uğur, Prof. Dr. Mehmet Yılmaz and Assist. Prof. Dr. Fulya Gökalp Yavuz, for their valuable suggestions.

I would like to express my endless gratitude towards Middle East Technical University for the opportunities it has provided during my undergraduate, master's and doctoral studies.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AR	Autoregressive
ARCH	Autoregressive Conditional Heteroscedasticity
ARIMA	Autoregressive Integrated Moving Average
ARIMAX	Autoregressive Integrated Moving Average with Exogenous Variables
ARMA	Autoregressive Moving Average
CBLOF	Cluster-Based Local Outlier Factor
CNN	Convolutional Neural Network
COF	Connectivity-based Outlier Factor
DBSCAN	Density Based Spatial Clustering of Applications with Noise
ESD	Extreme Studentized Deviate
EWMA	Exponentially Weighted Moving Average
FindCBLOF	Find Cluster-Based Local Outlier Factor
FN	False Negative

FP	False Positive
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GMM	Gaussian Mixture Models
iForest	Isolation Forest
INFLO	Influenced Outlierness
iTree	Isolation Tree
LOF	Local Outlier Factor
MAD	Median Absolute Deviation
MBM	Moving Blocks Matrix
MBMs	Moving Blocks Matrices
MCC	Matthew's Correlation Coefficient
MCD	Minimum Covariance Determinant
MD	Mahalanobis Distance
MVE	Minimum Volume Ellipsoid
OCSVM	One Class Support Vector Machine

OLS	Ordinary Least Squares
STL	Seasonal-Trend Decomposition Procedure Based on Loess
TN	True Negative
TP	True Positive

CHAPTER 1

INTRODUCTION AND MOTIVATION

Oxford English Dictionary defines an outlier as:

"*An observation whose value lies outside the set of values considered likely according to some hypothesis (usually one based on other observations); an isolated point.*"

There are many similar but not identical definitions of an outlier in the literature. Hawkins (1980) gives the intuitive definition of an outlier as follows:

"*An observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*"

Similarly, Barnett and Lewis (1994) define an outlier as:

"*An observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data.*"

In modern society, availability and reliability of data have become crucial. Hence, one important task for a variety of fields is to detect outliers. Depending on the specific applications, outliers are also referred to as anomalies, peculiarities, aberrations, contaminants, discordant observations, surprises, exceptions (Chandola et al., 2009), abnormalities, deviants (Aggarwal, 2013).

The statistics community has studied detection of anomalies in data as early as the 19th century (Edgeworthl, 1887). There are basically two main purposes of anomaly detection.

One is to detect outlying data points, leading to new discoveries. In this case, anomalies themselves are of primary interest. The other is to detect anomalies in the preprocessing step in statistical analysis that may otherwise lead to model misspecification, biased parameter estimation and incorrect results (Ben-Gal, 2005; Hadi et al., 2009).

Today, anomaly detection is applied to various fields such as fraud detection, medical condition monitoring, industry damage detection, intrusion detection, sensor network surveillance (Han et al., 2011), detection of unexpected entries in databases, pharmaceutical research (Hodge & Austin, 2004). An example of anomaly detection related to daily life is credit card fraud detection. Anomaly detection in credit card usage concerns the detection of abnormality which may result from unauthorized use of the credit card. Detection of such abnormality will prevent further loss for credit card owners and ensure financial security (Bolton & Hand, 2002). To give another example, anomaly detection in flight data aims to track the flight risks such as component faults of an aircraft. Detection of such anomaly will enhance flight safety (Igenewari et al., 2019). A more detailed list of applications that use anomaly detection is provided below (Aggarwal, 2013; Hodge & Austin, 2004).

- Detection of unexpected entries in databases – detecting valid but unexpected entries, errors.
- Fault detection – identifying faults in systems such as pipelines, motors, generators or aviation tools.
- Activity monitoring – detecting suspicious transactions in stock markets.
- Intrusion detection – recognizing unauthorized access to computer networks.
- Loan application processing – identifying fraudulent applications or clients that are potentially problematic.
- Structural defect detection – monitoring production lines to identify defective production runs.
- Fraud detection – identifying fraudulent state benefit applications, fraudulent credit card applications, fraudulent use of mobile phones or credit cards.
- Network performance – performance monitoring in computer networks.

- Pharmaceutical research – recognizing novel structures of molecules.
- Medical diagnosis – identifying unusual patterns in medical data.

Anomaly detection has gained growing attention as an active research subject in machine learning and data mining. At first glance, anomaly detection appears to be one of classification, i.e., the separation of observations into two classes: anomalous and non-anomalous. To solve the problem of detecting anomalies, it is tempting to use well-known machine learning and classification algorithms, e.g., support vector machines, decision trees and neural networks. However, these algorithms will rarely be successful as the data to be analyzed is severely imbalanced. This means that anomalies are much rarer than normal observations. Therefore, the algorithms cannot find enough cases to learn anomaly situation, and results obtained by these algorithms will often lead to too many false negatives, i.e., not detecting anomalies (Mehrotra et al., 2017). Besides, these algorithms need labelled data, which is difficult to obtain in real life.

Data points which have been recorded in an ordered manner and which are correlated in time form a time series. Recent advances in technology enable large amounts of data to be collected over time. In the past few years, mining this data has become a significant activity for researchers and practitioners. Nowadays, anomaly detection is one of the main tasks of time series data mining (Bl'azquez-Garc'ia et al., 2020). Obtaining forecasts using time series is a subject that has never lost its popularity, and it is becoming ever more important with the development of technology (Lim & Zohren, 2021; Mahalakshmi et al., 2016). The presence of even a few abnormal observations in the series, on the other hand, has a negative effect on these forecasts (Vishwakarma et al., 2020). As a result, identifying possible anomalies in time series and taking appropriate action against them has a significant impact on the accuracy of forecasts to be obtained.

Although anomaly detection in time series data has been an area of interest for many researchers and practitioners, most of the available methods of time series anomaly detection are based on fitting a model either implicitly or explicitly (Bl'azquez-Garc'ia et al., 2020). However, model parameters or distribution of data have a significant effect on

the results of anomaly detection (Mehrotra et al., 2017). The accurate assumption of distribution is the main issue of these methods because the distribution can only be defined with the existence of anomalies in the data which may prevent to capture the true data generating mechanism. A single model may not be adequate to capture the true underlying statistical features of the data (Eskin, 2000; Eskin et al., 2002).

Several factors make the detection of anomalies challenging. Firstly, the precise definition of an anomaly is different for various application areas. For instance, a small deviation from normal may be an anomaly in the medical field, while a similar deviation in the stock market may be considered normal. Moreover, the availability of labelled data for training of anomaly detection methods is often a major problem (Chandola et al., 2009). Furthermore, when there are anomalies at the beginning or at the end of the time series, it is seen that the current anomaly detection methods do not give successful results, and they give excessive amount of false positive rates.

In view of the above problems, it is understood that there is a need for widely applicable and effective anomaly detection methods.

Our main goal is to provide model-free, unsupervised, efficient and accurate detection of anomalous observations in time series data. For this reason, two novel time series anomaly detection methods have been proposed. These methods will not require anomalous samples in their training datasets. This skill has considerable practical value since the collection of labelled data can be difficult. In addition, the location of the anomaly does not affect the performance of the methods. The concept we call the Moving Blocks Matrix (MBM) is the core idea of the proposed methods. Since we use the moving blocks and their distances, the location of the anomaly becomes unimportant.

The rest of the thesis is organized as follows:

- Chapter 2 reviews the concepts and previous studies related to anomaly detection.
- In Chapter 3, we propose two novel time series anomaly detection methods and illustrate the operation of the proposed methods and deciding on the anomaly/anomalies.
- In Chapter 4, the proposed two novel time series anomaly detection methods are compared to the best-known time series anomaly detection methods in the literature using real and simulated time series.
- Chapter 5 concludes the thesis and discusses the potential extensions in the future.

CHAPTER 2

BACKGROUND

In this chapter, the concepts and previous studies related to anomaly detection are presented. Types of anomalies are addressed in Section 2.1. Anomaly detection methods are covered in Section 2.2. Detailed information about the robust Mahalanobis distance is given in Section 2.3. The known best time series anomaly detection methods in the literature are described in Section 2.4. Performance measures to evaluate the effectiveness of anomaly detection methods are mentioned in Section 2.5.

2.1 Types of Anomalies

It is possible to classify anomalies into three groups (Han et al., 2011):

Point Anomalies – An observation which is abnormal compared to the rest of the data is called a point anomaly. See Figure 2.1. Sometimes, point anomalies are referred to as global anomalies. Most anomaly detection methods aim to find point anomalies.

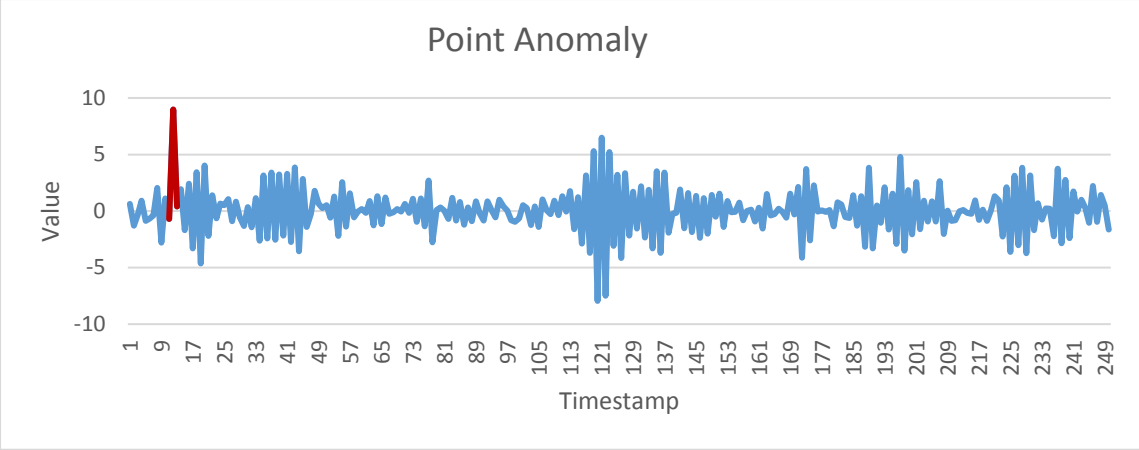

Figure 2.1: An example of a point anomaly

Contextual Anomalies – If a data instance is anomalous in a particular context but not otherwise, then it is called a contextual anomaly. See Figure 2.2. As they are conditional on the chosen context, contextual anomalies are also known as conditional anomalies. To give an example, increasing sales of ice cream in summer are accepted as common behavior, but it is not expected in winter. In this particular context, increased sales of ice cream in winter are accepted as contextual anomalies. Contextual anomalies are a generalization of local outliers. An observation in a dataset is a local anomaly if its density deviates significantly from the local region in which it occurs.

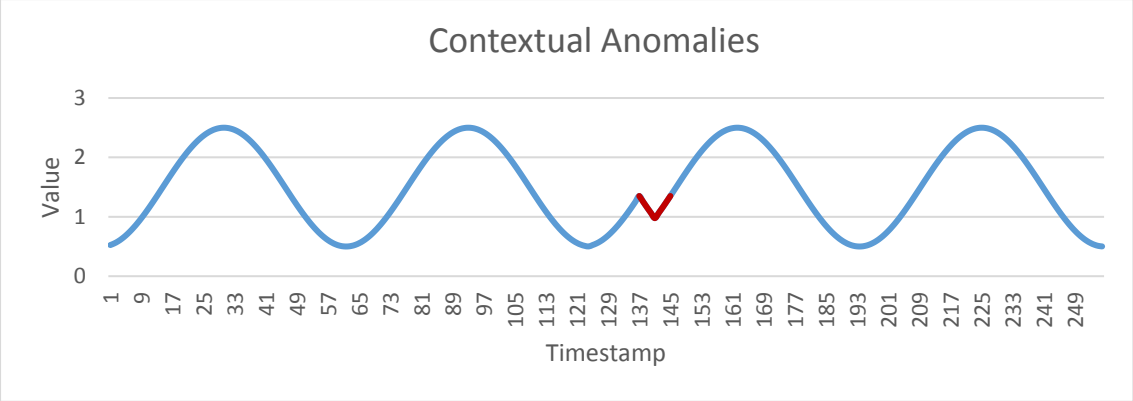

Figure 2.2: An example of contextual anomalies

Collective Anomalies – A subset of observations constitutes a collective anomaly if the observations as a whole deviate substantially from the entire data. Crucially, the individual observations may not be abnormal. See Figure 2.3.

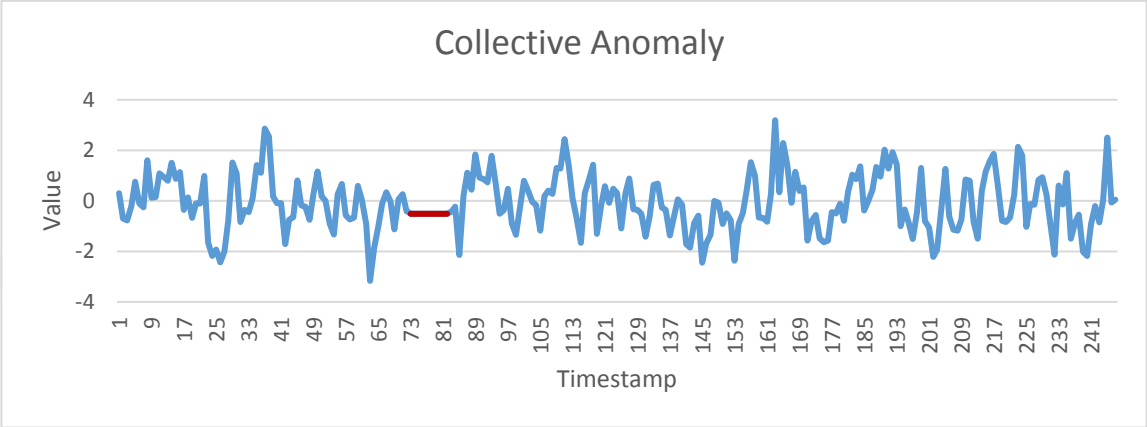

Figure 2.3: An example of a collective anomaly

2.2 Anomaly Detection Methods

Methods for anomaly detection can be classified under three headings: Statistical methods, proximity-based methods and clustering-based methods. In addition to these categories, there are isolation-based algorithms like Isolation Forest. For each of these methods, the nature of the data may be supervised, unsupervised or semi-supervised (Han et al., 2011; Mehrotra et al., 2017).

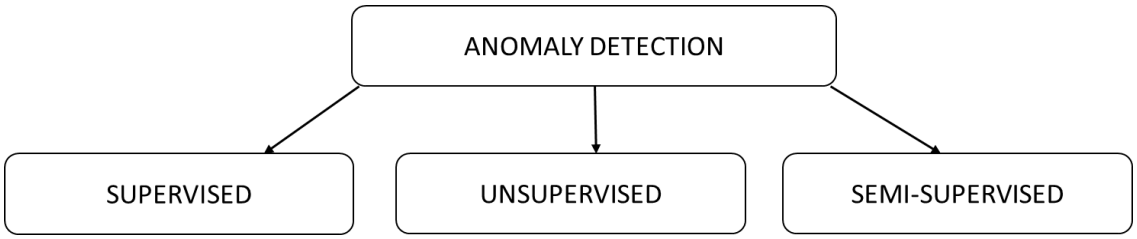

Figure 2.4: Anomaly detection approaches according to data labels

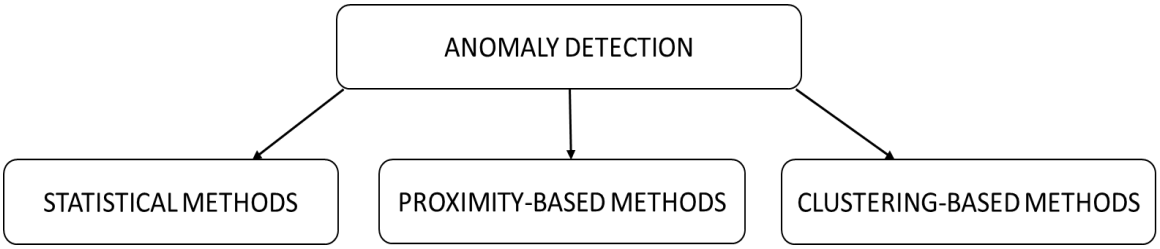

Figure 2.5: Anomaly detection approaches

2.2.1 Supervised, Semi-Supervised and Unsupervised Methods

The labels associated with a data instance indicate whether this instance is normal or anomalous. Labelling is often performed manually by a human specialist and so a significant effort is needed to achieve a labelled training dataset. Collecting labelled data that is reliable as well as descriptive of all types of anomalous behaviors is often prohibitively costly. In addition, new types of anomalies can occur for which no training data is labelled. Based on the availability of label, detection of anomalies can be done in one of the following three modes (Chandola et al., 2009), as shown in Figure 2.4.

Supervised methods – These methods require a training dataset that has labelled instances for anomaly as well as normal classes. Anomaly detection can then be treated as a classification problem. In some applications, only normal observations can be labelled and any other observations that do not match the model of normal observations are reported as anomalies. Other methods model the anomalies and treat observations that do not match the anomaly pattern as normal (Han et al., 2011).

In supervised anomaly detection, there are two significant issues that occur. Firstly, normal objects versus anomalies are imbalanced. That is, the abnormal instances are far less than the normal cases in the training data. In this case, the distribution of anomalous observations may not be sufficiently represented. Secondly, it is generally difficult to obtain correct and representative labels, particularly for the anomaly class (Chandola et al., 2009).

Due to the assumption that the anomalies are known and correctly labelled, these methods are not applicable in practice (Goldstein & Uchida, 2016).

Semi-supervised methods – These methods assume that a small set of the normal and/or abnormal observations are labelled in training data, but most of the data are unlabelled. When some labelled normal observations are available, they can be used together with unlabelled observations that are close by to train a model for normal observations. Then, those observations that do not fit the model of normal observations are classified as anomalies.

If only a few labelled anomalies are available, semi-supervised anomaly detection is trickier. It is unlikely that a small number of labelled anomalies can represent all possible anomalies. Building a model for anomalies based on just a few labelled anomalies is therefore unlikely to be efficient (Han et al., 2011).

Unsupervised methods – These methods do not require a training dataset. Thus, they are most widely applicable. Implicitly, the methods in this category assume that normal observations are far more common than abnormal observations in the dataset. This means that normal observations are somewhat grouped together. Normal observations do not have to fall into a high similarity group. Instead, with each group having different characteristics, several groups can be formed by normal observations. However, an anomaly is supposed to occur far away in feature space from any of these groups. If this assumption is not valid, such methods may not detect many actual anomalies and may detect many normal observations as anomalies (Chandola et al., 2009; Han et al., 2011).

2.2.2 Statistical Methods, Proximity-Based Methods and Clustering-Based Methods

Unlike the last subsection in which anomaly detection methods are categorized according to the availability of data labels, this section classifies the methods in three main categories: statistical methods, proximity-based methods and clustering-based methods, as shown in Figure 2.5.

2.2.2.1 Statistical Methods

Statistical methods for anomaly detection assume that a stochastic process (a generative model) produces the normal observations in a dataset. As a consequence, in regions of high probability for the stochastic model, normal observations occur and observations in the regions of low probability are anomalies. The general concept behind statistical methods for anomaly detection is to learn a model that fits the given dataset and identify observations in low probability regions of the model as anomalies. It is possible to classify statistical methods for anomaly detection into two main categories: Parametric and nonparametric methods.

While a parametric method assumes that a parametric distribution produces the normal observations, an a priori statistical model is not assumed by a nonparametric method. A nonparametric approach instead attempts to determine the model from the input data. Note that most nonparametric approaches do not assume that the model is entirely free of parameters. Nonparametric methods often take the view that nature and number of the parameters are flexible and not predetermined (Han et al., 2011).

Parametric methods can also be categorized as regression model-based, Gaussian model-based, parametric distribution mixture based depending on the type of distribution accepted. Gaussian model-based methods assume that a Gaussian distribution generates the data and the model parameters are estimated using maximum likelihood estimation. Parametric distribution mixture based models use a combination of parametric statistical distributions to model the data. Regression model-based methods, as the name implies, are based on a regression model (Chandola et al., 2009).

Most of the available methods of time series anomaly detection are based on fitting a model either implicitly or explicitly. In a univariate time series, an observation at time t can be defined as an anomaly, if the distance to its expected value is greater than a predefined threshold c:

$$|y_t - \hat{y}_t| > \mathrm{c} \tag{2.1}$$

where y_t is the observation at time *t* and $\hat{y}_t$ is its expected value. In the literature, anomaly detection methods based on the strategy described in Equation (2.1) are also known as model-based methods. Although the expected value $\hat{y}_t$ and the threshold *c* are determined differently by each method, they are all based on fitting a model either explicitly or implicitly. $\hat{y}_t$ can be obtained in the estimation or prediction stage of the method. The method is within the estimation model-based techniques, if $\hat{y}_t$ is obtained in the estimation stage of the method. In contrast, if $\hat{y}_t$ is obtained in the prediction stage of the method, then the method is within the prediction model-based techniques. The difference between using estimation or prediction methods is that prediction methods can determine whether or not a new observation is an anomaly as soon as it arrives. When estimation methods are used, this can only be done by adding the new observation to the dataset and fitting the model again (Bl'azquez-Garc'ia et al., 2020).

Basic statistics like median, median absolute deviation or mean are used in some estimation models. Basu and Meckesheimer (2007) propose time series anomaly detection method that uses the median of a neighborhood of observations to obtain the expected value $\hat{y}_t$. Similarly, to obtain the expected value $\hat{y}_t$, median absolute deviation is used (Mehrang et. al, 2015). Additionally, for time series anomaly detection, Dani et al. (2015) use segmentation and local means.

Some estimation based methods aim to identify anomalies in time series using a particular fitted model. For instance, Carter and Streilein (2012) suggest a probabilistic approach to an exponentially weighted moving average (EWMA) for the detection of anomalies in time series data. This approach learns a parametric statistical model that adapts to the changing distribution of the data. Moreover, Reddy et al. (2017) present an algorithm to detect anomalies using Gaussian mixture models (GMM). These methods use Equation (2.1) directly, once a model is fitted, to determine whether or not an observation is an anomaly.

Some time series anomaly detection methods first use time series decomposition techniques to determine the seasonal and trend components of the time series to be analyzed. For instance, the techniques proposed by Hochenbaum et al. (2017) first use a seasonal decomposition technique to filter the trend and seasonal components of the time series and then use statistical metrics to detect anomalies.

Furthermore, the combination of two statistical techniques for detection and imputation of anomalies in time series data is proposed by Akouemo and Povinelli (2014). To extract the characteristics of the time series and find the residuals, an autoregressive integrated moving average with exogenous variables (ARIMAX) model is used in this method. By performing hypothesis testing on the extrema of the residuals, the anomalies are detected. Then, the anomalous observations are imputed using another ARIMAX model and the process is performed in an iterative manner.

In comparison to estimation model-based techniques, prediction model-based techniques fit a model to the time series and obtain $\hat{y}_t$ using only the past data. Observations that vary significantly from their predicted values are identified as anomalies. Some prediction model-based methods use a fixed model and therefore are not able to adapt to the changes that occur in the data over time (Bl'azquez-Garc'ia et al., 2020). For example, Munir et al. (2019) propose an anomaly detection method that applies a fixed convolutional neural network (CNN) to predict future values. Likewise, Hill and Minsker (2010) use an autoregressive (AR) model to predict values for the future. Other methods are suited to the development of the time series by retraining the model. For instance, Zhou et al. (2018) build an autoregressive integrated moving average (ARIMA) model with seafloor observatory data using a sliding window so that the model is refitted with the sliding window movement.

Histogram-based techniques can also be covered in this category as a nonparametric approach. This type of approach is based on the identification of observations whose exclusion from the univariate time series results in a lower error than the original histogram representation. The histogram is created by calculating the average of the values

within each bin where the order of observations is retained. Given a sequence of observations *Y* and a number of bins *K*, *S* □ Y is a deviant set if

$$E_Y(H_K^*) > E_{Y-S}\left(H_{K-|S|}^*\right) \tag{2.2}$$

where $E_Y()$ is the total error in the approximation, H_K^* is the optimal histogram on *Y* with *K* bins and $H_{K-|S|}^*$ is the optimal histogram on $Y - S$ with $K - |S|$ bins. Optimal histogram refers to the histogram with lowest approximation error (Bl'azquez-Garc'ia et al., 2020).

Jagadish et al. (1999) propose a dynamic programming algorithm that uses optimal histograms to detect deviants in time series data. Later, an approximation to the dynamic programming-based approach is suggested by Muthukrishnan et al. (2004).

In detection of anomalies using histogram-based techniques, the size of the bin which is needed when constructing the histogram is the main challenge. Many normal observations will fall in rare or empty bins if the bins are small. This will lead to a lot of false positives. On the other hand, many abnormal observations will fall in frequent bins, if the bins are large. This will prevent abnormalities from being identified (Chandola et al., 2009).

Briefly, there are advantages and disadvantages of statistical anomaly detection techniques. If the distribution of the data to be examined is determined correctly, statistical methods offer a reasonable solution for the detection of anomalies. However, statistical methods depend on the assumption that a particular distribution generates the data. This assumption is often not correct. Another advantage of using statistical algorithms is that statistical methods are associated with statistical hypotheses and confidence intervals whose values may be affected by the abnormal data. That is the reason to search for robust methods. These components can provide flexibility while making inferences about the data instances. However, choosing the best statistic for hypothesis testing is often not an easy task (Chandola et al., 2009).

2.2.2.2 Proximity-Based Methods

Proximity-based methods assume that an observation is an anomaly if the nearest neighbors of the observation are far away in the feature space. There are two main types of proximity-based anomaly detection methods: density-based and distance-based anomaly detection.

The neighborhood of an observation, identified by a given radius, is consulted by a distance-based anomaly detection method. An observation is then regarded as an anomaly if there are not enough other points in its neighborhood. The density of an observation and that of its neighbors are investigated by a density-based anomaly detection method. If the density of an observation is relatively much lower than that of its neighbors, the observation is then identified as an anomaly (Han et al., 2011).

The distance of each data point to its *k*th nearest neighbor is determined in basic nearest neighbor methods. These methods calculate the anomaly score of a data instance based on its distance to its *k*th nearest neighbor. There are several variants of the basic nearest neighbor methods. Some methods calculate the anomaly score of an observation in the dataset as the sum of its distances from its *k* nearest neighbors. Another way to calculate the anomaly score of an observation is to count the number of nearest neighbors that are not more than *d* distance away from the given observation. Some methods prune the search space either by focusing on observations that are most likely to be abnormal or by ignoring instances that cannot be anomalous (Chandola et al., 2009).

In order to calculate the distance of a data instance to its nearest neighbor, these algorithms use a threshold value. The performance of the methods is dependent on the choice of *k* and the distance metric used. The value of *k* can generally be taken as the square root of the number of observations in the training dataset (Mittal et al., 2019). Euclidean distance, Minkowski distance and Mahalanobis distance are common distance measures.

If the different dimensions in the dataset $\Re$ are not commensurable and have different mutual correlations, a preferred measure is the Mahalanobis distance,

$$\sqrt{(P-Q)^T \sum^{-1} (P-Q)} \tag{2.3}$$

where $\sum$ is the covariance matrix measuring the mutual correlations between dimensions for all observations in the dataset $\Re$. If the covariance matrix $\sum$ is the identity matrix, this means the most frequently used Euclidean distance measure,

$$\sqrt{\sum_{i=1}^{n} (P_i - Q_i)^2}. \tag{2.4}$$

The Minkowski distance of order k between two points $P = (P_1, \dots, P_n)$ and $Q = (Q_1, \dots, Q_n) \in \Re$ is defined as,

$$(\sum_{i=1}^{n} |P_i - Q_i|^k)^{\frac{1}{k}}. \tag{2.5}$$

One and two are the most commonly used values of *k*. Minkowski distance is equivalent to the Euclidean distance for $k = 2$. For $k = 1$, this distance is equal to $(\sum_{i=1}^{n} |P_i - Q_i|)$, which is known as the Manhattan distance (Mehrotra et al., 2017).

The density of the neighborhood of each data instance is estimated by density-based anomaly detection methods. While an observation that lies in a dense neighborhood is said to be normal, an observation that lies in a low density neighborhood is said to be abnormal. If the data has regions of differing densities, density-based methods perform poorly. A series of techniques has been suggested to compute the density of observations relative to the density of their neighbors in order to deal with the issue of differing densities in the dataset (Chandola et al., 2009). Local outlier factor (LOF) is a degree of being an anomaly. The degree depends on how isolated the observation is relative to its neighborhood. Reachability distance of an observation is computed to calculate LOF. Reachability distance of an observation *m* with respect to observation *n* is the maximum value of the *k*-distance of observation *n* and *d (m, n)*. Essentially, the reachability distance of observations *m* and *n* will be the *k*-distance of observation *n* if observation *m* is within the *k* neighbors of observation *n*. In the other case, it will be the real distance of *m* and *n* (Breunig et al.,

2000). For the best performance of LOF, the choice of the parameter *k* and a reasonable threshold for the LOF value is important (Goldstein & Uchida, 2016).

Variants of the LOF technique have been suggested by many researchers. The connectivity-based outlier factor (COF) aims to increase the performance of the LOF technique when the density of nearby data instances is similar to that of the anomaly (Tang et al., 2002). The COF algorithm operates like the LOF algorithm except that the COF density approximation works differently for the data points. The *k* nearest neighbors is calculated for the LOF by using the Euclidean distance, while the COF uses a different approach called the chaining distance to calculate the *k* nearest neighbors. Specifically, the LOF assumes that the observations are distributed in a spherical form while the COF assumes that the data points are linearly distributed (Alghushairy et al., 2021). The influenced outlierness (INFLO) algorithm is designed to detect anomalies when abnormal observations are in the location where the density distributions in the neighborhood are substantially different (Jin et al., 2006). The LOF algorithm cannot score the data points accurately when the data points contain clusters with diverse densities that are close to each other. In the INFLO technique, the set of the reverse nearest neighbors is considered a neighbor and the *k* nearest neighbors method is used (Alghushairy et al., 2021).

Density-based anomaly detection methods have been widely discussed for non-temporal data. Also, the concept of neighborhood is more complex in time series data since the data is ordered (Bl'azquez-Garc'ia et al., 2020). Ishimtsev et al. (2017) proposed a time series anomaly detection method which is based on the *k* nearest neighbors and time-delay embedding. Angiulli and Fassetti (2010) suggested a distance-based anomaly detection method taking temporality into account. These methods adopt sliding windows.

A major advantage of proximity-based methods is that they are unsupervised in nature and they do not make any assumptions about the generative distribution of the data to be analyzed (Chandola et al., 2009).

The performance of nearest neighbor methods depends on the distance measure which can discriminate between normal and abnormal observations effectively (Chandola et al., 2009). Furthermore, the dataset plays an important role in deciding the perfect score for *k* nearest neighbors methods. The choice of *k* also affects the results obtained from these methods. Moreover, choosing an acceptable threshold when it is needed is one of the most complex tasks for most of the distance-based algorithms (Wang et al., 2019).

Although some density-based anomaly detection methods are shown to have improved performance, in most cases, they are more complex and computationally costly compared to statistical methods in particular. They are sensitive to parameter settings, such as the size of the neighbors being determined. For regions with different densities, they become more complex and result in poor performance (Wang et al., 2019).

2.2.2.3 Clustering-Based Methods

Methods based on clustering assume that the normal observations belong to dense and large clusters while anomalies belong to, or do not belong to, sparse or small clusters. There are many approaches for clustering. Hence, there are many methods for clustering-based anomaly detection as well.

Clustering-based anomaly detection methods can be classified into three categories according to the way they treat normal and abnormal observations. The first category of clustering-based methods assumes that while anomalies do not belong to any cluster, normal observations belong to a cluster in the dataset. The second category of clustering-based methods assumes that while anomalies are far away from their nearest cluster centroid, normal observations lie close to their nearest cluster centroid. The third category of clustering-based methods assumes that while anomalies either belong to small or sparse clusters, normal observations belong to large and dense clusters (Chandola et al., 2009).

Density Based Spatial Clustering of Applications with Noise (DBSCAN) is an example of the first category of clustering-based methods. The method requires two input parameters,

Eps and MinPts. Eps is the maximum distance between two points to consider them as neighbors and MinPts is the minimum number of points to form a cluster. If the number of points within the Eps-neighborhood of a point is greater than or equal to MinPts, this point is defined as a core instance (Ester et al., 1996). If the number of points within the Eps-neighborhood of a point is less than MinPts and no point in its Eps-neighborhood is a core instance, this point is defined as an anomaly.

The *k*-means algorithm is an example of the second category of clustering-based methods. An input to the algorithm is the number of clusters *k*. An observation belongs to a cluster if, compared to all other cluster centers, its distance to the corresponding cluster center is minimized. The *k*-means algorithm operates by choosing *k* random initial cluster centers and computing the distances between these cluster centers and each observation in the dataset. Then, observations are allocated to the clusters. With the updated cluster membership, new cluster centers are calculated and the process is repeated until no observations can be moved between clusters (Steinley, 2006).

Techniques based on the third category of clustering-based methods identify observations of clusters whose density and/or size is below a threshold as abnormal. Find cluster-based local outlier factor (FindCBLOF) is an example of the third category of clustering-based methods. This method calculates the anomaly score based on cluster-based local outlier factor (CBLOF). The CBLOF of an observation is calculated by both the distance between the observation and its closest cluster or its cluster and the size of the cluster the observation belongs to (He et al., 2003).

An advantage of clustering-based methods is that they can be operated in an unsupervised mode. Another advantage of these methods is that they can often be adapted to different data types (Chandola et al., 2009).

Although clustering-based methods have some advantages, these methods also have many disadvantages. Many clustering-based methods classify anomalies as byproduct of clustering and therefore they are not optimized for anomaly detection. Clustering is a costly

process in data mining. A basic adaptation of a clustering method for the identification of anomalies can be very expensive. Moreover, many clustering-based methods require users to determine the number of clusters in advance, which is not an easy task. Furthermore, some clustering methods force some clusters to be allocated to every data instance. This might cause anomalies to be marked as normal. Additionally, some clustering-based methods are only effective when abnormal observations do not generate meaningful clusters among themselves (Chandola et al., 2009; Han et al., 2011; Wang et al., 2019).

In terms of distance calculation, a similarity can be considered between clustering-based methods and proximity-based methods. Several clustering-based methods use distance measures, and the performance of these methods depends, therefore, on the distance measure chosen. However, clustering-based methods identify each observation with respect to the cluster it belongs to while proximity-based methods check each observation with respect to its neighborhood (Chandola et al., 2009).

If a training dataset with class labels is available, anomaly detection can be treated as a classification problem. Training a classification model that can distinguish normal data from anomalies is the general concept of classification-based anomaly detection methods. However, for anomaly detection, these techniques do not perform well because the training dataset is usually heavily biased. In other words, in the training dataset, the number of normal observations is substantially higher than the number of abnormal observations. This imbalance may prevent the development of an accurate classifier. Classification-based anomaly detection methods often use a one-class model to overcome this challenge. In this way, a classifier is intended to describe only the normal class and any observations that do not belong to the normal class are identified as anomalies (Han et al., 2011). One Class Support Vector Machine (OCSVM) is an example of classification-based anomaly detection methods. OCSVM defines a decision boundary. If a new observation is detected within the boundary, it is classified as normal. Otherwise, it is identified as an anomaly (Schölkopf et al., 2001). OCSVM is conventionally used in a semi-supervised way. OCSVM can also be used in an unsupervised setting, but the method is sensitive to anomalies in the data (Amer et al., 2013).

The availability and quality of the training dataset have significant influence on the performance of classification-based anomaly detection methods. Obtaining representative and high-quality training data is difficult in many applications. This situation restricts the applicability of classification-based anomaly detection methods (Han et al., 2011).

2.3 Robust Mahalanobis Distance

Masking effect: One anomaly is said to mask a second anomaly if the second anomaly can be regarded as an outlier only by itself, but not in the presence of the first anomaly. Therefore, the second observation appears as an anomaly after the deletion of the first anomaly. Masking happens when the mean and covariance estimates are distorted by a cluster of abnormal observations. For instance, this may occur if there is a small cluster of anomalies which draw the sample mean and inflate the covariance estimate in its direction. In this case, the resulting distances of the abnormal observations from the mean become smaller.

Swamping effect: One anomaly is said to swamp a second observation if the second observation can be regarded as an anomaly only under the presence of the first anomaly. In other words, the second observation becomes normal after the deletion of the first anomaly. Swamping happens when the mean and covariance estimates are distorted by a cluster of abnormal observations. For instance, this may occur if there is a small cluster of anomalies which draw the sample mean and inflate the covariance estimate toward it and away from the pattern of the normal observations. In this case, the resulting distances of the observations from the mean become larger, which makes them appear to be anomalies (Penny & Jolliffe, 2001).

In multivariate statistics, the Mahalanobis distance is a well-known measure. By taking into account the dispersion of the data around the center, it calculates the distance of a group of data points to the center (Roizman et al., 2021). Denote the mean vector by μ and the covariance matrix by $\sum$. The Mahalanobis distance (MD) can be written as (Iqbal et al., 2020):

$$MD(y, \hat{\mu}, \hat{\sum}) = \sqrt{(y - \hat{\mu})^T \hat{\sum}^{-1} (y - \hat{\mu})}. \quad (2.6)$$

μ and $\sum$ are calculated as following.

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} Y_i \quad (2.7)$$

$$\hat{\sum} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{\mu})(Y_i - \hat{\mu})^T \quad (2.8)$$

A large MD can mean that an anomaly is the corresponding observation (Aguinis et al., 2013). Masking and swamping effects play an important role in the ability of MD to detect anomalies. Masking effect may reduce the MD of an anomaly. On the other hand, swamping effect may increase the MD of a normal observation. Using robust estimates of the location and covariance matrix, these problems can be avoided (Penny & Jolliffe, 2001).

To deal with these problems, Hadi (1992) suggests replacing the mean vector with coordinate-wise medians and thus calculating MD using robust estimators of location and covariance matrix. Similarly, Cabana et al. (2019) propose robust estimators based on shrinkage for defining a robust MD in order to identify anomalies. For similar purpose, Draskovic & Pascal (2017) calculates MD using M-estimators.

The use of robust estimators such as S-estimators, M-estimators, minimum volume ellipsoid (MVE) estimator, MM-estimators, minimum covariance determinant (MCD) estimator detects anomalies better compared to the use of classical estimators (Rousseeuw & Hubert, 2017; Rousseeuw & Leroy, 1987).

Rousseeuw (1985) shows that combining a high breakdown point with affine equivariance is certainly possible. It is proved that MVE estimator and MCD estimator both are affine equivariant with a high breakdown point.

The breakdown point is the smallest fraction of contamination that can cause arbitrarily large aberrant values to be taken by an estimator. It is said that an estimator B is affine (location-scale) equivariant if and only if

$$B(My_1 + b, \dots, My_n + b) = MB(y_1, \dots, y_n) + b \qquad (2.9)$$

for any nonsingular matrix M and any vector b (Rousseeuw, 1985).

Rousseeuw and van Zomeren (1990) suggest calculating distances based on very robust location and covariance estimates to prevent the masking effect. They calculate MD using MVE estimators of mean and covariance. They demonstrate that this approach detects anomalies more efficiently compared to MD.

For the MVE, the location of a sample $Y = \{y_1, \dots, y_n\}$ of n observations in p dimensions is estimated using center of the minimal volume ellipsoid covering at least h observations of Y. For the MCD, it is estimated using mean of the h observations of Y for which the determinant of the covariance matrix is minimal (Rousseeuw, 1985).

The MCD attempts to find h observations (from n) whose classical covariance matrix has the lowest determinant. In this case, the MCD estimate of location is the mean of these h observations and the MCD estimate of dispersion is the covariance matrix of these h observations. The breakdown value of the MCD equals that of the MVE, but over the MVE, the MCD has several advantages (Rousseeuw & van Driessen, 1999). The breakdown value of the MCD is $(n - h + 1)/n$. The number h, therefore, determines the robustness of the estimator. h should be selected close to $0.5n$ if a large proportion of contamination is expected. Otherwise, to achieve greater finite-sample efficiency, an intermediate value for h, such as $0.75n$, is suggested (Rousseeuw & Hubert, 2017).

The MCD has better statistical efficiency compared to the MVE. The MCD is asymptotically normal (Butler et al., 1993). In addition, the MCD has a higher convergence rate than the MVE (Davies, 1992). Furthermore, robust distances which are

based on the MCD are more precise than those which are based on the MVE (Rousseeuw & van Driessen, 1999). Hence, in this thesis, MCD is used.

2.4 Selected Anomaly Detection Algorithms

The best known time series anomaly detection methods in the literature are described below and will be used in performance comparison studies.

2.4.1 Isolation Forest

The Isolation Forest (iForest) is an anomaly detection algorithm that explicitly isolates abnormal observations in the dataset. The algorithm uses a binary tree structure called the isolation tree (iTree) to separate all observations. The detection of anomalies with iForest is a two-step operation. In the first stage, the training set is recursively partitioned to create iTrees. In the second stage, the test instances are passed through isolation trees to obtain an anomaly score for each one.

The assumption is that abnormal observations are rare, and that they have properties which differ significantly from those of normal observations. As a consequence, anomalies are closer to the root of the iTree and hence have a shorter path length. Normal observations, on the other hand, are located at the deeper end of the tree and are more difficult to isolate since they occur in dense regions of the input space. The anomaly score Ψ of an observation *y* is calculated as follows (Liu et al., 2008):

$$\Psi(y, n) = 2^{-\frac{E(h(y))}{b(n)}} \tag{2.10}$$

where $h(y)$ is the path length of observation *y*, $E(h(y))$ is the average of $h(y)$ from a collection of isolation trees, *n* is the number of external nodes and

$$b(n) = 2F(n-1) - \left(\frac{2(n-1)}{n}\right) \tag{2.11}$$

where $F(i)$ is the harmonic number and it can be estimated by $\ln(i) + 0.5772156649$. Using the anomaly score Ψ, the following assessments are made.

- If observations have anomaly scores very close to 1, then they are definitely anomalies.
- If observations have anomaly scores much smaller than 0.5, then they are normal instances.
- If all observations return anomaly scores that are close to 0.5, then the dataset does not really have any obvious abnormal observation (Liu et al., 2008).

2.4.2 Twitter's Method

Twitter Inc. made its anomaly detection package open source in 2015. The method uses time series decomposition to detect anomalies. The technique is built upon the generalized Extreme Studentized Deviate (ESD) test with robust statistics. The method approximates the underlying long-term trend using a piecewise method and employs robust statistical metrics such as median and median absolute deviation (MAD) together with generalized ESD (Vallis et al., 2014).

For the *h* most extreme values in the dataset, ESD computes the following test statistic.

$$D_h = \frac{max_h|y_h - \bar{y}|}{s}. \tag{2.12}$$

To assess whether an instance is abnormal, the test statistic is compared to a critical value calculated using Equation (2.13). If the instance is anomalous, it is excluded from the dataset and the critical value is recalculated using the remaining dataset. With the number of anomalies equal to the largest *h* such that $D_h > \mathrm{C}_h$, ESD repeats this process *h* times (Hochenbaum et al., 2017) where

$$\mathrm{C}_h = \frac{(n-h)t_{p,n-h-1}}{\sqrt{(n-h-1+t_{p,n-h-1}^2)(n-h+1)}}. \tag{2.13}$$

A single anomaly can distort the sample mean $\bar{y}$ with the distortion increasing as $y_t \to \pm\infty$. The sample median, on the other hand, is robust to such distortions and can withstand up to 50% of the data being abnormal. For a univariate dataset $y_1, y_2, \ldots, y_n$, MAD is calculated as,

$$median_i(|y_i - median_j(y_j)|). \tag{2.14}$$

MAD is robust to anomalies in the dataset and can be used to estimate standard deviation.

$$\hat{\sigma} = a * MAD \tag{2.15}$$

where $a = 1.4826$ is employed for normally distributed data and $a = \frac{1}{Quantile(0.75)}$ is used for other distributions (Hochenbaum et al., 2017).

The technique decomposes the time series into seasonal, trend and residual components. The periodic variation of the time series is explained by the seasonal component and the long-term non-periodic variation is described by the trend component. After seasonal and trend components are eliminated, the residual component is defined as the remainder of the time series.

As opposed to other methods, the method employs the median of the time series to represent the trend value. The method approximates the underlying long-term trend using a piecewise combination of short-term medians. The method removes the seasonal component by employing Seasonal-Trend Decomposition Procedure Based on Loess (STL). Then, the technique applies generalized ESD test with robust statistics to the residual component in order to detect anomalies (Vallis et al., 2014).

2.4.3 STL

Based on locally weighted scatterplot smoothing referred to as loess, STL is a Seasonal-Trend Decomposition Technique. A time series is decomposed into trend, seasonal and remainder components using the STL filtering procedure. STL is a technique that involves a sequence of smoothing operations.

Two recursive procedures, an inner loop and an outer loop, are used in the method. The seasonal and trend components are updated once in each of the passes through the inner loop. In the inner loop, there are six steps (Cleveland et al., 1990).

Step 1: Detrending.

Assume the following additive model:

$$Z_t = S_t + T_t + L_t \tag{2.16}$$

where S_t, T_t, L_t denotes the seasonal component, the trend component and the remainder component respectively for $t = 1$ to N. In this step, a detrended series, $Z_t - T_t^{(k)}$ is calculated.

Step 2: Cycle-subseries smoothing.

Loess smoother is applied to each cycle-subseries of the detrended series calculated in the first step. Then, the collection of smoothed values for all of the cycle-subseries, $C_t^{(k+1)}$ is calculated.

Step 3: Low-pass filtering of smoothed cycle-subseries.

A low-pass filter is implemented to $C_t^{(k+1)}$. The filtering procedure consists of moving averages and a loess smoothing. In this step, the output, $M_t^{(k+1)}$ is determined at time positions $t = 1$ to N.

Step 4: Detrending of smoothed cycle-subseries.

In this step, the seasonal component is calculated as below.

$$S_t^{(k+1)} = C_t^{(k+1)} - M_t^{(k+1)} \tag{2.17}$$

Step 5: Deseasonalizing.

In this step, a seasonally adjusted series is obtained subtracting the seasonal component calculated in the fourth step from the original data. A deseasonalized series is calculated as follows.

$$Z_t - S_t^{(k+1)}. \tag{2.18}$$

Step 6: Trend smoothing.

In this step, the smoothing process is applied to the deseasonalized sequence once more and the trend component, $T_t^{(k+1)}$ is achieved.

In an iteration of the outer loop, estimates of the seasonal and trend components obtained in the inner loop are used to calculate the remainder component. Assume that an initial run of the inner loop is completed to get estimates of the seasonal and trend components. Then, the remainder component is calculated as below (Cleveland et al., 1990).

$$L_t = Z_t - S_t - T_{t.} \tag{2.19}$$

The technique applies generalized ESD test to the residual component in order to detect anomalies. STL uses the same approach as the Twitter's method to remove the seasonal component. The main difference is that while the Twitter's method approximates the underlying long-term trend using a piecewise combination of short-term medians, STL removes the trend component fitting a smoother.

2.4.4 F-test (Generalized Chow Test)

F-test is the generalization of the Chow test (Chow, 1960) and actually it is used to detect the structural changes in the series. The standard linear regression model can be defined as below.

$$y_i = x_i^T \theta_i + \varepsilon_i \qquad (i = 1, \dots, n) \tag{2.20}$$

where at time i, $x_i = (1, x_{i2}, \dots, x_{ik})^T$ is a $k \times 1$ vector of observations of the independent variables, ε_i are iid(0, σ^2), θ_i is the $k \times 1$ vector of regression coefficients and y_i is the observation of the dependent variable. The null hypothesis of "no structural change" is the subject of the structural change tests.

$$H_o: \theta_i = \theta_0 \qquad (i = 1, \dots, n) \tag{2.21}$$

versus the alternative that the coefficient vector varies over time (Hornik et al., 2002).

The regressors are assumed to be nonstochastic with $\|x_i\| = O(1)$ and it is assumed that

$$\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T \to P \tag{2.22}$$

for some finite regular matrix P. Then, $\hat{\theta}^{(i,j)}$ is the ordinary least squares (OLS) estimate of the regression coefficients based on the observations $i + 1, \dots, i + j$ and $\hat{\theta}^{(i)} = \hat{\theta}^{(0,i)}$ is the OLS estimate based on all observations up to i. Thus, in the linear regression model,

$\hat{\theta}^{(n)}$ is the common OLS estimate. In the same way, based on all observations up to i, $X^{(i)}$ is the regressor matrix.

