



UPPSALA
UNIVERSITET

Detecting Performance Degradation in Code Based on Execution Times of Unit Tests

Johan Venemalm

Project in Computational Science

January 23, 2014

PROJECT REPORT



Abstract

In many applications there is a need to determine when a change in the statistical properties of a series of data points occurs. The process of detecting the temporal location of a time instance is called change point detection. There are two fundamentally different approaches to change point detection today: parametric models and non-parametric models.

Within the parametric framework the goal is to find a model that fits data to sufficient extent. In most cases, however, this approach does not work well since it might be impossible to derive a closed-form analytic formula for the underlying statistical distribution. The most frequent situation in real-world applications is that the data generating mechanism is unknown. Hence, one must rely entirely on non-parametric methods.

This paper discusses essentially two different non-parametric approaches to change point detection in univariate time series data. First, it discusses the bootstrap method and a modification of it called the stationary bootstrap. These methods fall into the class of resampling methods in applied statistics. Next, it discusses the Kullback-Leibler Importance Estimation Procedure (KLIEP) which is a density estimation algorithm developed by Japanese researches. Finally, a general algorithm for change point detection is outlined and performance of the three methods is evaluated.

Keywords

Change Point Detection, Non-Parametric Modeling, Bootstrap, Stationary Bootstrap, Kullback-Leibler Importance Estimation Procedure (KLIEP)

Acknowledgements

I would like to thank Schlumberger for the initiative to shine light on this increasingly important problem and to grant me the opportunity to exercise the project. In particular, I am thankful for my supervisor Ivar Bratberg's cooperation during the project phase, who has responded to my emails with perseverance.

Contents

1	Introduction	1
1.1	Background	1
1.2	Basic Assumptions	2
1.3	The Advantage of Non-Parametric Modeling	3
1.4	Data Preprocessing	4
2	The Non-Parametric Approach to Change Point Detection	7
2.1	Problem Formulation	7
2.2	The Bootstrap as a Tool for Non-Parametric Inference	8
2.3	The Stationary Bootstrap	9
2.4	Change Point Detection with Bootstrap	10
2.5	Kullback-Leibler Importance Estimation Procedure (KLIEP)	12
2.6	Change Point Detection with KLIEP	14
2.7	A General Algorithm for Change Point Detection	16
3	Results	18
3.1	Parameter Selection	18
3.2	Type I Errors	19
3.3	Type II Errors	20
4	Discussion	25
4.1	Type I Errors	25
4.2	Type II Errors	25
4.3	General Topics	26
5	Conclusion	28

List of Figures

1.1	Univariate outliers can be eliminated using a boxplot technique. By construction, this is a robust outlier reduction method. In the figure, the red line represents the median of the data set and the blue lines on the boundary of the box denote the lower and upper quartiles. The whiskers extend to the most extreme data points and can be controlled by changing k as in (1.2). All observations located outside the whiskers (marked with the $+$ symbol) are classified as outliers.	5
3.1	Type I errors computed with respect to a purified stationary data set for the standard bootstrap, stationary bootstrap and KLIEP as a function of the ratio between the length of the test and reference intervals. The errors are plotted for two choices of the level of significance, namely $\alpha = 5\%$ and $\alpha = 1\%$, respectively.	19
3.2	Type II errors rates for a purified stationary data set plotted as a function of the test interval factor of increase K for different combinations of the reference and test interval lengths. The indicator (1) corresponds to the interval length combination $(n_{rf}, n_{te}) = (50, 50)$ whilst the indicator (2) corresponds to $(n_{rf}, n_{te}) = (50, 100)$. Here the level of significance $\alpha = 5\%$ is fixed.	20
3.3	A realization of an artificial non-stationary process for Model (1) ($\Leftrightarrow f = \mu_t$) for the data set defined by equation (3.6). The black zigzag line shows the local mean of the process.	21
3.4	Type II errors rates for a realization of the first artificial data set defined by equation (3.6) (Model (1)) plotted as a function of the test interval factor of increase K for $(n_{rf}, n_{te}) = (60, 90)$. Here the level of significance is 1% and 5% , respectively.	22
3.5	Type II errors rates for a realization of the second artificial data set defined by equation (3.6) (Model (2)) plotted as a function of test interval factor of increase K for $(n_{rf}, n_{te}) = (60, 90)$. Here the level of significance is 1% and 5% , respectively.	23

1 Introduction

1.1 Background

In today's fast-paced environment, tech companies maintaining large code packages are facing the challenge of steady addition and modification of code. To measure the effect of regular addition, removal and modification of existing code, multiple measurements are often performed on a daily basis, where each run measures the execution time of a small fraction of the total code. These tests are called unit tests. If the code package is very large, this means that many tests are run. Moreover, if the measurement frequency is high a lot of data is generated and must be subsequently analysed.

To manually decide whether an increase in execution time has occurred for a particular unit test requires going through all unit tests and visually inspect at what time instance the mean level has shifted upwards (if any such shift can be detected). This might be extremely time-consuming or even impossible to achieve in practice. The problem of knowing whether performance degradation (an increase in observed execution time) has really occurred also persists, since infrequent random fluctuations might have caused some observed abnormal behaviour. It is evident that statistical methods are needed to resolve the issue of whether a sequence of observations is significantly different from a preceding sequence of observations. It is therefore natural to implement an automatic detection scheme to be able to cope with potentially non-stationary time series data sequences.

The problem of detecting changes in time series data has been widely studied in literature. Within the data mining community, a change point is defined as a time instance where the statistical properties of the underlying stochastic process change. There are many models proposing different approaches on how to tackle this problem, but many of them rely on very limited assumptions about the nature of the data. As I was doing preliminary research, I observed that many articles were assuming artificial data sets and particular mathematical properties for the time series being investigated. In this project, the data is completely unknown and, additionally, the number of data series is extremely large. It is therefore impossible to find a general parametric model in any way or in any substep of the resulting change point algorithm to resolve the problem. It might even be impossible to fit a single parametric model to a data series due to complete lackness of the underlying probability distribution(s)¹.

After having convinced myself that parametric modeling is unable to resolve the issues arising from unknown data, I had to turn around to look at non-parametric alternatives; in particular, the bootstrap procedure. The bootstrap method belongs to a class of resampling algorithms that treat the data sample as an approximation of the true distribution, and attempts to make inferences about a statistic by re-drawing samples with replacement from the observed

¹It cannot be excluded that an observed data series comes from several underlying probability distributions. It is, more likely, a relatively probable case for many unit test data series.

data sequence. Being promising in terms of solid mathematical properties that make the method attractive, it still cannot be applied directly to non-stationary data without minor modification and a few basic assumptions.

There was, however, one paper that caught my interest as I was looking for potential candidate solutions. Authored by two Japanese researchers named Yoshinobu Kawahara and Masashi Sugiyama, their paper Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation proposes an entirely new way of estimating whether a change in a distribution has occurred without any parametric assumptions [1]. As pointed out earlier, parametric assumptions often tend to produce inaccurate and wrong results when those assumptions in reality cannot be met. Their method is principally based on an algorithm outlined in an earlier paper by Sugiyama *et al.* ([2]) that approximates the ratio of probability density functions without going through direct density estimation, and applies a statistical test to the result. This density estimation algorithm, referred to as Kullback-Leibler Importance Estimation Procedure (KLIEP), borrows several concepts from machine learning but can be considered non-parametric in the sense that one avoids the need for estimation of the individual densities in the two sample sets. Many ideas from the change point detection approach stated in their paper ([1]) can be found in the algorithm given in this paper.

A natural way to look for abrupt deviations in a time series is to partition the data into smaller intervals or segments. Two consecutive intervals are then analyzed separately; hence the change point is taken to be between the last observation in the first interval and before the first observation in the second interval. This implies that we might detect a region of consecutive change points. In those situations, with large confidence one can say that a change in the statistical properties of the data sequence occurred somewhere in that region.

In the next subsection, we state and justify a few basic assumptions that the analysis requires before dealing with the real problem.

1.2 Basic Assumptions

Statistical papers rely on a basic set of assumptions of a certain kind. Even though we exclusively rely on non-parametric measures in the analyses, a few conditions must be fulfilled for the solutions to be realistic.

First, we conjecture that each unit test can be analysed independently. In probabilistic terms, this means that the intersection between two unit tests is zero².

Secondly, each unit test can be considered to be sampled from a single distribution during a limited time interval, such that the statistical properties are approximately intact. This assumption is strongly affected by the sample length and the frequency for non-stationary oscillations of the execution time measurements. The motivation for this assumption is that during sufficiently small sam-

²This means that each unit test tests a unique part of the code, *i.e.*, they are atomic.

ple intervals and a limited performance decrease (increase) of the code within a certain unit test, the distributional properties will not deviate considerably from the previous underlying distribution. If the code is essentially optimized a priori it is reasonable to assume that the change from one interval to the subsequent is not too large.

Finally, we assume an existing set of observations, the training set, that is supposed to have been generated identically and independently (i.i.d.) from a stationary distribution for a given unit test. In other words, given a unit test, the expected value is assumed to be constant and the autocorrelation function is assumed to be zero for all observations belonging to the stationary training set.

We summarize the above assumptions in the following item list.

- Each unit test can be considered independent from the rest. This means that each test treats a non-overlapping piece of the code and that we expect no correlation between any two unit tests.
- During a sufficiently small interval, the sequence of observations for a given unit test can be considered approximately stationary. If a minor change occurs and the sample is not too large, the distribution will hopefully not change significantly. The extreme situation where this assumption becomes weak occurs in a situation where a large jump in measured execution time takes place. Processes that exhibit high non-stationarity would be problematic to handle.
- The training set, that is, the set of observations generated by repeated measurements without any modification of existing code, is assumed to be an i.i.d. sample³.

