TÉCNICO LISBOA

Online anomaly detection in univariate data streams

Carlos Manuel Mendes Branco

Thesis to obtain the Master of Science Degree in

Engenharia Informática e de Computadores

Supervisor: Prof. Cláudia Martins Antunes

Examination Committee

Chairperson: Prof. Alberto Manuel Rodrigues da Silva
Supervisor: Prof. Cláudia Martins Antunes
Member of the Committee: Prof. Rui Miguel Carrasqueiro Henriques

July 2020

Acknowledgments

This work was supported by national funds by Fundacão para a Ciência e Tecnologia (FCT) through project GameCourse (PTDC/CCI-CIF/28939/2017).

I would like to thank my mother for her friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this path would not be possible.

I would also like to acknowledge my dissertation supervisor Prof. Cláudia Antunes for her insight, support and sharing of knowledge that has made this thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

Anomaly detection is a crucial field nowadays, ranging from fault detection in machinery, surveillance, fraud detection, and others. However, it is not always easy to tackle the volume and speed of arriving data, and issue anomaly scores on it in an instantaneous manner, as usual in datastreams. The present work centers on the problem of detecting anomalies over continuous and endless univariate time series - *datastreams*. In particular, we propose to use the Matrix Profile method for identifying discords, and managing them both probabilistic and similarity wise, over both the original and differentiated datastream. Additionally, we used the Fourier analysis to identify the main frequencies within the time series, in order to define the window size parameter. Finally, we validate our method using experimental results over well-known dataset in anomaly detection.

Keywords

Anomaly detections; timeseries; datastreams; All Pair Similarity; Discord Management; Matrix Profile.

Resumo

A detecção de anomalias é um campo crucial hoje em dia, desde a detecção de falhas em máquinas, vigilância, detecção de fraudes, entre outros. No entanto, nem sempre é fácil lidar com volume e a velocidade de chegada dos mesmos equanto se emite atempadamente notas de anómalia de forma instantânea, como é habitual nos fluxos de dados. O presente trabalho centra-se no problema da detecção de anomalias em séries temporais univariadas contínuas e intermináveis - *datastreams*. Em particular, propomos a utilização do método Matrix Profile por forma a identificar pontos discordantes. De seguida, gerimo-los quer probabilísticamente, quer a nivel da semelhança de sequências, sobre o datastream original e diferenciado. Além disso, utilizamos a análise fourier para identificar as principais frequências dentro das séries temporais, a fim de definir o parâmetro de tamanho da janela, necessário para o Matrix Profile. Finalmente, validamos o nosso método utilizando resultados experimentais sobre um conjunto de dados bem conhecido na detecção de anomalias.

Palavras Chave

Detecção de Anomalia; Fluxo de Dados; Séries Temporais; All Pair Similarity; Gestão de Sequências; Matrix Profile

Contents

List of Figures

List of Tables

List of Algorithms

1

Introduction

Finding interesting behaviors and events is hardwired into our brains due to millennia of evolution. But what characterises this difference and distinctness in events? Is it something coarse, or more subtle to a point where one might almost not distinguish it from what is normal?

The following work centers on the problem of detecting anomalous behaviors over continuous and endless univariate time series, usually known as *datastreams*.

Anomaly detection has garnered considerable attention in the last couple of years [1]. In the case of datastreams, several problems are frequent, such as the volume of data generated, possible endlessness of data, timely detection of change and anomaly, lack of standards for classifying said streams, and the ever-changing nature of data (phenomena known as *concept drift*) are recurrent.

In order to address the problem at hand, detailed objectives for the task should be set out to guarantee the quality of the model. Metric definition, which should encompass spacial or time constraints ought to be planed. Numerous works have emerged in this relatively young area of study over the past years, each of which tackling its unique problems. Putting things briefly, the main problem is on how to maintain knowledge of the past in order to infer whether the present is normal or abnormal while keeping as few data as possible, answering in minimal time.

In this work, we propose an unsupervised algorithm capable of learning the normal behavior of a datastream, calculating an anomaly score at each time step, with minimal, and constant overhead. This learning mechanism will work on top of Eammon Keogh's Matrix Profile, using it's resulting distances to identify anomalies.

The main contributions of this work are the adaptation of Eammon Keogh Matrix Profiles algorithm to work over a univariate datastream, for the detection of anomalies in a given domain.

This is done via two main approaches for the management of discords reported by the computed matrix profiles, namely one probabilistic, and one distance based. In addition, we also propose to apply the methodology not only over the original domain but also on the differentiated sequences.

Finally, a method for automating the choice of an appropriate window size via the Fourier is also introduced and tested.

Furthermore, we have found a way not only to detect anomalies, but also deal with concept drift, as the proposed method inherently detect deviating concepts due the similarity function used, and the maintenance discord algorithm thoroughly demonstrated.

The aforementioned work is organised as follows: first, we will explore the concept of anomaly in data science, defining the scope and target of our identification procedures in chapter 2. As previously mentioned, anomalous behaviors can come from underlying problems in the system or from the malfunctioning of data-gathering devices.

Then, we explore and compare different methods for the identification of underlying problems in data, and whether they do, or do not, fit the nature of the data at hand. More precisely, datastreams

and problems associated with the inherent nature of time and volume of the data are also discussed in chapter 2.

Thirdly, we will discuss evaluation metrics for the proposed task in chapter 3 and see when, and under which conditions they are fit for each model, closing with a case study analysis. This case study will be revisited further ahead when testing our algorithm, such that we can discuss the seen behavior given a known and explored dataset.

Next, we present our proposal to address the problem described - the adaptation of an all pair similarity search method to detect annotated anomalies in a given time-series, over a datastream, in chapter 4. Here, two distinct methods are proposed, one being a simple statistical approach, and the other a behavior learning, sequence comparison mechanism.

In chapter 5 the classifiers proposed in chapter 4 are tested. Benchmarking of scores and quality metrics are recorded during and results are discussed and compared.

Finally, we conclude the document reporting our main findings, results, and guidelines for future work.

2

Knowledge discovery in data streams

Contents

2.1 Basic concepts

Data science, data mining, and knowledge discovery (KD) are some of the terms used along the years to describe the process of acquiring implicit, previously unknown, and potentially useful highly abstracted information from data.

The knowledge discovery in databases processes, usually called KD process are first introduced by Piatetsky-Shapiro & Frawley in 1991 [2], and later formalized by Fayyad et al, 1996 [3], leading to the creation of the first de facto standard in data science, CRISP-DM, four years later [4].

The five steps defined by Fayyad [3] consist of a continuous pipeline comprising five phases.