$\hat{\varepsilon}_i = y_i - x_i^T \hat{\theta}^{(n)}$ denotes the OLS residuals with the variance estimate $\hat{\sigma}^2 = \frac{1}{n-k}\sum_{i=1}^{n} \hat{\varepsilon}_i^2$. The recursive residuals are another type of residuals that are often used in structural change tests and calculated as below (Hornik et al., 2002).

$$\tilde{\varepsilon}_i = \frac{y_i - x_i^T \hat{\theta}^{(i-1)}}{\sqrt{1 + x_i^T (X^{(i-1)^T} X^{(i-1)})^{-1} x_i}} \qquad (i = k+1, \ldots, n) \tag{2.23}$$

which have variance σ^2 and zero mean under the null hypothesis. The corresponding estimate of variance is $\tilde{\sigma}^2 = \frac{1}{n-k}\sum_{i=k+1}^{n}(\tilde{\varepsilon}_i - \bar{\tilde{\varepsilon}})^2$.

Unlike other tests, the *F* tests which are often referred to as Chow tests in the literature (Andrews, 1993) are designed to evaluate a single shift alternative. Thus, using the standard linear regression model in Equation (2.20), the alternative can be defined as follows (Hornik et al., 2002).

$$\theta_i = \begin{cases} \theta_A & (1 \le i \le i_o) \\ \theta_B & (i_0 \le i \le n) \end{cases} \tag{2.24}$$

where i_0 denotes a change point in the interval $(k, n-k)$.

For the case where the potential change point i_0 is known, fitting two separate regressions for the two subsamples identified by i_0 is proposed (Chow, 1960). The null hypothesis is rejected when the below test statistic is too large.

$$F_{i0} = \frac{\hat{\varepsilon}^T \hat{\varepsilon} - \hat{u}^T \hat{u}}{\hat{u}^T \hat{u}/(n-2k)} \tag{2.25}$$

where $\hat{u} = (\hat{\varepsilon}_A, \hat{\varepsilon}_B)^T$ are the residuals from the full model for which the subsample coefficients are estimated separately and $\hat{\varepsilon}$ are the residuals from the restricted model, where for all observations the parameters are only fitted once. With k degrees of freedom, the test statistic F_{i0} has an asymptotic χ^2 distribution. F_{i0}/k has an exact F distribution with k and $n - 2k$ degrees of freedom under the normality assumption. The disadvantage of this *F* test is that the change point must be defined in advance. However, there are also tests based on Chow statistics (*F* statistics) that do not necessitate a specification of a precise change point. The idea behind these tests is to compute the *F* statistics for all potential change points. Similarly, the null hypothesis is rejected when any of these statistics is excessively high. The function Fstats in strucchange package in R makes this easy (Hornik et al., 2002).

2.4.5 Chen and Liu's Procedure

A stationary autoregressive moving average (ARMA) process of orders (p, q) for a sequence z_t observed at time points $t = 1, 2, \dots, n$ can be defined as below:

$$z_t = \sum_{i=1}^{p} \beta_i z_{t-i} + e_t + \sum_{i=1}^{q} \phi_i e_{t-i} \tag{2.26}$$

where $e_t \sim NID(0, \sigma_e^2)$. More succinctly,

$$\beta(B) z_t = \phi(B) e_t \tag{2.27}$$

where B is the lag operator. The roots of the autoregressive polynomial, $\beta(B)$, and the moving average polynomial, $\phi(B)$, are all outside of the unit circle. For non-stationary data, a differencing filter is used to transform the observed sequence z_t. The differenced data is then used to construct an ARMA model. For the observed series z_t an ARIMA model is defined as follows (López-de-Lacalle, 2019):

$$\beta(B)\alpha(B)z_t = \phi(B)e_t \tag{2.28}$$

$\alpha(B)$, an autoregressive polynomial with all its roots on the unit circle, provides the differencing filter that makes the data stationary.

Let z_t^* be the observed series subject to k anomalies with weights w. For the observed sequence, the ARIMA model is written as:

$$z_t^* = \sum_{j=1}^{k} w_j L_j(B) I_t(t_j) + \frac{\phi(B)}{\beta(B)\alpha(B)} e_t \tag{2.29}$$

where $I(t_j)$ is an indicator variable that takes the value 1 when the anomaly appears at time $t = j$ and the value 0 everywhere else and $L(B)$ is a lag operators polynomial. The outlier-contaminated estimated residuals are calculated as follows:

$$\pi(B)z_t^* \equiv \hat{\mathfrak{E}}_t = \sum_{j=1}^{k} w_j \pi(B) L_j(B)\, I_t\big(t_j\big) + e_{t.} \tag{2.30}$$

The coefficients of the power series expansion $\pi(B) = \sum_{i=0}^{\infty} \pi_i B^i$ can be calculated from the relation below.

$$\pi(B) = \frac{\beta(B)\alpha(B)}{\phi(B)}. \tag{2.31}$$

The coefficients π_i satisfy $\sum_{i=0}^{\infty} |\pi_i| < \infty$ and vanish to zero since all the roots of $\phi(B)$ lie outside the unit circle.

Equations (2.29) and (2.30) are the basis of the anomaly detection procedure in Chen and Liu's test (1993) which is available in tsoutliers package in R. The steps of this procedure are as follows (López-de-Lacalle, 2019).

Step 1: Locating anomalies.

The significance of all types of anomalies is checked at all possible time points to identify and locate anomalies. The corresponding t-statistics in the regression Equation (2.30) are used to do this.

Step 2: Removing anomalies.

Equation (2.29) is fitted for the observed sequence including the anomaly regressors found in Step 1 to prevent identifying spurious anomalies induced by an incorrect choice or estimate of a model. Any anomalies that turn out to be insignificant are then excluded from the list of potential anomalies.

Step 3: Iterating steps.

In this step, the Steps 1 and 2 are repeated for the adjusted series.

2.5 Performance Measures

To evaluate the effectiveness of anomaly detection methods, the confusion matrix can be used.

Table 2.1: Confusion matrix for anomaly detection

		Detection	
		Anomaly	Normal
Real	Anomaly	True Positive	False Negative
	Normal	False Positive	True Negative

True Positive (TP): This is the count of correctly detected anomalies.

True Negative (TN): This is the count of correctly detected negative values, that is, normal observations are detected as normal.

False Positive (FP): This is the count of wrongly detected normal observation which occurs when normal observations are detected as anomaly.

False Negative (FN): This is the count of undetected anomalies which occurs when anomalies are detected as normal (Fawcett, 2006).

Confusion matrix derived metrics such as accuracy, precision, recall, F1 score (Mund, 2015) and Matthew's correlation coefficient (MCC) (Akosa, 2017) can be used to quantify how good a particular algorithm is at detecting anomalies.

Accuracy: It is simply a ratio of correctly detected observations to the total observations.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)}. \tag{2.32}$$

Precision: It is the ratio of correctly detected positive observations to the total detected positive observations.

$$Precision = \frac{TP}{(TP + FP)}. \tag{2.33}$$

Recall: It is the ratio of correctly detected positive observations to all observations in reality – anomaly. Recall is also named as true positive rate, sensitivity or hit rate.

$$Recall = \frac{TP}{(TP + FN)}. \tag{2.34}$$

F1 score: It is the weighted average of Recall and Precision.

$$F1\ score = 2 \times \frac{(Recall \times Precision)}{(Recall + Precision)}. \quad (2.35)$$

MCC: Its score varies from -1 to 1, where +1 indicates a perfect classification, 0 represents no better than random classification and -1 shows the worst possible classification (Akosa, 2017).

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}. \quad (2.36)$$

CHAPTER 3

PROPOSED METHODS

In this chapter, the methods which we propose are presented. General concepts about the proposed methods are addressed in Section 3.1. The Yakamoz anomaly detection method is covered in Section 3.2. The Sunshine anomaly detection method is described in Section 3.3.

3.1 General Concepts About the Proposed Methods

In the first stage of algorithm design, it is thought that anomaly/anomalies in time series should be made more visible somehow and anomaly detection methods to be developed should be made more robust to achieve better results. Two concepts used in the proposed methods to achieve these goals are described below.

3.1.1 Moving Blocks Matrix

The concept we call the Moving Blocks Matrix (MBM) is the core idea of the proposed methods. By creating MBM, anomaly/anomalies are made more visible. For a sample of size n and block size of m, there are $n-m+1$ recursive vectors creating $n-m+1 \times m$ moving blocks matrix. The illustrative example of derivation of the MBM with the number of columns 5 is given in Figure 3.1 for 10 observations. Assuming that the 4th observation is an anomaly, it is enabled to see this anomaly on 4 separate rows with the creation of MBM.

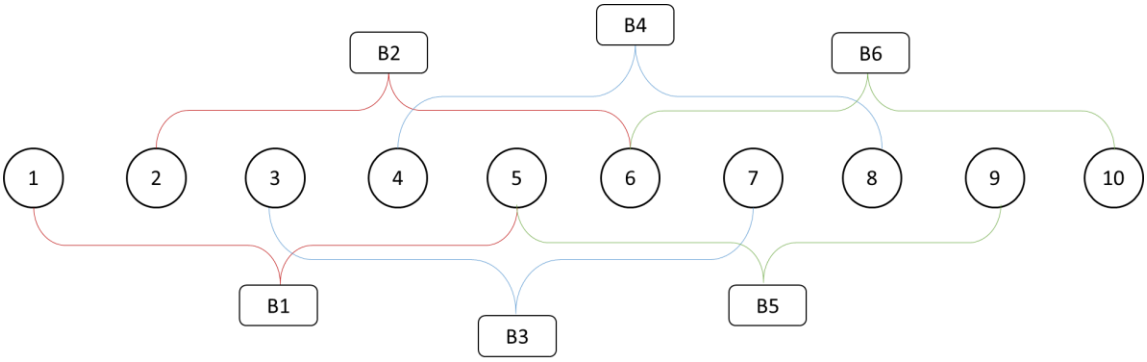

Figure 3.1: An example of the moving blocks procedure

Then, MBM will be as follows.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \\ 6 & 7 & 8 & 9 & 10 \end{bmatrix}_{6\times5}$$

After obtaining the MBM, distances between the rows of the MBM are calculated to detect the possible anomalies. If the calculated distance is high, this might indicate an anomaly. At this point, the choice of the distance measure plays a crucial role.

3.1.2 Calculating Robust Mahalanobis Distance

One of the basic anomaly detection methods can be constructed by calculating z-scores of observations and classifying observations as anomalies if their z-scores exceed a pre-defined threshold. For example, observations with z-scores greater than 3 or smaller than -3 can be classified as anomalies if normally distributed data are available. However, in this method, anomalies will affect z-scores of observations because they will also be used in the mean and standard deviation calculation of z-scores. As a result, the success of the

method will be affected by this situation. Therefore, anomaly detection methods can be made more robust to achieve better results.

In the proposed methods, the Minimum Covariance Determinant (MCD) estimators of mean vector and covariance matrix of MBM are used to achieve robustness for the Mahalanobis distance calculation. MCD looks for the subset of the rows of MBM whose covariance matrix has the smallest determinant, and it computes the mean vector and the covariance matrix of this subset. The goal is to find the "most central" subsample as that one will correspond to the one having least variability among the observations. Hence, there is no anomalous observation in this subsample. The aim here is to create an anomaly free series so that we can compare the distance measures for the series with an anomaly.

After creating MBM and calculating the robust Mahalanobis distances, the cutoff point is determined for detecting abnormal observations. Thus, anomalies are located in the rows of MBM for which the calculated statistics are above the cutoff point.

$$((x_i - \widehat{\mu_{MCD}})^T (\widehat{\sum_{MCD}})^{-1} (x_i - \widehat{\mu_{MCD}}))^{1/2} > c_k \qquad (3.1)$$

where $c_k = (chi - square\ (k, 1 - \alpha))^{1/2}$, the square-root of the upper-α quantile of the chi-square distribution with k degrees of freedom, and μ and $\sum$ denote the mean vector and covariance matrix, respectively. k is the number of columns of MBM (Rousseeuw & Hubert, 2017; Rousseeuw & van Zomeren, 1990).

We propose two new anomaly detection algorithms for time series by the help of above tools. We named the first method as "Yakamoz" because anomaly detection with this method resembles the behavior of sea sparkle that radiates light by the movement of water. Yakamoz is a method for the users who have at least some knowledge on the time series data to be examined. If the user has some knowledge about the behavior of the series, capturing the anomaly points with this method will be obvious. The second method that we propose is named "Sunshine" because anomaly detection with this method resembles sunlight that illuminates the environment without any need. The Sunshine method is a

fully automated anomaly detection tool. Therefore, anyone can use this method very easily. Both proposed methods use the MBM and have tuning parameters to obtain the correct anomaly points. The main difference between them is that the Yakamoz method uses the robust Mahalanobis distances and its critical values in a sequential manner to detect anomalies whereas the Sunshine method uses the Mahalanobis distances and its critical values for only two MBMs. The Sunshine method then uses the summed *k*-nearest neighbors distances obtained by the normal observations to detect anomalies.

The following part gives a detailed explanation of the steps of the algorithms and examples which illustrate the operation of the methods.

3.2 Yakamoz Anomaly Detection Method

This proposed method is more suitable for the users who have at least some knowledge on the time series data to be examined. It enables dynamic exploration of the anomalies in the time series data. It is an iterative method based on creating MBM and calculating the Mahalanobis distances of rows of this MBM using MCD estimators of mean vector and covariance matrix of the observed MBM. The iteration continues until enough number of observations which passed the cutoff point of the Mahalanobis distances are available to calculate MCD estimators of the mean vector and covariance matrix of the next MBM. The method reveals the rows of the Moving Blocks Matrices (MBMs) which include the anomaly/anomalies.

Step 1. Create the MBM:

Choose the numbers of columns of MBMs (length≥2). The number of columns of the first MBM can be chosen randomly. It is a good idea to choose the number of columns of the second, the third, … MBMs as 2.

Step 2. Find MCD estimators of the mean vector and covariance matrix of MBM:

Determine the minimum numbers of rows of MBMs, *m*. MCD will look for the subset of the rows of MBM whose number of rows is equal to *m* and will compute the mean vector and the covariance matrix of the subset whose covariance matrix has the smallest determinant. When there is one anomaly in the time series data, it has been seen that choosing this number as 75% of the total row length of MBM is a good choice. When there is more than one anomaly in the time series data or when the variance of the time series data is high, it has been seen that reducing this number to 50% or even 30% of the total row length of MBM is a good choice.

Step 3. Calculate the Mahalanobis distances of rows of MBM using MCD estimators of the mean vector and covariance matrix of MBM:

Determine α quantile of the chi-square distribution which is used to calculate the cutoff point of the Mahalanobis distances. Natural choices for α are 0.1, 0.05. If variance of the time series data is high, values greater than 0.05 can be selected.

Step 4. Repeat Steps 1-3 until enough number of observations which passed the cutoff point of the Mahalanobis distances are available to calculate MCD estimators of the mean vector and covariance matrix of the next MBM:

Explore the anomaly/anomalies examining observations that passed each MBMs. Note that if all possible MBMs are created in case of multiple anomalies, it can be observed that the rows which include normal observations pass the cutoff point of the Mahalanobis distances towards the last MBMs due to the fact that few normal observations can become abnormal in these multiple anomalies. For this reason, it is a good idea to examine not only the observations that passed the last MBM but also the observations that passed the last two or three MBMs.

To prevent this situation, the maximum number of MBMs that can be created can also be restricted. While working with real and simulated time series datasets, the maximum number of MBMs that can be created is restricted to 7. If the restriction is

made with a smaller number, more false positives may be encountered, especially for large time series data.

Step 5. Change the number of columns of the first MBM and run the method again to be absolutely sure of the anomaly/anomalies.

The user will change the number of columns of the first MBM and run the method again to be absolutely sure of the anomaly/anomalies.

The summary of the algorithm is given in Figure 3.2.

Step 1. Create the MBM
Step 2. Find MCD estimators of the mean vector and covariance matrix of MBM
Step 3. Calculate the Mahalanobis distances of rows of MBM using MCD estimators of the mean vector and covariance matrix of MBM
Step 4. Repeat Steps 1-3 until enough number of observations which passed the cutoff point of the Mahalanobis distances are available to calculate MCD estimators of the mean vector and covariance matrix of the next MBM
Step 5. Change the number of columns of the first MBM and run the method again to be certain about anomaly/anomalies.

Figure 3.2: Steps of the Yakamoz anomaly detection algorithm

3.2.1 Pseudocode for the Yakamoz Anomaly Detection Algorithm

```
Define the number of columns as l_c
Define the mahalanobis distances of rows as d_m
Define the MCD estimator of mean vector as mean_MCD
Define the MCD estimator of covariance matrix as covariance_MCD
Define cutoff point as c

yakamoz <- function(Arg_1, Arg_2, Arg_3, Arg_4, Arg_5)
{
  SET l_c{MBM_2, MBM_3,…} = 2

  ################### MBM_1 ################

  Create MBM_1 using Arg_1 and Arg_2
  Find mean_MCD and covariance_MCD of MBM_1 using Arg_3
  Calculate d_m of MBM_1 using mean_MCD and covariance_MCD of MBM_1
  Calculate c_1 of d_m using Arg_5 and find the rows which passed c_1
  ################### Observations passed MBM_1 ################

  if there is no row passing c_1 then
  {
    print "Ano_MCD_1 No row passed cutoff_1! Second Moving Blocks
Matrix cannot be created"

  }else if there is one row passing c_1 then
  {
    ndata ← the observations in this row
    print "Ano_MCD_1 Only one row passed the first cutoff! Second
Moving Blocks Matrix is not required"
    Return ndata
  }else
  {
    ndata ← the observations in the rows which passed c_1
  }

  try({

    for each movingblocksmatrixnumber
    {

      ################### MBM_2,...,MBM_7 ################

      Create MBM
      Find mean_MCD and covariance_MCD of MBM using Arg_4
      Calculate d_m of MBM using mean_MCD and covariance_MCD of MBM
      Calculate c of d_m using Arg_5 and find the rows which passed c

      ################### Observations passed MBM_2,...,MBM_7
################

      if there is no row passing c then
      {
        obs_passed_cutoff_2 ← NA
      }else if there is one row passing c then
```

```
      {
        obs_passed_cutoff_2 ← the observations in this row

      }else
      {
        obs_passed_cutoff_2 ← the observations in the rows which passed
c

      }
      First column of a matrix ← ndata #Observations passed MBMs will
be in columns of a matrix.
      ndata = obs_passed_cutoff_2
    } #movingblocksmatrixnumber

    Last column of the matrix ← ndata #Observations passed MBMs are in
columns of the matrix.

  }, silent = TRUE) #try

  Return observations passed penultimate MBM
  Return observations passed last MBM

} #yakamoz
```

The following examples have been prepared to illustrate the anomaly detection with this method.

3.2.2 Illustrative Example 1

In this example, the operation of the method and deciding on the anomaly will be illustrated when there is one anomaly in the time series data to be examined.

The user-selected values will be entered and the method will progress. These user-selected values are

- number of columns of the first MBM,
- number of columns of the second, the third, … MBM,
- minimum number of the rows of the first MBM,
- minimum number of the rows of the second, the third, … MBM
- α quantile of the chi-square distribution.

Then the user will change the length of the first MBM and run the method again to be absolutely sure of the anomaly.

It is a good idea to choose the number of columns of the second, the third, … MBM as 2.

It is a good choice to choose the minimum number of the rows of the first, the second, the third, … MBM as 75% of the total row length of the first, the second, the third, … MBM when there is one anomaly in the time series data. Natural choices for α are 0.1, 0.05.

3.2.2.1 Generation of the Time Series Data with One Anomaly

250 observations are generated from a stationary AR(1) model with parameter 0.5 using arima.sim function in R programming. The time series data is generated from a stationary AR(1) model with parameter 0.5 because the logic of the method can be easily explained

using this simple data. set.seed(123) function in R programming is used to reproduce the same observations. The highest observation of the data is 2.636 (highest top five observations: 2.636, 2.547, 2.484, 2.206, 2.140) and the smallest observation of the data is -2.756 (smallest top five observations: -2.756, -2.416, -2.364, -2.309, -2.223). A random observation, such as the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 0.5 unit increment of the highest observation of the series. As a result, too small to be seen by the eye, an anomaly is created. After creating the anomaly, the highest top five observations in the time series data are 3.136, 2.636, 2.547, 2.484 and 2.206. After creating the anomaly, the smallest top five observations in the time series data are -2.756, -2.416, -2.364, -2.309 and -2.223.

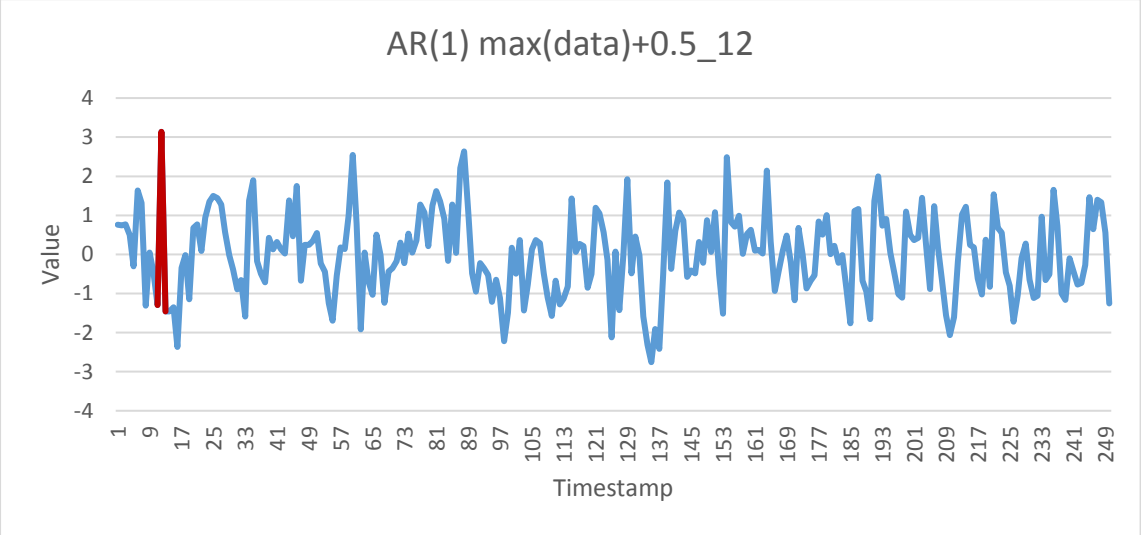

Figure 3.3: The time series plot of the simulated stationary AR(1) max(data)+0.5_12 Data

3.2.2.2 Operation of the Method and Deciding on the Anomaly

The following user-defined values are used to apply the method:

- Number of columns of the 1st MBM: 12
- Number of columns of the 2nd, 3rd, … MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the first MBM): 0.75

- Minimum number of the rows of the 2nd, 3rd, … MBM (of total row length of the 2nd, the 3rd, … MBM): 0.75
- α quantile of the chi-square distribution: 0.05

Then, the method will give the following results.

```
> Observations passed 1st MBM
[1] 0.761 0.741 0.771 0.496 -0.308 1.633 1.314 -1.309 0.047 -0.449
[11] -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.344 -0.019 -1.147 0.680
[21] 0.767 0.088 0.939 -0.140 -2.123 0.070 -1.426 0.027 1.923 -0.483
[31] 0.460 -0.032 -1.588 -2.309 -2.756 -1.909 -2.416 -0.520 1.840 -0.367
[41] 0.604 1.071 0.868 -0.574 -0.407 -0.484
> Observations passed 2nd MBM
[1] -1.293 3.136 -1.458 -2.309 -2.756
> Observations passed 3rd MBM
[1] -1.293 3.136
```

Different colors represent the values that are passed to the second and the third MBM. It should be noted that there will be an extra observation next to the anomaly because the method reveals the rows of MBMs which include the anomaly. From the above results, it is seen that only three MBMs are produced. The user will explore the anomaly thinking as follows.

-1.293, **3.136**, **-1.458** in **"**Observations passed 2nd MBM" are also next to each other in "Observations passed 1st MBM". This means that one of these observations can be the anomaly. **-2.309**, **-2.756** in **"**Observations passed 2nd MBM" are also next to each other in "Observations passed 1st MBM". This means that one of these observations can be the anomaly. **-1.293**, **3.136** are also in "Observations passed 3rd MBM". It can be said that **-1.293** or **3.136** is the anomaly. At this stage the user will choose **3.136** as the anomaly using some knowledge about the data or looking at the observations passed MBMs. Even by controlling the 5 observations in **"**Observations passed 2nd MBM" from 250 observations by eye, it can be said that **3.136** is the anomaly. However, the algorithm will run again under different user-defined values and make sure that **3.136** appears at the last

step with a different number as its neighbor. It should be noted that **-2.756** is the smallest observation in the time series data.

When the method is run again with the same user-selected values, there may be little difference in the number of observations passed the first MBMs. The difference is due to the fact that the number of random sub-samples that are drawn for calculating MCD estimators of the mean vector and covariance matrix of MBM is limited. The default number of random sub-samples is 500. When the method is run again with the same user-selected values, the following results can be found.

```
> Observations passed 1st MBM
[1] 0.761 0.741 0.771 0.496 -0.308 1.633 1.314 -1.309 0.047 -0.449
[11] -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.344 -0.019 -1.147 0.680
[21] 0.767 0.088 0.939 -0.140 -2.123 0.070 -1.426 0.027 1.923 -0.483
[31] 0.460 -0.032 -1.588 -2.309 -2.756 -1.909 -2.416 -0.520 1.840 -0.367
[41] 0.604 1.071 0.868 -0.574 -0.407 -0.484 -1.514 2.484 0.825 0.711
[51] 0.992 0.012 0.523 0.630 0.100 0.115 0.024 2.140
> Observations passed 2nd MBM
1] -1.293 3.136 -1.458 -2.309 -2.756 -1.909 -1.514 2.484
> Observations passed 3rd MBM
[1] 3.136 -1.458
```

From the above results, it is seen that three MBMs are produced again. From the results, it is also seen that the number of observations passed the 1st and 2nd MBMs is slightly different than the first run. The anomaly will be found by thinking the same approach given above.

3.2.2.3 Running the Method Again with the Different Number of Columns of the First MBM to Be Absolutely Sure of the Anomaly

The user is expected to change the number of columns of the first MBM and run the method again to be absolutely sure of the anomaly.

Suppose that the user chooses the length of the first MBM as 22 at this step. The other user-selected values will be the same with the user-selected values selected in the previous step. The user-selected values will be as follows at this step.

- Number of columns of the 1st MBM: 22
- Number of columns of the 2nd, 3rd, … MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the first MBM): 0.75
- Minimum number of the rows of the 2nd, 3rd, … MBM (of total row length of the 2nd, the 3rd, … MBM): 0.75
- α quantile of the chi-square distribution: 0.05

Then, the method will give the following results.

```
> Observations passed 1st MBM
[1] 0.761 0.741 0.771 0.496 -0.308 1.633 1.314 -1.309 0.047 -0.449
[11] -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.344 -0.019 -1.147 0.680
[21] 0.767 0.088 0.939 1.348 1.495 1.436 1.272 0.574 -0.019 -0.390
[31] -0.890 -0.653 -1.592 0.212 -0.856 -0.499 1.195 1.049 0.566 -0.140
[41] -2.123 0.070 -1.426 0.027 1.923 -0.483 0.460 -0.032 -1.588 -2.309
[51] -2.756 -1.909 -2.416 -0.520 1.840 -0.367 0.604 1.071 0.868 -0.574
[61] -0.407 -0.484 0.321 -0.212 0.871 0.061 1.083 -0.508 -1.514 2.484
[71] 0.825 0.711 0.992 0.012
> Observations passed 2nd MBM
[1] -1.293 3.136 -1.458 -2.309 -2.756 -1.514 2.484
> Observations passed 3rd MBM
[1] 3.136 -1.458
```

From the above results, it is seen that at this stage three MBMs are produced again. From the above results, it is also seen that the number of observations passed the 1st MBM and the number of observations passed the 2nd MBM are different than the first stage. The anomaly will be found by using the same approach given in the first step.

It can be said that **3.136** or **-1.458** is the anomaly. At this stage the user will choose **3.136** as the anomaly using some knowledge about the data or looking at the observations passed the MBMs. Even by controlling the 7 observations in **"**Observations passed 2nd MBM" from 250 observations by eye, it can be said that **3.136** is the anomaly. It should be noted that **-2.309** and **-2.756** are the fourth smallest and the smallest observation of the data, respectively. It should also be noted that **2.484** is the fourth largest observation in the time series data.

When the method is run again with the same user-selected values at this stage, the following results can be found.

```
> Observations passed 1st MBM
[1] 0.761 0.741 0.771 0.496 -0.308 1.633 1.314 -1.309 0.047 -0.449
[11] -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.344 -0.019 -1.147 0.680
[21] 0.767 0.088 0.939 1.348 1.495 1.436 1.272 0.574 -0.019 -0.390
[31] -0.890 -0.653 0.212 -0.856 -0.499 1.195 1.049 0.566 -0.140 -2.123
[41] 0.070 -1.426 0.027 1.923 -0.483 0.460 -0.032 -1.588 -2.309 -2.756
[51] -1.909 -2.416 -0.520 1.840 -0.367 0.604 1.071 0.868 -0.574 -0.407
[61] -0.484 0.321 -0.212 0.871 0.061 1.083 -0.508 -1.514 2.484 0.825
[71] 0.711 0.992
> Observations passed 2nd MBM
[1] -1.293 3.136 -1.458 -2.309 -2.756 -1.514 2.484
> Observations passed 3rd MBM
[1] 3.136 -1.458
```

From the above results, it is seen that three MBMs are produced again. When the method is run again with the same user-selected values, two fewer observations passed the first MBM. Similar methodology is applied to detect the anomaly.

3.2.3 Illustrative Example 2

In this example, the operation of the method and decision on the anomalies will be illustrated when there are multiple anomalies in the time series data to be examined.

The user-selected values will be entered and the method will progress again with the same values that are used in one anomaly case.

It is a good idea to choose the number of columns of the second, the third, … MBM as 2.

It is a good choice to choose the minimum number of the rows of the first, the second, the third, … MBM as 0.50 or even 0.30 (if the variance of the time series data is high) of total row length of the first, the second, the third, … MBM when there are multiple anomalies in the time series data. Natural choices for α are 0.1, 0.05.

3.2.3.1 Generation of the Time Series Data with Multiple Anomalies

250 observations are generated from a stationary AR(1) model with parameter 0.5 using arima.sim function in R programming. set.seed(123) function in R programming is used to reproduce the same observations. The highest observation of the data is 2.636 (highest top five observations: 2.636, 2.547, 2.484, 2.206, 2.140) and the smallest observation of the data is -2.756 (smallest top five observations: -2.756, -2.416, -2.364, -2.309, -2.223). Random observations, such as the 12th observation, the 90th observation and the 199th are made higher than the highest observation of the series to create the multiple anomalies. The anomalies are created by 0.5, 1.0 and 1.5 unit increments of the highest observation of the series. As a result, too small to be seen by the eye, anomalies are created. After creating the multiple anomalies, the highest top five observations in the time series data are 4.136, 3.636, 3.136, 2.636 and 2.547. After creating the multiple anomalies, the smallest top five observations in the time series data are -2.756, -2.416, -2.364, -2.309 and -2.223.

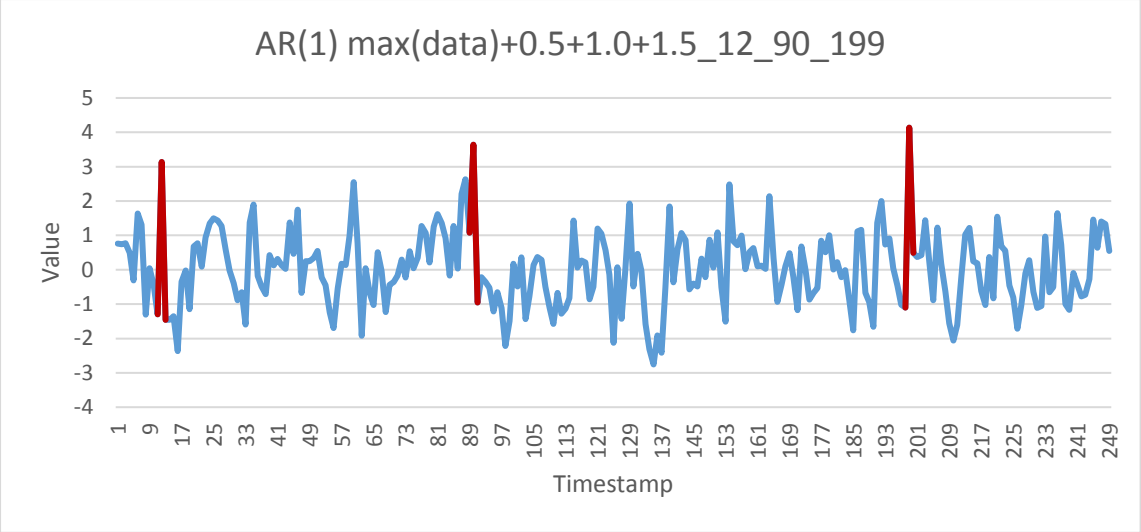

Figure 3.4: The time series plot of the simulated stationary AR(1) max(data)+0.5+1.0+1.5_12_90_199 Data

3.2.3.2 Operation of the Method and Deciding on the Anomalies

The user-defined values are chosen wisely. It is a good choice to choose the minimum number of the rows of MBM as 50% or even 30% (if the variance of the time series data is high) of total row length of MBM when there are multiple anomalies in the time series data.

User-defined values are selected as the following:

- Number of columns of the 1st MBM: 12
- Number of columns of the 2nd, 3rd, … MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the first MBM): 0.50
- Minimum number of the rows of the 2nd, 3rd, … MBM (of total row length of the 2nd, the 3rd, … MBM): 0.50
- α quantile of the chi-square distribution: 0.05

Then, the method will give the following results.

```
> Observations passed 1st MBM
[1] 0.761 0.741 0.771 0.496 -0.308 1.633 1.314 -1.309 0.047 -0.449
[11] -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.344 -0.019 -1.147 0.680
[21] 0.767 0.088 0.939 0.210 1.254 1.620 1.359 0.918 -0.169 1.276
[31] 0.038 2.206 2.636 1.082 3.636 -0.953 -0.220 -0.357 -0.526 -1.215
[41] -0.652 -1.111 -2.223 -1.492 0.173 -0.489 -0.140 -2.123 0.070 -1.426
[51] 0.027 1.923 -0.483 0.460 -0.032 -1.588 -2.309 -2.756 -1.909 -2.416
[61] -0.520 1.840 -0.367 0.604 1.071 0.868 -0.574 -0.407 -0.484 -0.672
[71] -0.947 -1.659 1.369 1.997 0.733 0.910 0.041 -0.456 -1.017 -1.103
[81] 4.136 0.496 0.367 0.427 1.446 0.207 -0.889 1.231 0.174 -0.636
[91] -1.554 -2.062
> Observations passed 2nd MBM
[1] -1.293 3.136 -1.458 -2.364 -0.344 1.082 3.636 -0.953 -2.123 0.070
[11] -2.756 -1.909 -2.416 -0.520 1.840 -1.659 1.369 -1.103 4.136 0.496
> Observations passed 3rd MBM
[1] -1.293 3.136 1.082 3.636 -1.103 4.136
```

It should be noted that there will be an extra observation next to the anomalies because the method reveals the rows of MBMs which include the anomalies. From the above results, it is seen that only three MBMs are produced. The user will explore the anomalies thinking as follows.

-1.293, **3.136**, **-1.458** in **"**Observations passed 2nd MBM" are also next to each other in "Observations passed 1st MBM". This means that one of these observations can be the anomaly. **1.082**, **3.636**, **-0.953** in **"**Observations passed 2nd MBM" are also next to each other in "Observations passed 1st MBM". This means that one of these observations can be the anomaly. **-1.103**, **4.136**, **0.496** in **"**Observations passed 2nd MBM" are also next to each other in "Observations passed 1st MBM". This means that one of these observations can be the anomaly.

-1.293, **3.136** are also in "Observations passed 3rd MBM". It can be said that **-1.293** or **3.136** is the anomaly. **1.082**, **3.636** are also in "Observations passed 3rd MBM". It can be said that **1.082** or **3.636** is the anomaly. **-1.103**, **4.136** are also in "Observations passed 3rd MBM". It can be said that **-1.103** or **4.136** is the anomaly. At this stage the user will choose

3.136, **3.636** and **4.136** as the anomaly using some knowledge about the data or looking at the observations passed MBMs. Even by controlling the 20 observations in **"**Observations passed 2nd MBM" from 250 observations by eye, it can be said that **3.136**, **3.636** and **4.136** are the anomalies. It should be noted that **-2.756**, **-2.416** and **-2.364** in "Observations passed 2nd MBM" are the smallest top 3 observations in the time series data.

When the method is run again with the same user-selected values, the following results can be found.

```
> Observations passed 1st MBM
[1] 0.761 0.741 0.771 0.496 -0.308 1.633 1.314 -1.309 0.047 -0.449
[11] -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.344 -0.019 -1.147 0.680
[21] 0.767 0.088 0.939 0.177 0.141 0.993 2.547 0.782 -1.918 0.047
[31] -0.686 -1.031 0.510 -0.030 -1.236 0.210 1.254 1.620 1.359 0.918
[41] -0.169 1.276 0.038 2.206 2.636 1.082 3.636 -0.953 -0.220 -0.357
[51] -0.526 -1.215 -0.652 -1.111 -2.223 -1.492 0.173 -0.489 -0.140 -
2.123
[61] 0.070 -1.426 0.027 1.923 -0.483 0.460 -0.032 -1.588 -2.309 -2.756
[71] -1.909 -2.416 -0.520 1.840 -0.367 0.604 1.071 0.868 -0.574 -0.407
[81] -0.484 -0.672 -0.947 -1.659 1.369 1.997 0.733 0.910 0.041 -0.456
[91] -1.017 -1.103 4.136 0.496 0.367 0.427 1.446 0.207 -0.889 1.231
[101] 0.174 -0.636 -1.554 -2.062
> Observations passed 2nd MBM
[1] -1.293 3.136 -1.458 2.206 2.636 1.082 3.636 -0.953 -2.756 -1.909
[11] -1.659 1.369 -1.103 4.136 0.496
> Observations passed 3rd MBM
[1] 3.136 -1.458 3.636 -0.953 4.136 0.496
```

When the method is run again with the same user-selected values, twelve more observations passed the first MBM and five fewer observations passed the second MBM. The same methodology is used to decide on the anomaly points. When we examine the last rows of two separate runs, anomaly points are the same, but the numbers next to them changed in the second run. Hence, the values which are stable at the end of each run are

anomalies. However, sometimes it is possible to catch the same numbers as the neighbors. To eliminate this, we recommend rerunning the process with different block sizes.

3.2.3.3 Running the Method Again with the Different Number of Columns of the First MBM to Be Absolutely Sure of the Anomalies

The user is expected to change the number of columns of the first MBM and run the method again to be absolutely sure of the anomalies.

Suppose that the user chooses the length of the first MBM as 5 at this step. The other user-selected values will be the same with the user-selected values selected in the previous step. The user-selected values will be as below at this step.

- Number of columns of the 1st MBM: 5
- Number of columns of the 2nd, 3rd, … MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the first MBM): 0.50
- Minimum number of the rows of the 2nd, 3rd, … MBM (of total row length of the 2nd, the 3rd, … MBM): 0.50
- α quantile of the chi-square distribution: 0.05

Then, the method will give the following results.

```
> Observations passed 1st MBM
[1] -1.309 0.047 -0.449 -1.293 3.136 -1.458 -1.458 -1.354 -2.364 -0.653
[11] -1.592 1.373 1.894 -0.176 0.141 0.993 2.547 0.782 -1.918 0.047
[21] -0.686 -0.169 1.276 0.038 2.206 2.636 1.082 3.636 -0.953 -0.220
[31] -0.357 -0.526 -2.123 0.070 -1.426 0.027 1.923 -1.588 -2.309 -2.756
[41] -1.909 -2.416 -0.520 1.840 -0.367 0.604 0.061 1.083 -0.508 -1.514
[51] 2.484 0.825 0.711 0.992 1.159 -0.672 -0.947 -1.659 1.369 1.997
[61] 0.733 0.041 -0.456 -1.017 -1.103 4.136 0.496 0.367 0.427 1.446
> Observations passed 2nd MBM
[1] -1.293 3.136 -1.458 1.082 3.636 -0.953 1.923 -1.588 -1.514 2.484
```

```
[11] -1.103 4.136 0.496
> Observations passed 3rd MBM
[1] 3.136 -1.458 3.636 -0.953 4.136 0.496
```

From the above results, it is also seen that the number of observations passed the 1st and the 2nd MBMs is different than the first stage. Hence, it can be said that unchanged values **3.136**, **3.636** and **4.136** are the anomalies.

3.3 Sunshine Anomaly Detection Method

This method can be used by anyone who wants to find the anomalies in the time series data without any effort. The method creates only two MBMs and calculates Mahalanobis distances of rows of each MBM using MCD estimators of the mean vector and covariance matrix of each MBM. Then, the method calculates summed *k*-nearest neighbors distance to find the cutoff point for the anomalies in the time series data.

Step 1. The method creates MBM:

Choose the numbers of columns of MBMs (length≥2). By default, the method changes the number of columns of the first MBM from 2 to 20, respectively. It is a good idea to choose the number of columns of the second MBMs as 2.

Step 2. The method finds MCD estimators of the mean vector and covariance matrix of MBM:

Determine the minimum numbers of the rows of the MBMs. MCD will look for the subset of the rows of MBM whose number of rows is equal to this number and will compute the mean vector and the covariance matrix of the subset whose covariance matrix has the smallest determinant.

Step 3. The method calculates Mahalanobis distances of rows of MBM using MCD estimators of the mean vector and covariance matrix of the MBM:

Determine α quantile of the chi-square distribution which is used to calculate the cutoff point of the Mahalanobis distances. Natural choices for α are 0.1, 0.05. If variance of the time series data is high, values greater than 0.05 can be selected.

Step 4. The method repeats above steps until two MBMs are produced:

The method uses the observations, which passed the second MBM, to calculate summed *k*-nearest neighbors distances to find the cutoff point for the anomalies.

Step 5. The method cleans the observations which passed the second MBM by omitting the outlying observations:

Determine a trim number. The method removes the highest and the lowest observations, which passed the second MBM, according to this trim number. The trim number can be interpreted as the number of observations to be cut off at each tail of the observations which passed the second MBM. This number can also be chosen as the fraction (0 to 0.5) of the observations which passed the second MBM.

Step 6. The method calculates the summed *k*-nearest neighbors distances of the cleaned observations to find the highest distance:

This distance is the cutoff point for the observations which passed the second MBM to detect anomalies.

Step 7. The method calculates the summed *k*-nearest neighbors distances of the all observations which passed the second MBM:

The observations which have a greater summed *k*-nearest neighbor distance than the cutoff point calculated in Step 6 are the potential anomalies.

Step 8. The method changes the number of columns of the first MBM and repeats the above steps to be absolutely sure of the anomaly/anomalies.

The method will change the number of columns of the first MBM and repeat the above steps to be absolutely sure of the anomaly/anomalies.

Changing the number of columns of the first MBM from 2 to 20 and providing the frequency of each possible anomaly point detected by the algorithm are the automation part of the algorithm.

The summary of the algorithm is given in Figure 3.5.

Step 1. Create MBM
Step 2. Find MCD estimators of the mean vector and covariance matrix of MBM
Step 3. Calculate Mahalanobis distances of rows of MBM using MCD estimators of the mean vector and covariance matrix of the MBM
Step 4. Repeat Steps 1-3 until two MBMs are produced
Step 5. Clean the observations which passed the second MBM by omitting the outlying observations
Step 6. Calculate the summed *k*-nearest neighbors distances of the cleaned observations to find the highest distance
Step 7. Calculate the summed *k*-nearest neighbors distances of the all observations which passed the second MBM
Step 8. Change the number of columns of the first MBM and run the method again to be absolutely sure of the anomaly/anomalies.

Figure 3.5: Steps of the Sunshine anomaly detection algorithm

3.3.1 Pseudocode for the Sunshine Anomaly Detection Algorithm

```
Define length as l
Define the number of columns as lc
Define the mahalanobis distances of rows as d_m
Define the MCD estimator of mean vector as mean_MCD
Define the MCD estimator of covariance matrix as covariance_MCD
Define cutoff point as c
Define summed k-nearest neighbors distance as d_s

sunshine <- function (Arg_1, Arg_2, Arg_3, Arg_4, Arg_5, Arg_6)
{
  SET l_c{MBM_2} = 2

  for each Arg_2
  {

    try({

      ################### MBM_1 ################

  Create MBM_1 using Arg_1 and Arg_2
  Find mean_MCD and covariance_MCD of MBM_1 using Arg_3
  Calculate d_m of MBM_1 using mean_MCD and covariance_MCD of MBM_1
  Calculate c_1 of d_m using Arg_5 and find the rows which passed c_1

      ################### MBM_2 ################

      if there is no row passing c_1 then
      {
        obs_passed_cutoff_2 ← NA

      }else if there is one row passing c_1 then
      {
        obs_passed_cutoff_2 ← the observations in this row

      }else #elseopenone
      {
        Get the observations in the rows which passed c_1
        Create MBM_2 using these observations
        Find mean_MCD and covariance_MCD of MBM_2 using Arg_4
        Calculate d_m of MBM_2 using mean_MCD and covariance_MCD of MBM_2
        Calculate c_2 of d_m using Arg_5 and find the rows which passed c_2


        if there is no row passing c_2 then
        {
          obs_passed_cutoff_2 ← NA

        }else if there is one row passing c_2 then
        {
          obs_passed_cutoff_2 ← the observations in this row
        }else
        {
```

```
          obs_passed_cutoff_2 ← the observations in these rows
        }
      } #elsecloseone

      ################### knn #############

      SET obs_passed_last_step to NA

      if l{obs_passed_cutoff_2} <= 3 then
      {
        obs_passed_last_step ← the observations in obs_passed_cutoff_2

      }else #elseopentwo
      {
        Remove the highest and the lowest observations in
obs_passed_cutoff_2 according to Arg_6 and inner_1 ← these observations

        Calculate d_s{observations in inner_1} to find the highest
distance
        Calculate d_s{all observations in obs_passed_cutoff_2}
        Determine observations in obs_passed_cutoff_2 which have a
greater d_s than the highest distance and obs_passed_last_step ← these
observations

        if l{obs_passed_last_step} == 0 then
        {
          obs_passed_last_step ← NA
        }

      } #elseclosetwo


      Combine the observations that can be obtained from each Arg_2 in
all_obs_passed_last_step

    }, silent = TRUE) #try

  } #for

  Find more than one observed observations in all_obs_passed_last_step

  Return more than one observed observations

} #sunshine
```

3.3.2 Illustrative Example 3

In this example, the operation of the method and decision on the anomaly will be illustrated when there is only one anomaly in the time series data to be examined.

The user-selected values will be entered and the method will progress. These user-selected values are

- number of columns of the first MBM,
- number of columns of the second MBM,
- minimum number of the rows of the first MBM,
- minimum number of the rows of the second MBM
- α quantile of the chi-square distribution.
- trim number

It is a good idea to choose the number of columns of the second MBM as 2. Natural choices for α are 0.1, 0.05.

3.3.2.1 Generation of the Time Series Data with One Anomaly

The time series data which are produced in Illustrative example 1 are also generated here.

3.3.2.2 Operation of the Method and Deciding on the Anomaly

The user will choose the number of columns of the first MBM and the number of columns of the second MBM, the user will determine the minimum numbers of the rows of MBMs, the user will determine α quantile of the chi-square distribution which is used to calculate the cutoff point of the Mahalanobis distances and the user will choose the trim number which is used to clean the observations in order to start the method.

The following values are used to apply the methodology.