We end this section by briefly discussing the advantages of using non-parametric modeling instead of the traditional parametric approach.

1.3 The Advantage of Non-Parametric Modeling

In statistics, non-parametric modeling is the use of models that lack specific assumptions about the characteristic structure of data or the model behind the data. This is in direct contrast to the traditional parametric modeling, which is almost only taught at universities. Parametric models assume that one knows the structure of the data a priori. To justify the assumption of a specific model, one can apply statistical tests (most of them to check whether data is normal distributed or not) and/or transform the data so that the statistical properties are improved. The non-parametric approach is crucial for this work since *i*) the underlying data displays large variety in terms of the shape of the distributions, and *ii*) even if parametric models would be possible to fit via application of goodness-of-fit methods, these procedures would be too complex to carry out in

³In practice, the training sample is not perfectly i.i.d. due to measurement errors. The assumption is nevertheless reasonable and necessary for testing the null hypothesis.

practice. Even within the non-parametric framework, the computational time quickly becomes large, and for continuously growing time series the problem will only increase substantially over time.

The advantage of using non-parametric modeling cannot be stressed enough due to the nature of the data generating process. Without making any vague assumptions about the nature of the data or the asymptotic properties of the time series, we can use non-parametric techniques to avoid this major drawback. Of course, minor assumptions have to be made (see subsection 1.2), but they can be limited with decent choices of input parameters in the algorithm that searches for statistically significant upward deviations.

1.4 Data Preprocessing

The data that has been used for training consists of a set of roughly 50,000 unit tests, each consisting of 202 nonnegative integer-valued observations. The latter means that the observations are only sampled with integer accuracy, even though the observations in reality are sampled from a continuous distribution. Due to the data measuring process, some observations are very small (in the extreme case, 0) and some are very large. The zero observations arise when the tests are not run, whilst extremely large observations can result if the measurement process is hanging⁴. Moreover, some unit tests contain a small fraction of non-zero observations and some of them also display poor variability and only consist of a few distinct entries. The original data set is thus contaminated with certain entries that would have a negative impact on the performance of the algorithms.

In order to capture the interesting part of the data, data cleansing and outlier removal is required. The zero observations are easily removed on the fly, whereas threshold parameters can filter out tests that do not meet the requirements of minimum variability and a certain minimum length. The remaining observations that need to be processed are the extreme observations and will be referred to as *outliers*.

Outlier deletion is a well-known problem in the data mining community and is also known as anomaly or outlier detection. In multi-dimensional cases, the situation would be far more complicated, but luckily we are dealing with univariate data.

To start with, let us define what we refer to as an outlier. According to the definition by Hawkins (1980), an outlier is "an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism". Inspired by this definition, one could as an example apply the definition to the hypothetical case of normal distributed data. Here, values falling far away from the mean in a normal distribution can be considered to be outliers if sufficient understanding of the underlying data generating process is known. In financial data, for instance, outliers should be treated with extreme care since they are often more important than the middle-range observations.

⁴The measurements are run on a server with varying loads. Sometimes, the measurements stall without halting the measurement of the execution time.

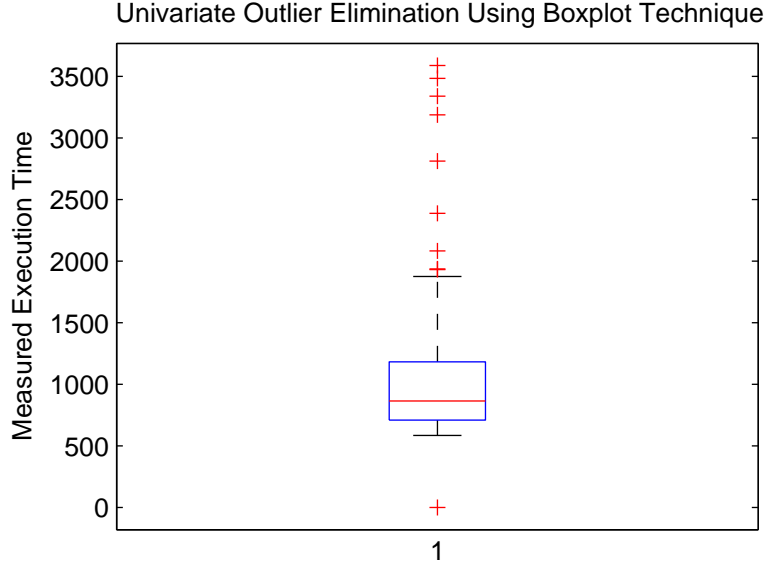


Figure 1.1: Univariate outliers can be eliminated using a boxplot technique. By construction, this is a robust outlier reduction method. In the figure, the red line represents the median of the data set and the blue lines on the boundary of the box denote the lower and upper quartiles. The whiskers extend to the most extreme data points and can be controlled by changing k as in (1.2). All observations located outside the whiskers (marked with the $+$ symbol) are classified as outliers.

However, let us assume that we have knowledge about the data generating mechanism. Then we can without hesitation eliminate observations that fall outside a pre-specified standard normal confidence interval.

Specifically, if x_1, \dots, x_n is a set of observations from the standard normal distribution $\mathcal{N}(0, 1)$ and \bar{x} , s_x denote the sample mean and sample standard deviation, respectively, then an observation x_o is declared an outlier given that

$$\frac{|x_o - \bar{x}|}{s_x} > \lambda_\alpha, \quad (1.1)$$

where λ_α denotes the standard normal quantile for a given α (≈ 3 if a 99 % confidence interval is sought). Unfortunately, data is not normal distributed in our case. However, the same idea can be applied if the data is visualized in a boxplot (Figure 1.1). This requires interchanging the mean for the first and third sample quartiles, Q_1 and Q_3 , which are robust measures of the center of the lower and upper ranges of a data sample, and to interchange the sample standard deviation for the interquartile range, $IQR := Q_3 - Q_1$, which is a robust measure of variability [3]. Thus, for data from any type of distribution,

an observation is classified as an outlier if

$$\frac{Q_1 - x_o}{IQR} > k, \text{ or } \frac{x_o - Q_3}{IQR} > k, \quad (1.2)$$

where k is a parameter with a similar role as the quantile function for the standard normal distribution. We let $k = 1.5$ classify an outlier as mild whilst $k = 3$ filters out extreme outliers from the data sample.

Outlier elimination is naturally executed after data cleansing.

2 The Non-Parametric Approach to Change Point Detection

Two mathematically different non-parametric models are proposed for detecting whether a performance degradation has occurred between two consecutive samples of given lengths. This section provides a walkthrough of the bootstrap and a modification of it called the stationary bootstrap, as well as the Kullback-Leibler Importance Estimation Procedure (KLIEP) method. The reader should note that the aim here is not to provide a thorough mathematical background for any of the three methods, but to give the basic ideas and equations behind them. The interested reader should therefore consult the bibliographic section in this paper for a more detailed and in-depth analysis of the tools that are used here.

2.1 Problem Formulation

Given is a cleansed, outlier-free, positive integer-valued univariate data set denoted by $\mathbf{X}_j = (x_{j,1}, \dots, x_{j,n}) \sim F_{j,n}$, where $j = 1, \dots, J$ and $\{\mathbf{X}_j \in \mathbf{Z}_+, j = 1, \dots, J\}$ is sampled from some true distribution $F_{j,n}$. In general, $F_{j,n}$ can contain a mixture of different distributions and is then a non-stationary time series. Here, the observations $x_{j,1}, x_{j,2}, \dots$ are thus not necessarily independent⁵. Choose next two consecutive non-overlapping subsets $\mathbf{Y}_{rf,j} = (y_{rf,j,1}, \dots, y_{rf,j,n_{rf}}) \subset \mathbf{X}_j$, $\mathbf{Y}_{te,j} = (y_{te,j,1}, \dots, y_{te,j,n_{te}}) \subset \mathbf{X}_j$ of length n_{rf} and n_{te} respectively. The first observations in each set were observed at time instances t_{rf}, t_{te} , with the current time instance $t = t_T = t_{rf} + n_{rf} + n_{te} - 1$. The first subset is called the *reference interval* and the second subset is called the *test interval*; the latter is a set where the statistical properties might have changed with respect to the reference set. If the length of the samples is not too large, they can be treated as approximately stationary under the assumption from subsection 1.2.

Next, introduce the *shift size* w_s , supported on the set $\{1, \dots, n - n_{rf} - n_{te}\}$. This parameter controls the number of elements that the reference and test intervals are shifted with after the computations are finished on an arbitrary reference-test sample pair. If t_{CP} denotes a candidate change point, then $t_{CP} + w_s$ is the next candidate change point under consideration.

The special case of $w_s = 1$ corresponds to the unit shift. By sliding with the size of one unit, it is possible to detect a consecutive sequence of change points. The larger the sequence, the stronger the evidence in favour of a change point scenario between the reference and test intervals.

The task is to find a time point between consecutive time intervals where a statistically significant degradation has occurred. We formulate the following two hypotheses

$$H_0 : \text{No change has occurred vs. } H_1 : \text{An upward change has occurred.} \quad (2.1)$$

⁵The stationary bootstrap takes into account dependencies between observations. It is a slight modification of the standard bootstrap method and is studied in subsection 2.3.