Selection corresponds to the first phase, where the dataset is created with the available variables on which the discovery process will be performed. Following selection comes a pre-processing stage, the point at which the data is cleaned and prepared in order to increase its quality [5]. Then a transformation step takes place, where dimensionality reduction and transformation methods are applied in order to further enrich the dataset or make it more manageable. Following data transformation, a step where the data mining process happens takes place. This step can be expanded into 3 consecutive sub-tasks [5]: choosing the data mining task, data mining algorithm and applying it on data followed by an interpretation , and evaluation of the mined patterns. Last but not the least, a consolidation step takes place where knowledge is used for further tasks [5].

The above steps can be resumed by the following figure.

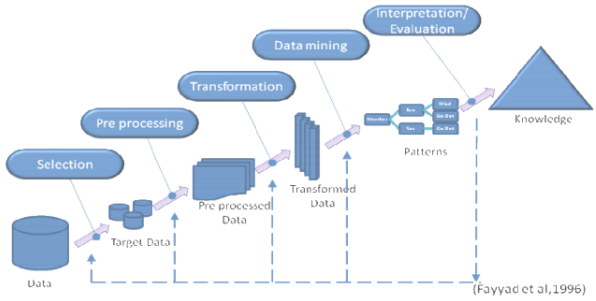


Figure 2.1: Fayyad 1996 Data processing pipeline

The SEMMA process (Sample, Explore, Modify, Model, and Assess) was later created by the SAS Institute and also considers a five-stage cycle, closely corresponding to the one proposed by Fayyad [6]. As this cycle was created by the SAS Institute and reflects a cycle present in their proprietary data mining tool, the SAS Enterprise Miner software, we shall not be viewing it in detail. It is, however, a notable mention in the consolidation process of the processes used by the data mining community, and a standard that further motivated research on the topic.

Standards provide people and organizations with a basis for mutual understanding and are used as tools to facilitate communication, measurement, commerce, and manufacturing. Furthermore, standards

play an important role in multiple fields by facilitating business interaction, enabling compliance with law and regulations, speeding up innovation, providing interoperability between systems, services, and processes.

The CRISP-DM methodology (Cross-Industry Standard Process for Data Mining) is a six-stage cycle successor of SEMMA, and was proposed as the first standard for the discovery process.

The first step, *Business Understanding*, concerns with the project objectives and requirements from a business perspective, converting the resulting knowledge into a data mining problem, culminating in a preliminary plan of objectives to achieve.

Then, *Data Understanding* is the second step in the CRISP-DM cycle, where an initial collection of data happens, data quality problems are identified, the first insights are derived and initial hypotheses for hidden information are formulated.

Data preparation follows, where the data is manipulated in order to create a final dataset. Next comes the *Modeling* stage, where different techniques are experimented, and parameters tuned for optimal results.

Before the last step, the *Evaluation* stage frames a thoroughly evaluation and comparison of the discovered information, in order to assess if they meet the earlier proposed objectives.

Finally is the *Deployment* step, where the models are deployed to production and used in real-life scenarios.

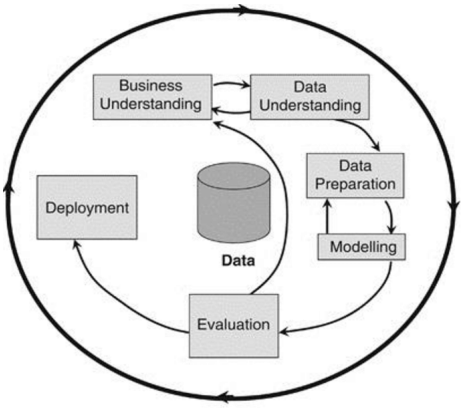


Figure 2.2: CRISP-DM life-cyle

More recently another standard appeared, the ASUM-DM (Analytics Solutions Unified Method) which attempts to bring enterprises closer in their research methods, thus promoting interoperability and portability within the data science industry, by combining classical data mining techniques with AGILE principles.

As ASUM is very similar to CRISP [6] and the higher we go in the abstraction (which is typical for business understanding) the more generalist concepts we find. ASUM is yet another framework that provides coarse steps for data science application from a business level, which does not meet our

initially set criteria, to disambiguate and provide formal methods on which we will base our investigation. For the more interested reader in business integration, the ASUM project can be found at [7].

2.2 Data streams

With the evolution of computers and information gathering processes, we have reached an era where observed signals from multiple sources are collected in a continuous ever-growing flux. Because of this, methods for handling such data and extracting valuable insights experience higher demand than ever.

Thus, the process of knowledge discovery gives rise to two additional problems namely 1) the reception and handling of the data at an increasing pace and volume and 2) the process of acquiring insight on it under constrained time and space.

New methods for data analysis appear every day and new problems and characteristics of said data keep being discovered, hence our need to specify the multiple steps from reception of the data up to issuing insights on said collections.

A data stream is a potentially unbounded and ordered sequence of instances that arrive over time [8]. For the remaining work, we will assume that the data at hand was generated from sources that do not experience byzantine faults, i.e. sensors will always emit ”correct” measurements, and do not fall under states where inaccurate data is generated for the mining process [9]. Consequently, we can state that the datasets are real and accurate, thus reflecting reliable and faithful information while allowing for noise from the generation process. If such was not the case and byzantine faults from sensors were allowed, not only it would be required to find knowledge in data, but also assess the quality of said generation mechanisms, which goes beyond the scope of the work. We will also assume that data is already presented in increasing time steps and does not suffer from delays, duplication, or information loss, thus implying a cleaning process took place previously or was never needed.

2.2.1 Nature of Data

Before going further ahead let us make the distinction between time-series and subsequences.

Definition 1. ***Time Series*** *A time series T is a sequence of real-valued numbers t_i: T= t_1, t_2, ..., t_n where n is the length of T.*

Definition 2. ***Subsequence*** *A subsequence $T_{i,m}$ of a T is a continuous subset of the values from T of length m starting from position i. $T_{i,m} = t_i , t_{i+1} ,..., t_{i+m-1}$, where $1 \leq i \leq n-m+1$.*

Data points can be seen as a collection of instances described by a set of attributes. These attributes originate from a wide variety of sensors, either physical or virtual, that monitor usage and performance

of various systems. Furthermore, and in the context of time-series, a timestamp is associated with said instances.

If the aforementioned data originates from one single sensor, thus reflecting one single attribute, the time series is said to be *univariate*, whereas if more than one attribute is associated with a single timestamp, the series is categorized as *multivariate*.

Observations from sensors can arrive out of order, duplicated, delayed, or maybe never arrive. Time points and associated measurements can be correlated or independent. Because of these problems, safeguarding steps should be taken to maximize the quality of the input data, and consequently of the applied classification technique. Guaranteeing that no instance arrives out of order, i.e. observation O at time O_{t-1} arrives before O_t, no duplicate observations are found and no loss of data occurs are identified as research subjects of their own. Moreover, most datasets researched throughout various works, some of which presented in the references section, derive either from the Numenta Benchmark, KDD competitions, or the UCR website [10]. Noise, however, seems to be present in all mentioned datasets and should be either removed or learned to be ignored.