- Number of columns of the 1st MBM: 2:20
- Number of columns of the 2nd MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the 1st MBM): 0.40
- Minimum number of the rows of the 2nd MBM (of total row length of the 2nd MBM): 0.95
- α quantile of the chi-square distribution: 0.05
- Trim number=1

Then, the method gave the following result.

```
> Anomalies
2.484   3.136
  6       11
```

The method prints the anomalous observations with the information on how many times they are identified as anomalies. From the above results, it is seen that **3.136** is identified as the anomaly 11 times out of 19 repetitions (the user selected the number of columns of the first MBM as 2:20), 2.484 is identified as the anomaly 6 times out of 19 repetitions. It should be noted that 2.484 is the fourth largest observation in the time series data.

If the method is run again with the same user-selected values, there may be little difference in the number of observations passed MBMs. The difference is due to the fact that the number of random subsamples that are drawn for calculating MCD estimators of mean vector and covariance matrix of MBM is limited. The default number of random subsamples is 500. When the method is run again with the same user-selected values, the following result can be found.

```
> Anomalies
2.484   3.136
  5       8
```

From the results, it is seen that the method prints the same observations with little difference in the information of how many times they are identified as anomalies. From the results, it is seen that **3.136** is identified as the anomaly 8 times out of 19 repetitions, 2.484 is identified as the anomaly 5 times out of 19 repetitions.

3.3.3 Illustrative example 4

In this example, the operation of the method and deciding on the anomalies will be illustrated when there are multiple anomalies in the time series data to be examined.

The user-selected values will be entered and the method will progress again with the same values that are used in one anomaly case.

It is a good idea to choose the number of columns of the second MBM as 2. Natural choices for α are 0.1, 0.05.

3.3.3.1 Generation of the Time Series Data with Multiple Anomalies

The time series data which are produced in Illustrative example 2 are also generated here.

3.3.3.2 Operation of the Method and Deciding on the Anomalies

Assume that user-defined values are as follows:

- Number of columns of the 1st MBM: 2:20
- Number of columns of the 2nd MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the 1st MBM): 0.45
- Minimum number of the rows of the 2nd MBM (of total row length of the 2nd MBM): 0.95
- α quantile of the chi-square distribution: 0.05

- Trim number=1

Then, the method produced the following results.

```
> Anomalies
0.496   1.082   3.136   3.636   4.136
  5       5       5       5      10
```

The method prints the anomalous observations with the information on how many times they are identified as anomalies. From the above results, it is seen that **4.136** is identified as the anomaly 10 times out of 19 repetitions (the user selected the number of columns of the first MBM as 2:20), **3.636** is identified as the anomaly 5 times out of 19 repetitions, **3.136** is identified as the anomaly 5 times out of 19 repetitions, 1.082 is identified as the anomaly 5 times out of 19 repetitions and 0.496 is identified as the anomaly 5 times out of 19 repetitions.

When the method is run again with the same user-selected values, the following result can be found.

```
> Anomalies
0.496   1.082   3.136    3.636   4.136
  4       4       4        4      11
```

From the above results, it is seen that the method prints the same observations with little difference in the information of how many times they are identified as anomalies. It is also seen that **4.136** is identified as the anomaly 11 times out of 19 repetitions, **3.636**, **3.136**, 1.082 and 0.496 are identified as the anomaly 4 times out of 19 repetitions.

CHAPTER 4

PERFORMANCE COMPARISON STUDIES

Performance comparisons of the proposed two novel time series anomaly detection methods and known the best time series anomaly detection methods in the literature explained in Chapter 2 are investigated in this chapter.

Success of the proposed methods in finding anomaly/anomalies is investigated using real and simulated time series. To evaluate the performance of the methods, we use the confusion matrix and some measures like sensitivity, precision or F1 score.

4.1 Real Datasets

In this part, we want to evaluate the performance of the proposed methods on labelled real time series data. First two series are obtained from Wei (2006). The others from Yahoo! which are used as examples of series with anomalies. These datasets are provided as part of the Yahoo! Webscope program.

For small length real data, W10 Beer Production Data and W14 Airline Passengers Data in Wei (2006) are used. W10 Beer Production Data is a quarterly series of beer production in the United States between 1975 and 1982. The series has a typing error. W14 Airline Passengers data is a series of monthly airline passengers in the United States from January 1995 to March 2002 and has one anomaly that corresponds to September 2001, the month of the World Trade Center tragedy.

For long real data, Yahoo!'s time series "real_1", "real_4", "real_5", "real_33", "real_49" and "real_51" are used. These series are hourly time series with labelled anomalies under

A1 benchmark dataset and chosen randomly. The series are based on the real production traffic to several Yahoo! properties.

4.1.1 Confusion Matrix and Evaluation Results for W10 Beer Production Data

W10 Beer Production Data consists of 32 observations and has one anomaly.

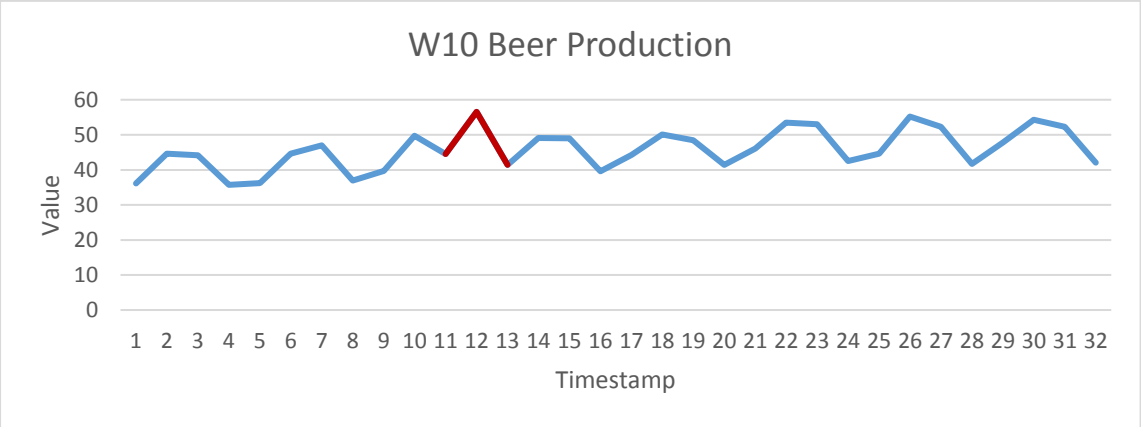

Figure 4.1: The time series plot of W10 Beer Production Data

Table 4.1: Confusion matrix for W10 Beer Production Data

	TP	FN	FP	TN
Sunshine	1.000	0.000	0.000	31.000
Twitter's method	0.000	1.000	0.000	31.000
STL	0.000	1.000	0.000	31.000
F-test	0.000	1.000	1.000	30.000
Yakamoz_last_moving_blocks_matrix	1.000	0.000	1.000	30.000
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	1.000	30.000
Isolation Forest	1.000	0.000	13.000	18.000
Chen and Liu's procedure	0.000	1.000	0.000	31.000

sunshine(data, 20, 0.5, 0.5, 0.05, 1)
yakamoz(data, 2, 0.5, 0.5, 0.05)

Table 4.2: Evaluation metrics for W10 Beer Production Data

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	1.000	1.000	1.000	1.000	1.000
Twitter's method	0.969	0.000	0.000	0.000	-
STL	0.969	0.000	0.000	0.000	-
F-test	0.938	0.000	0.000	0.000	-0.032
Yakamoz_last_moving_blocks_matrix	0.969	0.500	1.000	0.667	0.696
Yakamoz_penultimate_moving_blocks_matrix	0.969	0.500	1.000	0.667	0.696
Isolation Forest	0.594	0.071	1.000	0.133	0.204
Chen and Liu's procedure	0.969	0.000	0.000	0.000	-

From the above tables, it is seen that the Sunshine method is the most successful algorithm in terms of all measurements and the Yakamoz method is the second most successful algorithm in terms of precision, recall, F1 score and MCC. From the tables, it is also seen that Twitter's method, Chen and Liu's procedure, STL and F-test cannot detect the abnormal observation.

4.1.2 Confusion Matrix and Evaluation Results for W14 Airline Passengers Data

W14 Airline Passengers Data consists of 87 observations and has one anomaly near the end of the series.

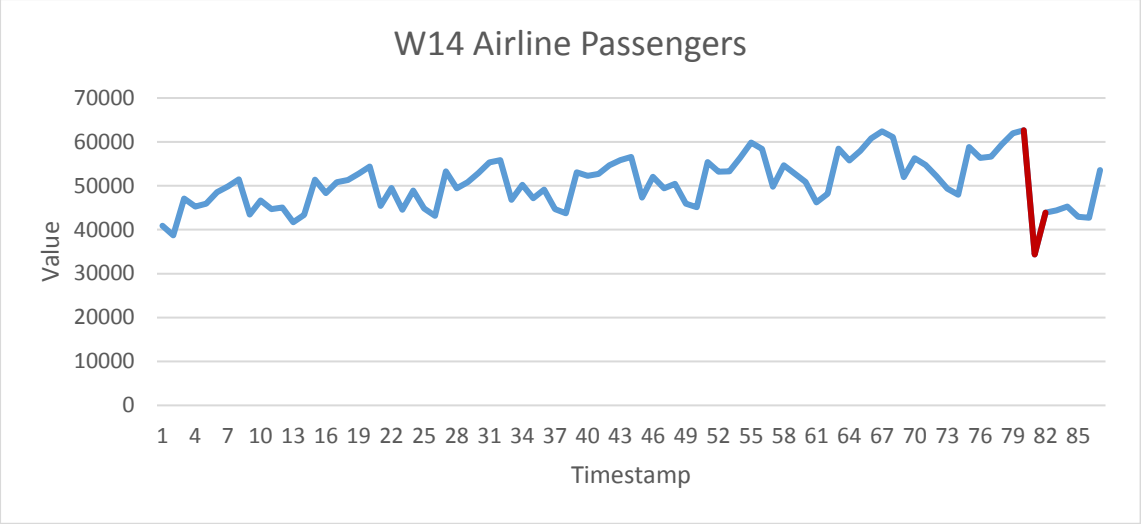

Figure 4.2: The time series plot of W14 Airline Passengers Data

Table 4.3: Confusion matrix for W14 Airline Passengers Data

	TP	FN	FP	TN
Sunshine	1.000	0.000	2.000	84.000
Twitter's method	0.000	1.000	0.000	86.000
STL	0.000	1.000	0.000	86.000
F-test	0.000	1.000	1.000	85.000
Yakamoz_last_moving_blocks_matrix	1.000	0.000	9.000	77.000
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	17.000	69.000
Isolation Forest	1.000	0.000	23.000	63.000
Chen and Liu's procedure	1.000	0.000	1.000	85.000

sunshine(data, 20, 0.5, 0.5, 0.05, 1)
yakamoz(data, 12, 0.95, 0.5, 0.05)

Table 4.4: Evaluation metrics for W14 Airline Passengers Data

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	0.977	0.333	1.000	0.500	0.571
Twitter's method	0.989	0.000	0.000	0.000	-
STL	0.989	0.000	0.000	0.000	-
F-test	0.977	0.000	0.000	0.000	-0.012
Yakamoz_last_moving_blocks_matrix	0.897	0.100	1.000	0.182	0.299
Yakamoz_penultimate_moving_blocks_matrix	0.805	0.056	1.000	0.105	0.211
Isolation Forest	0.736	0.042	1.000	0.080	0.175
Chen and Liu's procedure	0.989	0.500	1.000	0.667	0.703

From the above tables, it is seen that the Sunshine method is the second most successful algorithm in terms of precision, recall, F1 score and MCC. From the tables, it is seen that the Yakamoz method is the third most successful algorithm in terms of precision, recall, F1 score and MCC. From the tables, it is also seen that Twitter's method, STL and F-test cannot detect the abnormal observation.

4.1.3 Confusion Matrix and Evaluation Results for Time Series "real_1"

Time series "real_1" consists of 1420 observations and has 2 anomalies next to each other.

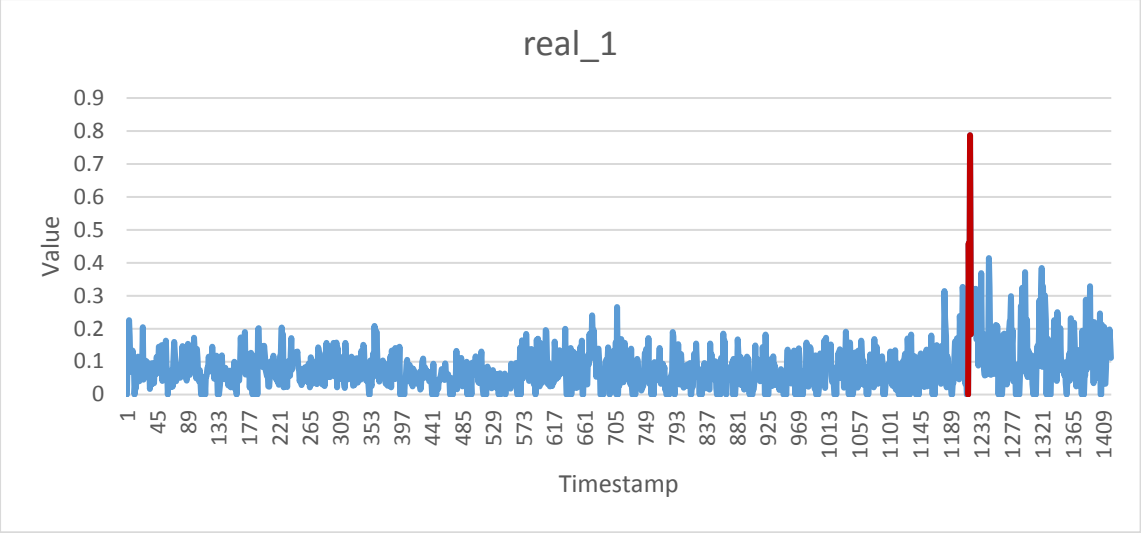

Figure 4.3: The time series plot of "real_1"

Table 4.5: Confusion matrix for time series "real_1"

	TP	FN	FP	TN
Sunshine	2.000	0.000	0.000	1418.000
Twitter's method	2.000	0.000	21.000	1397.000
STL	2.000	0.000	16.000	1402.000
F-test	0.000	2.000	1.000	1417.000
Yakamoz_last_moving_blocks_matrix	2.000	0.000	5.000	1413.000
Yakamoz_penultimate_moving_blocks_matrix	2.000	0.000	106.000	1312.000
Isolation Forest	2.000	0.000	267.000	1151.000
Chen and Liu's procedure	2.000	0.000	25.000	1393.000

sunshine(data, 20, 0.5, 0.5, 0.05, 1)
yakamoz(data, 12, 0.5, 0.5, 0.05)

Table 4.6: Evaluation metrics for time series “real_1”

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	1.000	1.000	1.000	1.000	1.000
Twitter’s method	0.985	0.087	1.000	0.160	0.293
STL	0.989	0.111	1.000	0.200	0.331
F-test	0.998	0.000	0.000	0.000	-0.001
Yakamoz_last_moving_blocks_matrix	0.996	0.286	1.000	0.444	0.534
Yakamoz_penultimate_moving_blocks_matrix	0.925	0.019	1.000	0.036	0.131
Isolation Forest	0.812	0.007	1.000	0.015	0.078
Chen and Liu’s procedure	0.982	0.074	1.000	0.138	0.270

From the above tables, it is seen that the proposed time series anomaly detection methods give better results compared to the other methods. From the tables, it is seen that although Twitter’s method, STL, Isolation Forest and Chen and Liu’s procedure detect the anomalies, they have high false positive rates. In particular, the Isolation Forest method produces too many false positives. It is also seen that the F-test method cannot find the anomalies.

4.1.4 Confusion Matrix and Evaluation Results for Time Series “real_4”

Time series “real_4” consists of 1423 observations and has 5 anomalies which are at the end of the series.

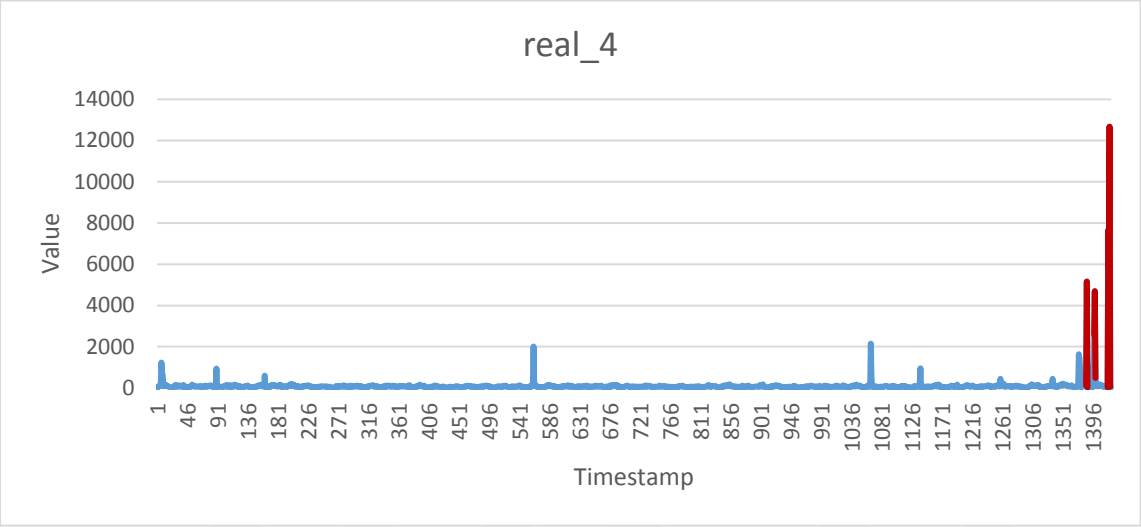

Figure 4.4: The time series plot of “real_4”

Table 4.7: Confusion matrix for time series “real_4”

	TP	FN	FP	TN
Sunshine	5.000	0.000	1.000	1417.000
Twitter’s method	5.000	0.000	22.000	1396.000
STL	5.000	0.000	65.000	1353.000
F-test	0.000	5.000	1.000	1417.000
Yakamoz_last_moving_blocks_matrix	2.000	3.000	0.000	1418.000
Yakamoz_penultimate_moving_blocks_matrix	5.000	0.000	5.000	1413.000
Isolation Forest	5.000	0.000	82.000	1336.000
Chen and Liu’s procedure	5.000	0.000	434.000	984.000

sunshine(data, 20, 0.5, 0.5, 0.05, 5)
yakamoz(data, 12, 0.5, 0.5, 0.05)

Table 4.8: Evaluation metrics for time series “real_4”

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	0.999	0.833	1.000	0.909	0.913
Twitter’s method	0.985	0.185	1.000	0.313	0.427
STL	0.954	0.071	1.000	0.133	0.261
F-test	0.996	0.000	0.000	0.000	-0.002
Yakamoz_last_moving_blocks_matrix	0.998	1.000	0.400	0.571	0.632
Yakamoz_penultimate_moving_blocks_matrix	0.996	0.500	1.000	0.667	0.706
Isolation Forest	0.942	0.057	1.000	0.109	0.233
Chen and Liu’s procedure	0.695	0.011	1.000	0.023	0.089

From the above tables, it is seen that the proposed time series anomaly detection methods give better results compared to the other methods. From the tables, it is seen that the proposed methods have many fewer false positives than Twitter’s method, Chen and Liu’s procedure, STL, Isolation Forest. From the tables, it is also seen that F-test cannot detect the anomalies.

4.1.5 Confusion Matrix and Evaluation Results for Time Series “real_5”

Time series “real_5” consists of 1439 observations and has 2 anomalies at the middle of the series.

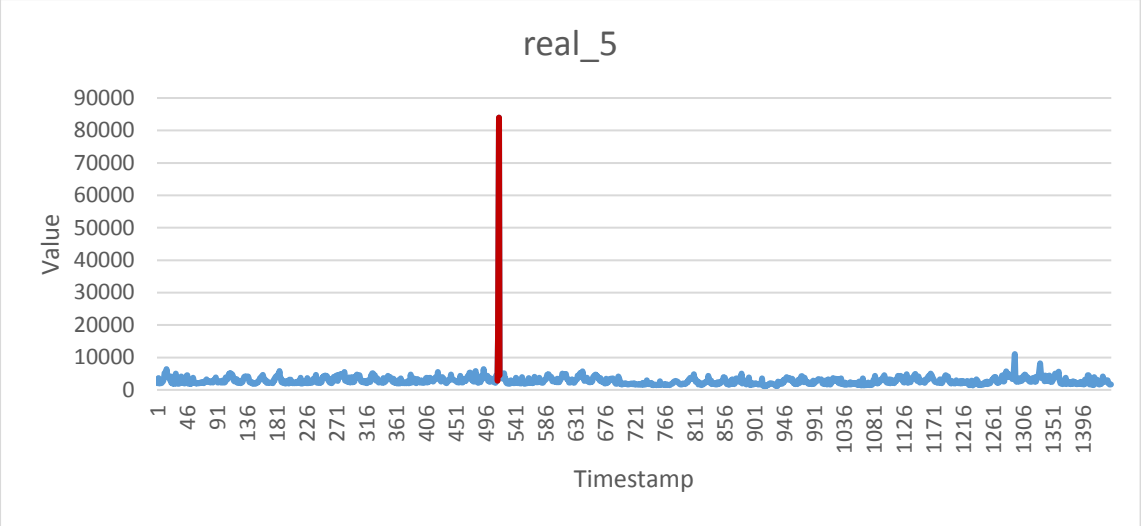

Figure 4.5: The time series plot of “real_5”

Table 4.9: Confusion matrix for time series “real_5”

	TP	FN	FP	TN
Sunshine	2.000	0.000	1.000	1436.000
Twitter’s method	2.000	0.000	5.000	1432.000
STL	2.000	0.000	14.000	1423.000
F-test	0.000	2.000	1.000	1436.000
Yakamoz_last_moving_blocks_matrix	2.000	0.000	9.000	1428.000
Yakamoz_penultimate_moving_blocks_matrix	2.000	0.000	25.000	1412.000
Isolation Forest	2.000	0.000	141.000	1296.000
Chen and Liu’s procedure	2.000	0.000	16.000	1421.000

sunshine(data, 20, 0.5, 0.5, 0.05, 2)
yakamoz(data, 12, 0.5, 0.5, 0.05)

Table 4.10: Evaluation metrics for time series “real_5”

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	0.999	0.667	1.000	0.800	0.816
Twitter’s method	0.997	0.286	1.000	0.444	0.534
STL	0.990	0.125	1.000	0.222	0.352
F-test	0.998	0.000	0.000	0.000	-0.001
Yakamoz_last_moving_blocks_matrix	0.994	0.182	1.000	0.308	0.425
Yakamoz_penultimate_moving_blocks_matrix	0.983	0.074	1.000	0.138	0.270
Isolation Forest	0.902	0.014	1.000	0.028	0.112
Chen and Liu’s procedure	0.989	0.111	1.000	0.200	0.331

From the above tables, it is seen that the Sunshine method gives better results compared to the other methods. From the tables, it is also seen that the Isolation Forest method has too many false positives compared to the other methods, and the F-test method cannot detect the abnormal observations.

4.1.6 Confusion Matrix and Evaluation Results for Time Series “real_33”

Time series “real_33” consists of 1439 observations and has 2 anomalies at the end of the series.

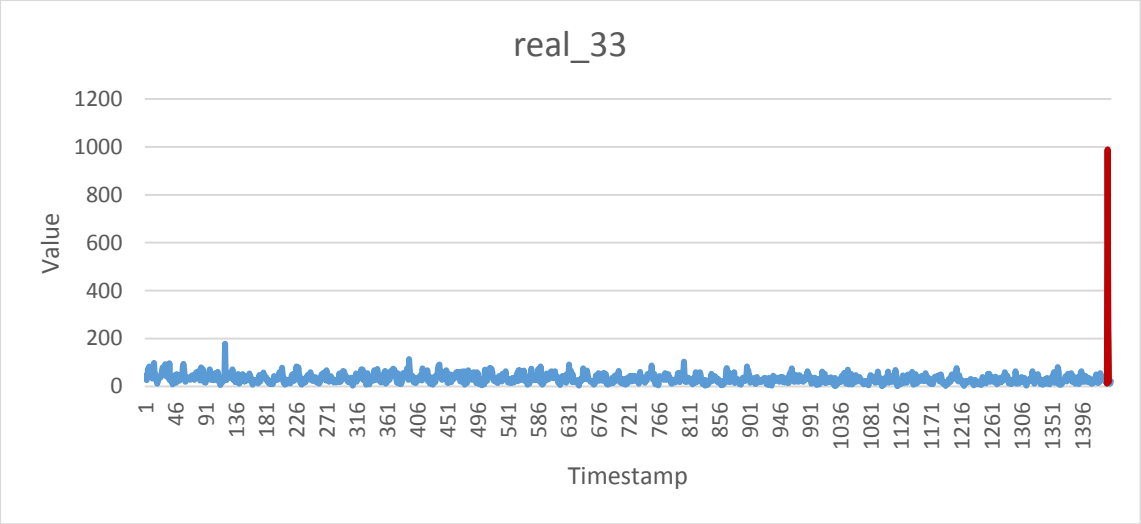

Figure 4.6: The time series plot of “real_33”

Table 4.11: Confusion matrix for time series “real_33”

	TP	FN	FP	TN
Sunshine	2.000	0.000	1.000	1436.000
Twitter’s method	2.000	0.000	6.000	1431.000
STL	2.000	0.000	5.000	1432.000
F-test	0.000	2.000	1.000	1436.000
Yakamoz_last_moving_blocks_matrix	2.000	0.000	7.000	1430.000
Yakamoz_penultimate_moving_blocks_matrix	2.000	0.000	13.000	1424.000
Isolation Forest	2.000	0.000	177.000	1260.000
Chen and Liu’s procedure	1.000	1.000	16.000	1421.000

sunshine(data, 20, 0.5, 0.5, 0.05, 2)
yakamoz(data, 12, 0.5, 0.5, 0.05)

Table 4.12: Evaluation metrics for time series “real_33”

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	0.999	0.667	1.000	0.800	0.816
Twitter’s method	0.996	0.250	1.000	0.400	0.499
STL	0.997	0.286	1.000	0.444	0.534
F-test	0.998	0.000	0.000	0.000	-0.001
Yakamoz_last_moving_blocks_matrix	0.995	0.222	1.000	0.364	0.470
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.133	1.000	0.235	0.363
Isolation Forest	0.877	0.011	1.000	0.022	0.099
Chen and Liu’s procedure	0.988	0.059	0.500	0.105	0.169

From the above tables, it is seen that the Sunshine method gives better results compared to the other methods, and F-test cannot find the anomalies. It is also seen that the Isolation Forest method has too many false positives compared to the other methods.

4.1.7 Confusion Matrix and Evaluation Results for Time Series "real_49"

Time series "real_49" consists of 1461 observations and has 3 anomalies.

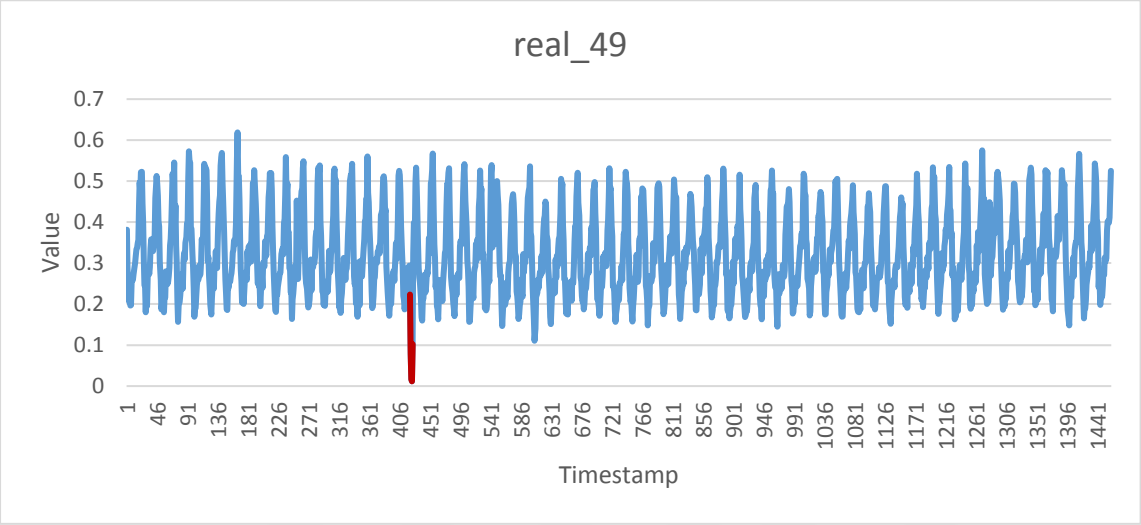

Figure 4.7: The time series plot of "real_49"

Table 4.13: Confusion matrix for time series "real_49"

	TP	FN	FP	TN
Sunshine	3.000	0.000	1.000	1457.000
Twitter's method	0.000	3.000	0.000	1458.000
STL	0.000	3.000	0.000	1458.000
F-test	0.000	3.000	1.000	1457.000
Yakamoz_last_moving_blocks_matrix	3.000	0.000	5.000	1453.000
Yakamoz_penultimate_moving_blocks_matrix	3.000	0.000	26.000	1432.000
Isolation Forest	3.000	0.000	361.000	1097.000
Chen and Liu's procedure	0.000	3.000	21.000	1437.000

sunshine(data, 20, 0.5, 0.5, 0.05, 3)
yakamoz(data, 12, 0.5, 0.5, 0.05)

Table 4.14: Evaluation metrics for time series “real_49”

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	1.000	0.750	1.000	0.857	0.866
Twitter’s method	0.998	0.000	0.000	0.000	-
STL	0.998	0.000	0.000	0.000	-
F-test	0.997	0.000	0.000	0.000	-0.001
Yakamoz_last_moving_blocks_matrix	0.997	0.375	1.000	0.545	0.611
Yakamoz_penultimate_moving_blocks_matrix	0.982	0.103	1.000	0.188	0.319
Isolation Forest	0.753	0.008	1.000	0.016	0.079
Chen and Liu’s procedure	0.984	0.000	0.000	0.000	-0.005

From the above tables, it is seen that the proposed time series anomaly detection methods give better results compared to the other methods and Twitter’s method, Chen and Liu’s procedure, STL and F-test cannot detect the anomalies. It is also seen that the Isolation Forest method has too many false positives compared to the other methods.

4.1.8 Confusion Matrix and Evaluation Results for Time Series “real_51”

Time series “real_51” consists of 1427 observations and has 4 anomalies at different places of the series.

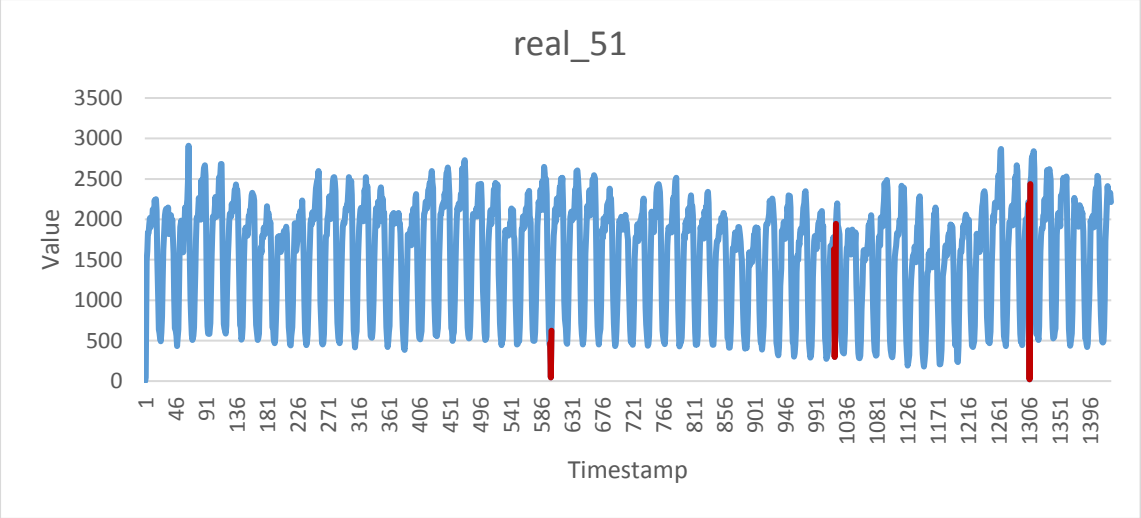

Figure 4.8: The time series plot of “real_51”

Table 4.15: Confusion matrix for time series “real_51”

	TP	FN	FP	TN
Sunshine	3.000	1.000	6.000	1417.000
Twitter’s method	0.000	4.000	0.000	1423.000
STL	2.000	2.000	16.000	1407.000
F-test	0.000	4.000	1.000	1422.000
Yakamoz_last_moving_blocks_matrix	3.000	1.000	6.000	1417.000
Yakamoz_penultimate_moving_blocks_matrix	3.000	1.000	21.000	1402.000
Isolation Forest	4.000	0.000	480.000	943.000
Chen and Liu’s procedure	4.000	0.000	15.000	1408.000

sunshine(data, 20, 0.5, 0.5, 0.05, 4)
yakamoz(data, 12, 0.5, 0.5, 0.05)

Table 4.16: Evaluation metrics for time series “real_51”

	Accuracy	**Precision**	**Recall**	**F1 score**	**MCC**
Sunshine	0.995	0.333	0.750	0.462	0.498
Twitter’s method	0.997	0.000	0.000	0.000	-
STL	0.987	0.111	0.500	0.182	0.232
F-test	0.996	0.000	0.000	0.000	-0.001
Yakamoz_last_moving_blocks_matrix	0.995	0.333	0.750	0.462	0.498
Yakamoz_penultimate_moving_blocks_matrix	0.985	0.125	0.750	0.214	0.302
Isolation Forest	0.664	0.008	1.000	0.016	0.074
Chen and Liu’s procedure	0.989	0.211	1.000	0.348	0.456

From the above tables, it is seen that the proposed time series anomaly detection methods give better results compared to the other methods. It is seen that Twitter’s method and F-test cannot detect any anomalies, and although the Isolation Forest method detects all the anomalies, it has too many false positives compared to the other methods.

4.2 Simulated Datasets under Different Setups

Time series data of 250 lengths are generated from stationary and non-stationary AR, ARMA and ARCH, GARCH models with different parameters and anomaly/anomalies are created. One anomaly case is simulated by making a random observation higher than the highest observation of the series. In one anomaly case, the anomaly is created by 0.5, 1.0, 1.5, 2.0, 2.5 and 3.0 unit increments of the highest observation of the series. The multiple anomalies case is simulated by making two random observations higher than the highest observation of the series. In multiple anomalies case, the anomalies are created by 1.0 and 1.5 unit increments of the highest observation of the series. The simulation number is set to 1000.

In one anomaly and multiple anomalies cases, the below user-selected values are fixed for the Yakamoz method for each simulation step.

- Number of columns of the 1st MBM: 12
- Number of columns of the 2nd, 3rd, … MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the first MBM): 0.50
- Minimum number of the rows of the 2nd, 3rd, … MBM (of total row length of the second, the third, … MBM): 0.50
- α quantile of the chi-square distribution: 0.05

In one anomaly and multiple anomalies cases, the below user-selected values are fixed for the Sunshine method for each simulation step. In multiple anomalies case, trim number is also chosen as 2 for some models to be able to show how better scores will be achieved for the Sunshine method by adjusting the user-selected values.

- Number of columns of the 1st MBM: 2:20
- Number of columns of the 2nd MBM: 2
- Minimum number of the rows of the 1st MBM (of total row length of the first MBM): 0.50
- Minimum number of the rows of the 2nd MBM (of total row length of the second MBM): 0.50
- α quantile of the chi-square distribution: 0.05
- Trim number=1

In the simulation study, we cannot tune the hyperparameters to be able to obtain better results; however, we can play with some parameters when we analyze a real-life dataset. Comparison results of the proposed two novel time series anomaly detection methods and the best known time series anomaly detection methods using simulated datasets are listed below for only one model type. Comprehensive simulation results for other types of models are included in Appendices.

4.2.1 Confusion Matrix and Evaluation Results for Stationary AR(1) max(data)+0.5_12 Data

Time series data of 250 lengths are generated from the stationary AR model with parameter -0.9 and a random observation, the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 0.5 unit increment of the highest observation of the series. The simulation number is set to 1000.

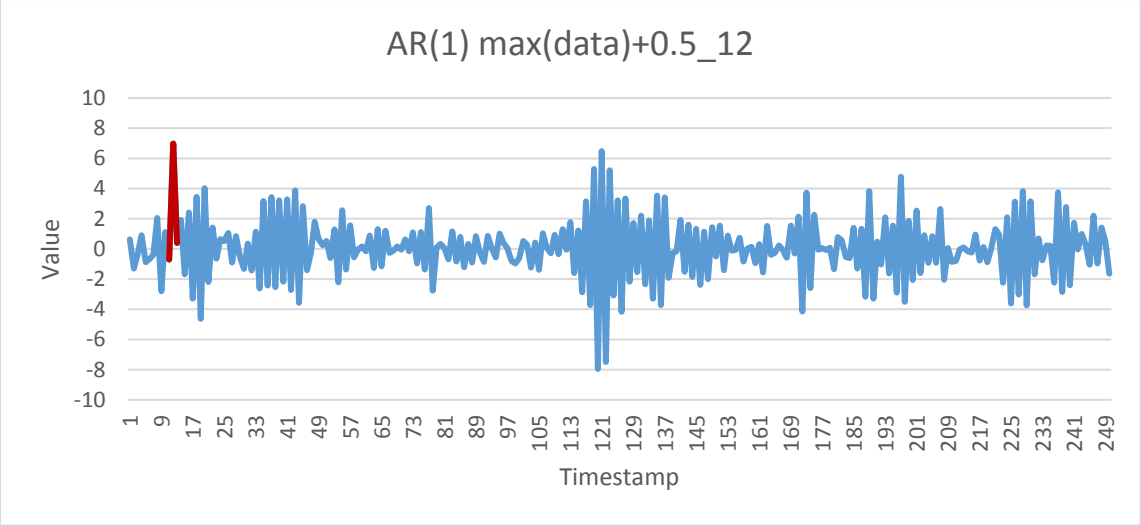

Figure 4.9: Time series plot of the one of the simulated stationary AR(1) max(data)+0.5_12 data

Table 4.17: Confusion matrix for stationary time series AR(1) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.982	0.018	4.526	244.474
Twitter's method	0.098	0.902	0.161	248.839
STL	0.142	0.858	1.486	247.514
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.946	0.054	14.034	234.966
Yakamoz_penultimate_moving_blocks_matrix	0.983	0.017	56.301	192.699
Isolation Forest	1.000	0.000	69.392	179.608

Table 4.18: Evaluation metrics for stationary time series AR(1) max(data)+0.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.982	0.218	0.982	0.343
Twitter's method	0.996	0.057	0.098	0.067
STL	0.991	0.034	0.142	0.048
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.944	0.084	0.946	0.152
Yakamoz_penultimate_moving_blocks_matrix	0.775	0.028	0.983	0.053
Isolation Forest	0.722	0.014	1.000	0.028

From the tables, it is seen that the Sunshine method and the Yakamoz method give better results compared to the other methods. It is seen that F-test can never detect the abnormal observation and Twitter's method and STL can detect the anomaly very few times. From the tables, it is seen that although the Isolation Forest method can detect the anomaly every time, it has too many false positives compared to the other methods. It should be noted that the better scores for the Yakamoz method will be achieved when the user will change the number of columns of the first MBM and reduce FP by using her/his knowledge about the time series data, and the better scores for the Sunshine method will be achieved when the user-selected values are set according to the structure of the data.

4.2.2 Confusion Matrix and Evaluation Results for Stationary AR(1) max(data)+1.0_12 Data

Time series data of 250 lengths are generated from the stationary AR model with parameter -0.9 and a random observation, the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 1.0 unit increment of the highest observation of the series. The simulation number is set to 1000.

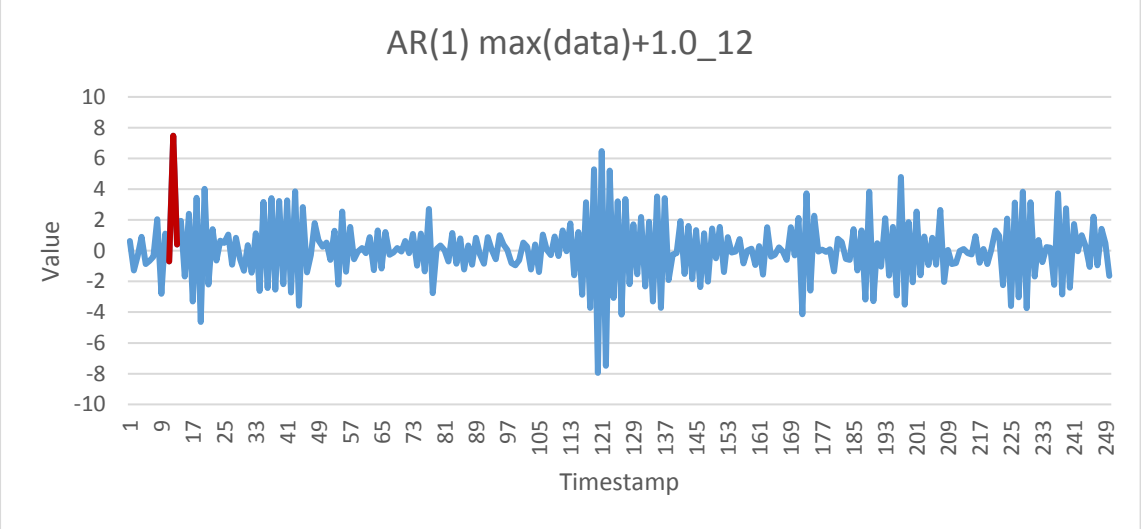

Figure 4.10: Time series plot of the one of the simulated stationary AR(1) max(data)+1.0_12 data

Table 4.19: Confusion matrix for stationary time series AR(1) max(data)+1.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.998	0.002	4.627	244.373
Twitter's method	0.154	0.846	0.149	248.851
STL	0.181	0.819	1.301	247.699
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.964	0.036	13.847	235.153
Yakamoz_penultimate_moving_blocks_matrix	0.994	0.006	56.210	192.790
Isolation Forest	1.000	0.000	68.381	180.619

Table 4.20: Evaluation metrics for stationary time series AR(1) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.981	0.221	0.998	0.347
Twitter's method	0.996	0.110	0.154	0.121
STL	0.992	0.059	0.181	0.076
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.944	0.086	0.964	0.156
Yakamoz_penultimate_moving_blocks_matrix	0.775	0.029	0.994	0.054
Isolation Forest	0.726	0.014	1.000	0.029

From the above tables, it is seen that the Sunshine method is the most successful algorithm in terms of precision and F1 score. It is seen that F-test can never detect the anomalous observation, and Twitter's method and STL can find the anomaly very few times. From the tables, it is seen that although the Isolation Forest method can detect the anomaly every time, it has too many false positives compared to the other methods.

4.2.3 Confusion Matrix and Evaluation Results for Stationary AR(1) max(data)+1.5_12 Data

Time series data of 250 lengths are generated from the stationary AR model with parameter -0.9 and a random observation, the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 1.5 unit increments of the highest observation of the series. The simulation number is set to 1000.

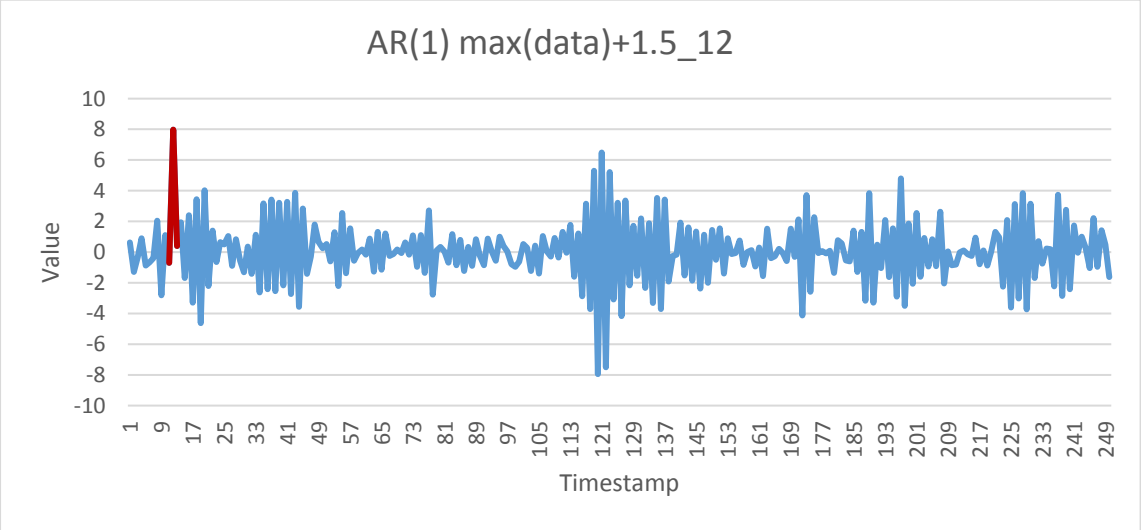

Figure 4.11: Time series plot of the one of the simulated stationary AR(1) max(data)+1.5_12 data

Table 4.21: Confusion matrix for stationary time series AR(1) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.870	244.130
Twitter's method	0.241	0.759	0.105	248.895
STL	0.262	0.738	1.126	247.874
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.978	0.022	13.860	235.140
Yakamoz_penultimate_moving_blocks_matrix	0.997	0.003	56.604	192.396
Isolation Forest	1.000	0.000	67.891	181.109

Table 4.22: Evaluation metrics for stationary time series AR(1) max(data)+1.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.981	0.210	1.000	0.334
Twitter's method	0.997	0.207	0.241	0.216
STL	0.993	0.135	0.262	0.156
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.944	0.088	0.978	0.160
Yakamoz_penultimate_moving_blocks_matrix	0.774	0.028	0.997	0.054
Isolation Forest	0.728	0.015	1.000	0.029

From the above tables, it is seen that the Sunshine method gives better results compared to the other methods in terms of precision, recall and F1 score. It is seen that the Isolation Forest method has too many false positives compared to the other methods, and F-test can never detect the anomaly. The better scores for the Yakamoz method and the Sunshine method will be achieved when the user-selected values are set according to the structure of the data.

From the tables, it is seen that there are very few changes in precision value, recall value and F1 score of the Sunshine method, the Yakamoz method and the Isolation Forest method with the increase in the anomaly from 0.5 unit to 1.5 unit compared to the values of Twitter's method and STL.

4.2.4 Confusion Matrix and Evaluation Results for Stationary AR(1) max(data)+2.0_12 Data

Time series data of 250 lengths are generated from the stationary AR model with parameter -0.9 and a random observation, the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 2.0 unit increments of the highest observation of the series. The simulation number is set to 1000.

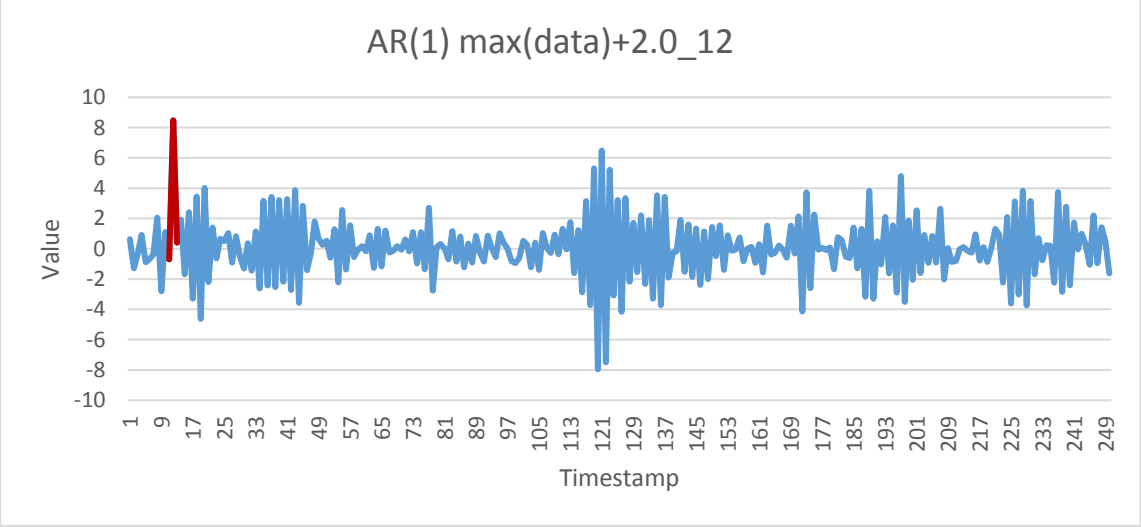

Figure 4.12: Time series plot of the one of the simulated stationary AR(1) max(data)+2.0_12 data

Table 4.23: Confusion matrix for stationary time series AR(1) max(data)+2.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.712	244.288
Twitter's method	0.386	0.614	0.145	248.855
STL	0.399	0.601	1.329	247.671
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.987	0.013	14.166	234.834
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	56.161	192.839
Isolation Forest	1.000	0.000	67.322	181.678

Table 4.24: Evaluation metrics for stationary time series AR(1) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.981	0.219	1.000	0.344
Twitter's method	0.997	0.342	0.386	0.353
STL	0.992	0.236	0.399	0.262
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.943	0.088	0.987	0.159
Yakamoz_penultimate_moving_blocks_matrix	0.775	0.029	0.999	0.055
Isolation Forest	0.731	0.015	1.000	0.029

From the above tables, it is seen that even if precision value and recall value of Twitter's method and STL start to increase when the anomaly becomes visible to the eye, the Sunshine method, the Yakamoz method and the Isolation Forest method have better recall values. It is seen that F-test can never detect the anomalous observation and Twitter's method and STL can detect the anomaly very few times. Better scores for the Yakamoz method and the Sunshine method will be achieved when the user-selected values are set according to the structure of the data.