If H_1 cannot be rejected, t_{CP} is alarmed as a potential time point of an upward shift in the time series. Depending on whether the bootstrap technique or KLIEP is used, the test statistic will be different. The complete formulation for both approaches is given in the corresponding sections.

2.2 The Bootstrap as a Tool for Non-Parametric Inference

The bootstrap method was originally invented in 1979 by Bradley Efron [4]. It belongs to a broader class of computer based statistical methods called computer intensive methods; these include mainly the Jackknife, Cross-Validation, and Permutation Tests. Although the first two methods are used for other purposes, they are important tools in the area of applied statistics today.

The bootstrap methodology quickly became popular due to its wide applicability. Its strength relies on assumptions that are often met in practice. The bootstrap approach is outlined as follows: Given an observed i.i.d. sample $\mathbf{X} = \{x_1, \dots, x_n\}$, one claims that the observed sample approximates the true distribution F by the empirical distribution function F_n . Important asymptotic properties state that the observed distribution converges to the true distribution as the sample size approaches infinity. If F_n poorly approximates the true distribution and if $\theta = \theta(X_1, \dots, X_n)$ is the statistic of interest, then the bootstrap estimate of a statistic, $\theta^* = \theta^*(x_1^*, \dots, x_n^*)$, with x_1^*, \dots, x_n^* being the bootstrapped sample under replacement, will not be much better. If the reverse is true, then the resampled distribution F_n^* is a good approximation of F . Repeated computation of the statistic of interest under F_n^* yields the bootstrap replicates θ_i^* , $i = 1, \dots, B$, and results in better and better approximation of $\hat{\theta}_B^* = \frac{1}{B} \sum_{i=1}^B \theta_i^*$. One can thus control the Monte Carlo error of the statistic under consideration by increasing the value of B . The bootstrap approach essentially mimics the traditional approach but generates pseudo observations based on already observed data.

One problem with the bootstrap method for time series data is the feature of non-stationarity. There are many papers proposing different bootstrap techniques for versions of non-stationary data, but none of them can be directly applied to data that might potentially change fast in time. The local block bootstrap proposed by E. Paparoditis and D. N. Politis is promising for slowly changing stochastic processes (typically evolutionary data or potentially a financial time series). It is based on the idea of resampling blocks of constant size in a local neighbourhood around a data point. For large time series, numerous local neighbourhoods emerge and consequently it would make sense to employ a local bootstrap technique. However, I have access to a very small data sample (less than 200 observations after the cleansing and outlier removal process is completed) which means that the local bootstrap would in practice mimic another version of the standard bootstrap called the *stationary bootstrap*. The most essential difference would be that in the latter case, the block size is

stochastic⁶.

The stationary bootstrap relies on the assumption of approximate stationarity of a data sequence. Also proposed by D. N. Politis and J. P. Romano for resolving issues involving dependency in data ([6]), the stationary bootstrap generates a pseudo time series that is actually stationary under the assumption of weak stationarity. The stationary bootstrap method is explored in subsection 2.3.

2.3 The Stationary Bootstrap

The stationary bootstrap was proposed by D. N. Politis and J. P. Romano ([6]) for resolving issues regarding dependency in data. In this paper, a subsample (reference- or test set) is assumed to be approximately stationary under some conditions. However, there might exist dependencies between data in the sample intervals. The stationary bootstrap method resolves these issues under the assumption of a weakly dependent time series. Intuitively, a series is weakly dependent if two observations x_t, x_{t+h} are almost uncorrelated as $h \rightarrow 0$. In our framework, that is reasonable to assume.

A brief overview of the stationary bootstrap is given in this subsection. All details can be found in the original paper by Politis *et al.*

Let $B_{I_1, L_1}, B_{I_2, L_2}, \dots$ be a sequence of blocks of lengths L_1, L_2, \dots starting from indices I_1, I_2, \dots . The integers L_1, L_2, \dots form a sequence of i.i.d. random variables following a geometric distribution. Thus, the probability of drawing a block of length m is given by

$$P(L_i = m) = (1 - p)^{m-1}p, \quad m = 1, 2, \dots \quad (2.2)$$

The integers I_1, I_2, \dots form a sequence of i.i.d. random variables from a discrete uniform distribution. The probability of choosing a starting index $i \in \{1, \dots, n_{obs}\}$ is inversely proportional to the number of observations in the sample, n_{obs} :

$$P(I_i = k) = \frac{1}{n_{obs}}, \quad k = 1, \dots, n_{obs}. \quad (2.3)$$

It should be noted that the choice for L_i having a geometric distribution and I_i having a discrete uniform distribution was made so that the resampled distribution remains stationary. There are, however, other possible resampling schemes that preserve stationarity.

At last, the pseudo generated series should be of the same size as the original series, that is

$$\sum_k L_k = n_{obs}. \quad (2.4)$$

⁶The starting index for a block in the local bootstrap algorithm does not necessarily have to be sampled from a uniform distribution as in the case of the stationary bootstrap.

It remains to choose a value for p which controls the geometric average block length. Previous studies on block resampling algorithms in statistics have led to conclusions that the optimal block length grows (c.f., *e.g.*, [7]) as

$$L_{opt} \propto N^{1/3}. \quad (2.5)$$

Hence, for a particular data sequence of size N the probability parameter p can be picked as

$$p = \frac{1}{N^{1/3}}. \quad (2.6)$$

In the situation under consideration, we simulate sequences of bootstrap blocks $Y_{rf,j}^* = B_{rf,j,I_1,L_1}^* \cup B_{rf,j,I_2,L_2}^* \cup \dots \cup B_{rf,j,I_k,L_k}^*$, $Y_{te,j}^* = B_{te,j,I_1,L_1}^* \cup B_{te,j,I_2,L_2}^* \cup \dots \cup B_{te,j,I_k,L_k}^*$, where the first subscript signifies the sample type (reference or test) and (\cdot) signifies either of the intervals, the second subscript specifies the particular unit test, and the last two subscripts are as before. An arbitrary bootstrap block in any sample can thus be expressed as

$$B_{(\cdot),j,I_i,L_i}^* = (b_{(\cdot),j,I_i}^*, b_{(\cdot),j,I_i+1}^*, \dots, b_{(\cdot),j,I_i+L_i-1}^*).$$

In the next subsection a formalized change point detection scheme using the bootstrap methodology is outlined. We will drop the block notation from this subsection to avoid multiple notations. Thus, observations $y_{(\cdot),j,i}^*$ could correspond to observations generated according to the stationary bootstrap scheme. The only difference is that a dependency structure between consecutive observations exists.

2.4 Change Point Detection with Bootstrap

Let as before $\mathbf{Y}_{rf,j} = (y_{rf,j,1}, \dots, y_{rf,j,n_{rf}}) \subset \mathbf{X}_j$, $\mathbf{Y}_{te} = (y_{te,j,1}, \dots, y_{te,j,n_{te}}) \subset \mathbf{X}_j$ denote the reference and test samples of length n_{rf} and n_{te} , respectively. Further, let

$$\hat{\mathbf{Y}}_{rf,j} := \frac{1}{n_{rf}} \sum_{i=1}^{n_{rf}} y_{rf,j,i}, \quad \hat{\mathbf{Y}}_{te,j} := \frac{1}{n_{te}} \sum_{i=1}^{n_{te}} y_{te,j,i} \quad (2.7)$$

denote the observed replicates of the arithmetic mean of reference- and test samples, respectively. Let also

$$T_j = T_j(\hat{\mathbf{Y}}_{rf,j}; \hat{\mathbf{Y}}_{te,j}) := \hat{\mathbf{Y}}_{te,j} - \hat{\mathbf{Y}}_{rf,j}, \quad j = 1, \dots, J, \quad (2.8)$$

be the test statistic symbolizing the mean difference between the reference and test samples.

The hypothesis then reads

$$H_0 : T_j = 0 \text{ vs. } H_1 : T_j > 0, j = 1, \dots, J.$$

Following the bootstrap methodology, resampling with replacement from $\mathbf{Y}_{rf,j}, \mathbf{Y}_{te}$ yields the bootstrap observations $\mathbf{Y}_{rf,j}^* = (y_{rf,j,1}^*, \dots, y_{rf,j,n_{rf}}^*) \sim \hat{F}_{rf,j,n_{rf}}$ and $\mathbf{Y}_{te,j}^* = (y_{te,j,1}^*, \dots, y_{te,j,n_{te}}^*) \sim \hat{F}_{te,j,n_{te}}$, where $\hat{F}_{(\cdot),j,n}$ denotes the observed empirical distribution function of a general distribution $F_{(\cdot),j,n}$. In case of applying the stationary bootstrap, the observations contained in $\mathbf{Y}_{rf,j}^*, \mathbf{Y}_{te,j}^*$ have been generated in a blockwise manner, taking local dependency into account.