2.2.2 Novelty

Due to the evolving nature of generated data and patterns present in it, we must differentiate between anomaly and novelty.

Novelty detection corresponds to identifying an incoming pattern as being hitherto unknown. This phenomena arises if we consider that data is expected to evolve over time, by being generated from dynamically changing environments where non-stationarity is typical [11].

If we blindly applied traditional anomaly detection methods to non-stationary time series, resulting conclusions would be deemed meaningless as values only have a meaning within specific context frames, and no meaning in others. Furthermore, we may want to integrate these new concepts in our models in order to better approximate real-life usage conditions.

2.2.3 Concept Drift

Novelty detection corresponds to detecting the appearance of new behaviors in the data that may initially appear as anomalous when, in fact, correspond to the introduction of a new concept that may arise from underlying possible changes in the generation concepts.

Concepts can change regarding prior assumptions or posterior concepts. This can be either by changing apriori distributions, or evolving posterior concepts (something that was once unusual has become normal) [11].

To better understand what is concept drift, we ask the reader to imagine the hypothetical scenario of water consumption in a dwelling over a period of time.

If we detect a sudden increase in water consumption we may want to warn the owner that a pipe might have broken. This oscillation in consumption accompanied by no changes in the ”useful” consumption (original concept), experienced by the owner, corresponds to an anomaly with no concept change. This is known as a *real drift*.

On the other hand, if the same dwelling doubled its tenant occupation, the oscillation does not correspond to an anomaly, but rather the introduction of a new concept. The new behavior of consumption and a new apriori assumption replace the old ones. Hence, the concept has changed from what was originally assumed and what was once an abnormally high consumption might be seen as normal. This is known as a *virtual drift*.

Finally, we have *population drifts* that correspond to changes in the sampling process. I.e. if we change the way we gather data (swap a sensor for a more accurate one) the resulting observations are expected to alter, with no change in prior distributions or posterior concepts.

Furthermore, drifts may be classified based on their behaviour being either *sudden*, *incremental*, *gradual*, *reoccurring* or *sporadic* [8, 11], thus domain driven information is required to specify which of the two concepts we want to detect, namely finding changes among patterns and within known patterns or simply finding abnormal values within unknown pattern(s) or stationary series.

Below, in figures 2.3 and 2.4, we provide a visual depiction of the latter phenomena for easier comprehension, where X is the prior and y the posterior.

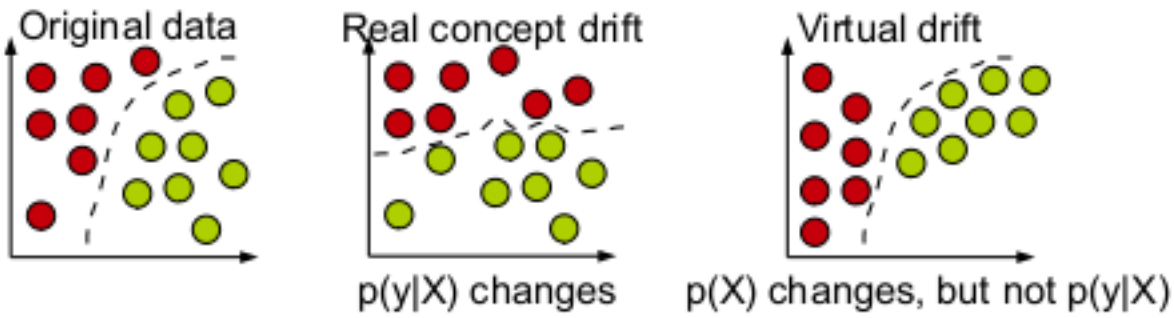


Figure 2.3: Types of drift

2.2.4 Anomaly Detection

Anomalies can be widely defined as patterns in data that do not conform to expected behaviour [1, 12]. These can fit into one of the following three types [1, 13]:

Figure 2.4: Reoccurring drift patterns

- `Point Anomalies`: if an individual data instance can be considered anomalous with respect to the remaining data. As a real-life example, consider credit card fraud detection. To simplify our example assume the data is defined using only one feature: the amount spent. A transaction for which the amount spent is very high compared to the ”normal” range of expenditure for that person will be a point anomaly.

- `Contextual Anomalies`: if a data instance is anomalous in a specific context. Take as an example, the temperature 35ºC. In summer one might think this as normal temperature, however, if recorded during winter (where temperatures are lower), this would imply a faulty sensor or at least cause an investigation as the value is non-habitual during that season.

- `Collective Anomalies`: if a collection of data points is anomalous with respect to the entire data set. Individual values may be normal however, when placed together in a sequence or context may reflect and abnormal behavior. For example, assume the execution log of services in a remote desktop. Buffer-overflow, SSH, FTP are all common instructions, however when presented in the previous sequence reflect an attack vector or rather common infected behavior in a machine.

3

Literature Review

Contents

The following section concerns with introducing multiple approaches for the anomaly detection domain. A brief overview of the major categories and outcomes will be made, focusing on the matrix profiles as a possible solution.

3.1 Algorithms for Anomaly Detection

The output of anomaly detection techniques algorithms usually falls under one of two categories. The outcome can be either one, or a mixture of:

1. A continuous *anomaly score* corresponding to the level of trust in the classification of the anomaly. The score allows the analyst to rank anomalies and, therefore, to be able to compare them qualitatively.

2. A binary *label* is issued, declaring if an anomaly took place or not. The method normally consists of the value of the analysed variable surpassing a previously set threshold, which is a tuning parameter.

The previous scoring mechanisms can also be adapted to work in conjunction. The analyst must be aware that the outcomes heavily depend on the nature of the available data. Henceforth, we will be looking at how each class of algorithms works in order to better understand which approach best suits our problem and needs. All methods can be adapted to issue either one of the previously mentioned scoring mechanisms, depending on the task at hand. Thereafter, we will be looking at how each class of algorithms works in order to better understand what model best suits our problem and needs.

Machine learning algorithms can learn in two different modes namely offline or online. *Offline learning* algorithms receive the complete sequence beforehand, whereas *online algorithms* are continuously presented with the data produced, up to the current moment, one point at a time [14, 15].

Because of this clear distinction, and given the nature of streaming data along with the necessity to issue instant decisions, online training modes are mandatory. Otherwise, true streaming would not be achieved (other approaches for approximating streaming are for example batching, however, this is not true streaming) [8, 12, 16]. Issues such as convergence arise from batching approaches when compared to streaming algorithms. Although batch does allow for avoiding some problems previously identified such as out of order arrival of information and duplication since we can reorganize the buffered data.