4.2.5 Confusion Matrix and Evaluation Results for Stationary AR(1) max(data)+2.5_12 Data

Time series data of 250 lengths are generated from the stationary AR model with parameter -0.9 and a random observation, the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 2.5 unit increments of the highest observation of the series. The simulation number is set to 1000.

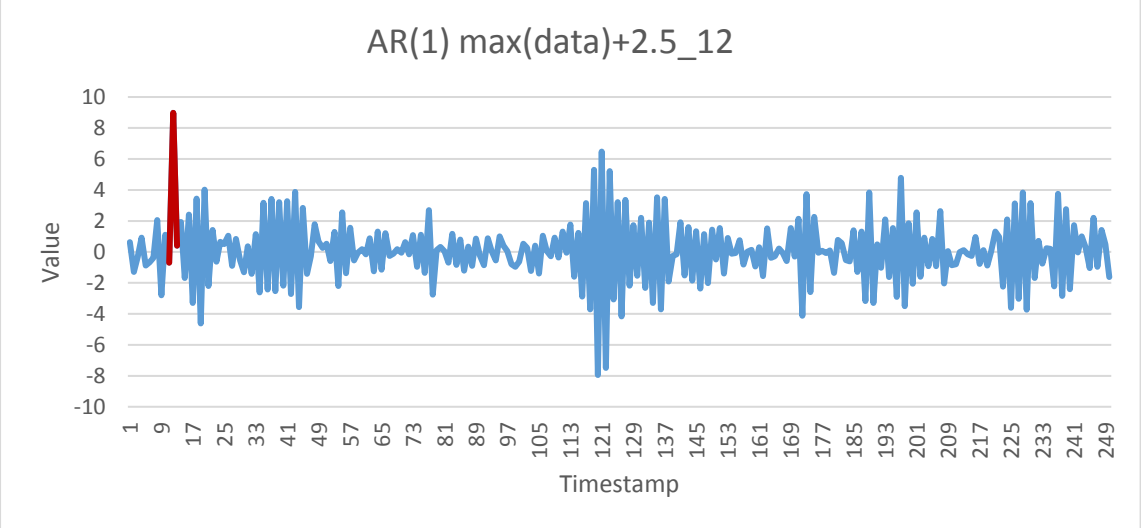

Figure 4.13: Time series plot of the one of the simulated stationary AR(1) max(data)+2.5_12 data

Table 4.25: Confusion matrix for stationary time series AR(1) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.965	244.035
Twitter's method	0.521	0.479	0.146	248.854
STL	0.517	0.483	1.291	247.709
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.989	0.011	14.117	234.883
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	57.392	191.608
Isolation Forest	1.000	0.000	66.671	182.329

Table 4.26: Evaluation metrics for stationary time series AR(1) max(data)+2.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.209	1.000	0.331
Twitter's method	0.998	0.477	0.521	0.488
STL	0.993	0.350	0.517	0.377
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.943	0.089	0.989	0.161
Yakamoz_penultimate_moving_blocks_matrix	0.770	0.028	0.999	0.053
Isolation Forest	0.733	0.015	1.000	0.030

From the above tables, it is seen that even if precision values of Twitter's method and STL are high when the anomaly is visible to the eye, the Sunshine method, the Yakamoz method and the Isolation Forest method have better recall values. From the tables, it is also seen that the Sunshine method and the Yakamoz method have higher precision values than the Isolation Forest method and F-test can never detect the anomaly.

4.2.6 Confusion Matrix and Evaluation Results for Stationary AR(1) max(data)+3.0_12 Data

Time series data of 250 lengths are generated from the stationary AR model with parameter -0.9 and a random observation, the 12th observation, is made higher than the highest observation of the series. The anomaly is created by 3.0 unit increments of the highest observation of the series. The simulation number is set to 1000.

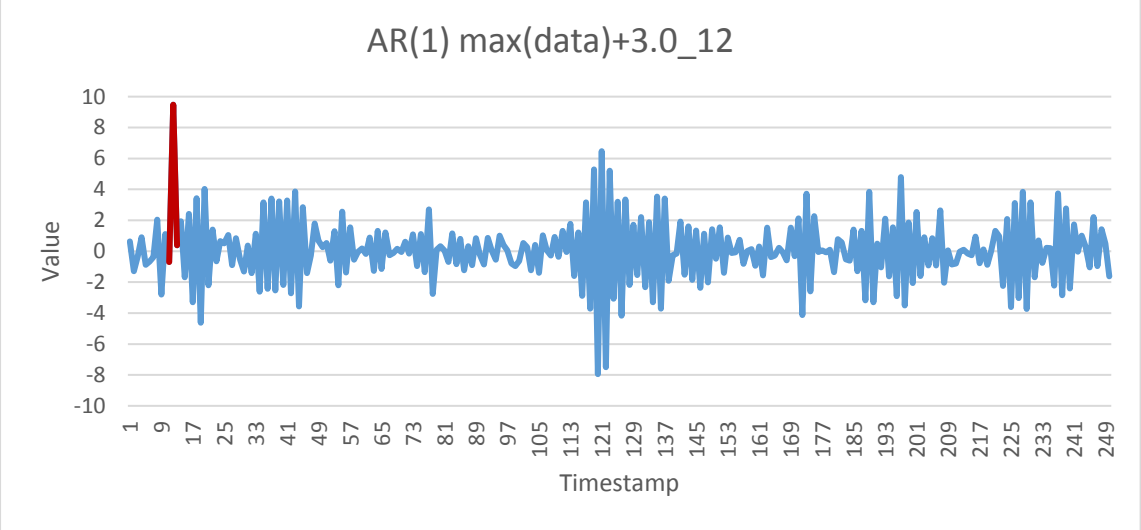

Figure 4.14: Time series plot of the one of the simulated stationary AR(1) max(data)+3.0_12 data

Table 4.27: Confusion matrix for stationary time series AR(1) max(data)+3.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.037	243.963
Twitter's method	0.660	0.340	0.149	248.851
STL	0.629	0.371	1.194	247.806
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.991	0.009	13.729	235.271
Yakamoz_penultimate_moving_blocks_matrix	0.998	0.002	56.972	192.028
Isolation Forest	1.000	0.000	65.944	183.056

Table 4.28: Evaluation metrics for stationary time series AR(1) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.201	1.000	0.323
Twitter's method	0.998	0.616	0.660	0.627
STL	0.994	0.459	0.629	0.487
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.945	0.090	0.991	0.163
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.029	0.998	0.055
Isolation Forest	0.736	0.015	1.000	0.030

From the above tables, it is seen that even if the precision values of Twitter's method and STL are high when the anomaly is visible to the eye, the Sunshine method, the Yakamoz method and the Isolation Forest method have better recall values. It is seen that F-test can never detect the abnormal observation and Twitter's method and STL can detect the anomaly a few times. From the above tables, it is also seen that the Sunshine method and the Yakamoz method have higher precision values than the Isolation Forest method. The better scores for the Yakamoz method and the Sunshine method will be achieved when the user-selected values are set according to the structure of the data.

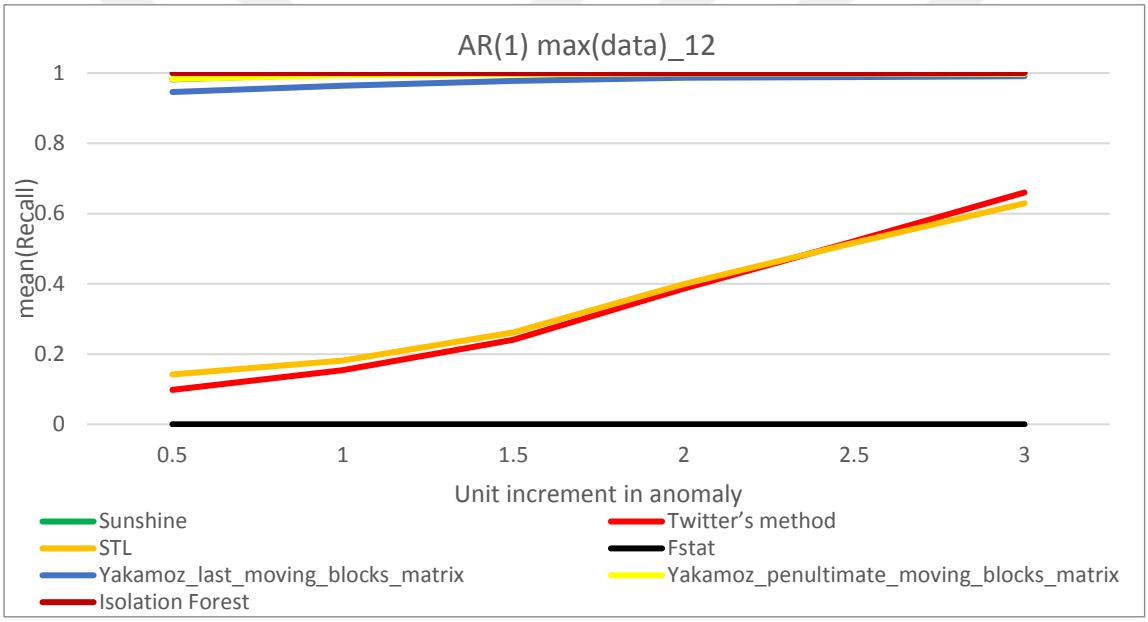

Figure 4.15: Changes in mean(Recall) for stationary AR(1) max(data)_12 data

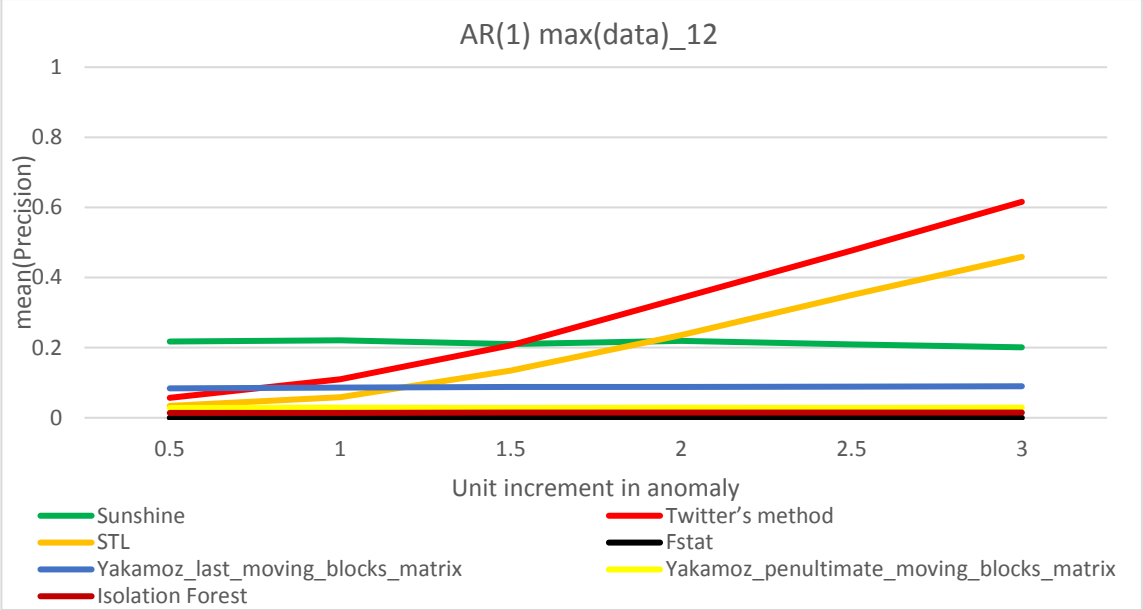

Figure 4.16: Changes in mean(Precision) for stationary AR(1) max(data)_12 data

From the above tables and figures, it is seen that there are very few changes in precision value, recall value and F1 score of the Sunshine method, the Yakamoz method and the Isolation Forest method with the increase in the anomaly from 0.5 unit to 3.0 units. Moreover, it is seen that these values of Twitter's method and STL change a lot with the increase in the anomaly from 0.5 unit to 3.0 units.

When we focus on the other simulation scenerios, we observe the similar results. The performance of the proposed methods is not changing by the model, high variability within data, position of the anomaly or how big the anomaly is. Hence, we can say that the proposed methods are very robust.

CHAPTER 5

CONCLUSION

Two novel time series anomaly detection methods have been proposed. The proposed methods provide assumption-free, unsupervised, efficient and accurate detection of anomalous observations in time series data. These methods do not require anomalous samples in their training datasets. This capability has notable practical value because it can be challenging to collect and label anomalous data.

Success of the proposed methods in finding anomaly/anomalies is investigated using real and simulated time series datasets. From the performance comparison results, it is seen that the proposed methods are very effective in finding anomaly/anomalies in these time series regardless of the variation in the data. From the real datasets, it is seen that the proposed time series anomaly detection methods give better results compared to the other methods for both long and short real time series, especially the Sunshine method.

The proposed methods are widely applicable in detecting anomalies. From the simulation studies, it is seen that the proposed methods are very effective in finding the anomaly regardless of the small or large deviation of the anomaly from normal observations. From the simulation studies, it is also seen that the location of the anomaly does not affect the performance of the methods.

From the simulation studies, it is seen that the Sunshine method and the Yakamoz method have better recall values compared to the recall values of Twitter's method and STL for almost all stationary and nonstationary time series models for both one anomaly and multiple anomalies cases. From the simulation studies, it is also seen that the Sunshine method and the Yakamoz method have higher precision values than the precision values

of the Isolation Forest method for all stationary and nonstationary time series models for both one and multiple anomalies.

It should be noted that the better scores for the Yakamoz method will be achieved when the user changes the number of columns of the first MBM and reduces FP by using her/his knowledge about the time series data and the better scores for the Sunshine method will be achieved when the minimum number of the rows of the first MBM, minimum number of the rows of the second MBM and trim number are set according to the structure of the data.

As the availability and reliability of data become increasingly important, research on anomaly detection is expanding into more complex and diverse application areas. The idea behind the proposed time series anomaly detection methods can be adapted to these application areas in the future.

REFERENCES

Aggarwal, C. C. (2013). *Outlier Analysis. [electronic resource]*. Springer New York.

Aguinis, H., Gottfredson, R. K., & Joo, H. (2013). Best-Practice Recommendations for Defining, Identifying, and Handling Outliers. *Organizational Research Methods*, *16*(2), 270–301. https://doi.org/10.1177/1094428112470848

Akouemo, H. N., & Povinelli, R. J. (2014). Time series outlier detection and imputation. *2014 IEEE PES General Meeting | Conference & Exposition, PES General Meeting | Conference & Exposition, 2014 IEEE*, 1–5. https://doi.org/10.1109/PESGM.2014.6939802

Akosa, J. S. (2017). Predictive Accuracy: A Misleading Performance Measure for Highly Imbalanced Data. Retrieved from http://support.sas.com/resources/papers/proceedings17/0942-2017.pdf

Alghushairy, O., Alsini, R., Soule, T., & Ma, X. (2021). A Review of Local Outlier Factor Algorithms for Outlier Detection in Big Data Streams. *Big Data and Cognitive Computing*, *5*(1), 1. https://doi.org/10.3390/bdcc5010001

Amer, M., Goldstein, M., & Abdennadher, S. (2013). Enhancing one-class support vector machines for unsupervised anomaly detection. *Outlier Detection and Description*, 8–15. https://doi.org/10.1145/2500853.2500857

Andrews, D. W. K. (1993). Tests for Parameter Instability and Structural Change with Unknown Change Point. *Econometrica*, *61*(4), 821–856. https://doi.org/10.2307/2951764

Angiulli, F., & Fassetti, F. (2010). Distance-based outlier queries in data streams: the novel task and algorithms. *Data Mining and Knowledge Discovery*, *20*(2), 290. https://doi.org/10.1007/s10618-009-0159-9

Ben-Gal I. (2005) Outlier Detection. In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA. https://doi.org/10.1007/0-387-25465-X_7

Bl'azquez-Garc'ia, A., Conde, A., Mori, U., & Lozano, J. (2020). *A review on outlier/anomaly detection in time series data*.

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data* (3rd ed.). Wiley.

Basu, S., & Meckesheimer, M. (2007). Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems: An International Journal*, *11*(2), 137. https://doi.org/10.1007/s10115-006-0026-6

Bolton, R. J., & Hand, D. J. (2002). Statistical Fraud Detection: A Review. *Statistical Science*, *17*(3), 235–249.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF : identifying density-based local outliers. *Management of Data*, 93–104. https://doi.org/10.1145/342009.335388

Butler, R. W., Davies, P. L., & Jhun, M. (1993). Asymptotics for the Minimum Covariance Determinant Estimator. *The Annals of Statistics*, *21*(3), 1385–1400.

Cabana, E., Lillo, R. E., & Laniado, H. (2019). *Multivariate outlier detection based on a robust Mahalanobis distance with shrinkage estimators*. https://doi.org/10.1007/s00362-019-01148-1

Carter, K. M., & Streilein, W. W. (2012). Probabilistic reasoning for streaming anomaly detection. *2012 IEEE Statistical Signal Processing Workshop (SSP), Statistical Signal Processing Workshop (SSP), 2012 IEEE*, 377–380. https://doi.org/10.1109/SSP.2012.6319708

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection : A survey. *ACM Computing Surveys (CSUR)*, *41*(3), 1–58. https://doi.org/10.1145/1541880.1541882

Chen, C. and Liu, Lon-Mu (1993). Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, *88*(421), 284–297. https://doi.org/10.2307/2290724

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1), 3. Retrieved from https://search.proquest.com/scholarly-journals/stl-seasonal-trend decomposition-procedure based/docview/1266805989/se-2?accountid=13014

Chow, G. C. (1960). Tests of Equality Between Sets of Coefficients in Two Linear Regressions. *Econometrica*, 28(3), 591–605. https://doi.org/10.2307/1910133

Dani, M., Jollois, F., Nadif, M., & Freixo, C. (2015). Adaptive Threshold for Anomaly Detection Using Time Series Segmentation. *ICONIP*.

Davies, L. (1992). The Asymptotics of Rousseeuw's Minimum Volume Ellipsoid Estimator. *The Annals of Statistics*, *20*(4), 1828–1843.

Draskovic, G., & Pascal, F. (2017). *New insights into the statistical properties of M-estimators*. https://doi.org/10.1109/TSP.2018.2841892

Edgeworth, F. Y. (1887). XLI. On discordant observations. *Philosophical Magazine Series 1*, *23*, 364-375.

Eskin, E. (2000). Anomaly Detection over Noisy Data using Learned Probability Distributions. *ICML*.

Eskin, E., Arnold, A.O., & Prerau, M. (2002). A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data.

Ester, M., Kriegel, H., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *KDD*.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Goldstein, M., & Uchida, S. (2016). A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE*, *11*(4), 1–31. https://doi.org/10.1371/journal.pone.0152173

Hadi, A. S. (1992). Identifying Multiple Outliers in Multivariate Data. *Journal of the Royal Statistical Society. Series B (Methodological)*, *54*(3), 761–771.

Hadi, A. S., Imon, A. H. M. R. and Werner, M. (2009), Detection of outliers. WIREs Comp Stat, 1: 57-70. https://doi.org/10.1002/wics.6

Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques: Vol. 3rd ed*. Morgan Kaufmann.

Hawkins, D. M. (1980). *Identification of outliers*. Chapman and Hall.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, *24*(9–10), 1641–1650. https://doi.org/10.1016/S0167-8655(03)00003-5

Hill, D. J., & Minsker, B. S. (2010). Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. *Environmental Modelling & Software*, *25*(9), 1014–1022. https://doi.org/10.1016/j.envsoft.2009.08.010

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

Hodge, V. J., & Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review: An International Science and Engineering Journal*, *22*(2), 85. https://doi.org/10.1007/s10462-004-4304-y

Hornik, K., Kleiber, C., Leisch, F., & Zeileis, A. (2002). Strucchange: An R Package for Testing for Structural Change in Linear Regression Models. *Journal of Statistical Software*. https://doi.org/10.17877/de290r-12194

Iqbal, M. Z., Habib, S., Khan, M. I., & Kashif, M. (2020). Comparison of Different Techniques for Detection of Outliers in Case of Multivariate Data. P*akistan Journal of Agricultural Sciences*, *57*(3), 865–869. https://doi.org/10.21162/PAKJAS/20.9369

Igenewari, V. R., Skaf, Z., & Jennions, I. (2019). *A Survey of Flight Anomaly Detection Methods: Challenges and Opportunities*.

Ishimtsev, V., Nazarov, I., Bernstein, A., & Burnaev, E. (2017). *Conformal k-NN Anomaly Detector for Univariate Data Streams*.

Jagadish, H. V., Koudas, N., & Muthukrishnan, S. (1999). Mining Deviants in a Time Series Database. *VLDB*.

Jin W., Tung A. K. H., Han J., Wang W. (2006) Ranking Outliers Using Symmetric Neighborhood Relationship. In: Ng WK., Kitsuregawa M., Li J., Chang K. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2006. Lecture Notes in Computer Science, vol 3918. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11731139_68

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, *379*(2194), 20200209. https://doi.org/10.1098/rsta.2020.0209

Liu, F. T., Ting, K. M., & Zhou. Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422. https://doi.org/10.1109/ICDM.2008.17

López-de-Lacalle, J. (2019). tsoutliers R Package for Detection of Outliers in Time Series. Retrieved from https://jalobe.com/doc/tsoutliers.pdf

Mahalakshmi, G., Sridevi, S., & Rajaram, S. (2016). A survey on forecasting of time series data. *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, 1–8. https://doi.org/10.1109/ICCTIDE.2016.7725358

Mehrang, S., Helander, E., Pavel, M., Chieh, A., & Korhonen, I. (2015). Outlier detection in weight time series of connected scales. *2015 IEEE International Conference on Bioinformatics & Biomedicine (BIBM)*, 1489.

Mehrotra, K. G., Mohan, C. K., & Huang, H. (2017). Anomaly Detection Principles and Algorithms. *Terrorism, Security, and Computation*. doi:10.1007/978-3-319-67526-8

Mittal, K., Aggarwal, G., & Mahajan, P. (2019). Performance study of K-nearest neighbor classifier and K-means clustering for predicting the diagnostic accuracy. *International Journal of Information Technology: An Official Journal of Bharati Vidyapeeth's Institute of Computer Applications and Management*, *11*(3), 535. https://doi.org/10.1007/s41870-018-0233-x

Mund, S. (2015). *Microsoft Azure Machine Learning*. Packt Publishing.

Munir, M., Siddiqui, S. A., Dengel, A., & Ahmed, S. (2019). DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*, *7*, 1991–2005. https://doi.org/10.1109/ACCESS.2018.2886457

Muthukrishnan, S., Shah, R., & Vitter, J. S. (2004). Mining deviants in time series data streams. *Proceedings of the International Conference on Scientific and Statistical Database Management, SSDBM*, *16*, 41–50.

Outlier. (n.d.). In Oxford English Dictionary. Retrieved from https://www.oed.com/view/Entry/133735?redirectedFrom=outlier#eid

Penny, K. I., & Jolliffe, I. T. (2001). A Comparison of Multivariate Outlier Detection Methods for Clinical Laboratory Safety Data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, *50*(3), 295–308.

Reddy, A., Ordway-West, M., Lee, M., Dugan, M., Whitney, J., Kahana, R., Ford, B., Muedsam, J., Henslee, A., & Rao, M. (2017). Using Gaussian Mixture Models to Detect Outliers in Seasonal Univariate Network Traffic. *2017 IEEE Security and Privacy Workshops (SPW)*, 229-234.

Roizman, V., Jonckheere, M., & Pascal, F. (2021). Robust clustering and outlier rejection using the Mahalanobis distance distribution. *European Signal Processing Conference (EUSIPCO), 2020 28th*, 2448–2452. https://doi.org/10.23919/Eusipco47968.2020.9287356

Rousseeuw, P. (1985). Multivariate Estimation with High Breakdown Point. *Mathematical Statistics and Applications Vol. B.* 283-297. 10.1007/978-94-009-5438-0_20.

Rousseeuw, P. J., & Hubert, M. (2017). *Anomaly Detection by Robust Statistics*. https://doi.org/10.1002/widm.1236

Rousseeuw, P. J., & Leroy, A. M. (1987). *Robust regression and outlier detection*. Wiley.

Rousseeuw, P. J., & van Driessen, K. (1999). A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics*, *41*(3), 212. https://doi.org/10.1080/00401706.1999.10485670

Rousseeuw, P. J., & van Zomeren, B.C. (1990). Unmasking Multivariate Outliers and Leverage Points. *Journal of the American Statistical Association, 85*, 633-639.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the Support of a High-Dimensional Distribution. *Neural Computation*, *13*(7), 1443–1471. https://doi.org/10.1162/089976601750264965

Steinley, D. (2006). K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, *59*(1), 1–34. https://doi.org/10.1348/000711005X48266

Tang, J., Chen, Z., Fu, A., & Cheung, D. (2002). Enhancing Effectiveness of Outlier Detections for Low Density Patterns. *PAKDD*.

Vallis, O., Hochenbaum, J., & Kejariwal, A. (2014). A Novel Technique for Long-Term Anomaly Detection in the Cloud. *HotCloud*.

Vishwakarma, G. K., Paul, C., & Elsawah, A. M. (2020). An algorithm for outlier detection in a time series model using backpropagation neural network. *Journal of King Saud University - Science*, *32*(8), 3328–3336. https://doi.org/10.1016/j.jksus.2020.09.018

Wang, H., Bah, M. J., & Hammad, M. (2019). Progress in Outlier Detection Techniques: A Survey. *IEEE ACCESS*, *7*, 107964–108000. https://doi.org/10.1109/ACCESS.2019.2932769

Wei, W. W. S. (2006). *Time series analysis : univariate and multivariate methods* (2nd ed.). Pearson Addison Wesley.

Yahoo! Webscope dataset ydata-labeled-time-series-anomalies-v1_0 [https://webscope.sandbox.yahoo.com/]

Zhou, Y., Qin, R., Xu, H., Sadiq, S., & Yu, Y. (2018). A Data Quality Control Method for Seafloor Observatories: The Application of Observed Time Series Data in the East China Sea. *SENSORS*, *18*(8). https://doi.org/10.3390/s18082628

APPENDICES

A. R CODE

A.1 R Code for the Yakamoz Method

```
yakamoz <- function(data, lngth_1, q1, q2, alpha)
{
  #install.packages("MASS") #For Minimum Covariance Determinant
  library(MASS)
  #install.packages("qpcR") #To construct a matrix of observations that
pass Moving Blocks Matrix
  library(qpcR)

  lngth = 2 #Length of the second, third... Moving Blocks Matrix

  obs_passed_moving_blocks_matrix_with_na = c() #Observations passed Mo
ving Blocks Matrices are in columns of the matrix. NAs are also in colu
mns to create the matrix


  mbb_1 = matrix(,
                 nrow = length(data) - lngth_1 + 1,
                 ncol = lngth_1,
                 byrow = TRUE) #Moving blocks in each row
  i = 1
  while (i <= length(data) - lngth_1 + 1)
  {
    mbb_1[i, ] = data[seq(i, length.out = lngth_1)]
    i = i + 1
  }

  ###### Minimum Covariance Determinant ######

  MCD_sample_1 = cov.mcd(mbb_1, quantile.used = nrow(mbb_1) * q1) #Usin
g this sub-sample mean and covariance will be calculated

  D1_MCD_Mahalanobis = mahalanobis(mbb_1, MCD_sample_1$center, MCD_samp
le_1$cov) #Mahalanobis distance using MCD estimates of mean and covaria
nce

  cutoff_1 = (qchisq(p = 1 - alpha, df = ncol(mbb_1)))

  Ano_MCD_1 = which(D1_MCD_Mahalanobis > cutoff_1) #Gives rows which pa
ssed the first cutoff
```

```
  ################################## Observations passed the first Movi
ng Blocks Matrix ##############################

  if (length(Ano_MCD_1) == 0)
  {
    print("Ano_MCD_1 No row passed cutoff_1! Second Moving Blocks Matri
x cannot be created")
  } else if (length(Ano_MCD_1) == 1)
  {
    ndata = mbb_1[Ano_MCD_1[1], ]
    print(
      "Ano_MCD_1 Only one row passed the first cutoff! Second Moving Bl
ocks Matrix is not required"
    )
    return(ndata)
  } else
    #Observations which passed the first cutoff
  {
    ndata = mbb_1[Ano_MCD_1[1], ]

    for (i in 1:(length(Ano_MCD_1) - 1))
    {
      if (Ano_MCD_1[i + 1] - Ano_MCD_1[i] < lngth_1)
      {
        ndata = append(ndata, mbb_1[Ano_MCD_1[i + 1], seq(from = (lngth
_1 - (Ano_MCD_1[i +

1] - Ano_MCD_1[i]) + 1), to = lngth_1)])
      } else
      {
        ndata = append(ndata, mbb_1[Ano_MCD_1[i + 1], ])
      }
    }

  }

  options(scipen = 999) #To prevent scientific notation

  try({
    for (movingblocksmatrixnumber in 2:7)
    {
      ###### Minimum Covariance Determinant for the Second Moving Block
s Matrix ######

      mbb_2 = matrix(
        ,
        nrow = length(ndata) - lngth + 1,
        ncol = lngth,
        byrow = TRUE
      ) #Moving blocks samples in each row
      i = 1
      while (i <= length(ndata) - lngth + 1)
      {
        mbb_2[i, ] = ndata[seq(i, length.out = lngth)]
        i = i + 1
      }
```

```
      MCD_sample_2 = cov.mcd(mbb_2, quantile.used = nrow(mbb_2) * q2) #
Using this sub-sample mean and covariance will be calculated

      D2_MCD_Mahalanobis = mahalanobis(mbb_2, MCD_sample_2$center, MCD_
sample_2$cov) #Mahalanobis distance using MCD estimates of mean and cov
ariance

      ##########################################

      cutoff_2 = (qchisq(p = 1 - alpha, df = ncol(mbb_2)))

      Ano_MCD_2 = which(D2_MCD_Mahalanobis > cutoff_2) #Gives rows whic
h passed the second cutoff

      ###### Observations passed the second,... Moving Blocks Matrix ##
####

      if (length(Ano_MCD_2) == 0)
      {
        obs_passed_cutoff_2 = NA
      } else if (length(Ano_MCD_2) == 1)
      {
        obs_passed_cutoff_2 = mbb_2[Ano_MCD_2[1], ]
      } else
      {
        obs_passed_cutoff_2 = NA
        obs_passed_cutoff_2 = mbb_2[Ano_MCD_2[1], ]
        for (i in 1:(length(Ano_MCD_2) - 1))
        {
          if (Ano_MCD_2[i + 1] - Ano_MCD_2[i] < lngth)
          {
            obs_passed_cutoff_2 = append(obs_passed_cutoff_2, mbb_2[Ano
_MCD_2[i + 1], seq(from = (lngth -

(Ano_MCD_2[i + 1] - Ano_MCD_2[i]) + 1), to = lngth)])
          } else
          {
            obs_passed_cutoff_2 = append(obs_passed_cutoff_2, mbb_2[Ano
_MCD_2[i + 1], ])
          }
        }
      }

      obs_passed_moving_blocks_matrix_with_na = qpcR:::cbind.na(obs_pas
sed_moving_blocks_matrix_with_na, ndata)

      ndata = obs_passed_cutoff_2

    } #movingblocksmatrixnumber

    obs_passed_moving_blocks_matrix_with_na = qpcR:::cbind.na(obs_passe
d_moving_blocks_matrix_with_na, ndata) #Observations passed Moving Bloc
ks Matrices are in columns of the matrix. NAs are also in columns to cr
eate the matrix
```

```
  }, silent = TRUE)
  #try

  obs_passed_moving_blocks_matrix_with_na_without_first_column = obs_pa
ssed_moving_blocks_matrix_with_na[,-1] #Remove the first column contain
ing NAs. Observations passed Moving Blocks Matrices are in columns of t
he matrix. NAs are also in columns to create the matrix

  if (is.null(dim(
    obs_passed_moving_blocks_matrix_with_na_without_first_column
  )))
  {
    obs_passed_last_moving_blocks_matrix = obs_passed_moving_blocks_mat
rix_with_na[,-1]

    obs_passed_penultimate_moving_blocks_matrix = obs_passed_moving_blo
cks_matrix_with_na[,-1]
  } else
  {
    obs_passed_last_moving_blocks_matrix = obs_passed_moving_blocks_mat
rix_with_na_without_first_column[, ncol(obs_passed_moving_blocks_matrix
_with_na_without_first_column)][!is.na(obs_passed_moving_blocks_matrix_
with_na_without_first_column[, ncol(obs_passed_moving_blocks_matrix_wit
h_na_without_first_column)])] #Observations passed the last Moving Bloc
ks Matrix

    obs_passed_penultimate_moving_blocks_matrix = obs_passed_moving_blo
cks_matrix_with_na_without_first_column[, ncol(obs_passed_moving_blocks
_matrix_with_na_without_first_column) -

1][!is.na(obs_passed_moving_blocks_matrix_with_na_without_first_column[
, ncol(obs_passed_moving_blocks_matrix_with_na_without_first_column) -

1])] #Observations passed the penultimate Moving Blocks Matrix
  }
  obs_passed_moving_blocks_matrix_with_na

  obs_passed_moving_blocks_matrix_with_na_without_first_column

  obs_passed_last_moving_blocks_matrix

  obs_passed_penultimate_moving_blocks_matrix

  return(
    list(
      obs_passed_last_moving_blocks_matrix,
      obs_passed_penultimate_moving_blocks_matrix,
      obs_passed_moving_blocks_matrix_with_na_without_first_column
    )
  )

} #yakamoz

#ykmz_res=yakamoz(data, 12, 0.5, 0.5, 0.05)
```

```r
#ykmz_res[[1]]
#ykmz_res[[2]]
```

A.2 R Code for the Sunshine Method

```r
sunshine <- function(data,
                     lngth_1s,
                     q1,
                     q2,
                     alpha,
                     trim_alpha_or_number_1)
{
  #install.packages("DDoutlier") #For knn
  library(DDoutlier)
  #install.packages("dbscan")
  library(dbscan)

  #install.packages("DescTools") #For Trim function
  library(DescTools)

  #install.packages("MASS") #For Minimum Covariance Determinant
  library(MASS)

  lngth_2 = 2 #Length of the second Moving Blocks Matrix

  all_obs_passed_cutoff_2 = c()
  all_obs_passed_last_step = c()
  all_not_enough_neighbors = c()
  all_change_trim_alpha_or_number_1 = c()
  obs_passed_cutoff_2 = c()


  for (lngth_1 in 2:lngth_1s)
  {
    try({
      not_enough_neighbors = 0

      change_trim_alpha_or_number_1 = 0

      obs_passed_cutoff_2 = c()

      mbb_1 = matrix(
        ,
        nrow = length(data) - lngth_1 + 1,
        ncol = lngth_1,
        byrow = TRUE
      ) #Moving blocks in each row
      i = 1
      while (i <= length(data) - lngth_1 + 1)
      {
        mbb_1[i, ] = data[seq(i, length.out = lngth_1)]
```

```
        i = i + 1
      }

      ###### Minimum Covariance Determinant ######

      MCD_sample_1 = cov.mcd(mbb_1, quantile.used = nrow(mbb_1) * q1) #
Using this sub-sample mean and covariance will be calculated
      D1_MCD_Mahalanobis = mahalanobis(mbb_1, MCD_sample_1$center, MCD_
sample_1$cov) #Mahalanobis distance using MCD estimates of mean and cov
ariance

      cutoff_1 = (qchisq(p = 1 - alpha, df = ncol(mbb_1)))
      Ano_MCD_1 = which(D1_MCD_Mahalanobis > cutoff_1) #Gives rows whic
h passed cutoff_1

      ################################ Second Moving Blocks Matrix ##
############################

      if (length(Ano_MCD_1) == 0)
      {
        obs_passed_cutoff_2 = NA
        print("Ano_MCD_1 No row passed cutoff_1! Second Moving Blocks M
atrix cannot be created")
      } else if (length(Ano_MCD_1) == 1)
      {
        obs_passed_cutoff_2 = mbb_1[Ano_MCD_1[1], ]
        print(
          "Ano_MCD_1 Only one row passed cutoff_1! Second Moving Blocks
Matrix is not required"
        )
      } else
        #Observations which passed cutoff_1
      {
        ndata_1 = mbb_1[Ano_MCD_1[1], ]

        for (i in 1:(length(Ano_MCD_1) - 1))
        {
          if (Ano_MCD_1[i + 1] - Ano_MCD_1[i] < lngth_1)
          {
            ndata_1 = append(ndata_1, mbb_1[Ano_MCD_1[i + 1], seq(from
= (lngth_1 -

(Ano_MCD_1[i + 1] - Ano_MCD_1[i]) + 1), to = lngth_1)])
          } else
          {
            ndata_1 = append(ndata_1, mbb_1[Ano_MCD_1[i + 1], ])
          }
        }

        mbb_2 = matrix(
          ,
          nrow = length(ndata_1) - lngth_2 + 1,
          ncol = lngth_2,
          byrow = TRUE
        ) #Moving blocks samples in each row
        i = 1
```

```
        while (i <= length(ndata_1) - lngth_2 + 1)
        {
          mbb_2[i, ] = ndata_1[seq(i, length.out = lngth_2)]
          i = i + 1
        }

        ###### Minimum Covariance Determinant for the second Moving Blo
cks Matrix ######

        MCD_sample_2 = cov.mcd(mbb_2, quantile.used = nrow(mbb_2) * q2)
#Using this sub-sample mean and covariance will be calculated
        D2_MCD_Mahalanobis = mahalanobis(mbb_2, MCD_sample_2$center, MC
D_sample_2$cov) #Mahalanobis distance using MCD estimates of mean and c
ovariance

        cutoff_2 = (qchisq(p = 1 - alpha, df = ncol(mbb_2)))

        Ano_MCD_2 = which(D2_MCD_Mahalanobis > cutoff_2) #Gives rows wh
ich passed the second cutoff

        if (length(Ano_MCD_2) == 0)
          #Observations which passed cutoff_2
        {
          obs_passed_cutoff_2 = NA
        } else if (length(Ano_MCD_2) == 1)
        {
          obs_passed_cutoff_2 = mbb_2[Ano_MCD_2[1], ]
        } else
        {
          obs_passed_cutoff_2 = mbb_2[Ano_MCD_2[1], ]
          for (i in 1:(length(Ano_MCD_2) - 1))
          {
            if (Ano_MCD_2[i + 1] - Ano_MCD_2[i] < lngth_2)
            {
              obs_passed_cutoff_2 = append(obs_passed_cutoff_2, mbb_2[A
no_MCD_2[i + 1], seq(from =

(lngth_2 - (Ano_MCD_2[i + 1] - Ano_MCD_2[i]) + 1), to = lngth_2)])
            } else
            {
              obs_passed_cutoff_2 = append(obs_passed_cutoff_2, mbb_2[A
no_MCD_2[i + 1], ])
            }
          }
        }
      } #else Observations which passed cutoff_1

      ###################### knn ##################

      obs_passed_last_step = NA

      if (length(obs_passed_cutoff_2) <= 3)
      {
        obs_passed_last_step = obs_passed_cutoff_2
        not_enough_neighbors = not_enough_neighbors + 1
        print("Not enough neighbors for knn!")
```

```
      } else
        #elseopen
      {
        inner_1 = Trim(obs_passed_cutoff_2, trim = trim_alpha_or_number
_1)

        if (length(is.na(inner_1)[!is.na(inner_1)]) <= 1)
        {
          change_trim_alpha_or_number_1 = change_trim_alpha_or_number_1
+ 1
          print("Not enough neighbors for knn! Change trim_alpha_or_num
ber_1!")
        }
        scores_knn_inner_1 = KNN_SUM(dataset = inner_1, k = sqrt(length
(inner_1)))
        obs_passed_last_step = obs_passed_cutoff_2[which(
          KNN_SUM(dataset = obs_passed_cutoff_2, sqrt(length(
            obs_passed_cutoff_2
          ))) > sort(scores_knn_inner_1, decreasing = TRUE)[1]
        )]
        if (length(obs_passed_last_step) == 0)
        {
          obs_passed_last_step = NA
        }

      } #elseclose

      all_obs_passed_cutoff_2 = c(all_obs_passed_cutoff_2, obs_passed_c
utoff_2)
      all_obs_passed_cutoff_2 = c(all_obs_passed_cutoff_2, "-")

      all_obs_passed_last_step = c(all_obs_passed_last_step, obs_passed
_last_step)
      all_obs_passed_last_step = c(all_obs_passed_last_step, "-")

      all_not_enough_neighbors = c(all_not_enough_neighbors, not_enough
_neighbors)
      all_not_enough_neighbors = c(all_not_enough_neighbors, "-")

      all_change_trim_alpha_or_number_1 = c(all_change_trim_alpha_or_nu
mber_1,
                                            change_trim_alpha_or_number
_1)
      all_change_trim_alpha_or_number_1 = c(all_change_trim_alpha_or_nu
mber_1, "-")

    }, silent = TRUE)
    #try

  } #for

  #all_obs_passed_cutoff_2
  #all_obs_passed_last_step
  #all_not_enough_neighbors
  #all_change_trim_alpha_or_number_1
```

```
  table(all_obs_passed_cutoff_2)
  table(all_obs_passed_last_step)

  table(all_not_enough_neighbors)
  table(all_change_trim_alpha_or_number_1)

  all_obs_passed_last_step = all_obs_passed_last_step[!all_obs_passed_l
ast_step %in%
                                                      "-"] #To remove
"-"s

  return(list(table(all_obs_passed_last_step)[which(table(all_obs_passe
d_last_step) !=
                                                    1)]))

} #sunshine
```