Introduce the test statistic for the mean difference between the reference and test bootstrap samples conditional on the original sample for the b :th bootstrap replicate:

$$\begin{aligned} T_{j,b}^* &= T_{j,b}^*(\hat{\mathbf{Y}}_{rf,j}^*; \hat{\mathbf{Y}}_{te,j}^* | \mathbf{Y}_{rf,j}; \mathbf{Y}_{te,j}) := \\ &\quad \frac{1}{n_{te}} \sum_{k=1}^{n_{te}} y_{te,j,b,k}^* - \frac{1}{n_{rf}} \sum_{k=1}^{n_{rf}} y_{rf,j,b,k}^* = \\ &\quad \hat{\mathbf{Y}}_{te,j,b}^* - \mathbf{Y}_{rf,j,b}^*, \text{ for } j = 1, \dots, J \text{ and } b = 1, \dots, B. \end{aligned} \quad (2.9)$$

To provide an explicit formula for the stationary bootstrap, the test statistic can be written as

$$\begin{aligned} T_{j,b}^* &= T_{j,b}^*(\hat{\mathbf{Y}}_{rf,j}^*; \hat{\mathbf{Y}}_{te,j}^* | \mathbf{Y}_{rf,j}; \mathbf{Y}_{te,j}) := \\ &\quad \frac{1}{n_{te}} \sum_{k=1}^{m_2} \sum_{l=0}^{L_k-1} b_{te,j,I_k+l}^* - \frac{1}{n_{rf}} \sum_{k=1}^{m_1} \sum_{l=0}^{L_k-1} b_{rf,j,I_k+l}^*, \\ &\quad \text{for } j = 1, \dots, J, b = 1, \dots, B, \end{aligned} \quad (2.10)$$

where m_1, m_2 denote the number of blocks in the two intervals.

Denote by t_{CP} a candidate change point, *i.e.*, $t_{rf} + n_{rf} - 1 < t_{CP} < t_{te}$. To check whether $\mathbf{Y}_{te,j}$ is significantly different from $\mathbf{Y}_{rf,j}$ with respect to t_{CP} , we state the following hypothesis involving the bootstrap statistic $T_{j,b}^*$:

$$H_0 : T_{j,b}^* = 0 \text{ vs. } H_1 : T_{j,b}^* > 0, j = 1, \dots, J, b = 1, \dots, B,$$

where $T_{j,b}^*$ is the bootstrap test statistic. Let now α denote the level of significance. By calculating B bootstrap replicates $T_{j,b}^*$ it is possible to obtain a $100(1 - \alpha)$ % percentile confidence interval for the mean difference between reference and test intervals by ordering the bootstrapped values in an increasing sequence, according to

$$T_{j,1}^* \leq T_{j,1}^* \leq \dots \leq T_{j,B}^*, j = 1, \dots, J.$$

Define the lower and upper cut-off points $k_L := \lfloor \frac{\alpha}{2}(B+1) \rfloor$ and $k_U := B+1 - k_L$. Then a $100(1 - \alpha)$ % percentile confidence interval for $T_{j,b}^*$ is

$$[T_{j,k_L}^*, T_{j,k_U}^*], j = 1, \dots, J. \quad (2.11)$$

A decision whether a degradation has occurred is thus taken if

$$0 \notin [T_{j,k_L}^* T_{j,k_U}^*] \implies \text{Reject } H_0, j = 1, \dots, J. \quad (2.12)$$

This confidence interval is known as a first order accurate confidence interval. It is nevertheless the most intuitive confidence interval to construct. Since it holds asymptotically that the bootstrap distribution converges to the normal distribution for the statistic of interest, it would be natural to construct a Student's t confidence interval. The problem is that this requires equal sample sizes. In general, the sample sizes might be different.

In the next subsection, we leave the resampling based methods for a moment and turn our attention to an algorithm with origins from a problem frequently occurring in machine learning.

2.5 Kullback-Leibler Importance Estimation Procedure (KLIEP)

This subsection gives a brief overview of the KLIEP algorithm. The reader should consult the corresponding paper found in the reference section for further details.

The starting point for the foundation of the Kullback-Leibler Importance Estimation Procedure (KLIEP) method is to find an efficient algorithm that estimates the probability density ratio of a test distribution, p_{te} , and a reference distribution, p_{rf} for some input data. In the general case, the input data follows continuous probability distributions. In our case, however, the data consists of a set of discrete integers. Hence, we aim at estimating the ratio $w = w(\mathbf{x})$, where the input data comes from an integer-valued domain $\mathcal{D}_k := \{k : k \in \mathbf{Z}_+\}$:

$$w(\mathbf{x}) := \frac{p_{te}(\mathbf{x})}{p_{rf}(\mathbf{x})}, \quad \mathbf{x} \in \mathcal{D}_k. \quad (2.13)$$

Here, $w(\mathbf{x})$ is called the *importance* and $p_{te}, p_{rf} > 0$ are probability distributions defined by some input data $\mathbf{x}_{te} = (x_1, \dots, x_{n_{te}})$ and $\mathbf{x}_{rf} = (x_1, \dots, x_{n_{rf}})$. By simply looking at the equation, one could think of the naive way of estimating the probability density functions directly, and then taking the ratio of the respective probability densities. However, since this is known to be a hard problem in practice, the goal of KLIEP is to avoid direct density estimation and instead find a solution that estimates the *ratio* only using the input data $\mathbf{x}_{te}, \mathbf{x}_{rf}$.

Let $K_\sigma(\mathbf{x}, \mathbf{x}')$ denote the Gaussian kernel function centered at the test point \mathbf{x}' . To begin, let us approximate the importance using an eigenfunction expansion with a Gaussian kernel function centered at the test input points:

$$\hat{w}(\mathbf{x}) = \sum_{l=1}^b \alpha_l K_\sigma(\mathbf{x}, \mathbf{x}_{te,l}). \quad (2.14)$$

Here, α_l are the parameters that need to be optimized. The output of KLIEP is the set of weights $\{w(\mathbf{x}) : \mathbf{x} \in \mathbf{x}_{te}\}$, i.e., the importance is evaluated at the corresponding test input sample points. The resulting importance is thus modeled as a Gaussian kernel expansion with kernel width σ , centered at the test sample. The Gaussian kernel is given by

$$K_\sigma(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{2\sigma^2}\right), \quad (2.15)$$

where $\sigma > 0$ is the kernel width. Using equation (2.13), the test input density can be approximated by

$$\hat{p}_{te}(\mathbf{x}) = \hat{w}p_{rf}(\mathbf{x}). \quad (2.16)$$

The goal is now to determine the parameters $\{\alpha_l\}_{l=1}^b$ such that the Kullback-Leibler divergence from p_{te} to \hat{p}_{te} is minimized. The Kullback-Leibler divergence can be used to measure the difference between two probability distributions over the same event space. Even though the measure does not fulfill all requirements for being a metric (the Kullback-Leibler divergence does not hold for symmetry), it is nevertheless a widely used theoretical measure in statistics and information theory. The discrete Kullback-Leibler divergence is given by

$$\begin{aligned} KL[p_{te}(\mathbf{x})||\hat{p}_{te}(\mathbf{x})] &= \sum_{\mathbf{x} \in \mathcal{D}_k} p_{te}(\mathbf{x}) \ln \frac{p_{te}(\mathbf{x})}{\hat{p}_{te}(\mathbf{x})} = \\ &= \sum_{\mathbf{x} \in \mathcal{D}_k} p_{te}(\mathbf{x}) \ln \frac{p_{te}(\mathbf{x})}{p_{rf}(\mathbf{x})} - \sum_{\mathbf{x} \in \mathcal{D}_k} p_{te}(\mathbf{x}) \ln \hat{w}(\mathbf{x}). \end{aligned} \quad (2.17)$$

The trick is to observe that the second summation involves the importance factor with coefficients α_l to be determined. This term can be expanded by approximating the probability distribution for the test set:

$$\begin{aligned} M := \sum_{\mathbf{x} \in \mathcal{D}_k} p_{te}(\mathbf{x}) \ln \hat{w}(\mathbf{x}) &\approx \frac{1}{n_{te}} \sum_{k=1}^{n_{te}} \ln \hat{w}(\mathbf{x}_{te,k}) = \\ &= \frac{1}{n_{te}} \sum_{k=1}^{n_{te}} \ln \left(\sum_{l=1}^b \alpha_l K_\sigma(\mathbf{x}, \mathbf{x}_{te,k}) \right). \end{aligned} \quad (2.18)$$

The last term is of concave form, and hence the Kullback-Leibler divergence is minimized when the last term is maximized. The resulting problem to be solved becomes a convex optimization problem where the objective function is given by M in the expression in (2.18). Imposing the constraints

$$\begin{aligned} \frac{1}{n_{rf}} \sum_{k=1}^{n_{rf}} \sum_{l=1}^b \alpha_l K_\sigma(\mathbf{x}, \mathbf{x}_{rf,k}) &= 1, \\ \hat{w}(\mathbf{x}) &\geq 0, \mathbf{x} \in \mathcal{D}_k, \end{aligned} \quad (2.19)$$

results in the convex optimization problem

$$\begin{aligned} & \max_{\{\alpha_l\}_{l=1}^b} \left[\sum_{k=1}^{n_{te}} \ln \left(\sum_{l=1}^b \alpha_l K_\sigma(\mathbf{x}, \mathbf{x}_{te,k}) \right) \right] \\ \text{s.t. } & \sum_{k=1}^{n_{rf}} \sum_{l=1}^b \alpha_l K_\sigma(\mathbf{x}, \mathbf{x}_{te,k}) = n_{rf}, \quad \alpha_1, \dots, \alpha_b \geq 0. \end{aligned} \quad (2.20)$$

The KLIEP algorithm solves this equation iteratively by using gradient ascent. Since the problem is convex, a unique global solution is guaranteed.

What remains is to perform model selection in order to obtain the kernel functions K_σ . In general, one does not need to choose Gaussian kernels, but they are mathematically convenient to work with and are proposed by the founders of the algorithm. The KLIEP algorithm is equipped with a likelihood cross validation (LCV) procedure that splits the test samples into R disjoint subsets and approximates M in equation (2.18) by a sample average. By repeating this procedure for different kernel widths, one obtains the kernel that minimizes M . What LCV does is to choose the parameter set that is optimal given the data at hand.