Furthermore, we can group algorithms based on their ability to handle received data, and whether or not it is labeled. *Unsupervised, Semi-Supervised, and Supervised* tasks are the go-to when we have no labels, one class labeling or full labeling of data, respectively.

The labeling process itself (required for some tasks) can be extremely costly with labels becoming available only after long periods in some cases [1], hence unsupervised learning is usually the most appropriate approach in the context of anomaly detection in data streams.

Thereafter, we provide detailed information on each of these methods, stating the pros and cons.

Supervised techniques assume the availability of fully labeled data for both normal and abnormal instances. However, two major problems arise with these approaches: 1) the dataset should be balanced otherwise detection/classification may be skewed towards the majority class and 2) obtaining accurate representative labels might be extremely difficult [1].

Semi-Supervised techniques assume that the data is labeled for part of the observations. In our context, it is frequent to have just one of the possible outcomes (either normal or abnormal) labeled. Even if partial labels are taken into account, it might be impossible to generate labels for all possible registered faults as some may be of unknown origin, or so subtle that a human may not instantly grasp them. In contrast, it is easier to acquire labeled data for normal instances however, a labeling process still has to happen which might incur a high cost, and may not aggregate all the possible outcomes leading to incomplete profiles of anomalous behavior.

Unsupervised techniques not only do not require any labeling of the data but also do not make the distinction between the train and test data. Algorithms in this category make assumptions that normal observations are more frequent than abnormal ones, and each of the two can be grouped. I.e. there is a clear separation boundary between what is normal and what is not. However, if the previous assumptions do not stand, techniques in this category may suffer from high false arm rates [1]. Said rates should be properly assessed due to leading analysts to disregard anomaly reports as they can be often incorrect and clutter true detections.

There has also been an increasing exploration of anomaly detection methods, approached in multiple works, and at multiple levels of integration (from the device-based and relatively simple strategies to Hadoop and map reduce techniques for large data flows) [1, 12, 15, 17–21].

Orthogonally, anomaly detection techniques can be grouped based on the major approach they follow [1]. Taking into account multiple works from Chandola, we will be viewing a set of approaches featuring multiple learning modes and techniques. In addition, we will be taking a closer look at the Numenta benchmark and it's intrinsic mechanisms for time-series such as profiles for weighting false positive and false negative outcomes differently, how multiple classifications around a given event are modeled within a sliding window to avoid repeated counts and other. Furthermore, this benchmark already provides a set of implementations to compare against that fall under one or more of the following subsections.

3.1.1 Model Based

Model based algorithms use classification methods to train a *model*, or rather a classifier, to distinguish between normal and anomalous instances. This training is made through the use of a set of labeled instances (*train set*) to then classify any given unseen instances. The general assumption for model based algorithms is that a classifier can be learned to distinguish between normal and abnormal classes in a given feature space.

One way to tackle the problem is via neural networks [21,22]. The basic idea for this set of algorithms is that during a first phase a neural network is trained on the normal training data to learn what the normal behavior is and then, provided a given input, the network will output if the said instance is normal or anomalous. The way the learning is done is via the gradient descent formula. Several models have been created around neural networks to either manipulate sound, image, and time taking into account specific characteristics of each problem under analysis (e.g. recurrent vs convolution networks).

Another way to approach anomaly detection is via a Bayesian approach [23]. A basic technique for a univariate categorical dataset, consisting of using a naive bays network, tries to estimate the posterior probability of observing a class label from a set of normal and abnormal instances, given a test observation. The likelihood of observing the test instance given a class prior is estimated from the data set and can be used to detect an anomaly. In order to avoid probabilities equal to zero techniques such as Laplace smoothing [24] can be used.

3.1.2 Parametric Methods

Another subset of model-based methods takes into account parametric methods. Parametric techniques assume that normal data is generated by a parametric distribution with an associated probability density function. The anomaly score of a test instance is the inverse of the probability density function. Alternatively, statistical hypothesis tests can be used to see if a given test accepts or rejects a value as an anomaly. A couple of examples of said methods are the ESD (extreme studentized deviate test) also known as Grubbs test [25] which is used for concept drift detection, and the mean confidence interval [26].

Gaussian based models [27] can be used to estimate maximum likelihood scores and others.

Regression models [28] can also be used, however, the presence of outliers and anomalies in the training data can influence the regression model learned.

3.1.3 Support Vector Machines

Support Vector Machines are another class of learning techniques, used to learn the separating boundary between normal and abnormal instances [29]. Kernels, such as radial basis functions, can be used

to learn complex spaces. Several domains make use of SVMs such as novelty detection in power generation, anomaly detection in temporal series, anomaly detection in audio, and others [1].

3.1.4 Prediction Based

Often depending on statistical methods, prediction based methods will attempt to predict future events or model tendencies. These predictions can, and often include, analysis of past events in order to infer a model or extrapolate the true value of some observation (if in the presence of noise).

Inductive reasoning or regression tasks are common to be found when searching for prediction based models. In addition, we may also want to extract (predict) the true value of an observation. If such is the case, Kalman filters [30] are usual approaches.

Similarly to statistical inference methods, we may want to predict some amount, however, we may have look at past observations as premises rather than a set of points upon which statistical analysis will be performed.

3.1.5 Rule Based

Rule based anomaly detection techniques [31, 32] learn rules that capture the normal behavior of a system, while unseen/abnormal behaviors are classified as anomalies. Association rule mining has been used for both network intrusion detection, and credit card fraud detection. Frequent itemsets are generated in the intermediate steps of association rule mining. A proposed approach is that for categorical data sets the anomaly score of a test instance is inversely proportional to the number of frequent itemsets in which it occurs.

3.1.6 Distance Based

Distance based approaches rely on distance metrics (like Euclidean, Jaccard, Mahalanobis, and others) to infer the outcome of an observation. The assumption here is that similar, thus close, observations will have the same outcome. Furthermore, we can use the notion of density as it is inherently distance related.

3.1.7 Nearest neighbor and Density Based

Nearest neighbor techniques [19,33,34] take into account the assumption that normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors, or in neighborhoods of their own.

Density based anomaly detection techniques [35, 36] estimate the density of the neighborhood of each data instance. Instances that lie in a neighborhood with low density are declared to be anomalous. For a given data instance the distance to its k^{th} nearest neighbor is equivalent to the radius of a hyper-sphere, centered at the given data instance. Density based techniques tend to perform poorly if the data has regions of varying densities. Inherently density assumes the use of a distance metric.

3.1.8 Clustering Based

Clustering is used to group similar data instances into clusters. Clustering-based anomaly detection techniques [1, 37–39] can follow one of three assumptions: Normal data instances belong to a cluster in the data while anomalies do not belong in any cluster; Normal data instances lie close to their closest cluster centroid while anomalies are far away from said centroid, or that anomalies form clusters by themselves different from the normal ones.