A.3 R Code for the Simulation Study

```
#install.packages("DDoutlier") #For knn
library(DDoutlier)
#install.packages("dbscan")
library(dbscan)
#install.packages("DescTools") #For Trim function
library(DescTools)

#install.packages("MASS") #For Minimum Covariance Determinant
library(MASS)

#install.packages("devtools") #Twitter
library(devtools)

#devtools::install_github("twitter/AnomalyDetection")
library(AnomalyDetection)

#install.packages("tibbletime") #Anomalize
#install.packages("anomalize")
library(tibbletime)
library(anomalize)

#install.packages("strucchange") #strucchange
library(strucchange)

#install.packages("TSA") #garch.sim
library(TSA)

Tru_positives_sunshine = c()
Fals_negatives_sunshine = c()
Fals_positives_sunshine = c()
Tru_negatives_sunshine = c()
```

```
Accuracy_sunshine = c()
Precision_sunshine = c()
Recall_sunshine = c()
F1_score_sunshine = c()

Tru_positives_twitter = c()
Fals_negatives_twitter = c()
Fals_positives_twitter = c()
Tru_negatives_twitter = c()
Accuracy_twitter = c()
Precision_twitter = c()
Recall_twitter = c()
F1_score_twitter = c()

Tru_positives_anomalize = c()
Fals_negatives_anomalize = c()
Fals_positives_anomalize = c()
Tru_negatives_anomalize = c()

Accuracy_anomalize = c()
Precision_anomalize = c()
Recall_anomalize = c()
F1_score_anomalize = c()

Tru_positives_strucchange = c()
Fals_negatives_strucchange = c()
Fals_positives_strucchange = c()
Tru_negatives_strucchange = c()

Accuracy_strucchange = c()
Precision_strucchange = c()
Recall_strucchange = c()
F1_score_strucchange = c()

Tru_positives_yakamoz_last_moving_blocks_matrix = c()
Fals_negatives_yakamoz_last_moving_blocks_matrix = c()
Fals_positives_yakamoz_last_moving_blocks_matrix = c()
Tru_negatives_yakamoz_last_moving_blocks_matrix = c()

Accuracy_yakamoz_last_moving_blocks_matrix = c()
Precision_yakamoz_last_moving_blocks_matrix = c()
Recall_yakamoz_last_moving_blocks_matrix = c()
F1_score_yakamoz_last_moving_blocks_matrix = c()

Tru_positives_yakamoz_penultimate_moving_blocks_matrix = c()
Fals_negatives_yakamoz_penultimate_moving_blocks_matrix = c()
Fals_positives_yakamoz_penultimate_moving_blocks_matrix = c()
Tru_negatives_yakamoz_penultimate_moving_blocks_matrix = c()

Accuracy_yakamoz_penultimate_moving_blocks_matrix = c()
Precision_yakamoz_penultimate_moving_blocks_matrix = c()
Recall_yakamoz_penultimate_moving_blocks_matrix = c()
F1_score_yakamoz_penultimate_moving_blocks_matrix = c()

mat_data = c() #iForest
```

```
for (simulation in 1:1000)
{
  ################## AR(1) ################

  T = 250
  p = -0.9
  data = arima.sim(model = list(order = c(1, 0, 0), ar = p), n = T)
  #data = arima.sim(model = list(order = c(1, 1, 0), ar = p), n = T)

  ################## AR(2) ################

  #T = 250
  #p = c(-0.5, 0.2)
  #data = arima.sim(model = list(order = c(2, 0, 0), ar = p), n = T)
  #data = arima.sim(model = list(order = c(2, 1, 0), ar = p), n = T)

  ################## ARMA(1,1) ############

  #T = 250
  #p1 = -0.9
  #p2 = 0.4
  #data = arima.sim(model = list(
  #order = c(1, 0, 1),
  #ar = p1,
  #ma = p2
  #), n = T)
  #data = arima.sim(model = list(
  #order = c(1, 1, 1),
  #ar = p1,
  #ma = p2
  #), n = T)

  ################## ARCH(2) ############

  #T = 250
  #alpa = c(0.9, 0.01)
  #data = garch.sim(alpha = alpa, n = T)

  ################## GARCH(1,2) ############

  #T = 250
  #alpa = c(0.9, 0.1)
  #bet = c(-0.09)
  #data = garch.sim(alpha = alpa, beta = bet, n = T)

  anomaly_place1 = 12
  #anomaly_place1 = 250
  data[anomaly_place1] = max(data) + 0.5
  #data[anomaly_place1] = max(data) + 1.0
  #data[anomaly_place1] = max(data) + 1.5
  #data[anomaly_place1] = max(data) + 2.0
  #data[anomaly_place1] = max(data) + 2.5
  #data[anomaly_place1] = max(data) + 3.0

  mat_data = cbind(mat_data, data) #iForest
```

```
  anomaly_vector = c(data[anomaly_place1]) #sunshine, twitter, yakamoz
  anomaly_place_vector = c(anomaly_place1) #anomalize, iForest, strucch
ange


  ###################### sunshine ################

  #lngth_1 Length of the first Moving Blocks Matrix
  lngth_2 = 2 #Length of the second Moving Blocks Matrix
  q1 = 0.5 #quantile.used=nrow(mbb_1)*q1 for the first Moving Blocks Ma
trix
  q2 = 0.5 #quantile.used=nrow(mbb_2)*q2 for the second Moving Blocks M
atrix


  alpha = 0.05 #To calculate the cutoff point of mahalanobis distances

  trim_alpha_or_number_1 = 1 #Decide trim number

  all_obs_passed_cutoff_2 = c()
  all_obs_passed_last_step = c()
  all_not_enough_neighbors = c()
  all_change_trim_alpha_or_number_1 = c()
  obs_passed_cutoff_2 = c()


  for (lngth_1 in 2:20)
  {
    try({
      not_enough_neighbors = 0

      change_trim_alpha_or_number_1 = 0

      obs_passed_cutoff_2 = c()

      mbb_1 = matrix(
        ,
        nrow = length(data) - lngth_1 + 1,
        ncol = lngth_1,
        byrow = TRUE
      ) #Moving blocks in each row
      i = 1
      while (i <= length(data) - lngth_1 + 1)
      {
        mbb_1[i,] = data[seq(i, length.out = lngth_1)]
        i = i + 1
      }

      ###### Minimum Covariance Determinant ######

      MCD_sample_1 = cov.mcd(mbb_1, quantile.used = nrow(mbb_1) * q1) #
Using this sub-sample mean and covariance will be calculated
      D1_MCD_Mahalanobis = mahalanobis(mbb_1, MCD_sample_1$center, MCD_
sample_1$cov) #Mahalanobis distance using MCD estimates of mean and cov
ariance
```

```
      cutoff_1 = (qchisq(p = 1 - alpha, df = ncol(mbb_1)))
      Ano_MCD_1 = which(D1_MCD_Mahalanobis > cutoff_1) #Gives rows whic
h passed cutoff_1

      ################# Second Moving Blocks Matrix ################

      if (length(Ano_MCD_1) == 0)
      {
        obs_passed_cutoff_2 = NA
        print("Ano_MCD_1 No row passed cutoff_1! Second Moving Blocks M
atrix cannot be created")
      } else if (length(Ano_MCD_1) == 1)
      {
        obs_passed_cutoff_2 = mbb_1[Ano_MCD_1[1],]
        print(
          "Ano_MCD_1 Only one row passed cutoff_1! Second Moving Blocks
Matrix is not required"
        )
      } else
        #Observations which passed cutoff_1
      {
        ndata_1 = mbb_1[Ano_MCD_1[1],]

        for (i in 1:(length(Ano_MCD_1) - 1))
        {
          if (Ano_MCD_1[i + 1] - Ano_MCD_1[i] < lngth_1)
          {
            ndata_1 = append(ndata_1, mbb_1[Ano_MCD_1[i + 1], seq(from
= (lngth_1 -

(Ano_MCD_1[i + 1] - Ano_MCD_1[i]) + 1), to = lngth_1)])
          } else
          {
            ndata_1 = append(ndata_1, mbb_1[Ano_MCD_1[i + 1],])
          }
        }

        mbb_2 = matrix(
          ,
          nrow = length(ndata_1) - lngth_2 + 1,
          ncol = lngth_2,
          byrow = TRUE
        ) #Moving blocks samples in each row
        i = 1
        while (i <= length(ndata_1) - lngth_2 + 1)
        {
          mbb_2[i,] = ndata_1[seq(i, length.out = lngth_2)]
          i = i + 1
        }

        ###### Minimum Covariance Determinant for the second Moving Blo
cks Matrix ######

        MCD_sample_2 = cov.mcd(mbb_2, quantile.used = nrow(mbb_2) * q2)
#Using this sub-sample mean and covariance will be calculated
        D2_MCD_Mahalanobis = mahalanobis(mbb_2, MCD_sample_2$center, MC
```

```
D_sample_2$cov) #Mahalanobis distance using MCD estimates of mean and c
ovariance

        ########################################

        cutoff_2 = (qchisq(p = 1 - alpha, df = ncol(mbb_2)))

        Ano_MCD_2 = which(D2_MCD_Mahalanobis > cutoff_2) #Gives rows wh
ich passed the second cutoff

        if (length(Ano_MCD_2) == 0)
          #Observations which passed cutoff_2
        {
          obs_passed_cutoff_2 = NA
        } else if (length(Ano_MCD_2) == 1)
        {
          obs_passed_cutoff_2 = mbb_2[Ano_MCD_2[1],]
        } else
        {
          obs_passed_cutoff_2 = mbb_2[Ano_MCD_2[1],]
          for (i in 1:(length(Ano_MCD_2) - 1))
          {
            if (Ano_MCD_2[i + 1] - Ano_MCD_2[i] < lngth_2)
            {
              obs_passed_cutoff_2 = append(obs_passed_cutoff_2, mbb_2[A
no_MCD_2[i + 1], seq(from =

(lngth_2 - (Ano_MCD_2[i + 1] - Ano_MCD_2[i]) + 1), to = lngth_2)])
            } else
            {
              obs_passed_cutoff_2 = append(obs_passed_cutoff_2, mbb_2[A
no_MCD_2[i + 1],])
            }
          }
        }
      } #else Observations which passed cutoff_1

      ###################### knn ##################

      obs_passed_last_step = NA

      if (length(obs_passed_cutoff_2) <= 3)
      {
        obs_passed_last_step = obs_passed_cutoff_2
        not_enough_neighbors = not_enough_neighbors + 1
        print("Not enough neighbors for knn!")

      } else
        #elseopen
      {
        inner_1 = Trim(obs_passed_cutoff_2, trim = trim_alpha_or_number
_1)

        if (length(is.na(inner_1)[!is.na(inner_1)]) <= 1)
        {
          change_trim_alpha_or_number_1 = change_trim_alpha_or_number_1
```

```
+ 1
          print("Not enough neighbors for knn! Change trim_alpha_or_num
ber_1!")

        }
        scores_knn_inner_1 = KNN_SUM(dataset = inner_1, k = sqrt(length
(inner_1)))
        obs_passed_last_step = obs_passed_cutoff_2[which(
          KNN_SUM(dataset = obs_passed_cutoff_2, sqrt(length(
            obs_passed_cutoff_2
          ))) > sort(scores_knn_inner_1, decreasing = TRUE)[1]
        )]
        if (length(obs_passed_last_step) == 0)
        {
          obs_passed_last_step = NA
        }

      } #elseclose

      all_obs_passed_cutoff_2 = c(all_obs_passed_cutoff_2, obs_passed_c
utoff_2)
      all_obs_passed_cutoff_2 = c(all_obs_passed_cutoff_2, "-")

      all_obs_passed_last_step = c(all_obs_passed_last_step, obs_passed
_last_step)
      all_obs_passed_last_step = c(all_obs_passed_last_step, "-")

      all_not_enough_neighbors = c(all_not_enough_neighbors, not_enough
_neighbors)
      all_not_enough_neighbors = c(all_not_enough_neighbors, "-")

      all_change_trim_alpha_or_number_1 = c(all_change_trim_alpha_or_nu
mber_1,
                                            change_trim_alpha_or_number
_1)
      all_change_trim_alpha_or_number_1 = c(all_change_trim_alpha_or_nu
mber_1, "-")

    }, silent = TRUE)
    #try

  } #for

  all_obs_passed_last_step = all_obs_passed_last_step[!all_obs_passed_l
ast_step %in%
                                                      "-"]

  Tru_positives_sunshine[simulation] = length(which(is.element(anomaly_
vector, names(
    table(all_obs_passed_last_step)[which(table(all_obs_passed_last_ste
p) !=
                                            1)]
  )) == TRUE)) #For anomalies-more than 1 observed values
  Fals_negatives_sunshine[simulation] = length(which(is.element(anomaly
_vector, names(
    table(all_obs_passed_last_step)[which(table(all_obs_passed_last_ste
```

```
p) !=
                                                    1)]
  )) == FALSE)) #For anomalies-more than 1 observed values
  Fals_positives_sunshine[simulation] = length(which(is.element(names(
    table(all_obs_passed_last_step)[which(table(all_obs_passed_last_ste
p) !=
                                                    1)]
  ), anomaly_vector) == FALSE)) #For anomalies-more than 1 observed val
ues
  Tru_negatives_sunshine[simulation] = length(data) - Tru_positives_sun
shine[simulation] -
    Fals_negatives_sunshine[simulation] - Fals_positives_sunshine[simul
ation] #For anomalies-more than 1 observed values

  Accuracy_sunshine[simulation] = (Tru_positives_sunshine[simulation] +
                                     Tru_negatives_sunshine[simulation]
) / (
                                       Tru_positives_sunshine[simulatio
n] + Fals_negatives_sunshine[simulation] +
                                         Fals_positives_sunshine[simula
tion] + Tru_negatives_sunshine[simulation]
                                     )
  Precision_sunshine[simulation] = Tru_positives_sunshine[simulation] /
    (Tru_positives_sunshine[simulation] + Fals_positives_sunshine[simul
ation])
  Recall_sunshine[simulation] = Tru_positives_sunshine[simulation] / (T
ru_positives_sunshine[simulation] +

Fals_negatives_sunshine[simulation])
  F1_score_sunshine[simulation] = 2 * (Recall_sunshine[simulation] * Pr
ecision_sunshine[simulation]) /
    (Recall_sunshine[simulation] + Precision_sunshine[simulation])


  ###################### Twitter ################

  generate_dates = seq(as.Date("2000/1/1"),
                       by = "year",
                       length.out = length(data))
  create_data_frame = data.frame(generate_dates, data) #Creates data fr
ame to use in the below row

  res = AnomalyDetectionTs(
    create_data_frame,
    max_anoms = 0.02,
    direction = 'both',
    plot = FALSE
  )
  dtframe_res = as.data.frame(do.call(rbind, res)) #From list to data f
rame


  Tru_positives_twitter[simulation] = length(which(is.element(anomaly_v
ector, dtframe_res$anoms) ==
                                                     TRUE))
  Fals_negatives_twitter[simulation] = length(which(is.element(anomaly_
```

```
vector, dtframe_res$anoms) ==
                                                          FALSE))
  Fals_positives_twitter[simulation] = length(which(is.element(dtframe_
res$anoms, anomaly_vector) ==
                                                          FALSE))
  Tru_negatives_twitter[simulation] = length(data) - Tru_positives_twit
ter[simulation] -
    Fals_negatives_twitter[simulation] - Fals_positives_twitter[simulat
ion]

  Accuracy_twitter[simulation] = (Tru_positives_twitter[simulation] + T
ru_negatives_twitter[simulation]) /
    (
      Tru_positives_twitter[simulation] + Fals_negatives_twitter[simula
tion] +
        Fals_positives_twitter[simulation] + Tru_negatives_twitter[simu
lation]
    )
  Precision_twitter[simulation] = Tru_positives_twitter[simulation] / (
Tru_positives_twitter[simulation] +

Fals_positives_twitter[simulation])
  Recall_twitter[simulation] = Tru_positives_twitter[simulation] / (Tru
_positives_twitter[simulation] +
                                                                    F
als_negatives_twitter[simulation])
  F1_score_twitter[simulation] = 2 * (Recall_twitter[simulation] * Prec
ision_twitter[simulation]) /
    (Recall_twitter[simulation] + Precision_twitter[simulation])


  ###################### anomalize ################

  generate_dates = seq(as.Date("2000/1/1"),
                       by = "year",
                       length.out = length(data))
  create_data_frame = data.frame(generate_dates, data) #Creates a data
frame to use in the below row
  object_have_time_index = as_tbl_time(create_data_frame, index = gener
ate_dates)

  time_decomp_stl = time_decompose(object_have_time_index,
                                   data,
                                   method = "stl",
                                   merge = FALSE) #"stl" method
  data_frame_for_anomalize_stl = data.frame(generate_dates, time_decomp
_stl$remainder) #Creates a data frame to use in the below row
  anomalize_have_time_index_stl = as_tbl_time(data_frame_for_anomalize_
stl, index =
                                                generate_dates)


  Tru_positives_anomalize[simulation] = length(which(is.element(
    anomaly_place_vector, which(
      anomalize(
        anomalize_have_time_index_stl,
```

```
        time_decomp_stl.remainder,
        method = "gesd"
      )$anomaly == "Yes"
    )
  ) == TRUE))
  Fals_negatives_anomalize[simulation] = length(which(is.element(
    anomaly_place_vector, which(
      anomalize(
        anomalize_have_time_index_stl,
        time_decomp_stl.remainder,
        method = "gesd"
      )$anomaly == "Yes"
    )
  ) == FALSE))
  Fals_positives_anomalize[simulation] = length(which(is.element(
    which(
      anomalize(
        anomalize_have_time_index_stl,
        time_decomp_stl.remainder,
        method = "gesd"
      )$anomaly == "Yes"
    ), anomaly_place_vector
  ) == FALSE))
  Tru_negatives_anomalize[simulation] = length(data) - Tru_positives_an
omalize[simulation] -
    Fals_negatives_anomalize[simulation] - Fals_positives_anomalize[sim
ulation]

  Accuracy_anomalize[simulation] = (Tru_positives_anomalize[simulation]
+
                                      Tru_negatives_anomalize[simulatio
n]) / (
                                        Tru_positives_anomalize[simulat
ion] + Fals_negatives_anomalize[simulation] +
                                          Fals_positives_anomalize[simu
lation] + Tru_negatives_anomalize[simulation]
                                      )
  Precision_anomalize[simulation] = Tru_positives_anomalize[simulation]
/
    (Tru_positives_anomalize[simulation] + Fals_positives_anomalize[sim
ulation])
  Recall_anomalize[simulation] = Tru_positives_anomalize[simulation] /
(Tru_positives_anomalize[simulation] +

Fals_negatives_anomalize[simulation])
  F1_score_anomalize[simulation] = 2 * (Recall_anomalize[simulation] *
Precision_anomalize[simulation]) /
    (Recall_anomalize[simulation] + Precision_anomalize[simulation])


  ###################### strucchange ################

  fs.data = Fstats(data ~ 1)

  Tru_positives_strucchange[simulation] = length(which(
    is.element(anomaly_place_vector, fs.data$breakpoint) == TRUE
```

```
  ))
  Fals_negatives_strucchange[simulation] = length(which(
    is.element(anomaly_place_vector, fs.data$breakpoint) == FALSE
  ))
  Fals_positives_strucchange[simulation] = length(which(
    is.element(fs.data$breakpoint, anomaly_place_vector) == FALSE
  ))
  Tru_negatives_strucchange[simulation] = length(data) - Tru_positives_
strucchange[simulation] -
    Fals_negatives_strucchange[simulation] - Fals_positives_strucchange
[simulation]


  Accuracy_strucchange[simulation] = (Tru_positives_strucchange[simulat
ion] +
                                        Tru_negatives_strucchange[simul
ation]) / (
                                          Tru_positives_strucchange[sim
ulation] + Fals_negatives_strucchange[simulation] +
                                            Fals_positives_strucchange[
simulation] + Tru_negatives_strucchange[simulation]
                                        )
  Precision_strucchange[simulation] = Tru_positives_strucchange[simulat
ion] /
    (Tru_positives_strucchange[simulation] + Fals_positives_strucchange
[simulation])
  Recall_strucchange[simulation] = Tru_positives_strucchange[simulation
] /
    (Tru_positives_strucchange[simulation] + Fals_negatives_strucchange
[simulation])
  F1_score_strucchange[simulation] = 2 * (Recall_strucchange[simulation
] *
                                            Precision_strucchange[simul
ation]) / (Recall_strucchange[simulation] + Precision_strucchange[simul
ation])


  ###################### yakamoz ################

  ykmz_res = NULL
  ykmz_res = yakamoz(data, 12, 0.5, 0.5, 0.05)

  Tru_positives_yakamoz_last_moving_blocks_matrix[simulation] = length(
which(is.element(anomaly_vector, ykmz_res[[1]]) ==

TRUE))
  Fals_negatives_yakamoz_last_moving_blocks_matrix[simulation] = length
(which(is.element(anomaly_vector, ykmz_res[[1]]) ==

FALSE))
  Fals_positives_yakamoz_last_moving_blocks_matrix[simulation] = length
(which(is.element(ykmz_res[[1]], anomaly_vector) ==

FALSE))
  Tru_negatives_yakamoz_last_moving_blocks_matrix[simulation] = length(
data) -
```

```
    Tru_positives_yakamoz_last_moving_blocks_matrix[simulation] - Fals_
negatives_yakamoz_last_moving_blocks_matrix[simulation] -
    Fals_positives_yakamoz_last_moving_blocks_matrix[simulation]

  Accuracy_yakamoz_last_moving_blocks_matrix[simulation] = (
    Tru_positives_yakamoz_last_moving_blocks_matrix[simulation] + Tru_n
egatives_yakamoz_last_moving_blocks_matrix[simulation]
  ) / (
    Tru_positives_yakamoz_last_moving_blocks_matrix[simulation] + Fals_
negatives_yakamoz_last_moving_blocks_matrix[simulation] +
      Fals_positives_yakamoz_last_moving_blocks_matrix[simulation] + Tr
u_negatives_yakamoz_last_moving_blocks_matrix[simulation]
  )
  Precision_yakamoz_last_moving_blocks_matrix[simulation] = Tru_positiv
es_yakamoz_last_moving_blocks_matrix[simulation] /
    (
      Tru_positives_yakamoz_last_moving_blocks_matrix[simulation] + Fal
s_positives_yakamoz_last_moving_blocks_matrix[simulation]
    )
  Recall_yakamoz_last_moving_blocks_matrix[simulation] = Tru_positives_
yakamoz_last_moving_blocks_matrix[simulation] /
    (
      Tru_positives_yakamoz_last_moving_blocks_matrix[simulation] + Fal
s_negatives_yakamoz_last_moving_blocks_matrix[simulation]
    )
  F1_score_yakamoz_last_moving_blocks_matrix[simulation] = 2 * (
    Recall_yakamoz_last_moving_blocks_matrix[simulation] * Precision_ya
kamoz_last_moving_blocks_matrix[simulation]
  ) / (
    Recall_yakamoz_last_moving_blocks_matrix[simulation] + Precision_ya
kamoz_last_moving_blocks_matrix[simulation]
  )


  Tru_positives_yakamoz_penultimate_moving_blocks_matrix[simulation] =
length(which(is.element(anomaly_vector, ykmz_res[[2]]) ==

TRUE))
  Fals_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation] =
    length(which(is.element(anomaly_vector, ykmz_res[[2]]) == FALSE))
  Fals_positives_yakamoz_penultimate_moving_blocks_matrix[simulation] =
    length(which(is.element(ykmz_res[[2]], anomaly_vector) == FALSE))
  Tru_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation] =
length(data) -
    Tru_positives_yakamoz_penultimate_moving_blocks_matrix[simulation]
- Fals_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation] -
    Fals_positives_yakamoz_penultimate_moving_blocks_matrix[simulation]

  Accuracy_yakamoz_penultimate_moving_blocks_matrix[simulation] = (
    Tru_positives_yakamoz_penultimate_moving_blocks_matrix[simulation]
+ Tru_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation]
  ) / (
    Tru_positives_yakamoz_penultimate_moving_blocks_matrix[simulation]
+ Fals_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation] +
      Fals_positives_yakamoz_penultimate_moving_blocks_matrix[simulatio
n] + Tru_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation]
```

```
  )
  Precision_yakamoz_penultimate_moving_blocks_matrix[simulation] = Tru_
positives_yakamoz_penultimate_moving_blocks_matrix[simulation] /
    (
      Tru_positives_yakamoz_penultimate_moving_blocks_matrix[simulation
] + Fals_positives_yakamoz_penultimate_moving_blocks_matrix[simulation]
    )
  Recall_yakamoz_penultimate_moving_blocks_matrix[simulation] = Tru_pos
itives_yakamoz_penultimate_moving_blocks_matrix[simulation] /
    (
      Tru_positives_yakamoz_penultimate_moving_blocks_matrix[simulation
] + Fals_negatives_yakamoz_penultimate_moving_blocks_matrix[simulation]
    )
  F1_score_yakamoz_penultimate_moving_blocks_matrix[simulation] = 2 * (
    Recall_yakamoz_penultimate_moving_blocks_matrix[simulation] * Preci
sion_yakamoz_penultimate_moving_blocks_matrix[simulation]
  ) / (
    Recall_yakamoz_penultimate_moving_blocks_matrix[simulation] + Preci
sion_yakamoz_penultimate_moving_blocks_matrix[simulation]
  )

  print(simulation)
} #simulation
Accuracy_sunshine[is.na(Accuracy_sunshine)] = 0
Precision_sunshine[is.na(Precision_sunshine)] = 0
Recall_sunshine[is.na(Recall_sunshine)] = 0
F1_score_sunshine[is.na(F1_score_sunshine)] = 0
mean(Accuracy_sunshine)
mean(Precision_sunshine)
mean(Recall_sunshine)
mean(F1_score_sunshine)

Accuracy_twitter[is.na(Accuracy_twitter)] = 0
Precision_twitter[is.na(Precision_twitter)] = 0
Recall_twitter[is.na(Recall_twitter)] = 0
F1_score_twitter[is.na(F1_score_twitter)] = 0
mean(Accuracy_twitter)
mean(Precision_twitter)
mean(Recall_twitter)
mean(F1_score_twitter)

Accuracy_anomalize[is.na(Accuracy_anomalize)] = 0
Precision_anomalize[is.na(Precision_anomalize)] = 0
Recall_anomalize[is.na(Recall_anomalize)] = 0
F1_score_anomalize[is.na(F1_score_anomalize)] = 0
mean(Accuracy_anomalize)
mean(Precision_anomalize)
mean(Recall_anomalize)
mean(F1_score_anomalize)

Accuracy_strucchange[is.na(Accuracy_strucchange)] = 0
Precision_strucchange[is.na(Precision_strucchange)] = 0
Recall_strucchange[is.na(Recall_strucchange)] = 0
F1_score_strucchange[is.na(F1_score_strucchange)] = 0
mean(Accuracy_strucchange)
mean(Precision_strucchange)
```

```
mean(Recall_strucchange)
mean(F1_score_strucchange)

Accuracy_yakamoz_last_moving_blocks_matrix[is.na(Accuracy_yakamoz_last_
moving_blocks_matrix)] =
  0
Precision_yakamoz_last_moving_blocks_matrix[is.na(Precision_yakamoz_las
t_moving_blocks_matrix)] =
  0
Recall_yakamoz_last_moving_blocks_matrix[is.na(Recall_yakamoz_last_movi
ng_blocks_matrix)] =
  0
F1_score_yakamoz_last_moving_blocks_matrix[is.na(F1_score_yakamoz_last_
moving_blocks_matrix)] =
  0
mean(Accuracy_yakamoz_last_moving_blocks_matrix)
mean(Precision_yakamoz_last_moving_blocks_matrix)
mean(Recall_yakamoz_last_moving_blocks_matrix)
mean(F1_score_yakamoz_last_moving_blocks_matrix)

Accuracy_yakamoz_penultimate_moving_blocks_matrix[is.na(Accuracy_yakamo
z_penultimate_moving_blocks_matrix)] =
  0
Precision_yakamoz_penultimate_moving_blocks_matrix[is.na(Precision_yaka
moz_penultimate_moving_blocks_matrix)] =
  0
Recall_yakamoz_penultimate_moving_blocks_matrix[is.na(Recall_yakamoz_pe
nultimate_moving_blocks_matrix)] =
  0
F1_score_yakamoz_penultimate_moving_blocks_matrix[is.na(F1_score_yakamo
z_penultimate_moving_blocks_matrix)] =
  0
mean(Accuracy_yakamoz_penultimate_moving_blocks_matrix)
mean(Precision_yakamoz_penultimate_moving_blocks_matrix)
mean(Recall_yakamoz_penultimate_moving_blocks_matrix)
mean(F1_score_yakamoz_penultimate_moving_blocks_matrix)


mean(Tru_positives_sunshine)
mean(Fals_negatives_sunshine)
mean(Fals_positives_sunshine)
mean(Tru_negatives_sunshine)
########################################

mean(Tru_positives_twitter)
mean(Fals_negatives_twitter)
mean(Fals_positives_twitter)
mean(Tru_negatives_twitter)
########################################

mean(Tru_positives_anomalize)
mean(Fals_negatives_anomalize)
mean(Fals_positives_anomalize)
mean(Tru_negatives_anomalize)
```

```
########################################

mean(Tru_positives_strucchange)
mean(Fals_negatives_strucchange)
mean(Fals_positives_strucchange)
mean(Tru_negatives_strucchange)
########################################

mean(Tru_positives_yakamoz_last_moving_blocks_matrix)
mean(Fals_negatives_yakamoz_last_moving_blocks_matrix)
mean(Fals_positives_yakamoz_last_moving_blocks_matrix)
mean(Tru_negatives_yakamoz_last_moving_blocks_matrix)
########################################

mean(Tru_positives_yakamoz_penultimate_moving_blocks_matrix)
mean(Fals_negatives_yakamoz_penultimate_moving_blocks_matrix)
mean(Fals_positives_yakamoz_penultimate_moving_blocks_matrix)
mean(Tru_negatives_yakamoz_penultimate_moving_blocks_matrix)

if (FALSE)
{
  ######################### iForest ################

  #install.packages("solitude") #iForest
  library(solitude)

  Tru_positives_iForest = c()
  Fals_negatives_iForest = c()
  Fals_positives_iForest = c()
  Tru_negatives_iForest = c()

  Accuracy_iForest = c()
  Precision_iForest = c()
  Recall_iForest = c()
  F1_score_iForest = c()

  for (simulation in 1:1000)
  {
    generate_dates = seq(as.Date("2000/1/1"),
                         by = "year",
                         length.out = length(data))
    create_data_frame = data.frame(generate_dates, mat_data[, simulatio
n]) #Creates data frame to use in the below row

    iforest = isolationForest$new(sample_size = length(data))
    iforest$fit(create_data_frame)
    data_pred = iforest$predict(create_data_frame)


    Tru_positives_iForest[simulation] = length(which(is.element(
      anomaly_place_vector,
      which(data_pred$anomaly_score >= 0.5)
    ) == TRUE))
    Fals_negatives_iForest[simulation] = length(which(is.element(
      anomaly_place_vector,
      which(data_pred$anomaly_score >= 0.5)
```

```
    ) == FALSE))
    Fals_positives_iForest[simulation] = length(which(is.element(
      which(data_pred$anomaly_score >= 0.5),
      anomaly_place_vector
    ) == FALSE))
    Tru_negatives_iForest[simulation] = length(data) - Tru_positives_iF
orest[simulation] -
      Fals_negatives_iForest[simulation] - Fals_positives_iForest[simul
ation]

    Accuracy_iForest[simulation] = (Tru_positives_iForest[simulation] +
                                      Tru_negatives_iForest[simulation]
) / (
                                        Tru_positives_iForest[simulatio
n] + Fals_negatives_iForest[simulation] +
                                          Fals_positives_iForest[simula
tion] + Tru_negatives_iForest[simulation]
                                      )
    Precision_iForest[simulation] = Tru_positives_iForest[simulation] /
      (Tru_positives_iForest[simulation] + Fals_positives_iForest[simul
ation])
    Recall_iForest[simulation] = Tru_positives_iForest[simulation] / (T
ru_positives_iForest[simulation] +

Fals_negatives_iForest[simulation])
    F1_score_iForest[simulation] = 2 * (Recall_iForest[simulation] * Pr
ecision_iForest[simulation]) /
      (Recall_iForest[simulation] + Precision_iForest[simulation])

    print(simulation)

  } #simulation

  Accuracy_iForest[is.na(Accuracy_iForest)] = 0
  Precision_iForest[is.na(Precision_iForest)] = 0
  Recall_iForest[is.na(Recall_iForest)] = 0
  F1_score_iForest[is.na(F1_score_iForest)] = 0
  mean(Accuracy_iForest)
  mean(Precision_iForest)
  mean(Recall_iForest)
  mean(F1_score_iForest)


  mean(Tru_positives_iForest)
  mean(Fals_negatives_iForest)
  mean(Fals_positives_iForest)
  mean(Tru_negatives_iForest)
  ########################################
} #if(FALSE)
```

A.4 Anomaly Informaton for the Real Datasets

```
######### W10 Beer Production Data #########

T = 32
1 anomaly
anomaly_vector = c(data[12])
anomaly_place_vector = c(12)


######### W14 Airline Passengers Data #########

T = 87
1 anomaly
anomaly_vector = c(data[81])
anomaly_place_vector = c(81)


######### real_1 #########

T = 1420
2 anomalies
anomaly_vector = c(data[1215], data[1217])
anomaly_place_vector = c(1215, 1217)


######### real_4 #########

T = 1423
5 anomalies
anomaly_vector  =  c(data[1387],  data[1399],  data[1419],  data[1420],
data[1421])
anomaly_place_vector = c(1387, 1399, 1419, 1420, 1421)


######### real_5 #########

T = 1439
2 anomalies
anomaly_vector = c(data[515], data[516])
anomaly_place_vector = c(515, 516)


######### real_33 #########

T = 1439
2 anomalies
anomaly_vector = c(data[1434], data[1435])
anomaly_place_vector = c(1434, 1435)
```

```
######### real_49 #########

T = 1461
3 anomalies
anomaly_vector = c(data[422], data[423], data[424])
anomaly_place_vector = c(422, 423, 424)


######### real_51 #########

T = 1427
4 anomalies
anomaly_vector = c(data[600], data[1019], data[1020], data[1307])
anomaly_place_vector = c(600, 1019, 1020, 1307)
```

B. SIMULATION RESULTS

B.1 Simulation results for nonstationary time series AR(1) with anomaly place 12

Simulation number: 1000
T=250 p=-0.9
data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.1: Confusion matrix for nonstationary time series AR(1) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.893	0.107	4.094	245.906
Twitter's method	0.066	0.934	0.304	249.696
STL	0.692	0.308	1.130	248.870
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.829	0.171	23.273	226.727
Yakamoz_penultimate_moving_blocks_matrix	0.933	0.067	59.224	190.776
Isolation Forest	1.000	0.000	69.636	180.364

Table B.2: Evaluation metrics for nonstationary time series AR(1) max(data)+0.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.983	0.221	0.893	0.337
Twitter's method	0.995	0.030	0.066	0.038
STL	0.994	0.525	0.692	0.559
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.907	0.064	0.829	0.116
Yakamoz_penultimate_moving_blocks_matrix	0.764	0.024	0.933	0.046
Isolation Forest	0.723	0.014	1.000	0.028

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.3: Confusion matrix for nonstationary time series AR(1) max(data)+1.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.970	0.030	4.076	245.924
Twitter's method	0.100	0.900	0.295	249.705
STL	0.771	0.229	1.025	248.975
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.914	0.086	22.382	227.618
Yakamoz_penultimate_moving_blocks_matrix	0.971	0.029	55.956	194.044
Isolation Forest	1.000	0.000	68.181	181.819

Table B.4: Evaluation metrics for nonstationary time series AR(1) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.248	0.970	0.372
Twitter's method	0.995	0.061	0.100	0.070
STL	0.995	0.586	0.771	0.625
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.910	0.074	0.914	0.134
Yakamoz_penultimate_moving_blocks_matrix	0.777	0.027	0.971	0.051
Isolation Forest	0.728	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.5: Confusion matrix for nonstationary time series AR(1) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.984	0.016	3.968	246.032
Twitter's method	0.124	0.876	0.245	249.755
STL	0.838	0.162	1.034	248.966
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.937	0.063	21.864	228.136
Yakamoz_penultimate_moving_blocks_matrix	0.984	0.016	56.705	193.295
Isolation Forest	1.000	0.000	67.490	182.510

Table B.6: Evaluation metrics for nonstationary time series AR(1) max(data)+1.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.254	0.984	0.383
Twitter's method	0.996	0.093	0.124	0.100
STL	0.995	0.639	0.838	0.683
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.913	0.077	0.937	0.140
Yakamoz_penultimate_moving_blocks_matrix	0.774	0.027	0.984	0.051
Isolation Forest	0.731	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.7: Confusion matrix for nonstationary time series AR(1) max(data)+2.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.994	0.006	4.235	245.765
Twitter's method	0.179	0.821	0.280	249.720
STL	0.889	0.111	1.127	248.873
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.953	0.047	22.934	227.066
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.009	57.687	192.313
Isolation Forest	1.000	0.000	66.647	183.353

Table B.8: Evaluation metrics for nonstationary time series AR(1) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.983	0.242	0.994	0.370
Twitter's method	0.996	0.140	0.179	0.148
STL	0.995	0.667	0.889	0.714
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.908	0.076	0.953	0.138
Yakamoz_penultimate_moving_blocks_matrix	0.770	0.026	0.991	0.050
Isolation Forest	0.734	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.9: Confusion matrix for nonstationary time series AR(1) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.997	0.003	4.281	245.719
Twitter's method	0.246	0.754	0.192	249.808
STL	0.939	0.061	1.107	248.893
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.972	0.028	20.954	229.046
Yakamoz_penultimate_moving_blocks_matrix	0.997	0.003	58.385	191.615
Isolation Forest	1.000	0.000	65.929	184.071

Table B.10: Evaluation metrics for nonstationary time series AR(1) max(data)+2.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.983	0.236	0.997	0.364
Twitter's method	0.996	0.210	0.246	0.218
STL	0.995	0.708	0.939	0.759
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.916	0.080	0.972	0.145
Yakamoz_penultimate_moving_blocks_matrix	0.767	0.026	0.997	0.050
Isolation Forest	0.737	0.015	1.000	0.030

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.11: Confusion matrix for nonstationary time series AR(1) max(data)+3.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.068	245.932
Twitter's method	0.292	0.708	0.214	249.786
STL	0.973	0.027	1.092	248.908
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.975	0.025	21.671	228.329
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	57.126	192.874
Isolation Forest	1.000	0.000	65.110	184.890

Table B.12: Evaluation metrics for nonstationary time series AR(1) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.243	1.000	0.374
Twitter's method	0.996	0.257	0.292	0.265
STL	0.996	0.736	0.973	0.789
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.914	0.081	0.975	0.146
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.027	0.999	0.052
Isolation Forest	0.741	0.015	1.000	0.030

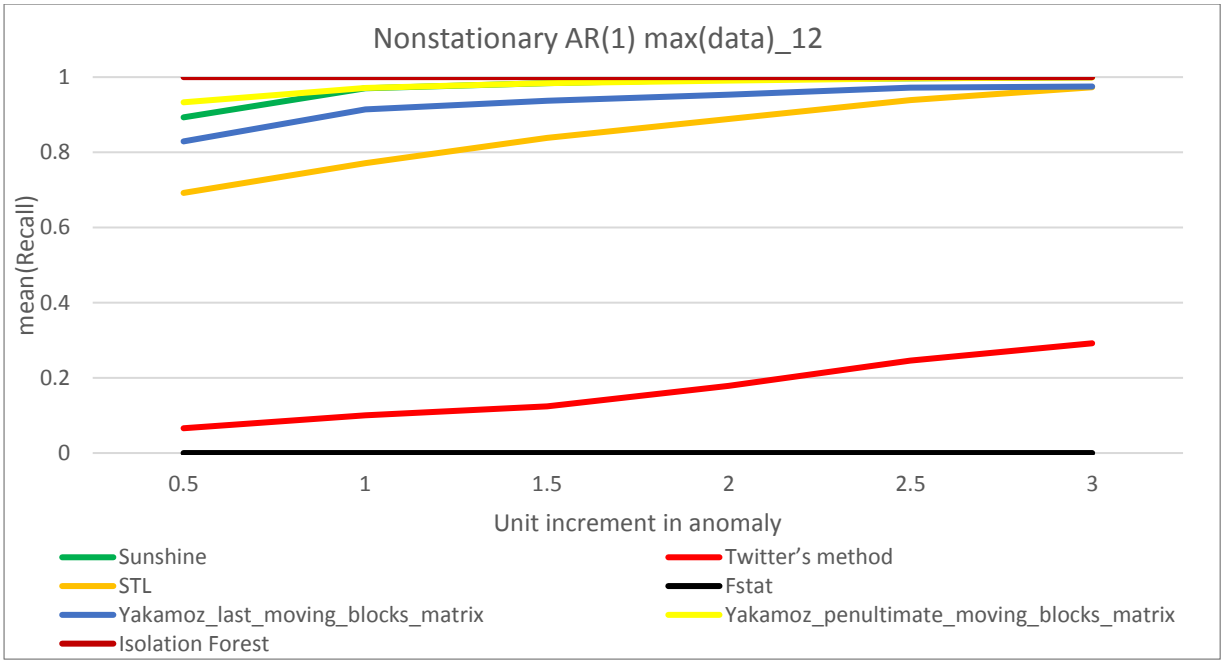

Figure B.1: Changes in mean(Recall) for nonstationary AR(1) max(data)_12 data

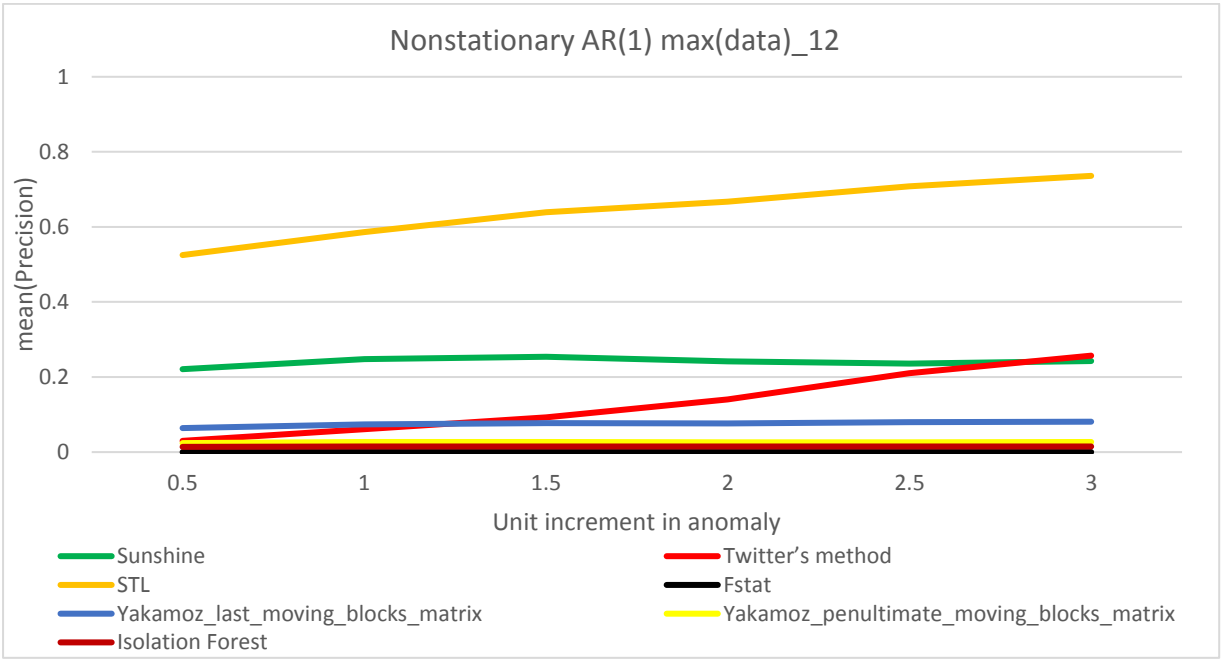

Figure B.2: Changes in mean(Precision) for nonstationary AR(1) max(data)_12 data

B.2 Simulation results for stationary time series AR(1) with anomaly place 250

Simulation number: 1000
T=250 p=-0.9
data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.13: Confusion matrix for stationary time series AR(1) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.943	0.057	4.393	244.607
Twitter's method	0.083	0.917	0.135	248.865
STL	0.118	0.882	1.275	247.725
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.862	0.138	18.318	230.682
Yakamoz_penultimate_moving_blocks_matrix	0.930	0.070	55.482	193.518
Isolation Forest	1.000	0.000	69.142	179.858

Table B.14: Evaluation metrics for stationary time series AR(1) max(data)+0.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.982	0.212	0.943	0.334
Twitter's method	0.996	0.048	0.083	0.057
STL	0.991	0.035	0.118	0.048
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.926	0.070	0.862	0.128
Yakamoz_penultimate_moving_blocks_matrix	0.778	0.026	0.930	0.050
Isolation Forest	0.723	0.014	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.15: Confusion matrix for stationary time series AR(1) max(data)+1.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.989	0.011	4.393	244.607
Twitter's method	0.157	0.843	0.137	248.863
STL	0.171	0.829	1.305	247.695
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.889	0.111	17.836	231.164
Yakamoz_penultimate_moving_blocks_matrix	0.961	0.039	52.181	196.819
Isolation Forest	1.000	0.000	68.131	180.869

Table B.16: Evaluation metrics for stationary time series AR(1) max(data)+1.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.982	0.222	0.989	0.350
Twitter's method	0.996	0.117	0.157	0.127
STL	0.991	0.064	0.171	0.080
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.928	0.074	0.889	0.135
Yakamoz_penultimate_moving_blocks_matrix	0.791	0.030	0.961	0.057
Isolation Forest	0.727	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.17: Confusion matrix for stationary time series AR(1) max(data)+1.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.998	0.002	4.440	244.560
Twitter's method	0.240	0.760	0.139	248.861
STL	0.257	0.743	1.237	247.763
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.916	0.084	17.541	231.459
Yakamoz_penultimate_moving_blocks_matrix	0.982	0.018	54.021	194.979
Isolation Forest	1.000	0.000	67.601	181.399

Table B.18: Evaluation metrics for stationary time series AR(1) max(data)+1.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.982	0.228	0.998	0.355
Twitter's method	0.996	0.197	0.240	0.208
STL	0.992	0.133	0.257	0.153
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.930	0.075	0.916	0.136
Yakamoz_penultimate_moving_blocks_matrix	0.784	0.029	0.982	0.055
Isolation Forest	0.730	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.19: Confusion matrix for stationary time series AR(1) max(data)+2.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	4.492	244.508
Twitter's method	0.383	0.617	0.158	248.842
STL	0.363	0.637	1.412	247.588
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.932	0.068	17.754	231.246
Yakamoz_penultimate_moving_blocks_matrix	0.986	0.014	54.199	194.801
Isolation Forest	1.000	0.000	66.852	182.148

Table B.20: Evaluation metrics for stationary time series AR(1) max(data)+2.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.982	0.220	1.000	0.348
Twitter's method	0.997	0.333	0.383	0.346
STL	0.992	0.211	0.363	0.234
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.929	0.077	0.932	0.139
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.030	0.986	0.057
Isolation Forest	0.733	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.21: Confusion matrix for stationary time series AR(1) max(data)+2.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.405	244.595
Twitter's method	0.536	0.464	0.161	248.839
STL	0.489	0.511	1.213	247.787
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.946	0.054	17.250	231.750
Yakamoz_penultimate_moving_blocks_matrix	0.994	0.006	54.136	194.864
Isolation Forest	1.000	0.000	66.457	182.543

Table B.22: Evaluation metrics for stationary time series AR(1) max(data)+2.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.982	0.225	1.000	0.354
Twitter's method	0.998	0.486	0.536	0.499
STL	0.993	0.340	0.489	0.366
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.931	0.079	0.946	0.144
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.029	0.994	0.056
Isolation Forest	0.734	0.015	1.000	0.030

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.23: Confusion matrix for stationary time series AR(1) max(data)+3.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	4.603	244.397
Twitter's method	0.643	0.357	0.123	248.877
STL	0.601	0.399	1.343	247.657
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.954	0.046	16.959	232.041
Yakamoz_penultimate_moving_blocks_matrix	0.994	0.006	54.310	194.690
Isolation Forest	1.000	0.000	65.773	183.227

Table B.24: Evaluation metrics for stationary time series AR(1) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.982	0.218	1.000	0.345
Twitter's method	0.998	0.604	0.643	0.614
STL	0.993	0.444	0.601	0.468
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.932	0.080	0.954	0.146
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.030	0.994	0.057
Isolation Forest	0.737	0.015	1.000	0.030

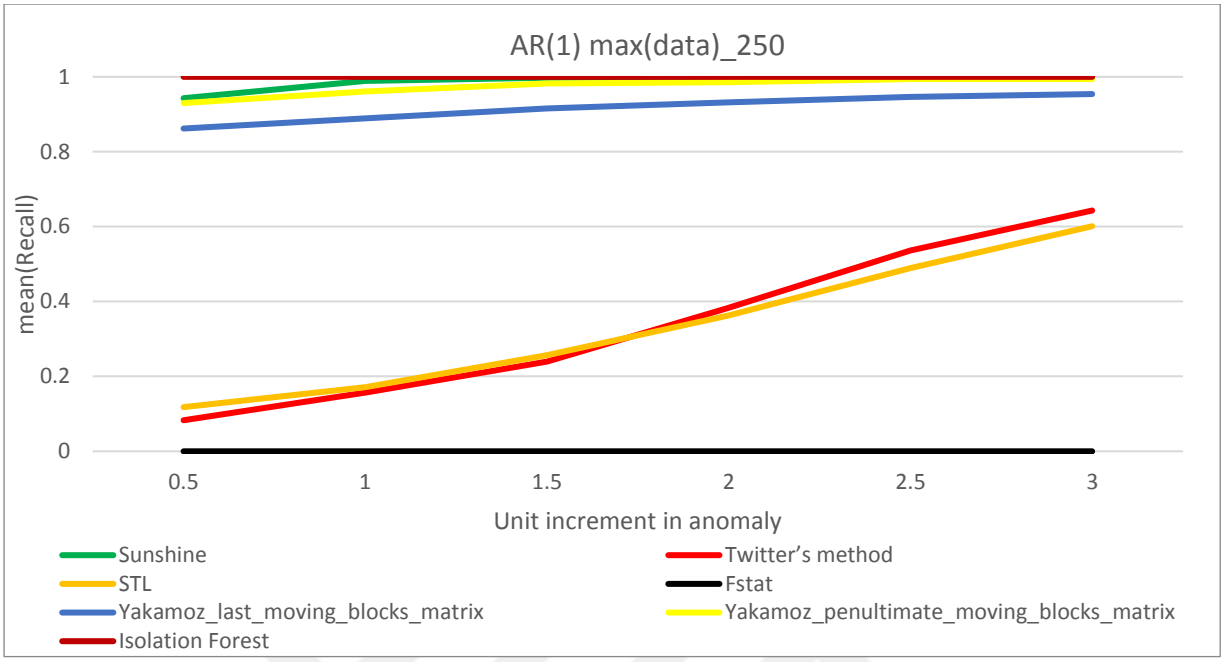

Figure B.3: Changes in mean(Recall) for stationary AR(1) max(data)_250 data

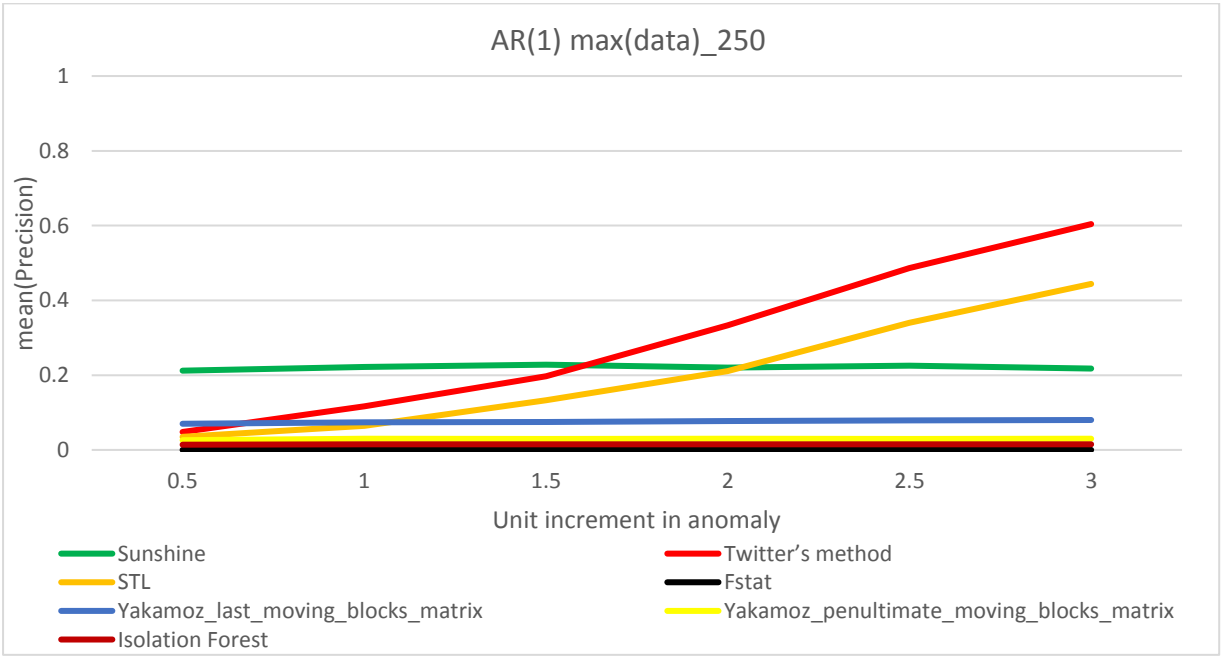

Figure B.4: Changes in mean(Precision) for stationary AR(1) max(data)_250 data

B.3 Simulation results for nonstationary time series AR(1) with anomaly place 250