There are a few complications that need to be understood when applying KLIEP. First, one needs to choose the number of kernels b heuristically. One can for example choose $b = \min(n_{te}, 100)$. Thus, the allocation of the kernel centers are in the test sample space, which was found to be computationally more efficient. Moreover, since in general the test sample size can be made larger than the reference sample size, the test density function will be larger than the corresponding reference density function. This means that the output function $w(\mathbf{x})$ takes on larger values (*i.e.*, the density is higher). Hence, allocating more kernels in the test sample space boosts the accuracy of the resulting algorithm.

That being said, KLIEP can handle the case where $n_{te} > n_{rf}$. How the reference and test interval lengths should be chosen depends on the degree of non-stationarity of the test interval. Note that it is essential that the intervals remain approximately stationary.

2.6 Change Point Detection with KLIEP

From this point it is straightforward to derive a change point detection algorithm using KLIEP. We mimic the overall terminology from the bootstrap sections with a few modifications.

As before, let $\mathbf{Y}_{rf,j}, \mathbf{Y}_{te,j}$ denote the *observed* reference and test intervals for unit test $j = 1, \dots, J$. The aim is to approximate the importance defined as the fraction

$$w(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{rf}(\mathbf{x})}.$$

Here, p_{rf}, p_{te} correspond to the empirical probability density functions from the reference and test intervals, respectively. Without estimating them separately, KLIEP is used to only estimate the fraction of the density function in equation (2.6)⁷.

Let $t_T = t_{rf} + n_{rf} + n_{te} - 1$ be the current time instance and let t_{CP} be a candidate change point as before between the reference and test intervals; that is, $t_{rf} + n_{rf} - 1 < t_{CP} < t_{te}$. Here, t_{rf}, t_{te} are the time instances marking the first observation in the reference and test intervals. To check whether $\mathbf{Y}_{te,j}$ is significantly different from $\mathbf{Y}_{rf,j}$, we state the hypothesis

$$\begin{aligned} H_0 : p(\mathbf{Y}_{rf,j}) &= p_{rf}(\mathbf{Y}_{rf,j}), t_{rf} \leq t < t_T \quad \text{vs.} \\ H_1 : p(\mathbf{Y}_{rf,j}) &= p_{rf}(\mathbf{Y}_{rf,j}), t_{rf} \leq t < t_{te}, \\ p(\mathbf{Y}_{te,j}) &= p_{te}(\mathbf{Y}_{te,j}), t_{te} \leq t < t_T, \\ \text{such that } p_{te}(\mathbf{Y}_{te,j}) &> p_{rf}(\mathbf{Y}_{rf,j}). \end{aligned} \quad (2.21)$$

The most natural test statistic to consider is the likelihood ratio between the two probability distributions. The likelihood ratio between H_1 and H_0 is given by

$$\Lambda = \frac{\prod_{i=1}^{n_{rf}} p_{rf}(\mathbf{Y}_{rf,j}(i)) \prod_{i=1}^{n_{te}} p_{te}(\mathbf{Y}_{te,j}(i))}{\prod_{i=1}^{n_{rf}} p_{rf}(\mathbf{Y}_{rf,j}(i)) \prod_{i=1}^{n_{te}} p_{rf}(\mathbf{Y}_{te,j}(i))} = \frac{\prod_{i=1}^{n_{te}} p_{te}(\mathbf{Y}_{te,j}(i))}{\prod_{i=1}^{n_{te}} p_{rf}(\mathbf{Y}_{te,j}(i))}. \quad (2.22)$$

Note that the sum ranges over the observations from the test interval, which means that the importance is computed at all test input sample points. Taking the logarithm of the likelihood yields the test statistic

$$S = \ln \Lambda = \sum_{k=1}^{n_{te}} \ln \frac{p_{te}(\mathbf{Y}_{te,j}(i))}{p_{rf}(\mathbf{Y}_{te,j}(i))}. \quad (2.23)$$

Now, the hypothesis (2.21) can be evaluated by using the previous statistic and a threshold parameter μ :

$$\begin{cases} \text{If } S \leq \mu \longrightarrow H_0 \text{ cannot be rejected,} \\ \text{otherwise reject } H_0. \end{cases} \quad (2.24)$$

Remark : The hypothesis test (2.21) is a one-sided test. Since candidate change points where an upward shift has occurred are only considered, it is equivalent to require that a potential candidate change point t_{CP} satisfies $\hat{\mathbf{Y}}_{te,j} > \hat{\mathbf{Y}}_{rf,j}$. That is, the observed mean of the test interval must exceed the observed mean of the reference interval.

⁷Note that by computing the fraction $w(\mathbf{x}) = \frac{p_{te}(\mathbf{x})}{p_{rf}(\mathbf{x})}$, one does not know the individual probability density functions p_{rf} and p_{te} . However, by knowing p_{rf}, p_{te} one can straightforwardly calculate $\frac{p_{te}}{p_{rf}}$. The rationale of density estimation is thus unidirectional.

What is left undetermined is the threshold parameter μ . It depends on the length of the test and reference intervals and, most importantly, on the data set. Before running KLIEP, the user must make sure to calibrate μ so that the number of false rejections of H_0 does not exceed the level of significance α . Mathematically, we require

$$P(S > \mu | H_0) = \alpha. \quad (2.25)$$

In practice, it is required that this conditional probability holds only approximately. The calibration is naturally exercised on stationary training data.

Note that it is of vital importance that (2.25) is fulfilled; otherwise the testing of the hypothesis is likely to perform poorly. Applying KLIEP with calibration with respect to (2.25) introduces automatically two main sources of errors that will affect the performance of the final algorithm. Firstly, the data set must be stationary under H_0 , *i.e.*, full belief in H_0 is required. If not, we erroneously conduct a type III error. Secondly, the interval lengths n_{rf}, n_{te} must be chosen appropriately. The more data points, the better the result after the calibration phase.

2.7 A General Algorithm for Change Point Detection

We finish this section by stating a general algorithm for change point detection that can be used in practice. The methods appearing in the algorithm have all been described thoroughly in this section: the bootstrap, the stationary bootstrap and the Kullback-Leibler Importance Estimation Procedure (KLIEP). In the algorithm given on the following page (Algorithm 2.1), comments are marked with a double slash (//).

Algorithm 2.1: AN ALGORITHM FOR CHANGE POINT DETECTION(\mathcal{U}, \dots, μ)

Input : The set of all unit tests \mathcal{U} , the set of unit test indices \mathcal{T} , candidate change point t_{CP} , outlier type out_type, level of significance α , interval lengths n_{rf}, n_{te} , time instance t_{rf} , number of bootstrap replications n_{boot} , shift size w_s , minimum required length of a unit test d_{thr} , threshold of minimum variability t_{thr} , outlier type out_type, solution method do_method
Optional : Change point detection threshold μ

main

```

for each  $t \in \mathcal{T}$ 
do // Perform a full pass through the unit test set
 $\mathcal{U}_{clean} = \text{CLEAN\_DATA}(\mathcal{U}(t))$  // Cleansed data set
 $\mathcal{U}_{clean} = \text{DELETE\_OUTLIERS}(\mathcal{U}_{clean}, \text{out\_type})$  // Outlier adjusted data set
 $q = \text{FILTERING}(\mathcal{U}_{clean}, d_{thr}, t_{thr})$  // Returns either 0 (false) or 1 (true)
if do_method == 'KLIEP'
{
if  $\mu$  unspecified // This parameter is optional
{
then  $\mu = \text{CALIBRATE\_MU}(\mathcal{U}_{clean}, n_{rf}, n_{te}, t_{rf}, \alpha)$ 
}
if  $q > 0$ 
{
if do_method == 'do_bootstrap'
{
Initialize  $S_n$  // Vector of bootstrap replicates
for  $n = 1$  to  $n_{boot}$ 
do
{
if do_method == 'std_bootstrapfnc'
{
then  $S_n(n) = \text{STD\_BOOTSTRAP}(\mathcal{U}_{clean}, n_{rf}, n_{te}, t_{rf}, n_{boot})$ 
}
else if do_method == 'stry_bootstrapfnc'
{
then  $S_n(n) = \text{STRY\_BOOTSTRAP}(\mathcal{U}_{clean}, n_{rf}, n_{te}, t_{rf}, n_{boot})$ 
}
}
Compute lower and upper percentiles  $S_{low}, S_{up}$  from  $S_n$  given  $\alpha$ 
if  $S_{low} > 0$  and  $S_{up} > 0$ 
{
then {Reject  $H_0$ 
Signal potential change point  $t_{CP}$  for unit test  $\mathcal{U}(t)$ 
}
else Accept  $H_0$ 
}
else if do_method == 'KLIEP'
{
 $w = \text{KLIEP}(\mathcal{U}_{clean}, n_{rf}, n_{te})$  //  $w = (w(x_1), \dots, w(x_{n_{te}}))$ ; vector of weights
 $S = \sum_{k=1}^{n_{te}} \ln w(x_k)$  // Compute test statistic
if  $S > \mu$ 
{
then {Reject  $H_0$ 
Signal potential change point  $t_{CP}$  for unit test  $\mathcal{U}(t)$ 
}
else Accept  $H_0$ 
}
}
}
// Update time instances
 $t_{CP} = t_{CP} + w_s$ 
 $t_{rf} = t_{rf} + w_s$ 
 $t_{te} = t_{te} + w_s$ 

```

3 Results

3.1 Parameter Selection

This section provides the numerical results obtained from various simulations of the three models. In the simulations, the same seed for random number generation was used in order to obtain a reliable comparison between the models. The results presented here are based on type I and type II errors⁸. The performance of the models can thus be inferred based on the type I and type II error curves.