These methods are similar to Nearest Neighbor based techniques as both require distance computation between pairs of instances. The key difference between the two techniques is that clustering based techniques evaluate each instance with respect to the cluster it belongs to, while Nearest Neighbor techniques analyze each instance with respects to its local neighborhood.

It is common to see distance based approaches combined with statistical/probabilistic methods to boost classification outcomes as the assumptions may only be true under specific constraints (with the ”restricted space” being identified by other methods).

Another way to look at the problem is via symbolic representations and their relative distances (which implicitly model the discovery of patterns in sequences with the distance as a measure of similarity/dissimilarity) [40].

3.1.9 All Pair Similarity

Multiple works [41, 42] have taken a look at the all pair similarity search problem for identifying patterns of normality and abnormality.

One way to approach this problem is via the all pairs similarity joins for time series [40] which makes extensive use of distance based methods, taking into the account the z-normalized Euclidean distance as a distance measure between pairs, as well as the Fast Fourier Transform in order to approach the time/spatial complexity of the all pair distance computation.

Take into account the following definitions from Eammon Keogh’s first work [40] on the matrix profiles building on the notions of time series and subsequences:

Definition 3. ***Distance Profile*** *A distance profile D is a vector of the Euclidean distances between a given query and each subsequent in an all-subsequences set.*

The distance profile can be considered a meta time-series that describes the time series T that was used to generate it. The first three definitions are illustrated in figure 3.1.

Definition 4. ***All Subsequences set*** *An all-subsequences set A of a time series T is an ordered set of all possible subsequences of T obtained by sliding a window of length m across T: A=*$\{T_{1,m}$ *,,* $T_{2,m}$ *,...,* $T_{n-m+1,m}\}$*, where m is a user-defined subsequence length. We use A[i] to denote* $T_{i,m}$*, the subsequence of T starting at position i.*

Definition 5. ***1-NN-join*** *Given two all-subsequences sets A and B, and two subsequences A[i] and B[j], a 1-NN-join function* $\theta_{1nn}(A[i], B[j])$ *is a Boolean function which only returns "true" if B[j] is the nearest neighbor of A[i] in the set B.*

Definition 6. ***Similarity join set*** *Given all-subsequences sets A and B, a similarity join set* J_{AB} *of A and B is a set containing pairs of each subsequence in A with its nearest neighbor in B:* $J_{AB} = \langle A[i], B[j]\rangle \,|\theta_{1nn}(A[i], B[j])$*. We denote this formally as* $J_{AB} = A \triangleright \triangleleft_{1nn} B$*.*

Definition 7. ***Matrix Profile***

A matrix profile (or just profile) P_{AB} *is a vector of the Euclidean distances between each pair in* $\mathbf{J}_{AB}$*.*

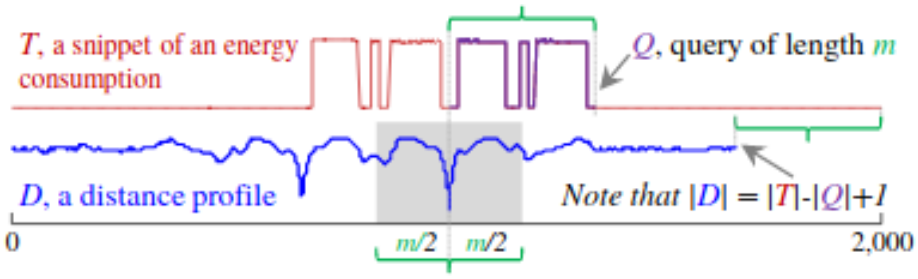


Figure 3.1: A subsequence Q extracted from a time series T is used as a query to every subsequence in T. The vector of all distances is a distance profile. M/2 corresponds to the exclusion zone around a distance profile maximum, and is half of the window size used for the analysis.

The basic task is that, given a collection of data objects, we want to retrieve the nearest neighbor for each object, where an object can be a single event/instance of a subsequence of events present in the time-series.

The output of the method is a series of distance profiles containing the distance between each sequence and all other possible contiguous sequences of a given size, with the highest values corresponding to the largest discords between the selected pattern at a given point in the time-series. The lower distances for points correspond to a high similarity between said sequence pair. Therefore, it is easy to infer that when looking for anomalies, we are actually looking for distance profiles with a majority of high distances, thus indicating the uniqueness (and disagreement) of the sequence with respect to the entire time series.

If we take into account an all-subsequences set A of a time-series T, corresponding to an ordered set of all possible subsequences with size M, by sliding a window across T, we can compute the nearest neighbor between this set and another, B, obtained the same way. The resulting vector will contain the distance between a subsequence starting at the i^{th} element in A and its closest match in B. If we get these vectors to contain, not only the distance but also the sequences, the resulting vector is called a similarity join set, while the distance vector is called a *matrix profile*. During the construction of said profile trivial matches (A[i] == A[j]) are excluded.

In this manner, a matrix profile contains the distance between the sequence starting at the i^{th} position, to each one of the subsequences in the all-subsequences set A, and the best match in another all-subsequences set B. However, we do not need to compare between two different time-series as we can construct a self similarity join set J_{AA} instead of J_{AB}. This way we can explore any given time-series for anomalies.

3.1.10 Other Nonparametric Techniques

Contrary to parametric techniques, in this category nonparametric statistical models are used , such that the model structure is not defined a priori [43, 44], but is instead determined from the data. Techniques under this class make fewer assumptions regarding the data such as smoothness of density when compared to parametric techniques. Here we can find histogram based techniques, kernel function based techniques, hidden markov models and expectation maximization.

3.2 Evaluation

The following section concerns with the evaluation of models. Despite a series of well known formulas for evaluating models being approached, none seems to tackle the problem of evaluating data streams as most of the methods are after the fact and do not take into account the continuous nature of time. However, Numenta has proposed a benchmark in order to aggregate a series of these formulas under a constrained environment, in order to provide a platform for model comparison. The Numenta benchmark seems to be the only one so far to aggregate these metrics and provide a benchmarking platform for online time series anomaly detection, thus we will be using it in order to compare our achieved solution with other proposals. This provides a uniform evaluation environment in parameters such as timely detection, and outcome profile which takes into account the weight of false positive *vs* false negatives.

3.2.1 Evaluation Environment

In order to evaluate our proposed solution several of the following methods will be used, despite being bundled in a single tool, namely the Numenta benchmark.