Simulation number: 1000
T=250 p=-0.9
data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.25: Confusion matrix for nonstationary time series AR(1) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.847	0.153	4.392	245.608
Twitter's method	0.072	0.928	0.240	249.760
STL	0.689	0.311	1.209	248.791
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.789	0.211	23.890	226.110
Yakamoz_penultimate_moving_blocks_matrix	0.865	0.135	54.419	195.581
Isolation Forest	1.000	0.000	68.873	181.127

Table B.26: Evaluation metrics for nonstationary time series AR(1) max(data)+0.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.982	0.196	0.847	0.306
Twitter's method	0.995	0.039	0.072	0.046
STL	0.994	0.509	0.689	0.547
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.904	0.061	0.789	0.111
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.024	0.865	0.046
Isolation Forest	0.726	0.014	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.27: Confusion matrix for nonstationary time series AR(1) max(data)+1.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.945	0.055	4.317	245.683
Twitter's method	0.094	0.906	0.252	249.748
STL	0.731	0.269	1.175	248.825
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.868	0.132	25.155	224.845
Yakamoz_penultimate_moving_blocks_matrix	0.917	0.083	57.823	192.177
Isolation Forest	1.000	0.000	67.357	182.643

Table B.28: Evaluation metrics for nonstationary time series AR(1) max(data)+1.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.983	0.224	0.945	0.344
Twitter's method	0.995	0.062	0.094	0.069
STL	0.994	0.543	0.731	0.581
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.899	0.066	0.868	0.119
Yakamoz_penultimate_moving_blocks_matrix	0.769	0.023	0.917	0.045
Isolation Forest	0.732	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.29: Confusion matrix for nonstationary time series AR(1) max(data)+1.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.985	0.015	4.215	245.785
Twitter's method	0.124	0.876	0.225	249.775
STL	0.769	0.231	1.073	248.927
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.911	0.089	20.875	229.125
Yakamoz_penultimate_moving_blocks_matrix	0.951	0.049	55.437	194.563
Isolation Forest	1.000	0.000	66.614	183.386

Table B.30: Evaluation metrics for nonstationary time series AR(1) max(data)+1.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.983	0.236	0.985	0.364
Twitter's method	0.996	0.091	0.124	0.099
STL	0.995	0.586	0.769	0.626
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.916	0.075	0.911	0.136
Yakamoz_penultimate_moving_blocks_matrix	0.779	0.026	0.951	0.051
Isolation Forest	0.735	0.015	1.000	0.029

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.31: Confusion matrix for nonstationary time series AR(1) max(data)+2.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.995	0.005	4.283	245.717
Twitter's method	0.188	0.812	0.254	249.746
STL	0.818	0.182	1.181	248.819
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.940	0.060	22.342	227.658
Yakamoz_penultimate_moving_blocks_matrix	0.977	0.023	54.367	195.633
Isolation Forest	1.000	0.000	65.658	184.342

Table B.32: Evaluation metrics for nonstationary time series AR(1) max(data)+2.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.983	0.233	0.995	0.361
Twitter's method	0.996	0.146	0.188	0.155
STL	0.995	0.613	0.818	0.656
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.911	0.077	0.940	0.139
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.028	0.977	0.053
Isolation Forest	0.738	0.015	1.000	0.030

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.33: Confusion matrix for nonstationary time series AR(1) max(data)+2.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.999	0.001	4.137	245.863
Twitter's method	0.254	0.746	0.272	249.728
STL	0.854	0.146	1.031	248.969
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.964	0.036	21.386	228.614
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.009	54.424	195.576
Isolation Forest	1.000	0.000	64.842	185.158

Table B.34: Evaluation metrics for nonstationary time series AR(1) max(data)+2.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.245	0.999	0.374
Twitter's method	0.996	0.207	0.254	0.218
STL	0.995	0.653	0.854	0.697
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.915	0.078	0.964	0.141
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.028	0.991	0.054
Isolation Forest	0.742	0.015	1.000	0.030

Simulation number: 1000
T=250 p=-0.9 data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.35: Confusion matrix for nonstationary time series AR(1) max(data)+3.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.997	0.003	4.081	245.919
Twitter's method	0.293	0.707	0.244	249.756
STL	0.921	0.079	1.140	248.860
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.984	0.016	21.995	228.005
Yakamoz_penultimate_moving_blocks_matrix	0.997	0.003	54.363	195.637
Isolation Forest	1.000	0.000	64.092	185.908

Table B.36: Evaluation metrics for nonstationary time series AR(1) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.243	0.997	0.374
Twitter's method	0.996	0.251	0.293	0.260
STL	0.995	0.698	0.921	0.745
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.912	0.081	0.984	0.147
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.029	0.997	0.055
Isolation Forest	0.745	0.016	1.000	0.031

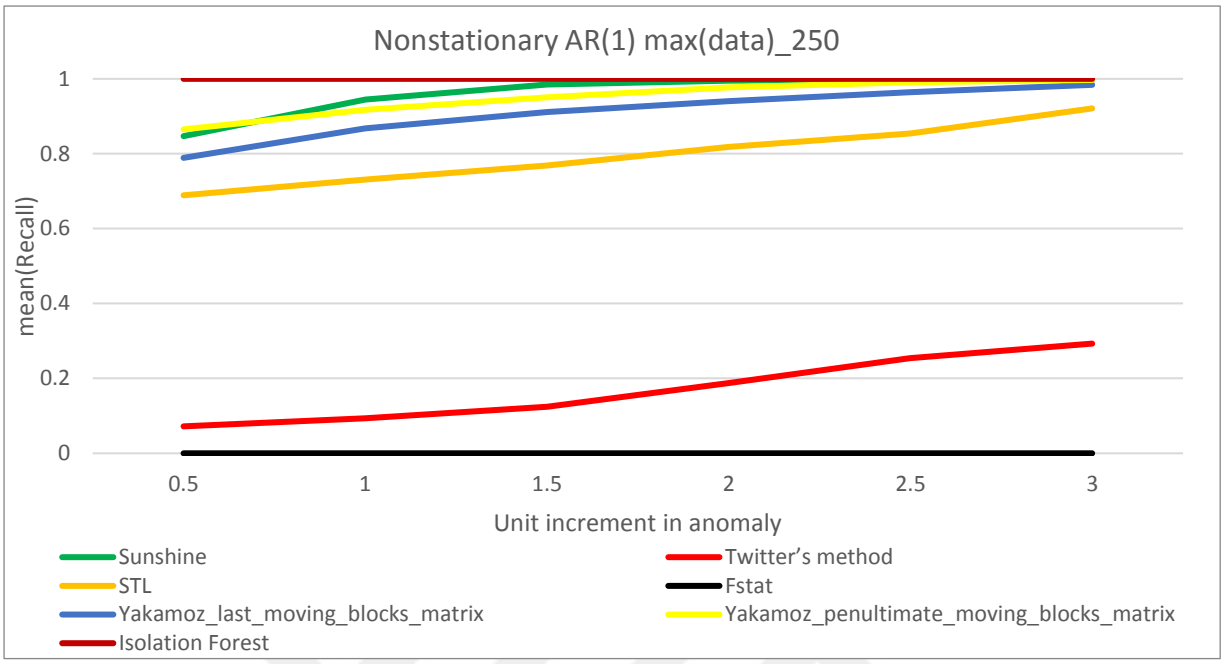

Figure B.5: Changes in mean(Recall) for nonstationary AR(1) max(data)_250 data

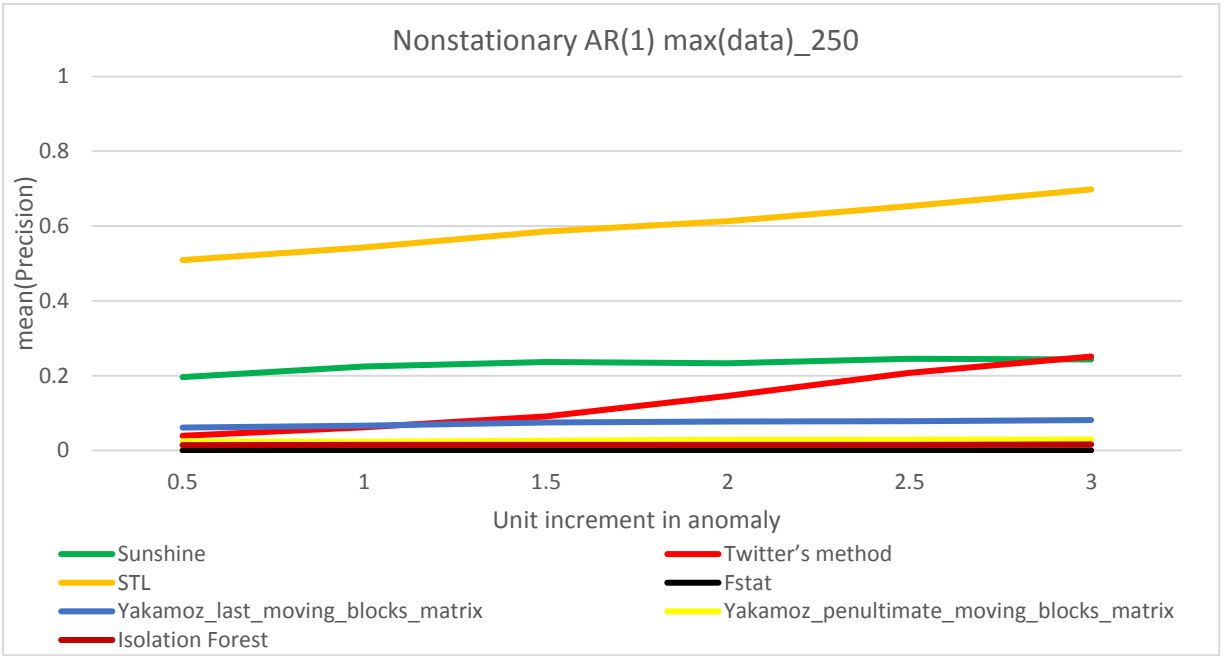

Figure B.6: Changes in mean(Precision) for nonstationary AR(1) max(data)_250 data

B.4 Simulation results for stationary time series AR(2) with anomaly place 12

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.37: Confusion matrix for stationary time series AR(2) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.972	0.028	4.955	244.045
Twitter's method	0.192	0.808	0.159	248.841
STL	0.210	0.790	0.347	248.653
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.915	0.085	17.002	231.998
Yakamoz_penultimate_moving_blocks_matrix	0.969	0.031	56.959	192.041
Isolation Forest	1.000	0.000	64.718	184.282

Table B.38: Evaluation metrics for stationary time series AR(2) max(data)+0.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.193	0.972	0.313
Twitter's method	0.996	0.144	0.192	0.158
STL	0.995	0.132	0.210	0.153
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.932	0.078	0.915	0.141
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.026	0.969	0.050
Isolation Forest	0.741	0.015	1.000	0.030

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.39: Confusion matrix for stationary time series AR(2) max(data)+1.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.998	0.002	5.126	243.874
Twitter's method	0.413	0.587	0.143	248.857
STL	0.450	0.550	0.320	248.680
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.956	0.044	17.300	231.700
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.009	58.032	190.968
Isolation Forest	1.000	0.000	64.087	184.913

Table B.40: Evaluation metrics for stationary time series AR(2) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.191	0.998	0.312
Twitter's method	0.997	0.361	0.413	0.376
STL	0.997	0.354	0.450	0.380
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.931	0.082	0.956	0.149
Yakamoz_penultimate_moving_blocks_matrix	0.768	0.027	0.991	0.051
Isolation Forest	0.744	0.016	1.000	0.031

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.41: Confusion matrix for stationary time series AR(2) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.215	243.785
Twitter's method	0.701	0.299	0.157	248.843
STL	0.750	0.250	0.341	248.659
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.977	0.023	16.319	232.681
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	58.164	190.836
Isolation Forest	1.000	0.000	63.205	185.795

Table B.42: Evaluation metrics for stationary time series AR(2) max(data)+1.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.192	1.000	0.311
Twitter's method	0.998	0.640	0.701	0.658
STL	0.998	0.632	0.750	0.666
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.935	0.085	0.977	0.154
Yakamoz_penultimate_moving_blocks_matrix	0.767	0.027	0.999	0.051
Isolation Forest	0.747	0.016	1.000	0.031

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.43: Confusion matrix for stationary time series AR(2) max(data)+2.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.371	243.629
Twitter's method	0.905	0.095	0.137	248.863
STL	0.911	0.089	0.313	248.687
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.985	0.015	16.610	232.390
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	58.139	190.861
Isolation Forest	1.000	0.000	61.945	187.055

Table B.44: Evaluation metrics for stationary time series AR(2) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.182	1.000	0.300
Twitter's method	0.999	0.850	0.905	0.867
STL	0.998	0.802	0.911	0.833
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.934	0.087	0.985	0.157
Yakamoz_penultimate_moving_blocks_matrix	0.767	0.028	1.000	0.053
Isolation Forest	0.752	0.016	1.000	0.032

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.45: Confusion matrix for stationary time series AR(2) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.320	243.680
Twitter's method	0.988	0.012	0.154	248.846
STL	0.985	0.015	0.333	248.667
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.995	0.005	16.497	232.503
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	57.880	191.120
Isolation Forest	1.000	0.000	60.959	188.041

Table B.46: Evaluation metrics for stationary time series AR(2) max(data)+2.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.189	1.000	0.308
Twitter's method	0.999	0.927	0.988	0.945
STL	0.999	0.868	0.985	0.901
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.934	0.087	0.995	0.157
Yakamoz_penultimate_moving_blocks_matrix	0.768	0.027	1.000	0.052
Isolation Forest	0.756	0.016	1.000	0.032

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.47: Confusion matrix for stationary time series AR(2) max(data)+3.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.474	243.526
Twitter's method	0.999	0.001	0.147	248.853
STL	0.998	0.002	0.330	248.670
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.997	0.003	16.304	232.696
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	56.954	192.046
Isolation Forest	1.000	0.000	60.000	189.000

Table B.48: Evaluation metrics for stationary time series AR(2) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.978	0.182	1.000	0.299
Twitter's method	0.999	0.938	0.999	0.957
STL	0.999	0.885	0.998	0.917
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.935	0.088	0.997	0.159
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.029	1.000	0.055
Isolation Forest	0.760	0.017	1.000	0.033

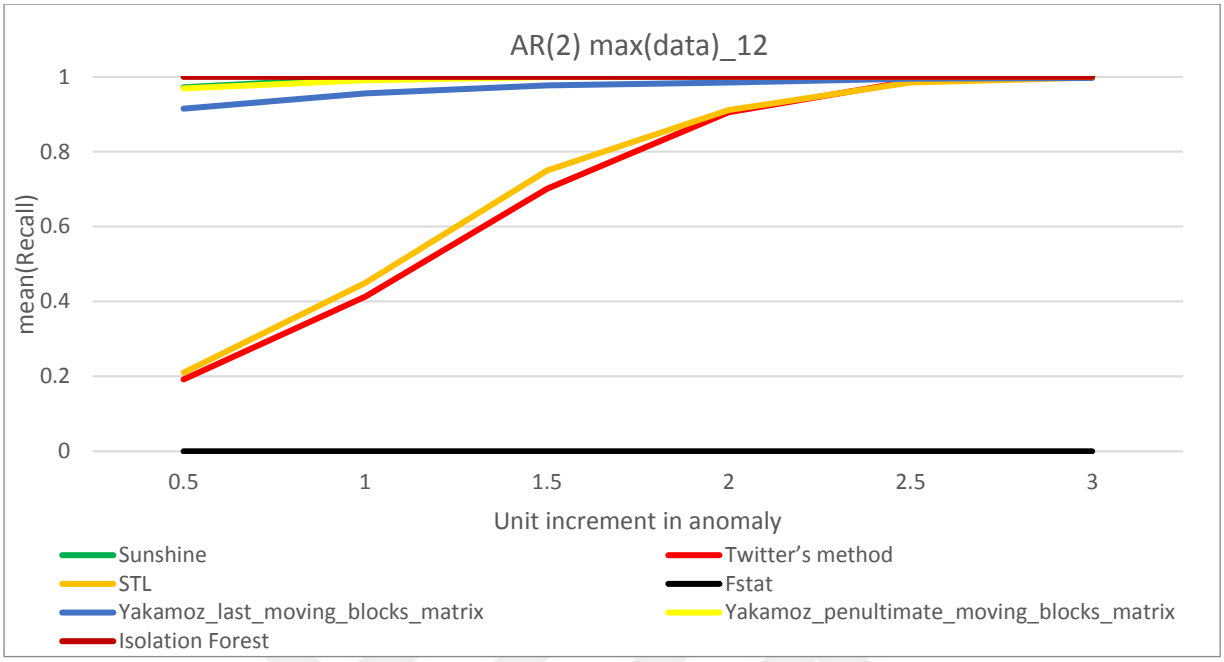

Figure B.7: Changes in mean(Recall) for stationary AR(2) max(data)_12 data

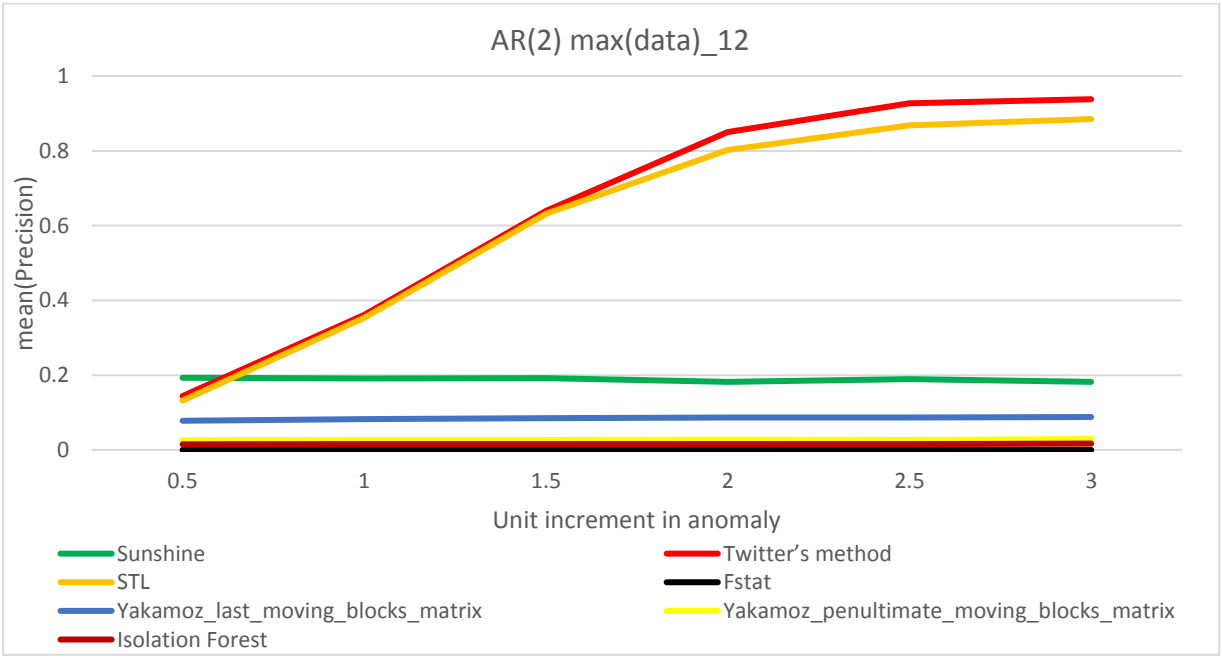

Figure B.8: Changes in mean(Precision) for stationary AR(2) max(data)_12 data

B.5 Simulation results for nonstationary time series AR(2) with anomaly place 12

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.49: Confusion matrix for nonstationary time series AR(2) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.921	0.079	3.760	246.240
Twitter's method	0.067	0.933	0.413	249.587
STL	0.832	0.168	0.511	249.489
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.923	0.077	15.688	234.312
Yakamoz_penultimate_moving_blocks_matrix	0.975	0.025	55.084	194.916
Isolation Forest	1.000	0.000	74.091	175.909

Table B.50: Evaluation metrics for nonstationary time series AR(2) max(data)+0.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.985	0.262	0.921	0.380
Twitter's method	0.995	0.027	0.067	0.035
STL	0.997	0.689	0.832	0.278
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.937	0.083	0.923	0.150
Yakamoz_penultimate_moving_blocks_matrix	0.780	0.027	0.975	0.051
Isolation Forest	0.705	0.014	1.000	0.027

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.51: Confusion matrix for nonstationary time series AR(2) max(data)+1.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.966	0.034	3.785	246.215
Twitter's method	0.083	0.917	0.401	249.599
STL	0.868	0.132	0.444	249.556
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.950	0.050	15.652	234.348
Yakamoz_penultimate_moving_blocks_matrix	0.990	0.010	53.875	196.125
Isolation Forest	1.000	0.000	72.320	177.680

Table B.52: Evaluation metrics for nonstationary time series AR(2) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.268	0.966	0.393
Twitter's method	0.995	0.039	0.083	0.048
STL	0.998	0.727	0.868	0.767
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.937	0.085	0.950	0.154
Yakamoz_penultimate_moving_blocks_matrix	0.785	0.028	0.990	0.054
Isolation Forest	0.712	0.014	1.000	0.027

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.53: Confusion matrix for nonstationary time series AR(2) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.987	0.013	3.777	246.223
Twitter's method	0.088	0.912	0.333	249.667
STL	0.917	0.083	0.559	249.441
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.971	0.029	15.258	234.742
Yakamoz_penultimate_moving_blocks_matrix	0.995	0.005	52.500	197.500
Isolation Forest	1.000	0.000	71.431	178.569

Table B.54: Evaluation metrics for nonstationary time series AR(2) max(data)+1.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.279	0.987	0.406
Twitter's method	0.995	0.049	0.088	0.056
STL	0.997	0.740	0.917	0.788
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.939	0.087	0.971	0.158
Yakamoz_penultimate_moving_blocks_matrix	0.791	0.028	0.995	0.054
Isolation Forest	0.715	0.014	1.000	0.028

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.55: Confusion matrix for nonstationary time series AR(2) max(data)+2.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.992	0.008	3.994	246.006
Twitter's method	0.132	0.868	0.400	249.600
STL	0.965	0.035	0.583	249.417
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.982	0.018	14.787	235.213
Yakamoz_penultimate_moving_blocks_matrix	0.997	0.003	52.934	197.066
Isolation Forest	1.000	0.000	70.830	179.170

Table B.56: Evaluation metrics for nonstationary time series AR(2) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.260	0.992	0.389
Twitter's method	0.995	0.084	0.132	0.094
STL	0.998	0.779	0.965	0.828
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.941	0.090	0.982	0.163
Yakamoz_penultimate_moving_blocks_matrix	0.789	0.029	0.997	0.056
Isolation Forest	0.718	0.014	1.000	0.028

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.57: Confusion matrix for nonstationary time series AR(2) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.996	0.004	3.777	246.223
Twitter's method	0.172	0.828	0.399	249.601
STL	0.994	0.006	0.467	249.533
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.987	0.013	14.591	235.409
Yakamoz_penultimate_moving_blocks_matrix	0.998	0.002	53.296	196.704
Isolation Forest	1.000	0.000	69.921	180.079

Table B.58: Evaluation metrics for nonstationary time series AR(2) max(data)+2.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.279	0.996	0.408
Twitter's method	0.995	0.121	0.172	0.130
STL	0.998	0.834	0.994	0.878
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.942	0.092	0.987	0.166
Yakamoz_penultimate_moving_blocks_matrix	0.788	0.028	0.998	0.054
Isolation Forest	0.721	0.014	1.000	0.028

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.59: Confusion matrix for nonstationary time series AR(2) max(data)+3.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.997	0.003	3.903	246.097
Twitter's method	0.184	0.816	0.374	249.626
STL	0.998	0.002	0.516	249.484
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.990	0.010	14.249	235.751
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	52.285	197.715
Isolation Forest	1.000	0.000	69.094	180.906

Table B.60: Evaluation metrics for nonstationary time series AR(2) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.266	0.997	0.395
Twitter's method	0.995	0.140	0.184	0.148
STL	0.998	0.823	0.998	0.871
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.943	0.094	0.990	0.169
Yakamoz_penultimate_moving_blocks_matrix	0.792	0.029	0.999	0.056
Isolation Forest	0.725	0.014	1.000	0.029

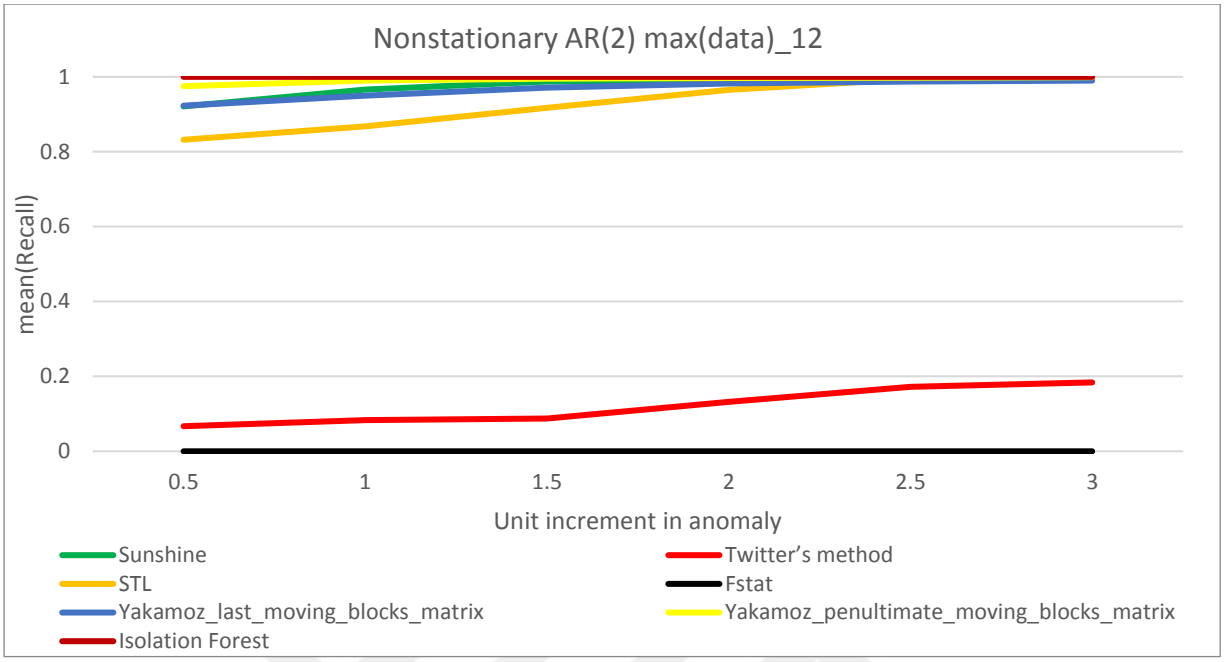

Figure B.9: Changes in mean(Recall) for nonstationary AR(2) max(data)_12 data

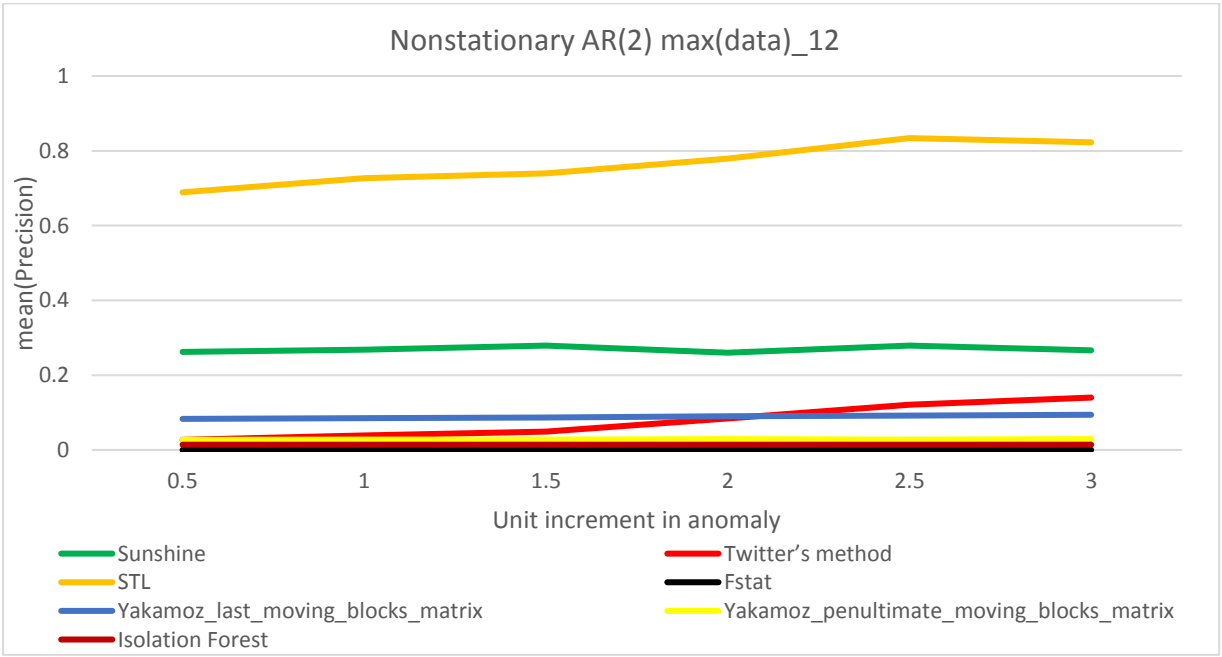

Figure B.10: Changes in mean(Precision) for nonstationary AR(2) max(data)_12 data

B.6 Simulation results for stationary time series AR(2) with anomaly place 250

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.61: Confusion matrix for stationary time series AR(2) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.890	0.110	4.992	244.008
Twitter's method	0.179	0.821	0.130	248.870
STL	0.206	0.794	0.366	248.634
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.725	0.275	21.067	227.933
Yakamoz_penultimate_moving_blocks_matrix	0.808	0.192	53.994	195.006
Isolation Forest	1.000	0.000	64.328	184.672

Table B.62: Evaluation metrics for stationary time series AR(2) max(data)+0.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.179	0.890	0.289
Twitter's method	0.996	0.141	0.179	0.152
STL	0.995	0.143	0.206	0.161
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.915	0.059	0.725	0.108
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.023	0.808	0.044
Isolation Forest	0.743	0.015	1.000	0.031

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.63: Confusion matrix for stationary time series AR(2) max(data)+1.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.986	0.014	4.934	244.066
Twitter's method	0.399	0.601	0.148	248.852
STL	0.451	0.549	0.364	248.636
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.860	0.140	21.186	227.814
Yakamoz_penultimate_moving_blocks_matrix	0.926	0.074	56.465	192.535
Isolation Forest	1.000	0.000	63.663	185.337

Table B.64: Evaluation metrics for stationary time series AR(2) max(data)+1.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.196	0.986	0.318
Twitter's method	0.997	0.348	0.399	0.363
STL	0.996	0.362	0.451	0.387
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.915	0.070	0.860	0.127
Yakamoz_penultimate_moving_blocks_matrix	0.774	0.026	0.926	0.049
Isolation Forest	0.745	0.016	1.000	0.031

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.65: Confusion matrix for stationary time series AR(2) max(data)+1.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.982	244.018
Twitter's method	0.695	0.305	0.144	248.856
STL	0.686	0.314	0.376	248.624
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.931	0.069	20.467	228.533
Yakamoz_penultimate_moving_blocks_matrix	0.984	0.016	55.537	193.463
Isolation Forest	1.000	0.000	62.907	186.093

Table B.66: Evaluation metrics for stationary time series AR(2) max(data)+1.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.199	1.000	0.322
Twitter's method	0.998	0.641	0.695	0.657
STL	0.997	0.580	0.686	0.609
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.918	0.076	0.931	0.137
Yakamoz_penultimate_moving_blocks_matrix	0.778	0.027	0.984	0.052
Isolation Forest	0.748	0.016	1.000	0.031

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.67: Confusion matrix for stationary time series AR(2) max(data)+2.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.062	243.938
Twitter's method	0.922	0.078	0.132	248.868
STL	0.872	0.128	0.323	248.677
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.923	0.077	19.474	229.526
Yakamoz_penultimate_moving_blocks_matrix	0.983	0.017	55.620	193.380
Isolation Forest	1.000	0.000	61.979	187.021

Table B.68: Evaluation metrics for stationary time series AR(2) max(data)+2.0_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.194	1.000	0.316
Twitter's method	0.999	0.864	0.922	0.882
STL	0.998	0.774	0.872	0.801
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.923	0.074	0.923	0.135
Yakamoz_penultimate_moving_blocks_matrix	0.777	0.028	0.983	0.053
Isolation Forest	0.752	0.016	1.000	0.032

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.69: Confusion matrix for stationary time series AR(2) max(data)+2.5_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.012	243.988
Twitter's method	0.986	0.014	0.144	248.856
STL	0.977	0.023	0.371	248.629
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.950	0.050	19.467	229.533
Yakamoz_penultimate_moving_blocks_matrix	0.995	0.005	55.226	193.774
Isolation Forest	1.000	0.000	61.091	187.909

Table B.70: Evaluation metrics for stationary time series AR(2) max(data)+2.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.198	1.000	0.321
Twitter's method	0.999	0.923	0.986	0.943
STL	0.998	0.857	0.977	0.891
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.922	0.079	0.950	0.144
Yakamoz_penultimate_moving_blocks_matrix	0.779	0.028	0.995	0.054
Isolation Forest	0.756	0.016	1.000	0.032

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.71: Confusion matrix for stationary time series AR(2) max(data)+3.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	4.927	244.073
Twitter's method	0.998	0.002	0.120	248.880
STL	0.987	0.013	0.340	248.660
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.957	0.043	19.993	229.007
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	55.278	193.722
Isolation Forest	1.000	0.000	60.236	188.764

Table B.72: Evaluation metrics for stationary time series AR(2) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.199	1.000	0.323
Twitter's method	1.000	0.945	0.998	0.961
STL	0.999	0.865	0.987	0.900
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.920	0.079	0.957	0.144
Yakamoz_penultimate_moving_blocks_matrix	0.779	0.028	0.999	0.054
Isolation Forest	0.759	0.016	1.000	0.032

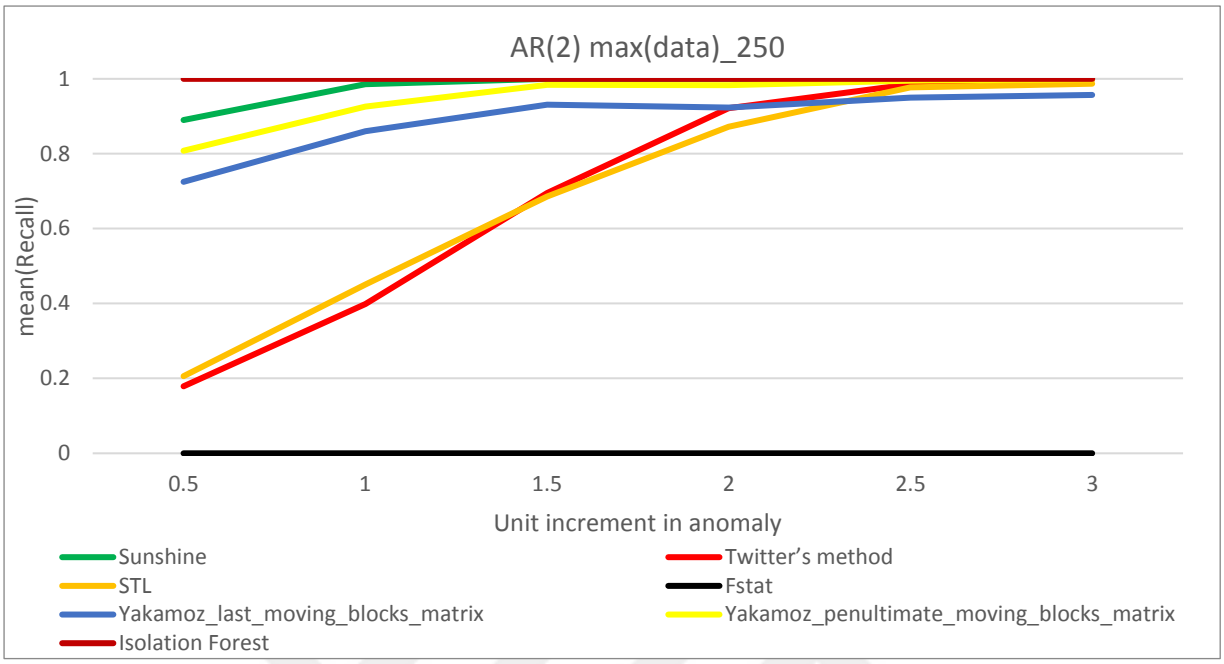

Figure B.11: Changes in mean(Recall) for stationary AR(2) max(data)_250 data

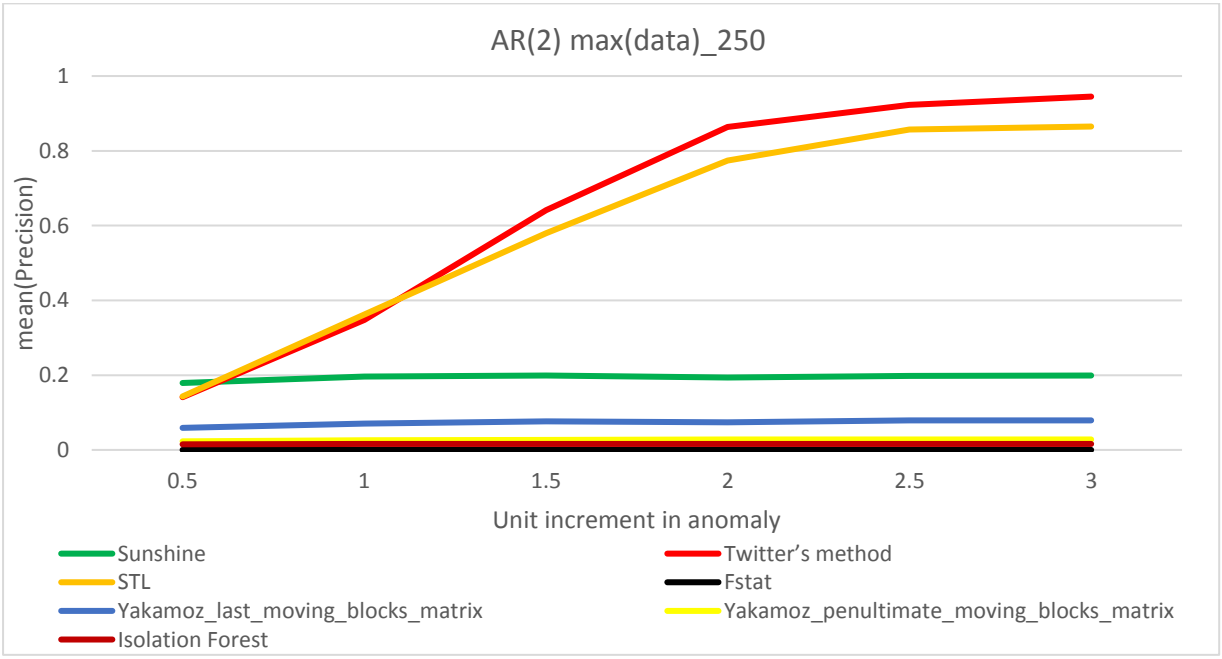

Figure B.12: Changes in mean(Precision) for stationary AR(2) max(data)_250 data

B.7 Simulation results for nonstationary time series AR(2) with anomaly place 250

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.73: Confusion matrix for nonstationary time series AR(2) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.853	0.147	3.896	246.104
Twitter's method	0.064	0.936	0.364	249.636
STL	0.747	0.253	0.602	249.398
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.850	0.150	16.220	233.780
Yakamoz_penultimate_moving_blocks_matrix	0.886	0.114	53.626	196.374
Isolation Forest	1.000	0.000	73.101	176.899

Table B.74: Evaluation metrics for nonstationary time series AR(2) max(data)+0.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.984	0.237	0.853	0.347
Twitter's method	0.995	0.025	0.064	0.033
STL	0.997	0.597	0.747	0.637
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.935	0.077	0.850	0.139
Yakamoz_penultimate_moving_blocks_matrix	0.786	0.024	0.886	0.047
Isolation Forest	0.709	0.014	1.000	0.027

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.75: Confusion matrix for nonstationary time series AR(2) max(data)+1.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.936	0.064	3.786	246.214
Twitter's method	0.101	0.899	0.406	249.594
STL	0.796	0.204	0.500	249.500
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.911	0.089	15.558	234.442
Yakamoz_penultimate_moving_blocks_matrix	0.947	0.053	51.459	198.541
Isolation Forest	1.000	0.000	71.419	178.581

Table B.76: Evaluation metrics for nonstationary time series AR(2) max(data)+1.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.261	0.936	0.382
Twitter's method	0.995	0.053	0.101	0.062
STL	0.997	0.659	0.796	0.697
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.938	0.085	0.911	0.154
Yakamoz_penultimate_moving_blocks_matrix	0.795	0.028	0.947	0.053
Isolation Forest	0.715	0.014	1.000	0.028

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.77: Confusion matrix for nonstationary time series AR(2) max(data)+1.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.970	0.030	3.797	246.203
Twitter's method	0.104	0.896	0.381	249.619
STL	0.850	0.150	0.465	249.535
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.938	0.062	15.310	234.690
Yakamoz_penultimate_moving_blocks_matrix	0.969	0.031	51.393	198.607
Isolation Forest	1.000	0.000	70.531	179.469

Table B.78: Evaluation metrics for nonstationary time series AR(2) max(data)+1.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.277	0.970	0.401
Twitter's method	0.995	0.057	0.104	0.066
STL	0.998	0.701	0.850	0.743
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.939	0.085	0.938	0.153
Yakamoz_penultimate_moving_blocks_matrix	0.795	0.028	0.969	0.054
Isolation Forest	0.719	0.014	1.000	0.028

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.79: Confusion matrix for nonstationary time series AR(2) max(data)+2.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.988	0.012	3.848	246.152
Twitter's method	0.136	0.864	0.324	249.676
STL	0.862	0.138	0.497	249.503
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.973	0.027	14.877	235.123
Yakamoz_penultimate_moving_blocks_matrix	0.985	0.015	52.553	197.447
Isolation Forest	1.000	0.000	69.553	180.447

Table B.80: Evaluation metrics for nonstationary time series AR(2) max(data)+2.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.267	0.988	0.395
Twitter's method	0.995	0.092	0.136	0.100
STL	0.997	0.712	0.862	0.753
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.941	0.089	0.973	0.161
Yakamoz_penultimate_moving_blocks_matrix	0.791	0.029	0.985	0.055
Isolation Forest	0.723	0.014	1.000	0.028

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.81: Confusion matrix for nonstationary time series AR(2) max(data)+2.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.996	0.004	3.935	246.065
Twitter's method	0.162	0.838	0.363	249.637
STL	0.897	0.103	0.483	249.517
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.981	0.019	14.263	235.737
Yakamoz_penultimate_moving_blocks_matrix	0.990	0.010	51.301	198.699
Isolation Forest	1.000	0.000	68.718	181.282

Table B.82: Evaluation metrics for nonstationary time series AR(2) max(data)+2.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.274	0.996	0.401
Twitter's method	0.995	0.115	0.162	0.124
STL	0.998	0.756	0.897	0.794
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.943	0.092	0.981	0.167
Yakamoz_penultimate_moving_blocks_matrix	0.796	0.030	0.990	0.057
Isolation Forest	0.726	0.015	1.000	0.029

Simulation number: 1000
T=250 p=c(-0.5,0.2) data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.83: Confusion matrix for nonstationary time series AR(2) max(data)+3.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.995	0.005	3.997	246.003
Twitter's method	0.218	0.782	0.421	249.579
STL	0.940	0.060	0.549	249.451
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.990	0.010	15.179	234.821
Yakamoz_penultimate_moving_blocks_matrix	0.998	0.002	51.022	198.978
Isolation Forest	1.000	0.000	67.870	182.130

Table B.84: Evaluation metrics for nonstationary time series AR(2) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.274	0.995	0.398
Twitter's method	0.995	0.165	0.218	0.175
STL	0.998	0.764	0.940	0.812
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.939	0.091	0.990	0.165
Yakamoz_penultimate_moving_blocks_matrix	0.797	0.029	0.998	0.056
Isolation Forest	0.730	0.015	1.000	0.029

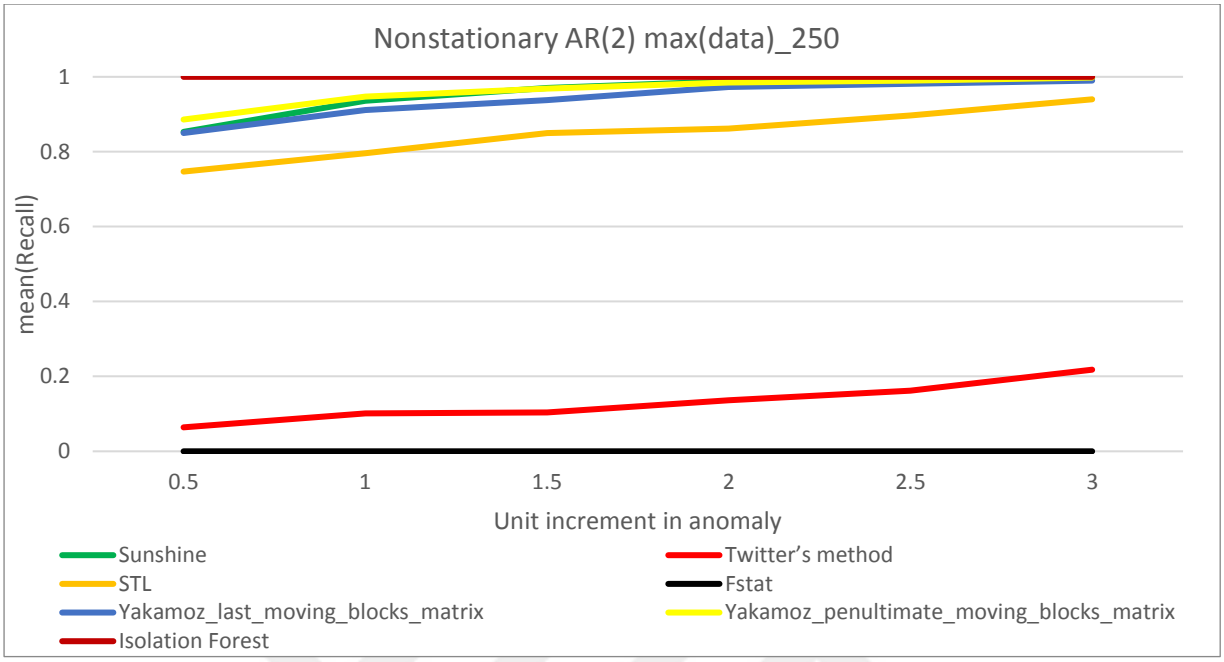

Figure B.13: Changes in mean(Recall) for nonstationary AR(2) max(data)_250 data

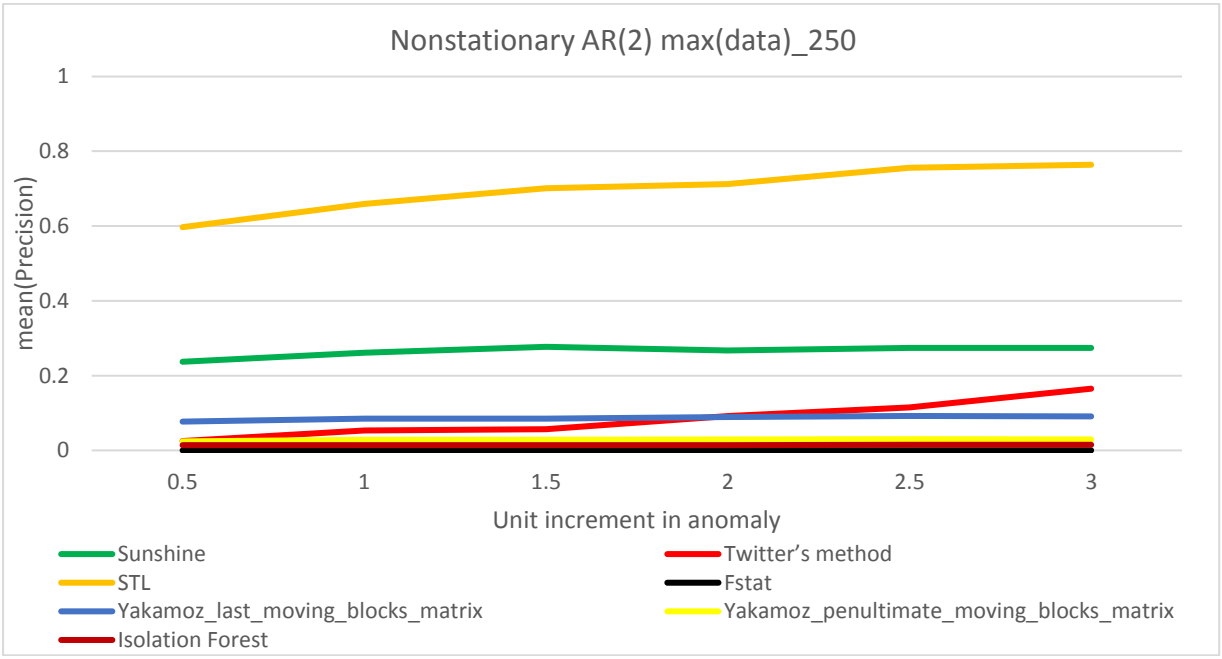

Figure B.14: Changes in mean(Precision) for nonstationary AR(2) max(data)_250 data