Before proceeding, denote by \mathbf{X} the stationary training data set:

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & & \dots \\ \vdots & \vdots & & \ddots \\ x_{J1} & x_{J2} & \dots & x_{Jn} \end{pmatrix} \in \text{Mat}_{Jn}. \quad (3.1)$$

Here, each element in \mathbf{X} is an integer and $n = 202$, $J = 50,000$. Since this matrix contains dirty data and outliers, data pre-processing in form of data cleansing and outlier deletion is executed prior to the analysis. We cannot write down the resulting matrix because the length of a cleansed data series depends on the number of 0 entries and how aggressively the outlier deletion process is carried out. In the following, a data point is an outlier if condition (1.2) is fulfilled for $k = 1.5$, *i.e.*, only mild outliers are considered.

It should also be stressed that the cleansed unit tests must fulfill some additional requirements. Let d_{min} and t_{min} denote the threshold parameters monitoring the minimum number of entries and unique entries that are required for the analysis. A cleansed unit test qualifies for analysis if and only if the minimum number of entries and unique entries in the resulting sequence exceed the threshold values. In the following, fix $d_{min} = 160$ and $t_{min} = 10$.

If the previous requirements are met, the simulations run through 200 different unit tests sampled randomly from the pool of available unit tests. That is, each unit test that is considered "representative enough" is treated identically with the same weight. Because performance evaluation of the models is restricted to type I and type II errors, simply take any point in a data set and consider it as an *i)* ordinary point in the case of investigating type I errors, and *ii)* a change point in the case of investigating type II errors.

In the following analyses, the number of bootstrap replicates $n_{boot} = 200$ is fixed in each resampling procedure. Moreover, KLIEP is always calibrated and run with likelihood cross validation for optimal performance⁹.

⁸In statistical data analysis, the performance of a model can be evaluated by studying type I and type II errors. The analyst determines the type I error threshold α based on the severeness of conducting type I and type II errors.

⁹The user can always try to choose the kernel widths appropriately, but the accuracy of the method can only decrease; likelihood cross validation returns the optimal bandwidth given the input reference and test data. Since the unit tests are sampled from characteristic distributions, it is not recommendable to choose one global kernel width.

3.2 Type I Errors

We start by studying type I errors or false positive errors. Given full belief in the null hypothesis, a type I error is defined as the fraction of false negatives or incorrect rejections of H_0 at the level of significance α . Recalling that each unit test that passes the filtering stage is treated equally, the fraction of type I errors for a given method and level of significance α is then given by

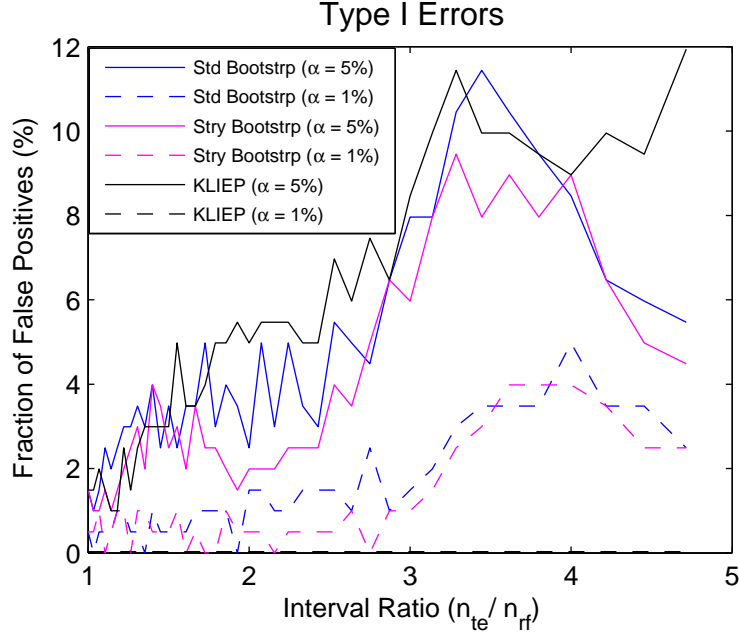


Figure 3.1: Type I errors computed with respect to a purified stationary data set for the standard bootstrap, stationary bootstrap and KLIEP as a function of the ratio between the length of the test and reference intervals. The errors are plotted for two choices of the level of significance, namely $\alpha = 5\%$ and $\alpha = 1\%$, respectively.

$$\text{Type I error} := \frac{\# \text{ false rejections of } H_0}{\# \text{ tests checked against } H_0} \quad (3.2)$$

To avoid spurious notation, it is understood that the numerator and denominator are counted conditionally on all tests that pass the filtering stage (otherwise, the unit test is not hypothesis tested). To study the sensitivity of the type I error under H_0 , it makes sense to study how the ratio between the test and reference interval changes when the interval lengths are increased/decreased. That is, we want to investigate how the fraction of false positives changes as the interval ratio

$$\frac{n_{te}}{n_{rf}} \neq 1 \quad (3.3)$$

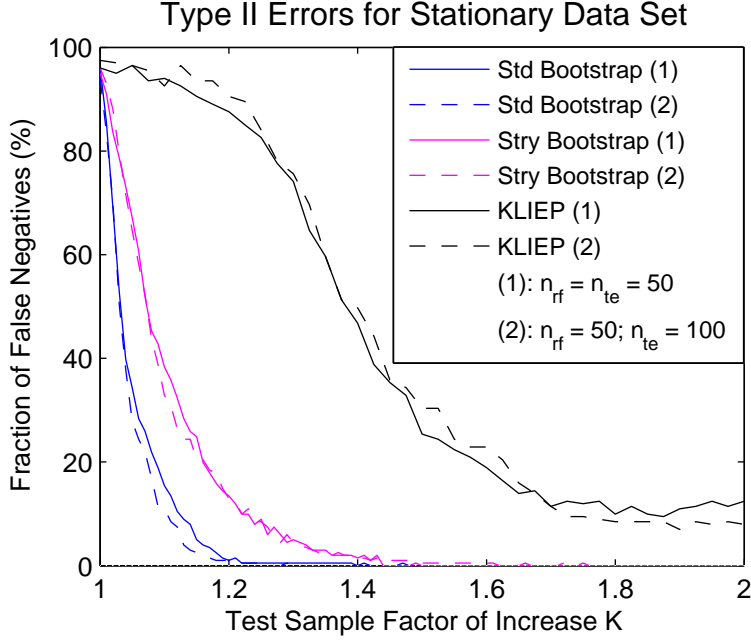


Figure 3.2: Type II errors rates for a purified stationary data set plotted as a function of the test interval factor of increase K for different combinations of the reference and test interval lengths. The indicator (1) corresponds to the interval length combination $(n_{rf}, n_{te}) = (50, 50)$ whilst the indicator (2) corresponds to $(n_{rf}, n_{te}) = (50, 100)$. Here the level of significance $\alpha = 5\%$ is fixed.

changes by increasing n_{te} and decreasing n_{rf} iteratively by one unit. Hence, the goal is to observe the type I error as a function of the fraction $\frac{n_{te}}{n_{rf}}$ given an appropriate choice of α .

Consider the purified stationary data set obtained originally from \mathbf{X} . Let $n_{te} = n_{rf} = 60$ initially. Now, construct the sequences of interval lengths $\{n_{te}, n_{te} + 1, \dots, n_{te} + 40\}$ and $\{n_{rf}, n_{rf} - 1, \dots, n_{rf} - 40\}$. Then, for a given ratio $\frac{n_{te}}{n_{rf}}$, Figure 3.1 shows the type I error as a function of the ratio between the test and sample interval lengths for $\alpha = 1\%$ and $\alpha = 5\%$, respectively.

3.3 Type II Errors

This subsection conducts studies on how type II errors vary for the standard bootstrap, stationary bootstrap and KLIEP given both stationary and non-stationary test data sets. Analogously, we start by defining a type II error as the fraction

$$\text{Type II error} := \frac{\# \text{ false acceptances of } H_0}{\# \text{ tests checked against } H_0}. \quad (3.4)$$

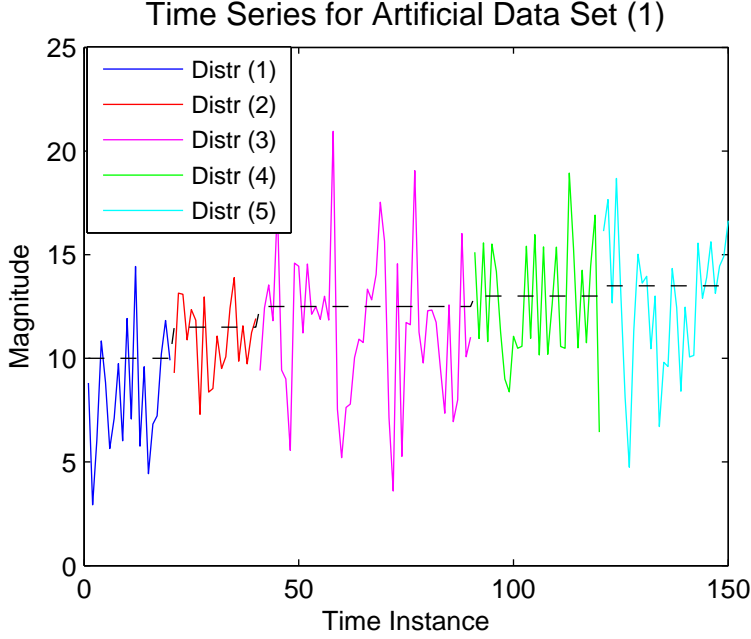


Figure 3.3: A realization of an artificial non-stationary process for Model (1) ($\Longleftrightarrow f = \mu_t$) for the data set defined by equation (3.6). The black zigzag line shows the local mean of the process.