Multiple evaluation metrics for models are approached in the literature [11, 45, 46], however some pitfalls should be avoided when choosing the evaluation steps, and methods as these may not reflect the true quality of the proposed model [19], additionally biasing the learning process, and model selection. While some metrics take into account class imbalance others do not, where some are pessimistic by reflecting errors made during a warmup phase others apply fading factors to circumvent this problem. Next, we will take a look at the most commonly found measures, their pros, cons, and domains of applicability, i.e. whether they do or do not fit the nature of streaming data (non-stationarity).

First of all, the problem at hand is a binary one, meaning we either detect a time point as anomalous or not. Multivariate evaluation metrics differ from univariate in the sense that they have to a capture an aggregate of multiple classification labels/scores for each attribute in a detailed way and combine them into a final result whereas univariate metrics are more straightforward since there is only one attribute under analysis and no combinations of intermediate results need to be done.

3.2.1.A Confusion Matrix

While evaluating results from a method there are four major possible outcomes. True positives are points reported as anomalies that are indeed anomalies whereas true negatives, similarly, are points reported and non-anomalous that are in fact normal. Then we have the false positives and false negatives which are respectively, points that were reported as anomalies when they were not and vice versa.

A Confusion Matrix is a table where differences between the observed prediction and the true outcome are registered. Comprising true positive (TP), true negative (TN), false positive (FP), and false negative (FN) classifications it is a widely used metric for binary classification performance. However, it has a reduced interpretation in cases where a class imbalance is present and, due to the nature of streaming data, does not reflect change detection in due time.

Furthermore, several metrics can be extracted from it such as:

Definition 8. *Accuracy*

Corresponds to the percentage of correct predictions over the total number of instances evaluated.

$$Acc = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.1}$$

Definition 9. *Error Rate*

Represents the misclassification error in contrast with accuracy.

$$ErrorRate = \frac{fp + fn}{tp + tn + fp + fn} \tag{3.2}$$

Definition 10. *Sensitivity or Recall*

Measures the fraction of positive patterns correctly classified.

$$Sen = \frac{tp}{tp + fn} \tag{3.3}$$

Definition 11. *Precision*

Measures the fraction of an identified event correctly classified.

$$Prec = \frac{tp}{tp + fp} \tag{3.4}$$

Definition 12. *Specificity*

Measures the fraction of negative patterns correctly classified.

$$Spec = \frac{tn}{tp + fn} \tag{3.5}$$

Definition 13. *F-Measure*

With β ranging in from $[0, +\infty[$ with higher β values putting more emphasis on false negatives. Default value is 1 where both false positives, and false negatives are weighted evenly. This may be particularly important in our case since anomalies are usually a minority instance nad F-Measure is widely used in imbalanced situations.

$$F - Measure = (1 + \beta^2)\frac{precision * recall}{(\beta^2 * precision) * recall} \tag{3.6}$$

Furthermore, we have many other measures that can be further used such as ROC AUC, Geometric Mean, Mean Squared Error and others. Not only it is important to measure the accuracy of model predictions but we must also make sure such decisions are issued in due time and models run in resource aware environments [20] as the possibility for never ending streams is a reality. Thus, we must be sure that the least amount of space is used (limited memory) and we can forecast errors in a reasonable time window, otherwise these would be useless if made after the fact or too late.

In addition, we want a low false positive rate as this might influence analysts into disregarding warnings if they are too common. In addition to this, a continuous monitoring of memory usage by the process taking place ought to be enforced in order to guarantee that the previously identified memory constraints are complied with.

3.3 Benchmarking Models

In order to monitor the accuracy of the model a solution found is the Numenta Anomaly Benchmark (NAB) [21], a software package addresses most of the aforementioned problems, apart from the time/space requirements.

First, it starts by providing a stream of points to an algorithm. For each point provided (via a handler function implemented by the algorithm), it stores the response of our solution for each data point.

When the program finishes giving time points, the benchmark tool scans the previously returned scores, using a window between 5% and 20% of the stream length around each found anomaly. The scoring function used takes into account the relative position around the true anomaly does determine the score of the detection. Anomalies detected outside these windows will have the lowest score possible and early detections are rewarded accordingly. The scoring function works as shown in image 3.2.

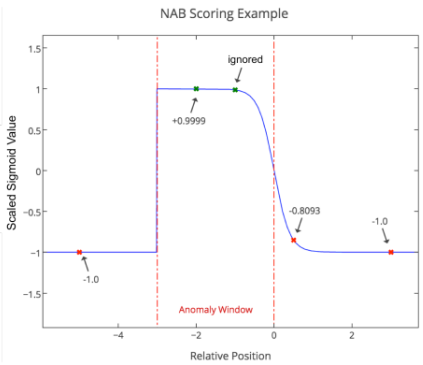


Figure 3.2: Scoring function

This is the most advanced benchmark for algorithm comparison (in the streaming anomaly detection domain) as it comprises not only annotated datasets but also incentives early detection, penalizes late/out of time detections while providing profiles for weighting false positives, and false negatives (for cases where false negatives are much more costly than a closer inspection triggered by a false positive).

The profiles work by giving different weights to false positives, and false negatives. Let A be the application profile under consideration, with A_{tp},A_{tn},A_{fp}, A_{fn} the corresponding weights for true positives, false positives, etc. These weights are bounded $0 \leq A_{fp}$, $A_{tn} \leq 1$ and $-1 \leq A_{fp}$, $A_{fn} \leq 0$.

Given the scoring function shown in equation 3.7, we can aggregate all of its results into a full score

via equation 3.8. The final score (taking into account the profiles) can then be obtained using formula 3.9, where *d* is a dataset.

$$\sigma^A(y) = (A_{tp} - A_{fp})\frac{1}{1+e^{5y}} - 1 \tag{3.7}$$

$$S_d^A = \sum \sigma^A(y) + A_{fn}\gamma_d \tag{3.8}$$

Then, the scores are scaled based on the raw scores of a "perfect" detector (one that outputs all true positives and no false positives) and a "null" detector (one that outputs no anomaly detections). This is called the standard score.

$$S_{NAB}^A = 100 * \frac{S^{\mathsf{A}} - S^{\mathsf{A}}{}_{\mathsf{null}}}{S^{\mathsf{A}}{}_{\mathsf{perfect}} - S^A{}_{\mathsf{null}}} \tag{3.9}$$

The following chapter demonstrates the potential of the matrix profiles when applied to univariate time series anomaly detection. A total of 6 parameters are tested and the reasoning behind the value ranges is detailed. At the end of this chapter, the final model, and prior experiments are used to derive conclusions on the behavior of the model, resulting in a series of pros and cons for the method.

3.4 Case Studies

From the Numenta Benchmark, multiple frequently seen datasets in the literature can be found. One with particularly simple and lay anomalies was chosen, namely the *nyc_taxi* dataset, which will be exhaustively described and characterized in the following subsections.