B.8 Simulation results for stationary time series ARMA(1,1) with anomaly place 12

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.85: Confusion matrix for stationary time series ARMA(1,1) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.975	0.025	4.896	244.104
Twitter's method	0.130	0.870	0.112	248.888
STL	0.159	0.841	0.785	248.215
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.925	0.075	17.320	231.680
Yakamoz_penultimate_moving_blocks_matrix	0.979	0.021	57.285	191.715
Isolation Forest	1.000	0.000	66.201	182.799

Table B.86: Evaluation metrics for stationary time series ARMA(1,1) max(data)+0.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.199	0.975	0.318
Twitter's method	0.996	0.098	0.130	0.107
STL	0.993	0.071	0.159	0.087
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.930	0.080	0.925	0.145
Yakamoz_penultimate_moving_blocks_matrix	0.771	0.027	0.979	0.051
Isolation Forest	0.735	0.015	1.000	0.030

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.87: Confusion matrix for stationary time series ARMA(1,1) max(data)+1.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.999	0.001	5.052	243.948
Twitter's method	0.286	0.714	0.111	248.889
STL	0.309	0.691	0.693	248.307
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.950	0.050	16.121	232.879
Yakamoz_penultimate_moving_blocks_matrix	0.990	0.010	55.540	193.460
Isolation Forest	1.000	0.000	65.888	183.112

Table B.88: Evaluation metrics for stationary time series ARMA(1,1) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.192	0.999	0.314
Twitter's method	0.997	0.249	0.286	0.260
STL	0.994	0.194	0.309	0.219
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.935	0.084	0.950	0.152
Yakamoz_penultimate_moving_blocks_matrix	0.778	0.028	0.990	0.054
Isolation Forest	0.736	0.015	1.000	0.030

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.89: Confusion matrix for stationary time series ARMA(1,1) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.194	243.806
Twitter's method	0.529	0.471	0.119	248.881
STL	0.555	0.445	0.708	248.292
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.982	0.018	15.583	233.417
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	57.715	191.285
Isolation Forest	1.000	0.000	64.183	184.817

Table B.90: Evaluation metrics for stationary time series ARMA(1,1) max(data)+1.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.195	1.000	0.315
Twitter's method	0.998	0.481	0.529	0.496
STL	0.995	0.417	0.555	0.448
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.938	0.086	0.982	0.156
Yakamoz_penultimate_moving_blocks_matrix	0.769	0.027	0.999	0.052
Isolation Forest	0.743	0.016	1.000	0.031

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.91: Confusion matrix for stationary time series ARMA(1,1) max(data)+2.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.394	243.606
Twitter's method	0.751	0.249	0.118	248.882
STL	0.766	0.234	0.829	248.171
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.990	0.010	15.955	233.045
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	58.403	190.597
Isolation Forest	1.000	0.000	63.943	185.057

Table B.92: Evaluation metrics for stationary time series ARMA(1,1) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.978	0.187	1.000	0.304
Twitter's method	0.999	0.704	0.751	0.718
STL	0.996	0.618	0.766	0.648
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.936	0.087	0.990	0.158
Yakamoz_penultimate_moving_blocks_matrix	0.766	0.027	0.999	0.052
Isolation Forest	0.744	0.016	1.000	0.031

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.93: Confusion matrix for stationary time series ARMA(1,1) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.347	243.653
Twitter's method	0.884	0.116	0.125	248.875
STL	0.879	0.121	0.810	248.190
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.997	0.003	16.607	232.393
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	58.611	190.389
Isolation Forest	1.000	0.000	62.626	186.374

Table B.94: Evaluation metrics for stationary time series ARMA(1,1) max(data)+2.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.184	1.000	0.303
Twitter's method	0.999	0.834	0.884	0.849
STL	0.996	0.723	0.879	0.756
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.934	0.088	0.997	0.159
Yakamoz_penultimate_moving_blocks_matrix	0.766	0.027	1.000	0.052
Isolation Forest	0.749	0.016	1.000	0.031

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.95: Confusion matrix for stationary time series ARMA(1,1) max(data)+3.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.331	243.669
Twitter's method	0.960	0.040	0.117	248.883
STL	0.961	0.039	0.738	248.262
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.994	0.006	14.864	234.136
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	58.440	190.560
Isolation Forest	1.000	0.000	61.848	187.152

Table B.96: Evaluation metrics for stationary time series ARMA(1,1) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.188	1.000	0.306
Twitter's method	0.999	0.916	0.960	0.929
STL	0.997	0.818	0.961	0.848
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.941	0.091	0.994	0.165
Yakamoz_penultimate_moving_blocks_matrix	0.766	0.028	1.000	0.053
Isolation Forest	0.753	0.016	1.000	0.032

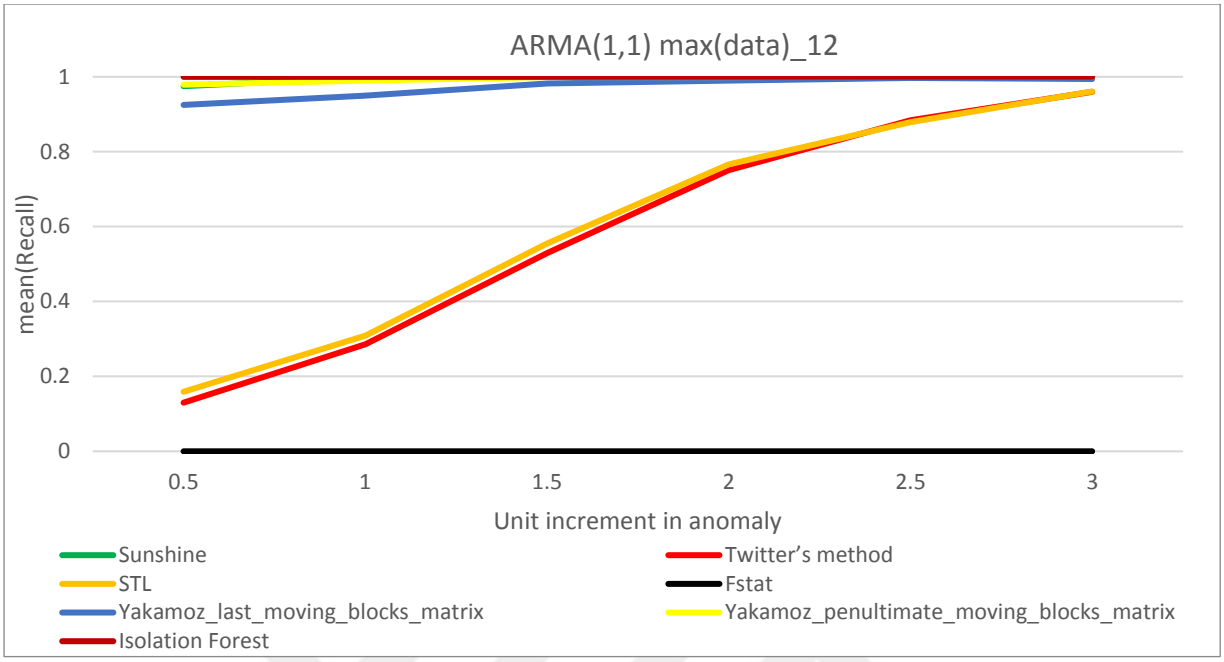

Figure B.15: Changes in mean(Recall) for stationary ARMA(1,1) max(data)_12 data

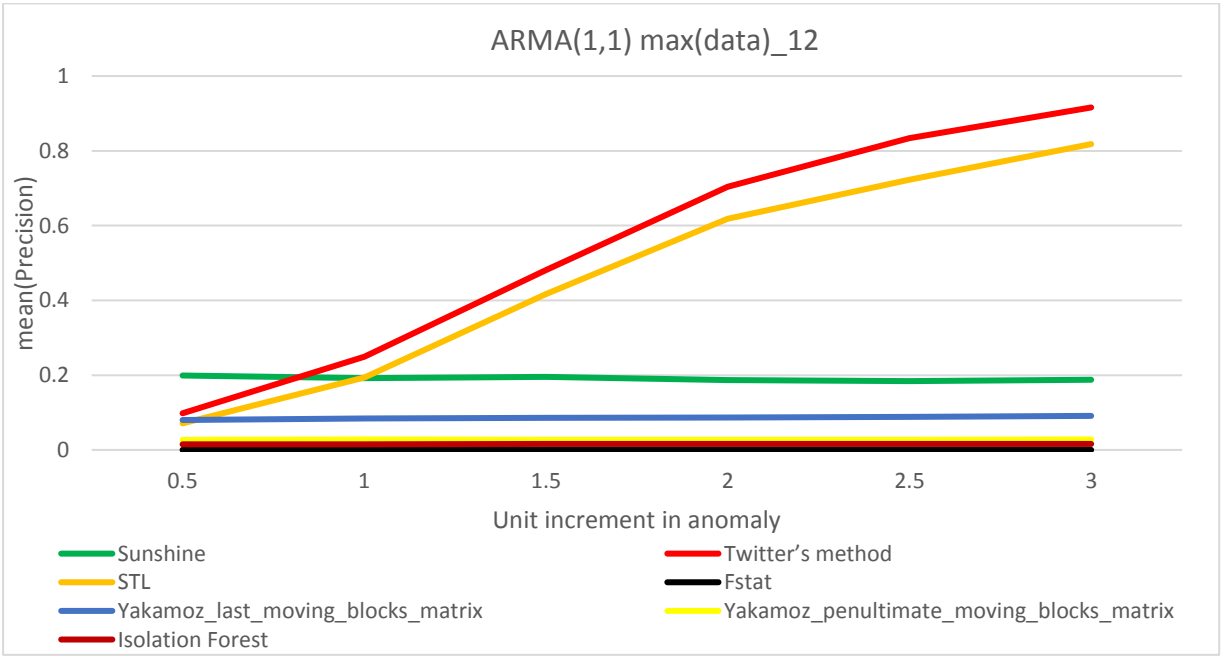

Figure B.16: Changes in mean(Precision) for stationary ARMA(1,1) max(data)_12 data

B.9 Simulation results for nonstationary time series ARMA(1,1) with anomaly place 12

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.97: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.897	0.103	3.879	246.121
Twitter's method	0.062	0.938	0.307	249.693
STL	0.790	0.210	0.510	249.490
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.909	0.091	18.851	231.149
Yakamoz_penultimate_moving_blocks_matrix	0.959	0.041	55.672	194.328
Isolation Forest	1.000	0.000	72.642	177.358

Table B.98: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+0.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.238	0.897	0.355
Twitter's method	0.995	0.025	0.062	0.033
STL	0.997	0.636	0.790	0.679
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.925	0.077	0.909	0.140
Yakamoz_penultimate_moving_blocks_matrix	0.778	0.025	0.959	0.049
Isolation Forest	0.711	0.014	1.000	0.027

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.99: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+1.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.956	0.044	3.883	246.117
Twitter's method	0.090	0.910	0.369	249.631
STL	0.850	0.150	0.519	249.481
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.933	0.067	16.326	233.674
Yakamoz_penultimate_moving_blocks_matrix	0.980	0.020	54.423	195.577
Isolation Forest	1.000	0.000	70.944	179.056

Table B.100: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.258	0.956	0.383
Twitter's method	0.995	0.046	0.090	0.055
STL	0.997	0.703	0.850	0.742
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.935	0.084	0.933	0.151
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.027	0.980	0.052
Isolation Forest	0.717	0.014	1.000	0.028

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.101: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.976	0.024	3.968	246.032
Twitter's method	0.108	0.892	0.393	249.607
STL	0.891	0.109	0.482	249.518
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.958	0.042	17.070	232.930
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.009	55.751	194.249
Isolation Forest	1.000	0.000	69.705	180.295

Table B.102: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+1.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.254	0.976	0.381
Twitter's method	0.995	0.065	0.108	0.073
STL	0.998	0.735	0.891	0.779
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.932	0.085	0.958	0.153
Yakamoz_penultimate_moving_blocks_matrix	0.778	0.027	0.991	0.052
Isolation Forest	0.722	0.014	1.000	0.028

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.103: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+2.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.987	0.013	3.855	246.145
Twitter's method	0.161	0.839	0.431	249.569
STL	0.949	0.051	0.475	249.525
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.977	0.023	17.512	232.488
Yakamoz_penultimate_moving_blocks_matrix	0.996	0.004	55.850	194.150
Isolation Forest	1.000	0.000	69.630	180.370

Table B.104: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.266	0.987	0.395
Twitter's method	0.995	0.112	0.161	0.122
STL	0.998	0.791	0.949	0.835
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.930	0.086	0.977	0.156
Yakamoz_penultimate_moving_blocks_matrix	0.777	0.027	0.996	0.052
Isolation Forest	0.723	0.014	1.000	0.028

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.105: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.996	0.004	3.950	246.050
Twitter's method	0.155	0.845	0.370	249.630
STL	0.987	0.013	0.484	249.516
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.981	0.019	17.840	232.160
Yakamoz_penultimate_moving_blocks_matrix	0.999	0.001	55.068	194.932
Isolation Forest	1.000	0.000	68.409	181.591

Table B.106: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+2.5_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.261	0.996	0.390
Twitter's method	0.995	0.110	0.155	0.119
STL	0.998	0.820	0.987	0.867
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.929	0.086	0.981	0.156
Yakamoz_penultimate_moving_blocks_matrix	0.781	0.028	0.999	0.053
Isolation Forest	0.727	0.015	1.000	0.029

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.107: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+3.0_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.998	0.002	3.878	246.122
Twitter's method	0.215	0.785	0.380	249.620
STL	0.995	0.005	0.470	249.530
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.988	0.012	16.522	233.478
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	54.692	195.308
Isolation Forest	1.000	0.000	67.516	182.484

Table B.108: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.262	0.998	0.393
Twitter's method	0.995	0.162	0.215	0.172
STL	0.998	0.829	0.995	0.876
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.934	0.091	0.988	0.164
Yakamoz_penultimate_moving_blocks_matrix	0.782	0.028	1.000	0.053
Isolation Forest	0.731	0.015	1.000	0.029

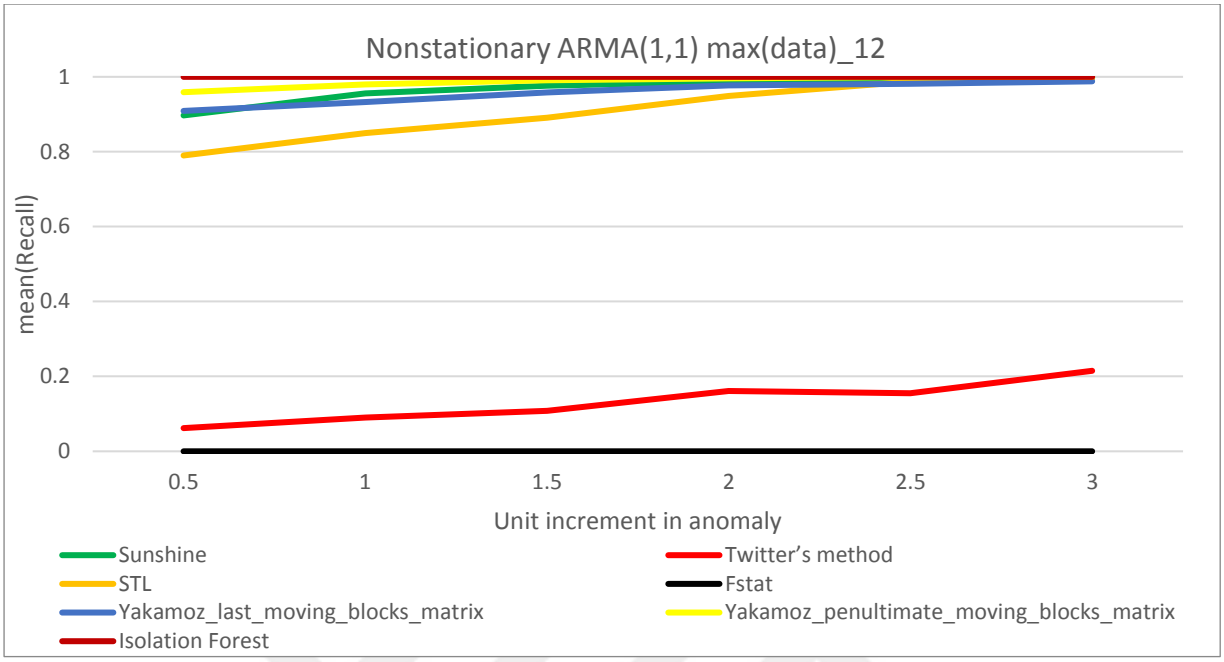

Figure B.17: Changes in mean(Recall) for nonstationary ARMA(1,1) max(data)_12 data

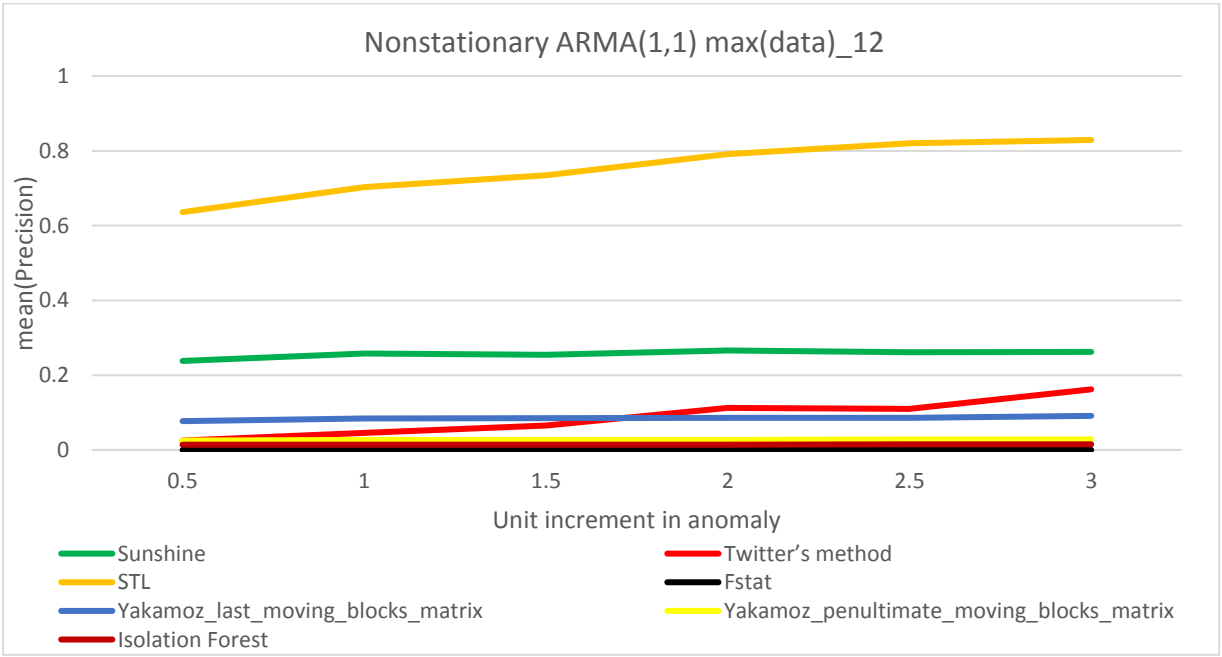

Figure B.18: Changes in mean(Precision) for nonstationary ARMA(1,1) max(data)_12 data

B.10 Simulation results for stationary time series ARMA(1,1) with anomaly place 250

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.109: Confusion matrix for stationary time series ARMA(1,1) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.905	0.095	4.789	244.211
Twitter's method	0.110	0.890	0.122	248.878
STL	0.132	0.868	0.763	248.237
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.767	0.233	21.071	227.929
Yakamoz_penultimate_moving_blocks_matrix	0.858	0.142	54.623	194.377
Isolation Forest	1.000	0.000	65.675	183.325

Table B.110: Evaluation metrics for stationary time series ARMA(1,1) max(data)+0.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.187	0.905	0.300
Twitter's method	0.996	0.076	0.110	0.086
STL	0.993	0.063	0.132	0.078
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.915	0.062	0.767	0.112
Yakamoz_penultimate_moving_blocks_matrix	0.781	0.025	0.858	0.047
Isolation Forest	0.737	0.015	1.000	0.030

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.111: Confusion matrix for stationary time series ARMA(1,1) max(data)+1.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.991	0.009	4.848	244.152
Twitter's method	0.286	0.714	0.133	248.867
STL	0.323	0.677	0.844	248.156
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.868	0.132	20.752	228.248
Yakamoz_penultimate_moving_blocks_matrix	0.938	0.062	54.457	194.543
Isolation Forest	1.000	0.000	65.169	183.831

Table B.112: Evaluation metrics for stationary time series ARMA(1,1) max(data)+1.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.981	0.203	0.991	0.327
Twitter's method	0.997	0.241	0.286	0.254
STL	0.994	0.199	0.323	0.224
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.916	0.069	0.868	0.126
Yakamoz_penultimate_moving_blocks_matrix	0.782	0.027	0.938	0.051
Isolation Forest	0.739	0.015	1.000	0.030

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.113: Confusion matrix for stationary time series ARMA(1,1) max(data)+1.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.998	0.002	4.791	244.209
Twitter's method	0.534	0.466	0.138	248.862
STL	0.516	0.484	0.845	248.155
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.921	0.079	18.793	230.207
Yakamoz_penultimate_moving_blocks_matrix	0.972	0.028	55.324	193.676
Isolation Forest	1.000	0.000	64.254	184.746

Table B.114: Evaluation metrics for stationary time series ARMA(1,1) max(data)+1.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.981	0.202	0.998	0.327
Twitter's method	0.998	0.484	0.534	0.498
STL	0.995	0.377	0.516	0.406
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.925	0.077	0.921	0.139
Yakamoz_penultimate_moving_blocks_matrix	0.779	0.028	0.972	0.053
Isolation Forest	0.743	0.016	1.000	0.031

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.115: Confusion matrix for stationary time series ARMA(1,1) max(data)+2.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.790	244.210
Twitter's method	0.744	0.256	0.135	248.865
STL	0.739	0.261	0.760	248.240
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.942	0.058	20.052	228.948
Yakamoz_penultimate_moving_blocks_matrix	0.989	0.011	54.896	194.104
Isolation Forest	1.000	0.000	63.349	185.651

Table B.116: Evaluation metrics for stationary time series ARMA(1,1) max(data)+2.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.981	0.205	1.000	0.330
Twitter's method	0.998	0.692	0.744	0.707
STL	0.996	0.598	0.739	0.630
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.920	0.077	0.942	0.140
Yakamoz_penultimate_moving_blocks_matrix	0.780	0.028	0.989	0.054
Isolation Forest	0.747	0.016	1.000	0.031

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.117: Confusion matrix for stationary time series ARMA(1,1) max(data)+2.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.782	244.218
Twitter's method	0.896	0.104	0.124	248.876
STL	0.851	0.149	0.706	248.294
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.952	0.048	19.772	229.228
Yakamoz_penultimate_moving_blocks_matrix	0.993	0.007	55.993	193.007
Isolation Forest	1.000	0.000	62.440	186.560

Table B.118: Evaluation metrics for stationary time series ARMA(1,1) max(data)+2.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.981	0.203	1.000	0.328
Twitter's method	0.999	0.848	0.896	0.862
STL	0.997	0.721	0.851	0.750
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.921	0.077	0.952	0.141
Yakamoz_penultimate_moving_blocks_matrix	0.776	0.028	0.993	0.053
Isolation Forest	0.750	0.016	1.000	0.031

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.119: Confusion matrix for stationary time series ARMA(1,1) max(data)+3.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	4.890	244.110
Twitter's method	0.954	0.046	0.128	248.872
STL	0.927	0.073	0.768	248.232
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.962	0.038	19.808	229.192
Yakamoz_penultimate_moving_blocks_matrix	0.997	0.003	56.192	192.808
Isolation Forest	1.000	0.000	61.799	187.201

Table B.120: Evaluation metrics for stationary time series ARMA(1,1) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.200	1.000	0.324
Twitter's method	0.999	0.904	0.954	0.919
STL	0.997	0.774	0.927	0.808
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.921	0.080	0.962	0.146
Yakamoz_penultimate_moving_blocks_matrix	0.775	0.028	0.997	0.054
Isolation Forest	0.753	0.016	1.000	0.032

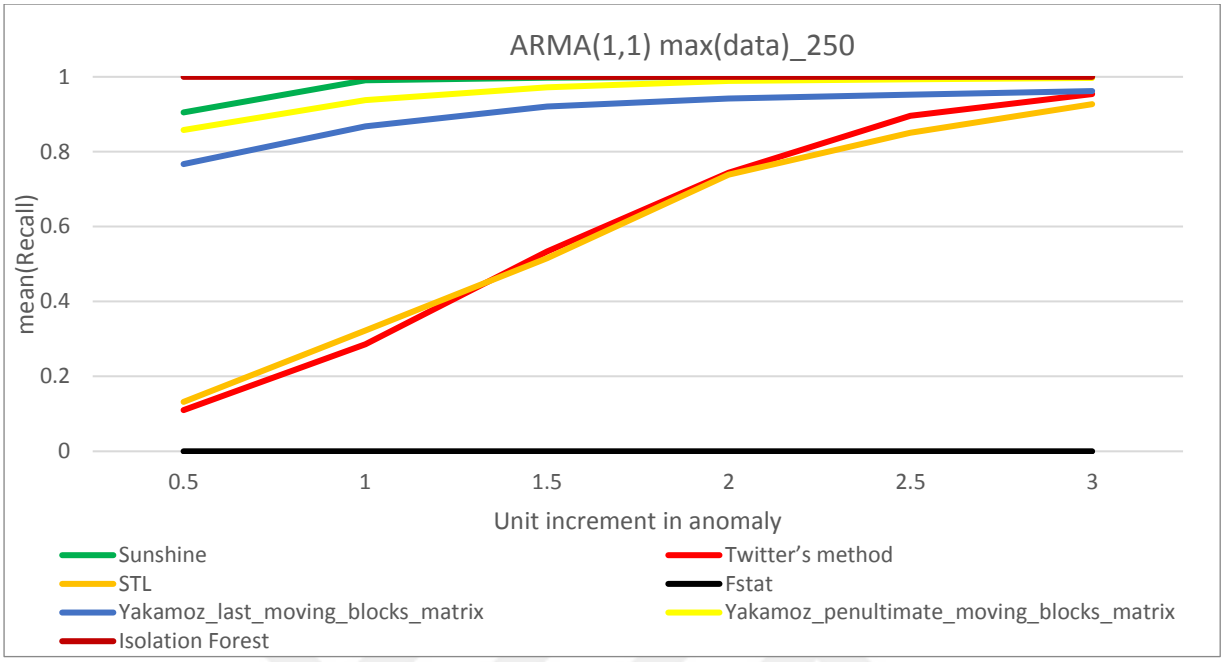

Figure B.19: Changes in mean(Recall) for stationary ARMA(1,1) max(data)_250 data

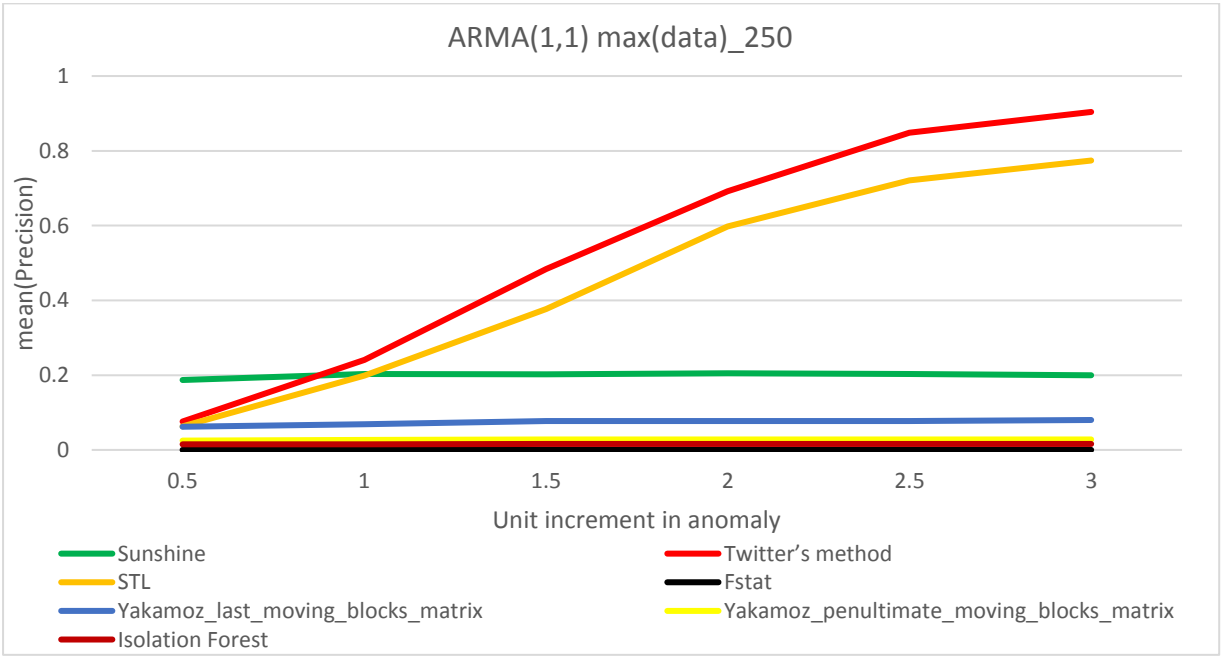

Figure B.20: Changes in mean(Precision) for stationary ARMA(1,1) max(data)_250 data

B.11 Simulation results for nonstationary time series ARMA(1,1) with anomaly place 250

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.121: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.876	0.124	4.040	245.960
Twitter's method	0.065	0.935	0.400	249.600
STL	0.741	0.259	0.526	249.474
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.858	0.142	19.098	230.902
Yakamoz_penultimate_moving_blocks_matrix	0.904	0.096	54.597	195.403
Isolation Forest	1.000	0.000	71.878	178.122

Table B.122: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+0.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.983	0.240	0.876	0.352
Twitter's method	0.995	0.028	0.065	0.036
STL	0.997	0.603	0.741	0.641
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.923	0.075	0.858	0.136
Yakamoz_penultimate_moving_blocks_matrix	0.782	0.025	0.904	0.049
Isolation Forest	0.714	0.014	1.000	0.027

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.123: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+1.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.929	0.071	3.875	246.125
Twitter's method	0.089	0.911	0.354	249.646
STL	0.789	0.211	0.484	249.516
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.905	0.095	16.937	233.063
Yakamoz_penultimate_moving_blocks_matrix	0.934	0.066	53.401	196.599
Isolation Forest	1.000	0.000	70.137	179.863

Table B.124: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+1.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.257	0.929	0.377
Twitter's method	0.995	0.046	0.089	0.054
STL	0.997	0.660	0.789	0.694
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.932	0.081	0.905	0.146
Yakamoz_penultimate_moving_blocks_matrix	0.787	0.026	0.934	0.050
Isolation Forest	0.721	0.014	1.000	0.028

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.125: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+1.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.979	0.021	3.850	246.150
Twitter's method	0.092	0.908	0.360	249.640
STL	0.829	0.171	0.440	249.560
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.945	0.055	16.633	233.367
Yakamoz_penultimate_moving_blocks_matrix	0.972	0.028	54.547	195.453
Isolation Forest	1.000	0.000	69.278	180.722

Table B.126: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+1.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.985	0.264	0.979	0.391
Twitter's method	0.995	0.057	0.092	0.064
STL	0.998	0.697	0.829	0.734
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.934	0.084	0.945	0.153
Yakamoz_penultimate_moving_blocks_matrix	0.783	0.027	0.972	0.052
Isolation Forest	0.724	0.014	1.000	0.028

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.127: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+2.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.985	0.015	3.747	246.253
Twitter's method	0.130	0.870	0.339	249.661
STL	0.866	0.134	0.573	249.427
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.954	0.046	16.665	233.335
Yakamoz_penultimate_moving_blocks_matrix	0.979	0.021	52.986	197.014
Isolation Forest	1.000	0.000	68.515	181.485

Table B.128: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+2.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.985	0.269	0.985	0.398
Twitter's method	0.995	0.091	0.130	0.099
STL	0.997	0.697	0.866	0.743
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.933	0.086	0.954	0.155
Yakamoz_penultimate_moving_blocks_matrix	0.789	0.028	0.979	0.054
Isolation Forest	0.727	0.015	1.000	0.029

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.129: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+2.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.994	0.006	3.911	246.089
Twitter's method	0.168	0.832	0.313	249.687
STL	0.894	0.106	0.558	249.442
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.993	0.007	16.214	233.786
Yakamoz_penultimate_moving_blocks_matrix	0.998	0.002	52.493	197.507
Isolation Forest	1.000	0.000	67.094	182.906

Table B.130: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+2.5_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.269	0.994	0.396
Twitter's method	0.995	0.123	0.168	0.132
STL	0.997	0.728	0.894	0.773
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.935	0.090	0.993	0.163
Yakamoz_penultimate_moving_blocks_matrix	0.791	0.029	0.998	0.056
Isolation Forest	0.733	0.015	1.000	0.029

Simulation number: 1000
T=250 p1=-0.9 p2=0.4 data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.131: Confusion matrix for nonstationary time series ARMA(1,1) max(data)+3.0_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.995	0.005	3.905	246.095
Twitter's method	0.197	0.803	0.323	249.677
STL	0.930	0.070	0.528	249.472
F-test	0.000	1.000	1.000	249.000
Yakamoz_last_moving_blocks_matrix	0.986	0.014	16.020	233.980
Yakamoz_penultimate_moving_blocks_matrix	0.998	0.002	51.802	198.198
Isolation Forest	1.000	0.000	66.542	183.458

Table B.132: Evaluation metrics for nonstationary time series ARMA(1,1) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.984	0.268	0.995	0.396
Twitter's method	0.996	0.151	0.197	0.160
STL	0.998	0.768	0.930	0.812
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.936	0.090	0.986	0.162
Yakamoz_penultimate_moving_blocks_matrix	0.794	0.029	0.998	0.056
Isolation Forest	0.735	0.015	1.000	0.030

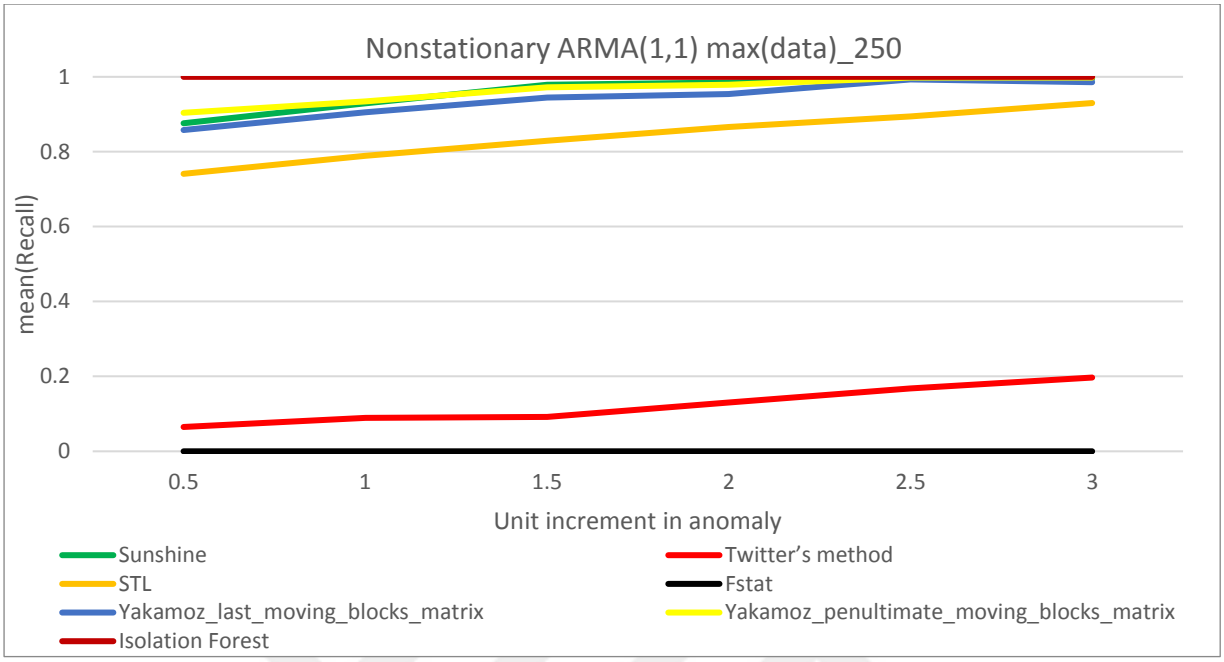

Figure B.21: Changes in mean(Recall) for nonstationary ARMA(1,1) max(data)_250 data

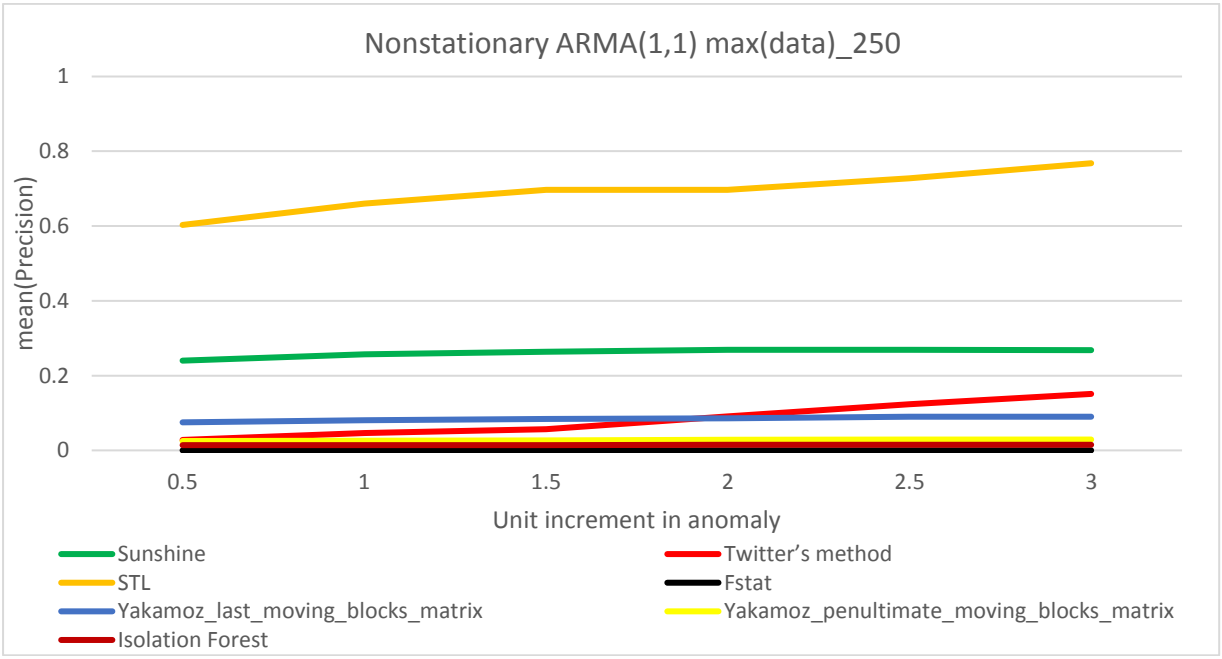

Figure B.22: Changes in mean(Precision) for nonstationary ARMA(1,1) max(data)_250 data

B.12 Simulation results for time series ARCH(2) with anomaly place 12

Simulation number: 1000
T=250 alpa=c(0.9,0.01)
data=garch.sim(alpha=alpa,n=T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.133: Confusion matrix for time series ARCH(2) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.962	0.038	5.106	243.894
Twitter's method	0.302	0.698	0.157	248.843
STL	0.399	0.601	0.331	248.669
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.820	0.180	20.663	228.337
Yakamoz_penultimate_moving_blocks_matrix	0.879	0.121	55.330	193.670
Isolation Forest	1.000	0.000	63.489	185.511

Table B.134: Evaluation metrics for time series ARCH(2) max(data)+0.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.190	0.962	0.306
Twitter's method	0.997	0.252	0.302	0.268
STL	0.996	0.303	0.399	0.331
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.917	0.065	0.820	0.119
Yakamoz_penultimate_moving_blocks_matrix	0.778	0.024	0.879	0.045
Isolation Forest	0.746	0.016	1.000	0.031

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.135: Confusion matrix for time series ARCH(2) max(data)+1.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.061	243.939
Twitter's method	0.700	0.300	0.135	248.865
STL	0.785	0.215	0.300	248.700
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.944	0.056	19.003	229.997
Yakamoz_penultimate_moving_blocks_matrix	0.981	0.019	57.051	191.949
Isolation Forest	1.000	0.000	62.412	186.588

Table B.136: Evaluation metrics for time series ARCH(2) max(data)+1.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.197	1.000	0.318
Twitter's method	0.998	0.641	0.700	0.660
STL	0.998	0.666	0.785	0.702
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.924	0.079	0.944	0.144
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.026	0.981	0.050
Isolation Forest	0.750	0.016	1.000	0.031

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.137: Confusion matrix for time series ARCH(2) max(data)+1.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.208	243.792
Twitter's method	0.949	0.051	0.144	248.856
STL	0.974	0.026	0.379	248.621
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.974	0.026	17.912	231.088
Yakamoz_penultimate_moving_blocks_matrix	0.998	0.002	60.396	188.604
Isolation Forest	1.000	0.000	61.041	187.959

Table B.138: Evaluation metrics for time series ARCH(2) max(data)+1.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.194	1.000	0.314
Twitter's method	0.999	0.884	0.949	0.905
STL	0.998	0.826	0.974	0.869
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.928	0.082	0.974	0.149
Yakamoz_penultimate_moving_blocks_matrix	0.758	0.026	0.998	0.050
Isolation Forest	0.756	0.016	1.000	0.032

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.139: Confusion matrix for time series ARCH(2) max(data)+2.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.260	243.740
Twitter's method	0.999	0.001	0.146	248.854
STL	0.999	0.001	0.324	248.676
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.987	0.013	17.912	231.088
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	59.060	189.940
Isolation Forest	1.000	0.000	59.456	189.544

Table B.140: Evaluation metrics for time series ARCH(2) max(data)+2.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.979	0.189	1.000	0.308
Twitter's method	0.999	0.934	0.999	0.954
STL	0.999	0.862	0.999	0.904
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.928	0.084	0.987	0.152
Yakamoz_penultimate_moving_blocks_matrix	0.764	0.028	1.000	0.053
Isolation Forest	0.762	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.141: Confusion matrix for time series ARCH(2) max(data)+2.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	5.372	243.628
Twitter's method	1.000	0.000	0.138	248.862
STL	1.000	0.000	0.307	248.693
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.999	0.001	18.035	230.965
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	60.458	188.542
Isolation Forest	1.000	0.000	58.348	190.652

Table B.142: Evaluation metrics for time series ARCH(2) max(data)+2.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.184	1.000	0.303
Twitter's method	0.999	0.938	1.000	0.958
STL	0.999	0.875	1.000	0.913
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.928	0.085	0.999	0.154
Yakamoz_penultimate_moving_blocks_matrix	0.758	0.026	1.000	0.050
Isolation Forest	0.767	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.143: Confusion matrix for time series ARCH(2) max(data)+3.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.420	243.580
Twitter's method	1.000	0.000	0.134	248.866
STL	1.000	0.000	0.303	248.697
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.998	0.002	17.167	231.833
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	59.744	189.256
Isolation Forest	1.000	0.000	57.867	191.133

Table B.144: Evaluation metrics for time series ARCH(2) max(data)+3.0_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.978	0.182	1.000	0.300
Twitter's method	0.999	0.940	1.000	0.959
STL	0.999	0.871	1.000	0.911
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.931	0.087	0.998	0.158
Yakamoz_penultimate_moving_blocks_matrix	0.761	0.026	1.000	0.050
Isolation Forest	0.769	0.017	1.000	0.034

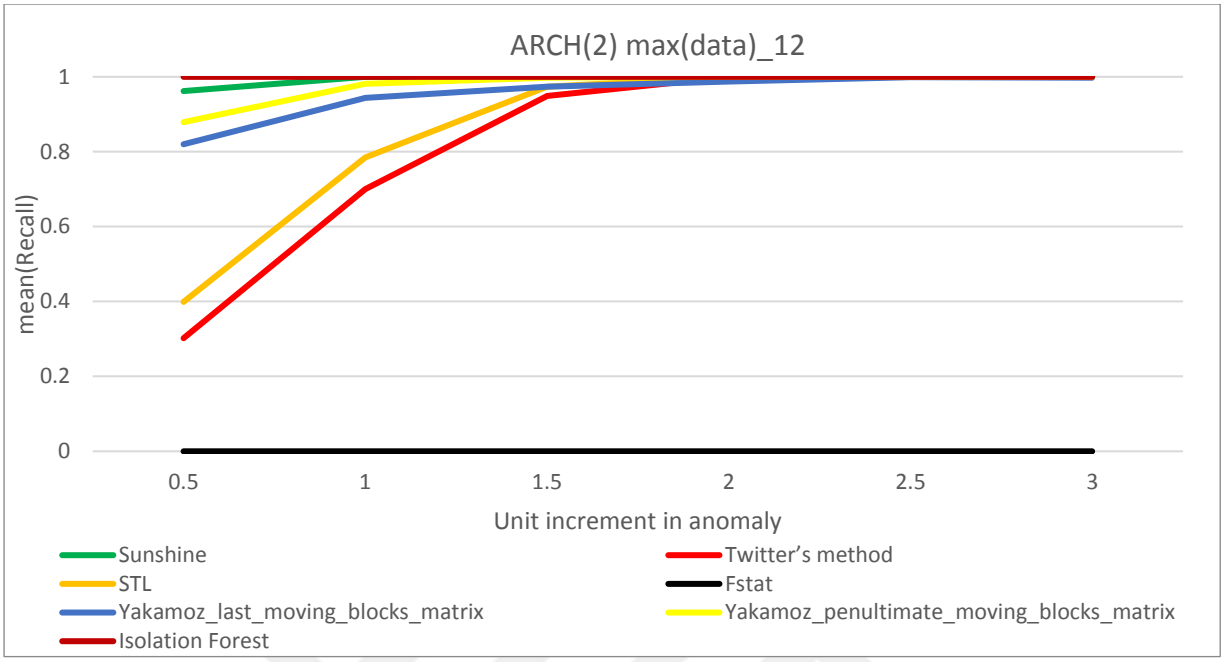

Figure B.23: Changes in mean(Recall) for ARCH(2) max(data)_12 data

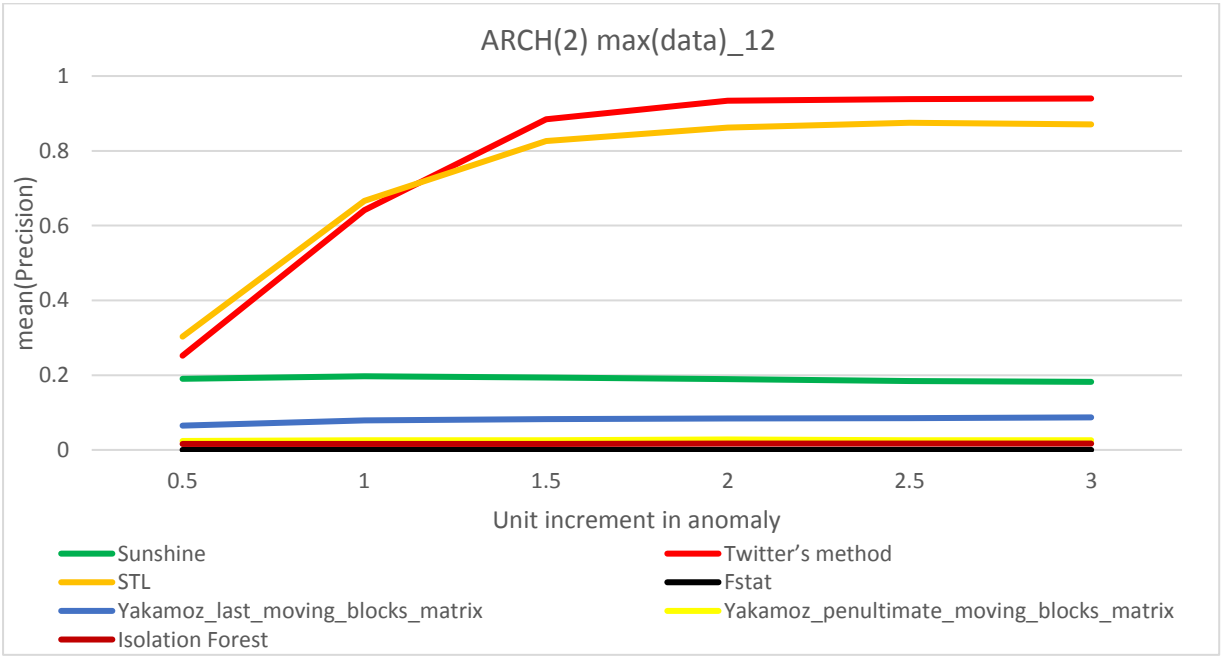

Figure B.24: Changes in mean(Precision) for ARCH(2) max(data)_12 data

B.13 Simulation results for time series ARCH(2) with anomaly place 250

Simulation number: 1000
T=250 alpa=c(0.9,0.01)
data=garch.sim(alpha=alpa,n=T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.145: Confusion matrix for time series ARCH(2) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.841	0.159	5.511	243.489
Twitter's method	0.266	0.734	0.138	248.862
STL	0.410	0.590	0.317	248.683
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.543	0.457	25.289	223.711
Yakamoz_penultimate_moving_blocks_matrix	0.616	0.384	55.913	193.087
Isolation Forest	1.000	0.000	63.487	185.513