An easy and well-motivated approach for studying type II errors is to take the given data series and multiply it by a constant factor K . In this way, the type II error can be studied as a function of K for a given degree of significance α ¹⁰. In the following, K ranges from 1 by steps of 0.01 to 2, *i.e.*, $K = 1.00, 1.01, \dots, 2.00$ and if otherwise not stated, the constant K denotes the multiplying factor of the entire test interval data sequence.

To begin with, consider type II errors for the purified stationary data set. We would like to study how the type II error depends on K . For a given change point t_{CP} , we pick two combinations of the parameters n_{rf}, n_{te} , namely $(n_{rf}, n_{te}) = (50, 50)$ and $(n_{rf}, n_{te}) = (50, 100)$ (that is, t_{CP} is located somewhere between the 50:th and 51:th entries in the data series). Then, for a fixed $\alpha = 0.05$, the fraction of type II errors as a function of K for each model under consideration is given by Figure 3.2.

Next, consider two artificially generated data sets. The two data sets only differ in terms of their individual variances. If X_t is an arbitrary non-stationary stochastic process, then the variance can be modeled as

¹⁰For artificially generated data sets, only the mean value is affected by the multiplying factor K . The magnitude of random fluctuations is kept constant.

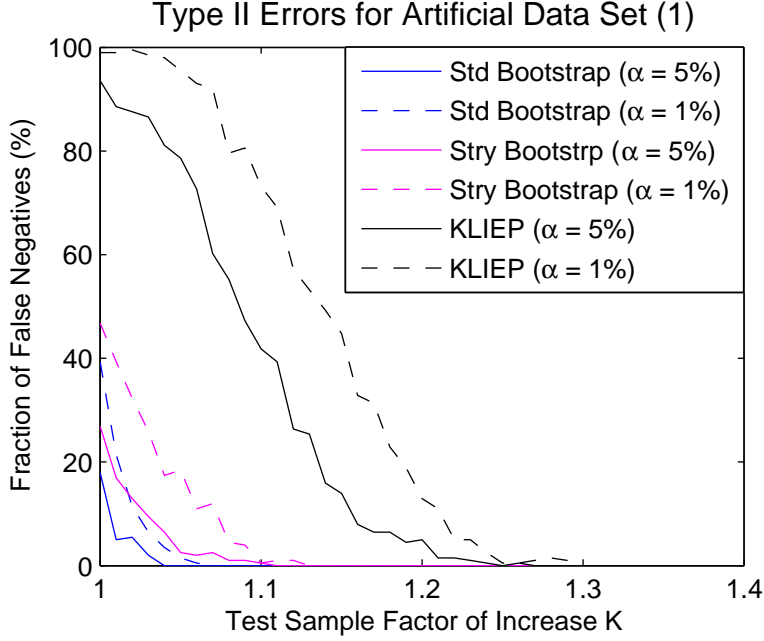


Figure 3.4: Type II errors rates for a realization of the first artificial data set defined by equation (3.6) (Model (1)) plotted as a function of the test interval factor of increase K for $(n_{rf}, n_{te}) = (60, 90)$. Here the level of significance is 1 % and 5 %, respectively.

$$\text{Var}[X_t] = kf(\mu_t) \quad (3.5)$$

where $k > 0$ and $f = f(\mu_t)$ is any smooth, monotonically increasing function singly depending on the mean level μ_t of the underlying process.

Consider next the following artificial data set where the reference and test intervals are subject to three different normal distributions each for a general f :

$$\mathbf{Y}_{rf}(i) = \begin{cases} K\mu_N + \sqrt{f(\mu_N)}Z & \text{if } i = 1, \dots, n_{rf}/3, \\ K(\mu_N + 1.5) + \sqrt{f(\mu_N + 1.5)}Z & \text{if } i = n_{rf}/3 + 1, \dots, 2n_{rf}/3, \\ K(\mu_N + 2.5) + \sqrt{f(\mu_N + 2.5)}Z & \text{if } i = 2n_{rf}/3 + 1, \dots, n_{rf}, \end{cases}$$

$$\mathbf{Y}_{te}(i) = \begin{cases} K(\mu_N + 2.5) + \sqrt{f(\mu_N + 2.5)}Z & \text{if } i = 1, \dots, n_{te}/3, \\ K(\mu_N + 3) + \sqrt{f(\mu_N + 3)}Z & \text{if } i = n_{te}/3 + 1, \dots, 2n_{te}/3, \\ K(\mu_N + 3.5) + \sqrt{f(\mu_N + 3.5)}Z & \text{if } i = 2n_{te}/3 + 1, \dots, n_{te}. \end{cases} \quad (3.6)$$

Here, $Z \in \mathcal{N}(0, 1)$, $\mu_N = 10$ and n_{rf}, n_{te} denote the interval lengths as before. This artificial data set was generated with a total of five different normal distributions, one of which overlaps both intervals. The introduction of multiple

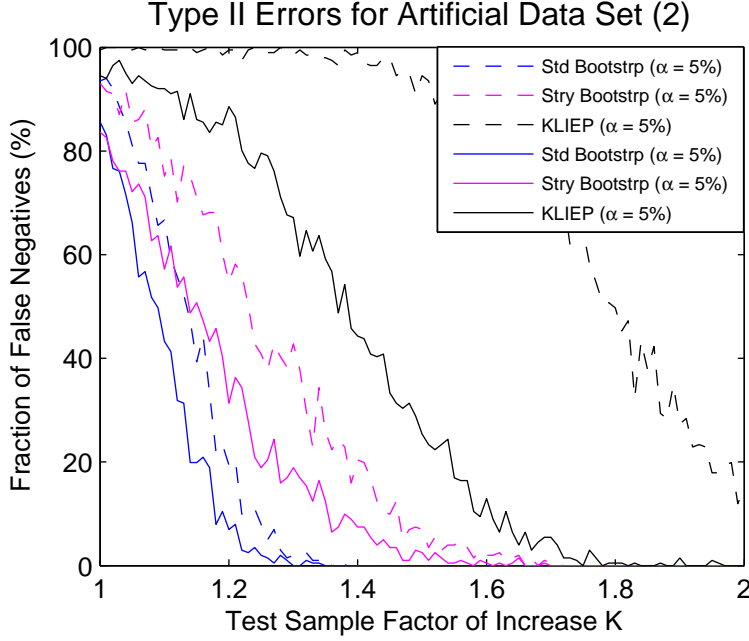


Figure 3.5: Type II errors rates for a realization of the second artificial data set defined by equation (3.6) (Model (2)) plotted as a function of test interval factor of increase K for $(n_{rf}, n_{te}) = (60, 90)$. Here the level of significance is 1 % and 5 %, respectively.

normal distributions in each time interval implies that $\mathbf{Y}_{rf}, \mathbf{Y}_{te}$ are two non-stationarity data series, but nevertheless the data set does not exhibit a highly non-stationary behaviour. The distributions were generated in a perturbative manner with respect to the base mean level μ_N and the variance is proportional to the mean according to (3.5).

The two models that are subject to (3.6) are now formed by choosing an appropriate f . Let

$$f = f(\mu_t) := \begin{cases} \mu_t & \text{(Model(1))}, \\ \mu_t^2 & \text{(Model (2))}. \end{cases} \quad (3.7)$$

First consider the case where $f = \mu_t$ (Model (1)). Figure 3.3 shows a realization of one data set generated with respect to the first artificial data set and Figure 3.4 displays the type II (false negative) errors for different K given two choices of the significance degree α (1 % and 5 %, respectively). The length of the reference and test intervals were chosen as $n_{rf} = 60$ and $n_{te} = 90$.

Consider now the case where $f(\mu_t) := \mu_t^2$ (Model (2)). That is, instead of the variance being proportional to the mean level, let the standard deviation be. Injecting more randomness into the model while keeping the mean levels

constant in each subinterval [in the reference and test intervals] implies that the models can be compared with additional noise. The type II error results obtained from running a simulation with the same parameter settings but with $f = \mu_t^2$ yields the type II error curves found in Figure 3.5.

4 Discussion

4.1 Type I Errors

Two comments are in order for explaining the type I error curves obtained in the previous section (Figure 3.1). First, the analysis is performed on a stationary data set with real-world data which we assume defines the correct null hypothesis; hence problems arising with non-stationarity is not an issue. Secondly, the results from running a type I error analysis proved to be different for the values of α that were chosen, namely $\alpha = 1\%$ and $\alpha = 5\%$ (Figure 3.2). In particular, KLIEP did not reject H_0 for any value of $\frac{n_{te}}{n_{rf}}$ given $\alpha = 1\%$, in contrast to the bootstrap methods. On the other hand, it produced the largest number of false positives at the level of significance 5 %.

The divergent results for KLIEP can partly be explained by the calibration parameter μ . If the data set under calibration is not fully representative, then μ is likely not to be optimally calibrated. Recall that a cleansing and outlier deletion process phase must be gone through for every unit test subject to calibration with KLIEP, and that the choice of the cleansing parameters are input-specific. With a low α , the parameter is likely chosen more accordingly, which can be observed in Figure 3.2.