3.4.1 NY City Taxi Domain

The New York City taxi dataset comprises half hour aggregations for the total number of passengers reported by the NYC Taxi and Limousine Commission (https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page), corresponding to a total of 10320 records, with domain minimum being zero and unknown possible maximum. The annotated anomalies correspond to the NYC marathon, Thanksgiving, Christmas, New Years day, and a snow storm. Anomalous points correspond respectively to the following dates:

1. "2014-11-01 19:00:00"
2. "2014-11-27 15:30:00"
3. "2014-12-25 15:00:00"

4. ”2015-01-01 01:00:00”

5. ”2015-01-27 00:00:00”

Although at first these might seem as 5 anomalies in a 10320 instance dataset, they will grow to

$$(10320 * \textit{window\_size\%})/\textit{tot\#anomalies} \approx 300 \tag{3.10}$$

anomalous points, due to the application of the scoring function and windows provided, as introduced in 3.3, considering a 10% window size with a total of 5 apriori known anomalies.

The datapoints corresponding to each timestamp are numerical, integer, and with non-occurring missing values. Furthermore, the mean, median, and standard deviation for the passenger count is *15137.57*, *16778* and *6939.49* respectively. In addition, the maximum value observed is *39197* while the minimum is *8*.

By boxplotting the dataset we can see that only two instances lay beyond the outlier boundary, defined by the following formulas, where IQR and nthQ represent the interquartile range and nth quartile respectively.

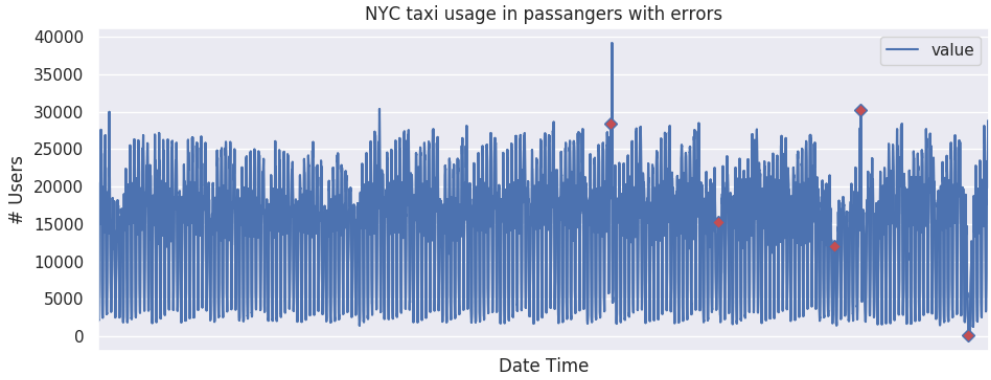
$$Outlier < 1stQuartile - 1.5IQR \tag{3.11}$$

$$Outlier > 3rdQuartile + 1.5IQR \tag{3.12}$$

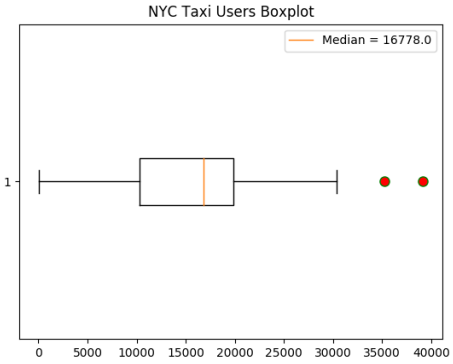
After application of the above formulas to the values on the dataset, we can easily see that the two instances that stand out as outliers are both within the detection window for the anomaly labeled on 2014-11-01 19:00:00 corresponding to the NYC marathon. As has been noted no further statistical outliers can be related to known events.

Furthermore, taxi usage data was plotted in a histogram in order to understand if there were any underlying distributions in usage numbers. This was not verified and can be seen in the histograms in figure 3.3.

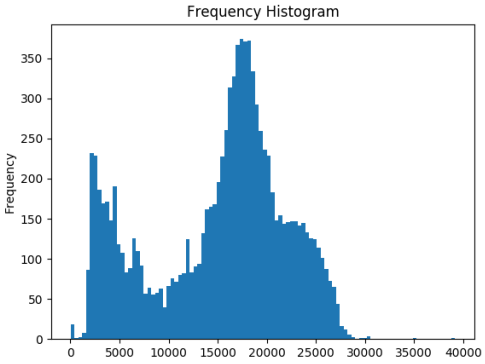
No problems regarding the duplication or lack of timestamps were detected, which reveals that problems such as daylight saving times were previously taken care of.



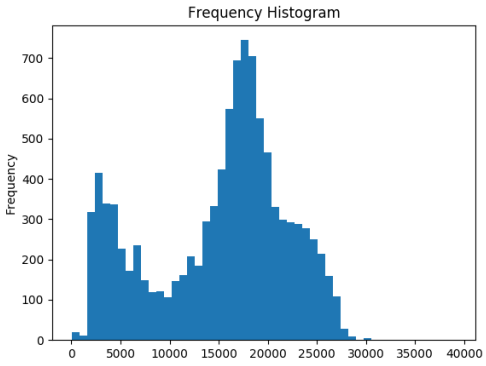
(a)



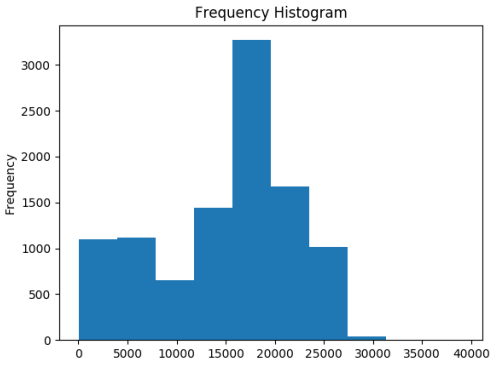
(b)



(c)



(d)



(e)

Figure 3.3: (a) Taxi dataset representation with annotated anomalies as red diamonds ; (b) Results for boxplotting the NYC taxi data. Two outliers are immediately identified, both of which corresponding to an annotated anomaly, the NYC marathon; (c) Passenger usage number histogram aggregated over 100 bins. No known distribution can be identified; (d) Passenger usage number histogram aggregated over 50 bins. No known distribution can be identified; (e) Passenger usage number histogram aggregated over 10 bins. No known distribution can be identified.

Given that the data seems to follow some cyclical trend (as shown in figure 3.3(a)), it would be interesting to extract the average running period underlying it. Visualization of this period can be achieved by running spectral analysis on the entire dataset.

In figure 3.4 we can see that an average week (one without anomalies) should be easily decomposed into multiple additive waves, as it seems to originate from a periodic wave pattern.