Table B.146: Evaluation metrics for time series ARCH(2) max(data)+0.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.977	0.151	0.841	0.250
Twitter's method	0.997	0.224	0.266	0.237
STL	0.996	0.322	0.410	0.349
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.897	0.040	0.543	0.073
Yakamoz_penultimate_moving_blocks_matrix	0.775	0.016	0.616	0.031
Isolation Forest	0.746	0.016	1.000	0.031

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.147: Confusion matrix for time series ARCH(2) max(data)+1.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.995	0.005	5.302	243.698
Twitter's method	0.670	0.330	0.150	248.850
STL	0.719	0.281	0.303	248.697
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.761	0.239	21.391	227.609
Yakamoz_penultimate_moving_blocks_matrix	0.838	0.162	56.291	192.709
Isolation Forest	1.000	0.000	62.298	186.702

Table B.148: Evaluation metrics for time series ARCH(2) max(data)+1.0_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.183	0.995	0.301
Twitter's method	0.998	0.611	0.670	0.629
STL	0.998	0.602	0.719	0.638
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.913	0.061	0.761	0.111
Yakamoz_penultimate_moving_blocks_matrix	0.774	0.022	0.838	0.042
Isolation Forest	0.751	0.016	1.000	0.031

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.149: Confusion matrix for time series ARCH(2) max(data)+1.5_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.222	243.778
Twitter's method	0.954	0.046	0.119	248.881
STL	0.930	0.070	0.298	248.702
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.889	0.111	21.277	227.723
Yakamoz_penultimate_moving_blocks_matrix	0.963	0.037	56.759	192.241
Isolation Forest	1.000	0.000	60.977	188.023

Table B.150: Evaluation metrics for time series ARCH(2) max(data)+1.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.187	1.000	0.307
Twitter's method	0.999	0.899	0.954	0.917
STL	0.999	0.809	0.930	0.846
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.914	0.072	0.889	0.130
Yakamoz_penultimate_moving_blocks_matrix	0.773	0.026	0.963	0.050
Isolation Forest	0.756	0.016	1.000	0.032

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.151: Confusion matrix for time series ARCH(2) max(data)+2.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.190	243.810
Twitter's method	0.998	0.002	0.149	248.851
STL	0.991	0.009	0.303	248.697
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.931	0.069	20.474	228.526
Yakamoz_penultimate_moving_blocks_matrix	0.993	0.007	58.767	190.233
Isolation Forest	1.000	0.000	59.689	189.311

Table B.152: Evaluation metrics for time series ARCH(2) max(data)+2.0_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.190	1.000	0.312
Twitter's method	0.999	0.931	0.998	0.952
STL	0.999	0.866	0.991	0.904
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.918	0.074	0.931	0.135
Yakamoz_penultimate_moving_blocks_matrix	0.765	0.025	0.993	0.049
Isolation Forest	0.761	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.153: Confusion matrix for time series ARCH(2) max(data)+2.5_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.120	243.880
Twitter's method	1.000	0.000	0.149	248.851
STL	0.999	0.001	0.305	248.695
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.933	0.067	20.998	228.002
Yakamoz_penultimate_moving_blocks_matrix	0.994	0.006	57.372	191.628
Isolation Forest	0.994	0.006	57.372	191.628

Table B.154: Evaluation metrics for time series ARCH(2) max(data)+2.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.191	1.000	0.313
Twitter's method	0.999	0.932	1.000	0.954
STL	0.999	0.871	0.999	0.910
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.916	0.074	0.933	0.134
Yakamoz_penultimate_moving_blocks_matrix	0.770	0.027	0.994	0.051
Isolation Forest	0.764	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.01) data=garch.sim(alpha=alpa,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.155: Confusion matrix for time series ARCH(2) max(data)+3.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.108	243.892
Twitter's method	1.000	0.000	0.152	248.848
STL	1.000	0.000	0.306	248.694
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.951	0.049	20.496	228.504
Yakamoz_penultimate_moving_blocks_matrix	0.992	0.008	56.901	192.099
Isolation Forest	1.000	0.000	58.324	190.676

Table B.156: Evaluation metrics for time series ARCH(2) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.190	1.000	0.312
Twitter's method	0.999	0.933	1.000	0.954
STL	0.999	0.874	1.000	0.912
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.918	0.077	0.951	0.140
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.026	0.992	0.050
Isolation Forest	0.767	0.017	1.000	0.033

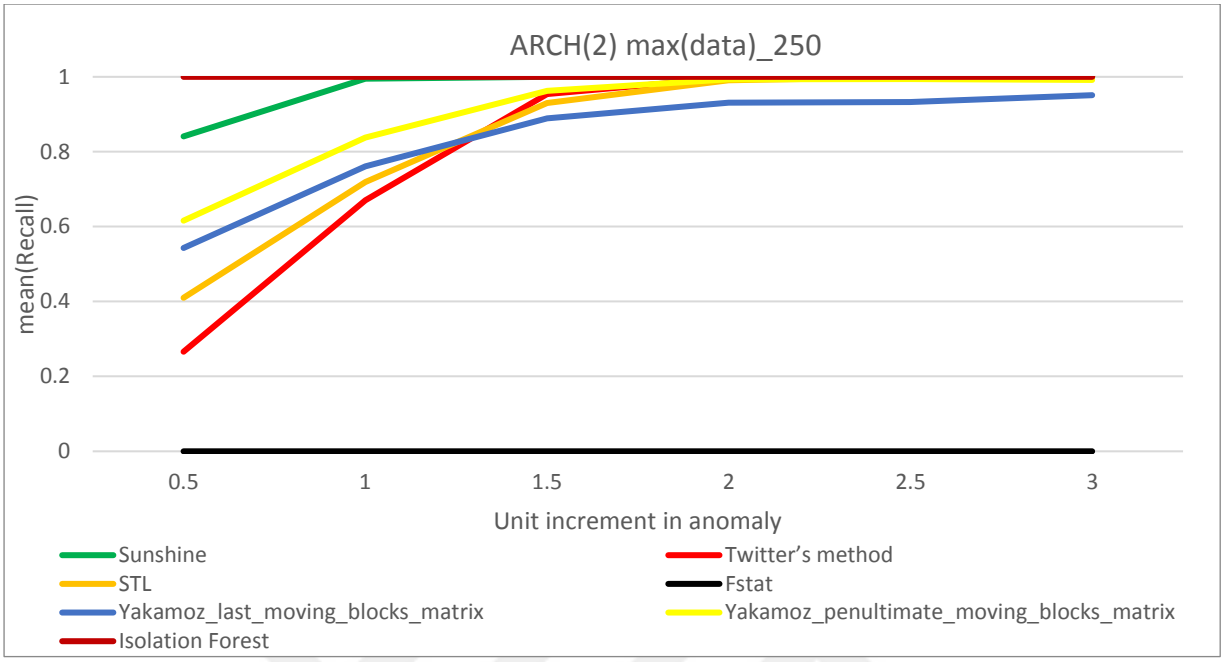

Figure B.25: Changes in mean(Recall) for ARCH(2) max(data)_250 data

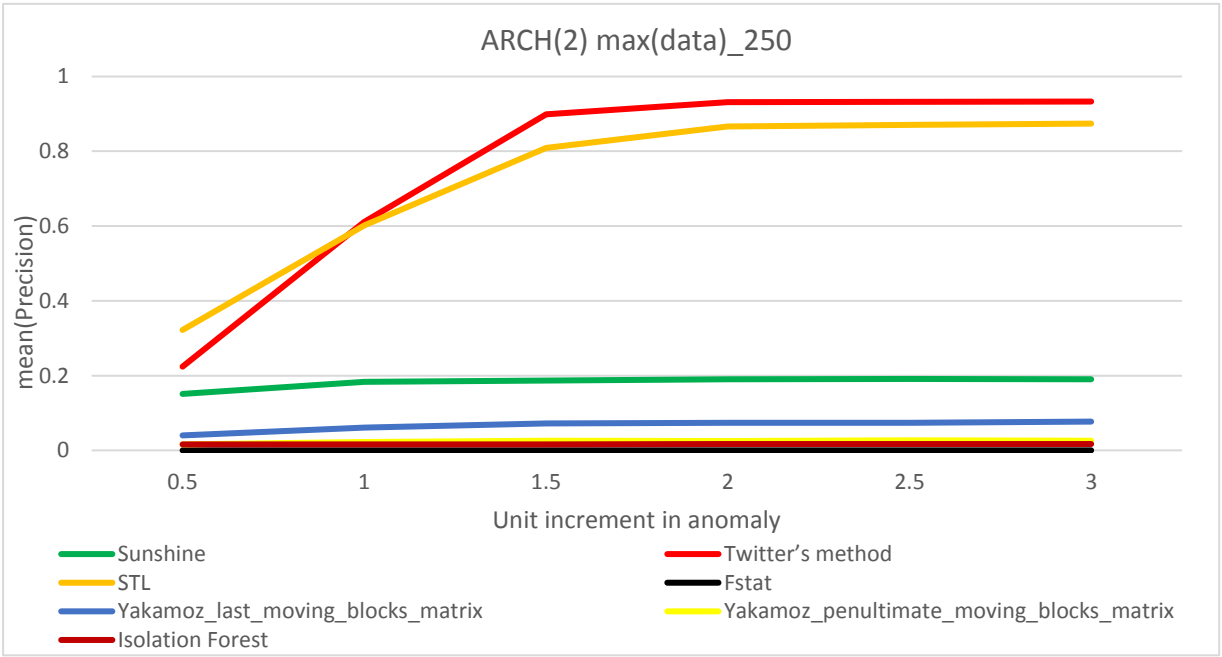

Figure B.26: Changes in mean(Precision) for ARCH(2) max(data)_250 data

B.14 Simulation results for time series GARCH(1,2) with anomaly place 12

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09)
data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=12
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.157: Confusion matrix for time series GARCH(1,2) max(data)+0.5_12 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.949	0.051	5.082	243.918
Twitter's method	0.338	0.662	0.209	248.791
STL	0.463	0.537	0.415	248.585
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.832	0.168	18.728	230.272
Yakamoz_penultimate_moving_blocks_matrix	0.895	0.105	56.964	192.036
Isolation Forest	1.000	0.000	63.310	185.690

Table B.158: Evaluation metrics for time series GARCH(1,2) max(data)+0.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.191	0.949	0.306
Twitter's method	0.997	0.269	0.338	0.290
STL	0.996	0.336	0.463	0.373
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.924	0.068	0.832	0.123
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.027	0.895	0.050
Isolation Forest	0.747	0.016	1.000	0.031

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.159: Confusion matrix for time series GARCH(1,2) max(data)+1.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.036	243.964
Twitter's method	0.717	0.283	0.180	248.820
STL	0.800	0.200	0.384	248.616
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.924	0.076	16.373	232.627
Yakamoz_penultimate_moving_blocks_matrix	0.973	0.027	58.865	190.135
Isolation Forest	1.000	0.000	61.922	187.078

Table B.160: Evaluation metrics for time series GARCH(1,2) max(data)+1.0_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.202	1.000	0.324
Twitter's method	0.998	0.644	0.717	0.667
STL	0.998	0.656	0.800	0.699
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.934	0.078	0.924	0.142
Yakamoz_penultimate_moving_blocks_matrix	0.764	0.028	0.973	0.053
Isolation Forest	0.752	0.016	1.000	0.032

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.161: Confusion matrix for time series GARCH(1,2) max(data)+1.5_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.213	243.787
Twitter's method	0.953	0.047	0.205	248.795
STL	0.975	0.025	0.408	248.592
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.969	0.031	15.670	233.330
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	59.850	189.150
Isolation Forest	1.000	0.000	60.684	188.316

Table B.162: Evaluation metrics for time series GARCH(1,2) max(data)+1.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.195	1.000	0.315
Twitter's method	0.999	0.866	0.953	0.893
STL	0.998	0.818	0.975	0.864
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.937	0.082	0.969	0.150
Yakamoz_penultimate_moving_blocks_matrix	0.761	0.028	1.000	0.054
Isolation Forest	0.757	0.016	1.000	0.032

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.163: Confusion matrix for time series GARCH(1,2) max(data)+2.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.307	243.693
Twitter's method	0.999	0.001	0.161	248.839
STL	1.000	0.000	0.338	248.662
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.983	0.017	15.516	233.484
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	62.191	186.809
Isolation Forest	1.000	0.000	59.247	189.753

Table B.164: Evaluation metrics for time series GARCH(1,2) max(data)+2.0_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.191	1.000	0.309
Twitter's method	0.999	0.924	0.999	0.948
STL	0.999	0.860	1.000	0.903
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.938	0.084	0.983	0.153
Yakamoz_penultimate_moving_blocks_matrix	0.751	0.027	1.000	0.052
Isolation Forest	0.763	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.165: Confusion matrix for time series GARCH(1,2) max(data)+2.5_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.268	243.732
Twitter's method	1.000	0.000	0.203	248.797
STL	1.000	0.000	0.430	248.570
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.995	0.005	15.688	233.312
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	61.108	187.892
Isolation Forest	1.000	0.000	58.069	190.931

Table B.166: Evaluation metrics for time series GARCH(1,2) max(data)+2.5_12 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.195	1.000	0.315
Twitter's method	0.999	0.909	1.000	0.938
STL	0.998	0.822	1.000	0.876
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.937	0.085	0.995	0.155
Yakamoz_penultimate_moving_blocks_matrix	0.756	0.027	1.000	0.052
Isolation Forest	0.768	0.017	1.000	0.034

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=12 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.167: Confusion matrix for time series GARCH(1,2) max(data)+3.0_12 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.402	243.598
Twitter's method	1.000	0.000	0.181	248.819
STL	1.000	0.000	0.354	248.646
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.998	0.002	15.870	233.130
Yakamoz_penultimate_moving_blocks_matrix	1.000	0.000	60.681	188.319
Isolation Forest	1.000	0.000	57.588	191.412

Table B.168: Evaluation metrics for time series GARCH(1,2) max(data)+3.0_12 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.978	0.189	1.000	0.306
Twitter's method	0.999	0.918	1.000	0.944
STL	0.999	0.856	1.000	0.899
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.937	0.086	0.998	0.157
Yakamoz_penultimate_moving_blocks_matrix	0.757	0.028	1.000	0.054
Isolation Forest	0.770	0.017	1.000	0.034

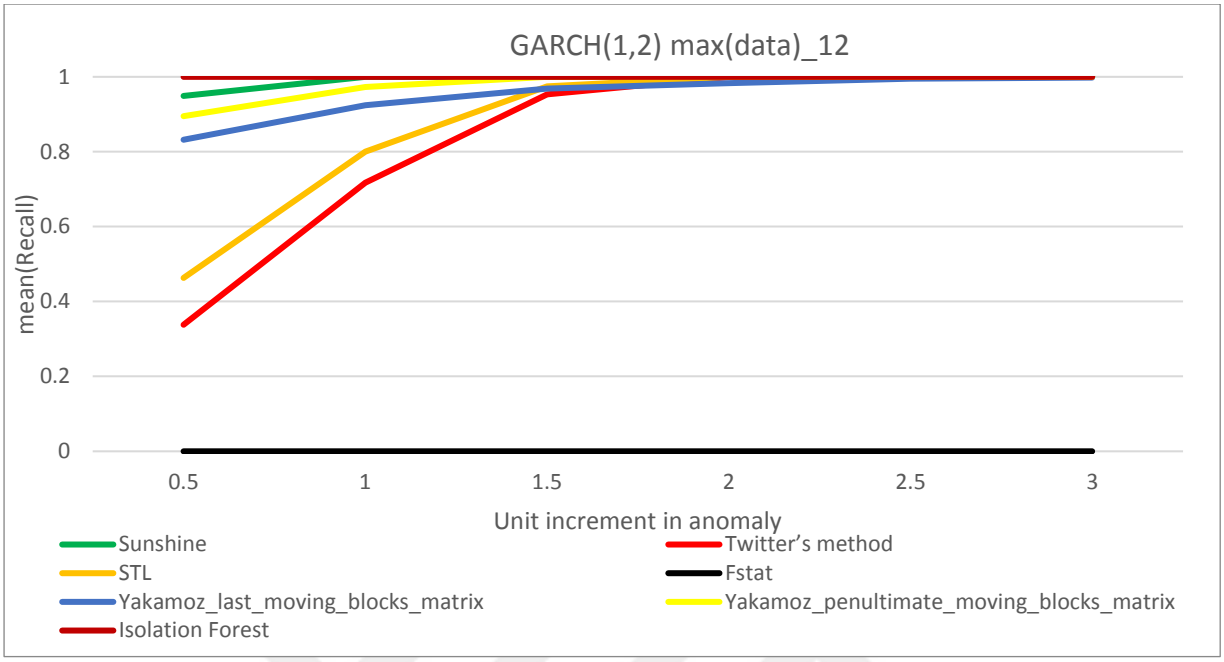

Figure B.27: Changes in mean(Recall) for GARCH(1,2) max(data)_12 data

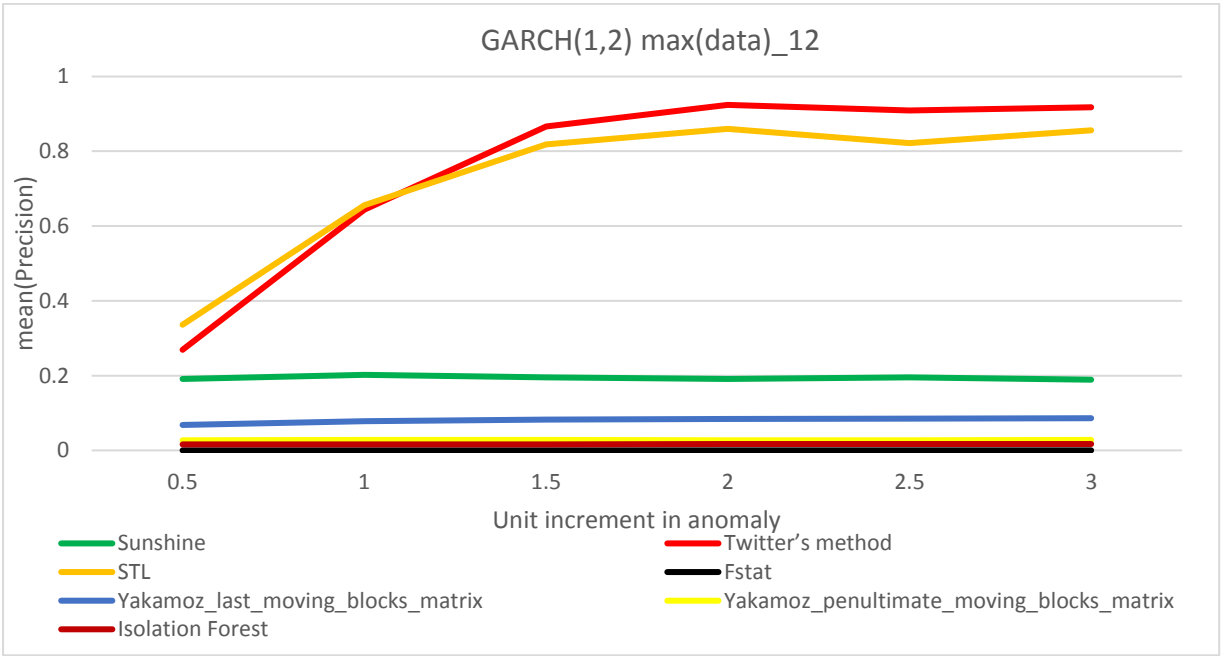

Figure B.28: Changes in mean(Precision) for GARCH(1,2) max(data)_12 data

B.15 Simulation results for time series GARCH(1,2) with anomaly place 250

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09)
data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=250
data[anomaly_place1]=max(data)+0.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.169: Confusion matrix for time series GARCH(1,2) max(data)+0.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	0.813	0.187	5.363	243.637
Twitter's method	0.306	0.694	0.184	248.816
STL	0.434	0.566	0.390	248.610
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.512	0.488	20.865	228.135
Yakamoz_penultimate_moving_blocks_matrix	0.606	0.394	56.178	192.822
Isolation Forest	1.000	0.000	63.285	185.715

Table B.170: Evaluation metrics for time series GARCH(1,2) max(data)+0.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.978	0.153	0.813	0.250
Twitter's method	0.996	0.246	0.306	0.264
STL	0.996	0.323	0.434	0.356
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.915	0.041	0.512	0.074
Yakamoz_penultimate_moving_blocks_matrix	0.774	0.017	0.606	0.032
Isolation Forest	0.747	0.016	1.000	0.031

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.171: Confusion matrix for time series GARCH(1,2) max(data)+1.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	0.989	0.011	5.263	243.737
Twitter's method	0.680	0.320	0.209	248.791
STL	0.733	0.267	0.396	248.604
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.720	0.280	19.961	229.039
Yakamoz_penultimate_moving_blocks_matrix	0.818	0.182	58.534	190.466
Isolation Forest	1.000	0.000	62.007	186.993

Table B.172: Evaluation metrics for time series GARCH(1,2) max(data)+1.0_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.979	0.185	0.989	0.304
Twitter's method	0.998	0.595	0.680	0.622
STL	0.997	0.590	0.733	0.632
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.919	0.058	0.720	0.105
Yakamoz_penultimate_moving_blocks_matrix	0.765	0.022	0.818	0.041
Isolation Forest	0.752	0.016	1.000	0.032

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+1.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.173: Confusion matrix for time series GARCH(1,2) max(data)+1.5_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.069	243.931
Twitter's method	0.953	0.047	0.173	248.827
STL	0.922	0.078	0.390	248.610
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.852	0.148	17.538	231.462
Yakamoz_penultimate_moving_blocks_matrix	0.951	0.049	59.078	189.922
Isolation Forest	1.000	0.000	60.660	188.340

Table B.174: Evaluation metrics for time series GARCH(1,2) max(data)+1.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.202	1.000	0.324
Twitter's method	0.999	0.877	0.953	0.901
STL	0.998	0.770	0.922	0.815
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.929	0.070	0.852	0.127
Yakamoz_penultimate_moving_blocks_matrix	0.763	0.026	0.951	0.049
Isolation Forest	0.757	0.016	1.000	0.032

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.175: Confusion matrix for time series GARCH(1,2) max(data)+2.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	5.016	243.984
Twitter's method	0.997	0.003	0.189	248.811
STL	0.992	0.008	0.384	248.616
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.914	0.086	17.728	231.272
Yakamoz_penultimate_moving_blocks_matrix	0.984	0.016	58.966	190.034
Isolation Forest	1.000	0.000	59.676	189.324

Table B.176: Evaluation metrics for time series GARCH(1,2) max(data)+2.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.200	1.000	0.321
Twitter's method	0.999	0.914	0.997	0.940
STL	0.998	0.841	0.992	0.885
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.929	0.077	0.914	0.139
Yakamoz_penultimate_moving_blocks_matrix	0.764	0.027	0.984	0.052
Isolation Forest	0.761	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+2.5
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.177: Confusion matrix for time series GARCH(1,2) max(data)+2.5_250 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.000	0.000	4.950	244.050
Twitter's method	1.000	0.000	0.195	248.805
STL	1.000	0.000	0.405	248.595
F-test	0.000	1.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	0.925	0.075	17.329	231.671
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.009	57.461	191.539
Isolation Forest	1.000	0.000	58.332	190.668

Table B.178: Evaluation metrics for time series GARCH(1,2) max(data)+2.5_250 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.980	0.209	1.000	0.331
Twitter's method	0.999	0.915	1.000	0.941
STL	0.998	0.836	1.000	0.885
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.930	0.077	0.925	0.140
Yakamoz_penultimate_moving_blocks_matrix	0.770	0.028	0.991	0.053
Isolation Forest	0.767	0.017	1.000	0.033

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09) data=garch.sim(alpha=alpa,beta=bet,n=T)
anomaly_place1=250 data[anomaly_place1]=max(data)+3.0
sunshine(data,20,0.5,0.5,0.05,1) yakamoz(data,12,0.5,0.5,0.05)

Table B.179: Confusion matrix for time series GARCH(1,2) max(data)+3.0_250 data

	mean(TP)	mean(FN)	mean(FP)	mean(TN)
Sunshine	1.000	0.000	4.918	244.082
Twitter's method	1.000	0.000	0.166	248.834
STL	1.000	0.000	0.389	248.611
F-test	0.000	1.000	1.0000	248.000
Yakamoz_last_moving_blocks_matrix	0.932	0.068	16.886	232.114
Yakamoz_penultimate_moving_blocks_matrix	0.991	0.009	58.133	190.867
Isolation Forest	1.000	0.000	58.332	190.668

Table B.180: Evaluation metrics for time series GARCH(1,2) max(data)+3.0_250 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.980	0.204	1.000	0.328
Twitter's method	0.999	0.925	1.000	0.949
STL	0.998	0.843	1.000	0.890
F-test	0.992	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.932	0.078	0.932	0.142
Yakamoz_penultimate_moving_blocks_matrix	0.767	0.028	0.991	0.054
Isolation Forest	0.767	0.017	1.000	0.033

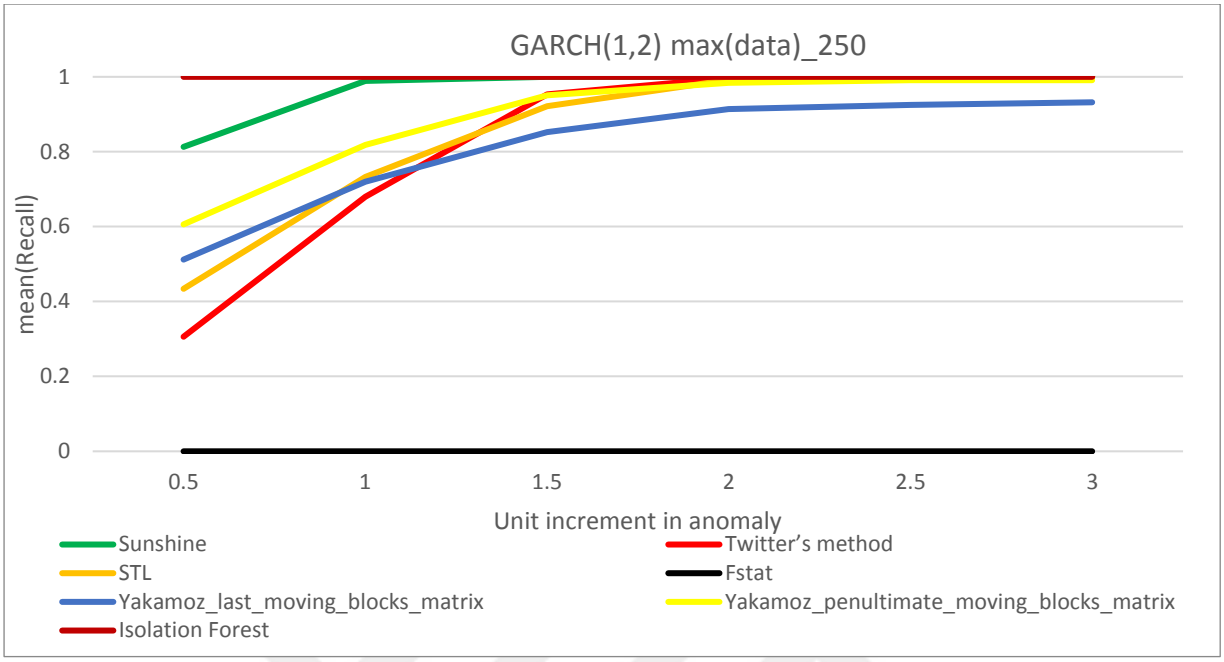

Figure B.29: Changes in mean(Recall) for GARCH(1,2) max(data)_250 data

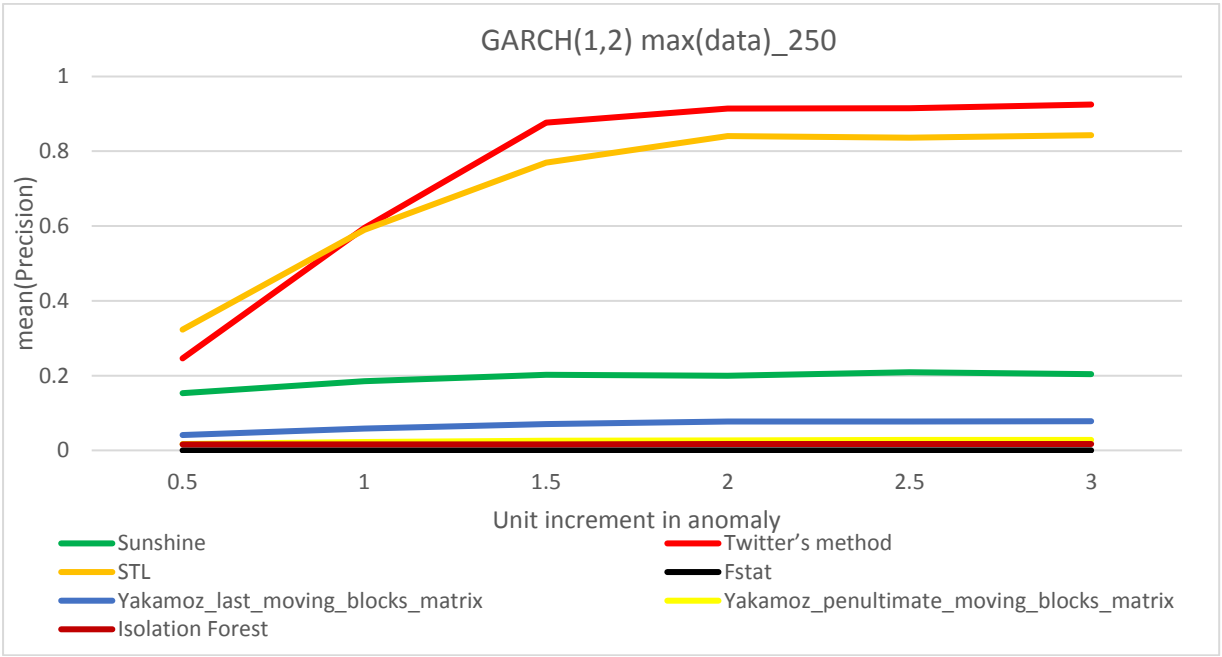

Figure B.30: Changes in mean(Precision) for GARCH(1,2) max(data)_250 data

B.16 Simulation results for stationary time series AR(1) with anomaly places 12 and 250

Simulation number: 1000
T=250 p=-0.9
data=arima.sim(model = list(order = c(1, 0, 0),ar=p),n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.181: Confusion matrix for stationary time series AR(1) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.801	0.199	2.158	245.842
Twitter's method	0.353	1.647	0.143	247.857
STL	0.401	1.599	1.117	246.883
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.803	0.197	11.635	236.365
Yakamoz_penultimate_moving_blocks_matrix	1.969	0.031	56.999	191.001
Isolation Forest	2.000	0.000	63.206	184.794

Table B.182: Evaluation metrics for stationary time series AR(1) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.991	0.518	0.901	0.632
Twitter's method	0.993	0.178	0.177	0.166
STL	0.989	0.140	0.201	0.141
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.953	0.153	0.902	0.257
Yakamoz_penultimate_moving_blocks_matrix	0.772	0.058	0.985	0.107
Isolation Forest	0.747	0.031	1.000	0.060

B.17 Simulation results for nonstationary time series AR(1) with anomaly places 12 and 250

Simulation number: 1000
T=250 p=-0.9
data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.183: Confusion matrix for nonstationary time series AR(1) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.496	0.504	2.254	246.746
Twitter's method	0.205	1.795	0.223	248.777
STL	1.560	0.440	1.054	247.946
F-test	0.000	2.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	1.730	0.270	17.634	231.366
Yakamoz_penultimate_moving_blocks_matrix	1.904	0.096	54.628	194.372
Isolation Forest	2.000	0.000	61.283	187.717

Table B.184: Evaluation metrics for nonstationary time series AR(1) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.989	0.464	0.748	0.544
Twitter's method	0.992	0.090	0.103	0.090
STL	0.994	0.786	0.780	0.730
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.929	0.150	0.865	0.249
Yakamoz_penultimate_moving_blocks_matrix	0.782	0.056	0.952	0.102
Isolation Forest	0.756	0.032	1.000	0.062

Simulation number: 1000
T=250 p=-0.9
data=arima.sim(model = list(order = c(1, 1, 0),ar=p),n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,2)
yakamoz(data,12,0.5,0.5,0.05)

Table B.185: Confusion matrix for nonstationary time series AR(1) 12_250_trim2 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.821	0.179	6.013	242.987
Twitter's method	0.191	1.809	0.206	248.794
STL	1.556	0.444	1.237	247.763
F-test	0.000	2.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	1.699	0.301	18.170	230.830
Yakamoz_penultimate_moving_blocks_matrix	1.886	0.114	53.876	195.124
Isolation Forest	2.000	0.000	60.984	188.016

Table B.186: Evaluation metrics for nonstationary time series AR(1) 12_250_trim2 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.975	0.264	0.911	0.396
Twitter's method	0.992	0.088	0.096	0.086
STL	0.993	0.768	0.778	0.716
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.926	0.147	0.850	0.243
Yakamoz_penultimate_moving_blocks_matrix	0.785	0.056	0.943	0.103
Isolation Forest	0.757	0.032	1.000	0.062

B.18 Simulation results for stationary time series AR(2) with anomaly places 12 and 250

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 0, 0),ar=p),n = T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.187: Confusion matrix for stationary time series AR(2) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.919	0.081	2.666	245.334
Twitter's method	1.067	0.933	0.131	247.869
STL	1.169	0.831	0.334	247.666
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.747	0.253	12.609	235.391
Yakamoz_penultimate_moving_blocks_matrix	1.943	0.057	57.503	190.497
Isolation Forest	2.000	0.000	58.719	189.281

Table B.188: Evaluation metrics for stationary time series AR(2) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.989	0.489	0.960	0.621
Twitter's method	0.996	0.635	0.534	0.557
STL	0.995	0.644	0.585	0.583
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.949	0.153	0.874	0.255
Yakamoz_penultimate_moving_blocks_matrix	0.770	0.059	0.972	0.106
Isolation Forest	0.765	0.033	1.000	0.064

B.19 Simulation results for nonstationary time series AR(2) with anomaly places 12 and 250

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.189: Confusion matrix for nonstationary time series AR(2) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.487	0.513	1.971	247.029
Twitter's method	0.189	1.811	0.318	248.682
STL	1.699	0.301	0.532	248.468
F-test	0.000	2.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	1.789	0.211	11.744	237.256
Yakamoz_penultimate_moving_blocks_matrix	1.942	0.058	52.559	196.441
Isolation Forest	2.000	0.000	64.158	184.842

Table B.190: Evaluation metrics for nonstationary time series AR(2) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.990	0.495	0.744	0.564
Twitter's method	0.992	0.069	0.095	0.073
STL	0.997	0.856	0.850	0.817
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.952	0.159	0.895	0.265
Yakamoz_penultimate_moving_blocks_matrix	0.790	0.059	0.971	0.109
Isolation Forest	0.744	0.031	1.000	0.060

Simulation number: 1000
T=250 p=c(-0.5,0.2)
data=arima.sim(model = list(order = c(2, 1, 0),ar=p),n = T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,2)
yakamoz(data,12,0.5,0.5,0.05)

Table B.191: Confusion matrix for nonstationary time series AR(2) 12_250_trim2 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.731	0.269	5.571	243.429
Twitter's method	0.183	1.817	0.348	248.652
STL	1.695	0.305	0.508	248.492
F-test	0.000	2.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	1.782	0.218	11.603	237.397
Yakamoz_penultimate_moving_blocks_matrix	1.938	0.062	50.820	198.180
Isolation Forest	2.000	0.000	64.119	184.881

Table B.192: Evaluation metrics for nonstationary time series AR(2) 12_250_trim2 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.977	0.276	0.866	0.398
Twitter's method	0.991	0.062	0.092	0.069
STL	0.997	0.856	0.848	0.816
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.953	0.160	0.891	0.267
Yakamoz_penultimate_moving_blocks_matrix	0.797	0.060	0.969	0.111
Isolation Forest	0.745	0.031	1.000	0.060

B.20 Simulation results for stationary time series ARMA(1,1) with anomaly places 12 and 250

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 0, 1),ar=p1,ma=p2),n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.193: Confusion matrix for stationary time series ARMA(1,1) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.932	0.068	2.491	245.509
Twitter's method	0.752	1.248	0.138	247.862
STL	0.819	1.181	0.712	247.288
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.778	0.222	11.737	236.263
Yakamoz_penultimate_moving_blocks_matrix	1.946	0.054	56.781	191.219
Isolation Forest	2.000	0.000	59.722	188.278

Table B.194: Evaluation metrics for stationary time series ARMA(1,1) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.990	0.519	0.966	0.648
Twitter's method	0.994	0.432	0.376	0.384
STL	0.992	0.420	0.410	0.378
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.952	0.156	0.889	0.261
Yakamoz_penultimate_moving_blocks_matrix	0.773	0.059	0.973	0.107
Isolation Forest	0.761	0.033	1.000	0.063

B.21 Simulation results for nonstationary time series ARMA(1,1) with anomaly places 12 and 250

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.195: Confusion matrix for nonstationary time series ARMA(1,1) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.499	0.501	2.029	246.971
Twitter's method	0.175	1.825	0.334	248.666
STL	1.676	0.324	0.484	248.516
F-test	0.000	2.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	1.791	0.209	11.715	237.285
Yakamoz_penultimate_moving_blocks_matrix	1.941	0.059	50.203	198.797
Isolation Forest	2.000	0.000	63.980	185.020

Table B.196: Evaluation metrics for nonstationary time series ARMA(1,1) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.990	0.489	0.750	0.560
Twitter's method	0.991	0.069	0.088	0.070
STL	0.997	0.866	0.838	0.818
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.952	0.165	0.896	0.275
Yakamoz_penultimate_moving_blocks_matrix	0.800	0.061	0.971	0.111
Isolation Forest	0.745	0.031	1.000	0.060

Simulation number: 1000
T=250 p1=-0.9 p2=0.4
data=arima.sim(model = list(order = c(1, 1, 1),ar=p1,ma=p2),n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,2)
yakamoz(data,12,0.5,0.5,0.05)

Table B.197: Confusion matrix for nonstationary time series ARMA(1,1) 12_250_trim2 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.760	0.240	5.487	243.513
Twitter's method	0.185	1.815	0.339	248.661
STL	1.669	0.331	0.534	248.466
F-test	0.000	2.000	1.000	248.000
Yakamoz_last_moving_blocks_matrix	1.777	0.223	12.494	236.506
Yakamoz_penultimate_moving_blocks_matrix	1.936	0.064	53.068	195.932
Isolation Forest	2.000	0.000	63.487	185.513

Table B.198: Evaluation metrics for nonstationary time series ARMA(1,1) 12_250_trim2 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.977	0.281	0.880	0.408
Twitter's method	0.991	0.074	0.093	0.076
STL	0.997	0.856	0.835	0.809
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.949	0.160	0.889	0.267
Yakamoz_penultimate_moving_blocks_matrix	0.788	0.059	0.968	0.108
Isolation Forest	0.747	0.031	1.000	0.060

B.22 Simulation results for time series ARCH(2) with anomaly places 12 and 250

Simulation number: 1000
T=250 alpa=c(0.9,0.01)
data=garch.sim(alpha=alpa,n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.199: Confusion matrix for time series ARCH(2) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.965	0.035	2.483	245.517
Twitter's method	1.627	0.373	0.147	247.853
STL	1.703	0.297	0.336	247.664
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.743	0.257	13.063	234.937
Yakamoz_penultimate_moving_blocks_matrix	1.925	0.075	59.800	188.200
Isolation Forest	2.000	0.000	56.494	191.506

Table B.200: Evaluation metrics for time series ARCH(2) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.990	0.523	0.983	0.655
Twitter's method	0.998	0.895	0.814	0.829
STL	0.997	0.865	0.852	0.830
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.947	0.151	0.872	0.252
Yakamoz_penultimate_moving_blocks_matrix	0.761	0.053	0.963	0.097
Isolation Forest	0.774	0.035	1.000	0.067

,

Simulation number: 1000
T=250 alpa=c(0.9,0.01)
data=garch.sim(alpha=alpa,n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,2)
yakamoz(data,12,0.5,0.5,0.05)

Table B.201: Confusion matrix for time series ARCH(2) 12_250_trim2 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.994	0.006	7.022	240.978
Twitter's method	1.627	0.373	0.147	247.853
STL	1.703	0.297	0.336	247.664
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.743	0.257	13.063	234.937
Yakamoz_penultimate_moving_blocks_matrix	1.925	0.075	59.800	188.200
Isolation Forest	2.000	0.000	56.494	191.506

Table B.202: Evaluation metrics for time series ARCH(2) 12_250_trim2 data

	mean(Accuracy)	mean(Precision)	mean(Recall)	mean(F1 score)
Sunshine	0.972	0.257	0.997	0.396
Twitter's method	0.998	0.895	0.814	0.829
STL	0.997	0.865	0.852	0.830
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.947	0.151	0.872	0.252
Yakamoz_penultimate_moving_blocks_matrix	0.761	0.053	0.963	0.097
Isolation Forest	0.774	0.035	1.000	0.067

B.23 Simulation results for time series GARCH(1,2) with anomaly places 12 and 250

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09)
data=garch.sim(alpha=alpa,beta=bet,n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,1)
yakamoz(data,12,0.5,0.5,0.05)

Table B.203: Confusion matrix for time series GARCH(1,2) 12_250_trim1 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.954	0.046	2.202	245.798
Twitter's method	1.669	0.331	0.195	247.805
STL	1.773	0.227	0.402	247.598
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.672	0.328	12.954	235.046
Yakamoz_penultimate_moving_blocks_matrix	1.917	0.083	60.858	187.142
Isolation Forest	2.000	0.000	56.111	191.889

Table B.204: Evaluation metrics for time series GARCH(1,2) 12_250_trim1 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.991	0.560	0.977	0.681
Twitter's method	0.998	0.894	0.835	0.838
STL	0.997	0.859	0.887	0.845
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.947	0.137	0.836	0.231
Yakamoz_penultimate_moving_blocks_matrix	0.756	0.054	0.959	0.098
Isolation Forest	0.776	0.035	1.000	0.067

Simulation number: 1000
T=250 alpa=c(0.9,0.1) bet=c(-0.09)
data=garch.sim(alpha=alpa,beta=bet,n=T)
kk=max(data)
anomaly_place1=12
data[anomaly_place1]=kk+1.0
anomaly_place2=250
data[anomaly_place2]=kk+1.5
sunshine(data,20,0.5,0.5,0.05,2)
yakamoz(data,12,0.5,0.5,0.05)

Table B.205: Confusion matrix for time series GARCH(1,2) 12_250_trim2 data

	mean(TP)	**mean(FN)**	**mean(FP)**	**mean(TN)**
Sunshine	1.998	0.002	7.084	240.916
Twitter's method	1.643	0.357	0.177	247.823
STL	1.732	0.268	0.377	247.623
F-test	0.000	2.000	1.000	247.000
Yakamoz_last_moving_blocks_matrix	1.671	0.329	12.702	235.298
Yakamoz_penultimate_moving_blocks_matrix	1.912	0.088	58.433	189.567
Isolation Forest	2.000	0.000	56.252	191.748

Table B.206: Evaluation metrics for time series GARCH(1,2) 12_250_trim2 data

	mean(Accuracy)	**mean(Precision)**	**mean(Recall)**	**mean(F1 score)**
Sunshine	0.972	0.256	0.999	0.395
Twitter's method	0.998	0.888	0.822	0.830
STL	0.997	0.841	0.866	0.828
F-test	0.988	0.000	0.000	0.000
Yakamoz_last_moving_blocks_matrix	0.948	0.139	0.836	0.234
Yakamoz_penultimate_moving_blocks_matrix	0.766	0.058	0.956	0.104
Isolation Forest	0.775	0.035	1.000	0.067

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name:
Nationality:
Date and Place of Birth:
Phone:
E-mail:

EDUCATION

Degree	**Institution**	**Year of Graduation**
Ph.D.	METU Statistics CGPA: 3.50/4	2021
M.S.	METU Statistics CGPA: 3.45/4	2015

Thesis Title: Time Series Analysis of Turkish Exports of Selected Ores and Concentrates

B.S.	METU Statistics CGPA: 3.71/4 (First rank graduate)	2013
High School	Darende High School	2007

MAJOR ACHIEVEMENTS

- Graduated as a High Honor Student from METU, Ranked 1st in the Department of Statistics (2013)

• Ranked 2nd (3rd Group) in the Written Exam of Assistant Specialist of Central Bank of the Republic of Turkey (2018)

• Ranked 1st (Statistics Section) in the Written Exam of Foreign Trade Assistant Expert of Rebuplic of Turkey Ministry of Economy (2013)

• Ranked 1st (Statistics Section) in the Interview of Foreign Trade Assistant Expert of Rebuplic of Turkey Ministry of Economy (2013)

• Ranked 1st (Statistics Section) in the Written Exam of Assistant Treasury Expert of Rebuplic of Turkey Prime Ministry Undersecretariat of Treasury (2013)

• Ranked 1st (Statistics Section) in the Interview of Assistant Treasury Expert of Rebuplic of Turkey Prime Ministry Undersecretariat of Treasury (2013)

• Ranked 3rd among 64025 Students (Statistics Score) in the Public Personnel Selection Exam (KPSS, 2014)

WORK EXPERIENCE

Year	**Place**	**Enrollment**
2019-Present	Central Bank of the Republic of Turkey	Assistant Specialist
2018-2019	Republic of Turkey Ministry of Economy	Foreign Trade Expert

Thesis Title:
Mineral Exportation, Industrial Production and Economic Growth: Cointegration and Causality Analysis for Turkey

2014-2018	Republic of Turkey Ministry of Economy	Foreign Trade Assistant Expert

FOREIGN LANGUAGES

English – Good command in reading and writing, fluent in speaking

Foreign Language Exam (YDS), March 2016: 90.00/100