Another source of error is that the length of the reference interval becomes very small since the change point is kept fixed when the lengths of the two intervals are changed. Since each observation is integer-valued, this requires a larger sample size than if the observations were drawn from a continuous probability distribution. This is actually one major justification for not applying the median as the test statistic within the bootstrap change point detection framework. Note that the possibility to detect changes using the median difference as the test statistic is likely to reduce performance of the change point detection algorithm since the data set contains integer data with potentially low variability. Moreover, it is not clear why the observations have been rounded off to the nearest integer. For accuracy reasons, it would be more sound to keep at least three decimals.

4.2 Type II Errors

The results from the previous section show that the standard bootstrap method handles stationary data and in particular non-stationary data sets surprisingly well when the reference and test intervals differ in distribution. That is, the standard bootstrap outperforms the stationary bootstrap and KLIEP with respect to type II errors for both stationary and non-stationary data sets (Figure 3.2, Figure 3.4, and Figure 3.5). Although the standard bootstrap method is based on independently and identically distributed data, the simulation results show that the method is capable of identifying a potential change point if the test interval differs significantly from the reference interval. Recall that the respective artificially generated data sets are not highly non-stationary and in addition, they are locally stationary in subintervals in the reference and test

intervals.

The stationary bootstrap performs slightly worse with respect to type II errors. The main reason could be that the reference interval is relatively small and that the stochastic block sizes could have an impact on the performance. For block resampling algorithms, the underlying sample size should be sufficiently large so that the sample becomes representative.

Turning over to type II errors for KLIEP, it is without doubt that it is least sensitive for detecting real changes in a data series. The type II error curves for the stationary data set shown in Figure 3.2 do not approach zero as K tends towards 2. This is in contrast to the type II errors for the standard and stationary bootstrap methods, which approach zero for $K \approx 1.2$ and $K \approx 1.5$, respectively. However, the situation is better given a realisation of the first artificial data set (Model (1)) as shown in Figure 3.4, but the method still lags behind the bootstrap methods. Adding randomness of squared the magnitude of the standard deviation in Model (1) also worsens performance. In particular as $K \rightarrow 2$, the type II error for KLIEP with $\alpha = 1\%$ has not yet approached zero.

If the goal is to minimize type II errors, a sound strategy would be to consider the standard bootstrap method.

4.3 General Topics

Some comments should be made regarding the assumptions stated in subsection 1.2 and the choice of parameters. The most important assumption concerns the belief in H_0 , which is based on the naive assumption of a completely stationary training set. In reality, this is not entirely true; non-zero autocorrelations and extreme values in the unit test samples occur to varying extent. Autocorrelation cannot be resolved, but outlier deletion is a user-controlled process and requires reasonable judgement from the analyst. In this paper, we could be more conservative regarding the threshold for determining whether a data point is considered an outlier or not, but the results would probably be less representative with a less aggressive outlier threshold.

The assumption of independency between unit tests cannot be checked, since one needs to examine the original code and check if it is actually true that the unit tests are atomic. Moreover, it still remains unclear to what degree non-stationarity affects the performance of the methods. The assumption of approximate stationarity was made to motivate and justify the incorporation of the models. Now, this assumption strongly relates to the lengths of the reference and test interval lengths. To enhance the approximation, the analyst must know the daily measurement frequency and how often changes in the code package are pushed from local systems to the main servers. Choosing n_{rf} and n_{te} appropriately is vital for the accuracy of the algorithm.

Above we touched the issues regarding contaminated data. This is one large source of error, since the type I and type II errors depend on how effectively non-representative tests are filtered out. Once passed the filtering stage, each unit test is treated identically and with the same weight. Another, at least as important error source is the relatively small amount of unit test data that was

used. After data cleansing and outlier elimination, the size of each test must exceed the threshold value of 160 observations. This is not much data. More data had without doubt produced more accurate results. In particular, the outlier elimination procedure would be more accurate.

The approach to change point detection taken in this paper does come with a few complications. Applying a bootstrap technique to non-stationary time series is of course risky; nevertheless, the bootstrap proved to perform well on the artificial data sets. Moreover, time dependency was not taken into account in the KLIEP algorithm. Incorporating local time dependency was actually attempted, but the results did not look promising and hence the approach was abandoned. The extension of the change point algorithm using KLIEP taking local time dependency into account is implemented by treating each observation in a given time interval as a vector of k components. This yields a Hankel matrix of size $k \times n$, where n is the number of observations in the given time interval.

It should also be noted that KLIEP was originally invented as an offline batch processing algorithm. An online version has been proposed by Sugiyama *et al.*, but it requires partly that one picks an appropriate value of the forgetting factor. Another, more recent algorithm developed by Sugiyama and one of his PhD students can be found in the bibliographic section ([8]). The paper proposes a method based on relative density ratio estimation using the relative Pearson divergence measure. Their method could *e.g.* be applied to this problem as a continuation of the current work.

The missing part in this project was a data set with potentially non-stationary properties that could be used for validation¹¹. Now the artificial data sets were generated for this purpose, but they were generated using different normal distributions; hence the performance of the algorithms was measured with respect to a *parametric* model when in reality observations might come from a distribution that has no closed form. In conclusion, investigating algorithms with *known* change points from any non-stationary time series data set could be a natural continuation of the current work when more data becomes available.

At last, the models used in this paper constitute only a small subset of the many approaches to change point detection. However, the limiting factor in many papers is the use of parametric models. The problem of finding an effective global change point detection algorithm needs a robust non-parametric algorithm that does not assume that data follows a certain probability model, nor that it requires the user to possess knowledge about specific input parameters. Our results, based on two bootstrap methods and a density ratio estimation model, indicate that the algorithms work as desired and with relatively promising results. However, one could as an alternative attempt non-parametric predictive modeling or perhaps a state-space model such as the Kalman filter. Nevertheless, these approaches would require careful pre-analysis so that knowledge about how to overcome their individual problems is known a priori.

¹¹Such a data set with known change points could therefore be used to "stress test" the algorithms without the need for artificial data sets.

5 Conclusion

Robust performance measurement in large code packages is a challenge tech companies are faced with today. The standard way of measuring performance in code is to run many independent atomic tests, unit tests, measure the execution time of each such test and store the results in a database. The fundamental problem that arises is whether a significant (statistically speaking) increase in observed execution time for a given unit test has occurred or not. In literature, this problem is called change point detection and is the subject of this study.

This paper proposes three models to approach non-parametric change point detection in time series data. The main idea is to partition a data sequence (sampled from any unit test) into two consecutive time intervals, where the intermediate point is a candidate change point. A change point is defined as a point where the statistical properties of a distribution changes. The null hypothesis thus reads that no change has occurred, in contrast to the alternative hypothesis stating that an upward shift in the mean level of the reference distribution has occurred. When a model has been selected, an appropriate test statistic is chosen and defined over both intervals to test whether a significant increase in observed execution time has occurred or not. Two of the three methods, the standard bootstrap and the stationary bootstrap, use statistical resampling to infer about whether the mean difference between the two time intervals is larger than zero. The third method, Kullback-Leibler Importance Estimation Procedure (KLIEP), estimates the density ratio between the test and reference time interval without going through direct (naive) density estimation. The test statistic used for KLIEP is based on the log likelihood ratio between the two distributions and tests whether the mean difference (larger than zero) is significant.

The performance of the three methods has been tested on a stationary training data set and on two artificially generated non-stationary data sets. It was found that all models satisfy the null hypothesis for a stationary data set up to a pre-specified error tolerance α . However, when the interval lengths are changed such that the test interval becomes larger than the reference interval, all models tend to invalidate the null hypothesis when the ratio between the interval lengths exceeds a model specific threshold. In particular, this behaviour is observed for all models if the level of significance $\alpha = 5\%$. By contrast if $\alpha = 1\%$, KLIEP proved to not reject the null hypothesis a single time whilst the bootstrap methods violated the null hypothesis when the respective ratios exceeded certain threshold values. Finally, a comparison between type II errors was conducted. The standard bootstrap was then shown to minimize type II errors for both the stationary data set and the non-stationary data sets, proceeding the standard bootstrap and KLIEP.

Whether one model is to be preferred against another depends on the tolerated fraction of type I and type II errors that is acceptable for the application and by human subjectiveness. The standard bootstrap method, however, turned out to work surprisingly well in the performance tests.

References

- [1] Y. Kawahara, M. Sugiyama. Change-Point Detection in Time-Series Data by Direct Density-Ratio Estimation, *2009 SIAM Int'l Conf. on Data Mining (SDM 09)*, pp. 389-400, *Nugget Avenue Sparks, NV*.
- [2] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. v. Bunau, M. Kawanabe. Direct Importance Estimation for Covariate Shift Adaptation, *Annals of the Institute of Statistical Mathematics*, vol 60, no .4, pp. 699-746, 2008.
- [3] E. Acua and C. Rodriguez. On Detection of Outliers and Their Effect in Supervised Classification.
- [4] K. Singh and M. Xie. Bootstrap: A Statistical Method.
- [5] E. Paparoditis and D. N. Politis. Local Block Bootstrap, *C. R. Acad. Sci. Paris, Ser. I* 335 (2002) 959-962.
- [6] D. N. Politis and J. P. Romano. The Stationary Bootstrap (1994), *Journal of the American Statistical Association*, 89, pp. 1303-1313.
- [7] Kunsch, H.R. (1989), The Jackknife and the Bootstrap for General Stationary Observations, *Annals of Statistics*, 17, 1217-1241.
- [8] S. Liu, M. Yamada, N. Collier, M. Sugiyama. Change-Point Detection in Time-Series Data by Relative Density-Ratio Estimation, *Journal Neural Networks archive Volume* 43, July, 2013 pp 72-83.