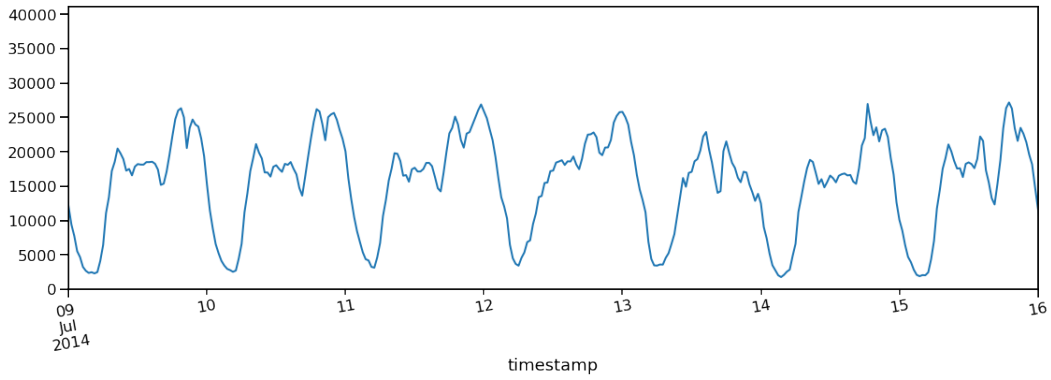


Figure 3.4: Average week of taxi usage. Data plotted reflected an interval between 9th and 16th of July.

First, we start by standardizing the time series. This process corresponds to generating a new set of points in which each point was subtracted by the mean and then divided by the standard deviation of the original points as described by formula 3.13. The results of this operation can be found in figure 3.5 and compared with the original in figure 3.3(a).

$$y_f = \frac{y_i - mean(y_i))}{standard\_deviation(y_i)} \tag{3.13}$$

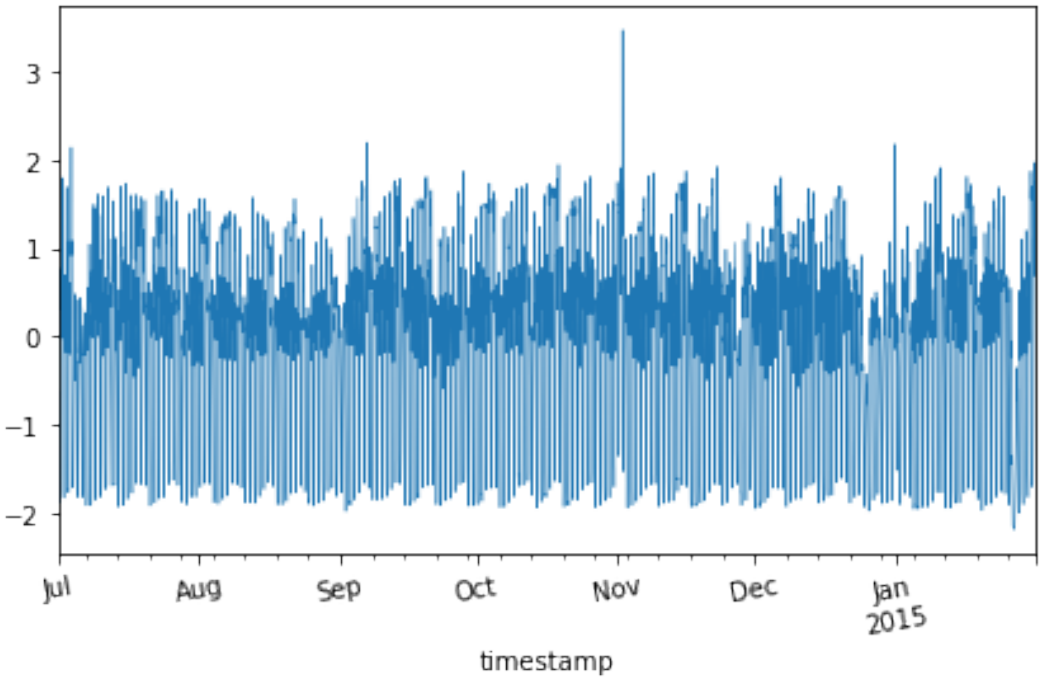


Figure 3.5: Standardize taxi dataset visualization.

After applying the the Fourier transform, if we plot the natural frequencies on an hourly scale, we find a peak of frequency at the 24-hour, and 48-hour mark as shown in figure 3.6, thus implying this is the most common operation regime. We can further infer that this is also the dominant trend throughout the analysis period, as shown in figure 3.7.

These frequencies correspond respectively to a 12 and 24-hour period, something expected of an event following a circadian cycle.

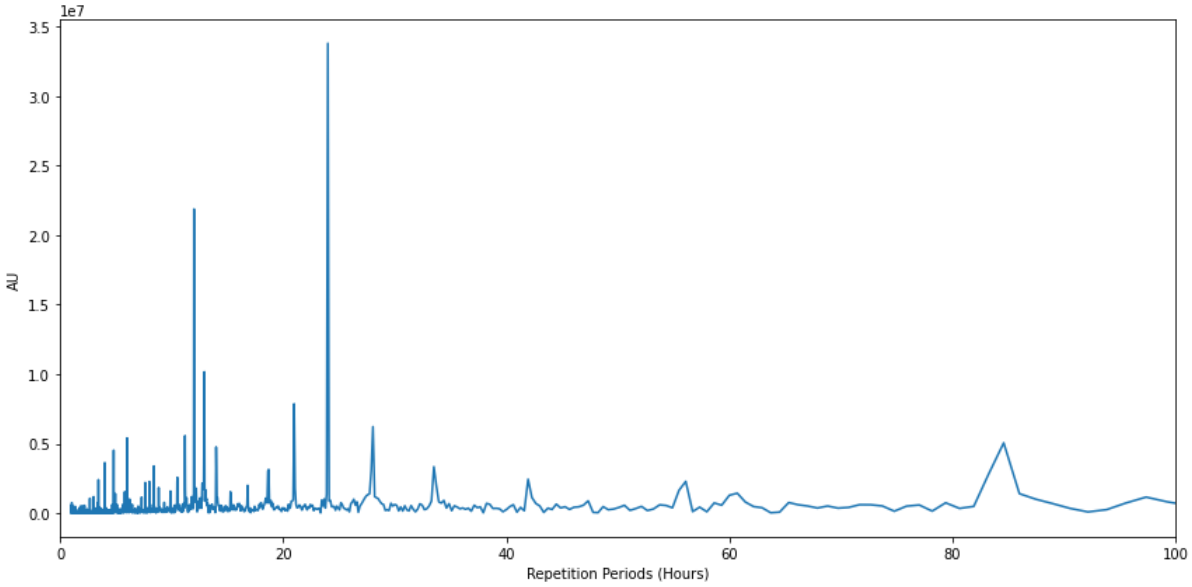


Figure 3.6: Taxi Fourier resuts plotted on an hourly scale.

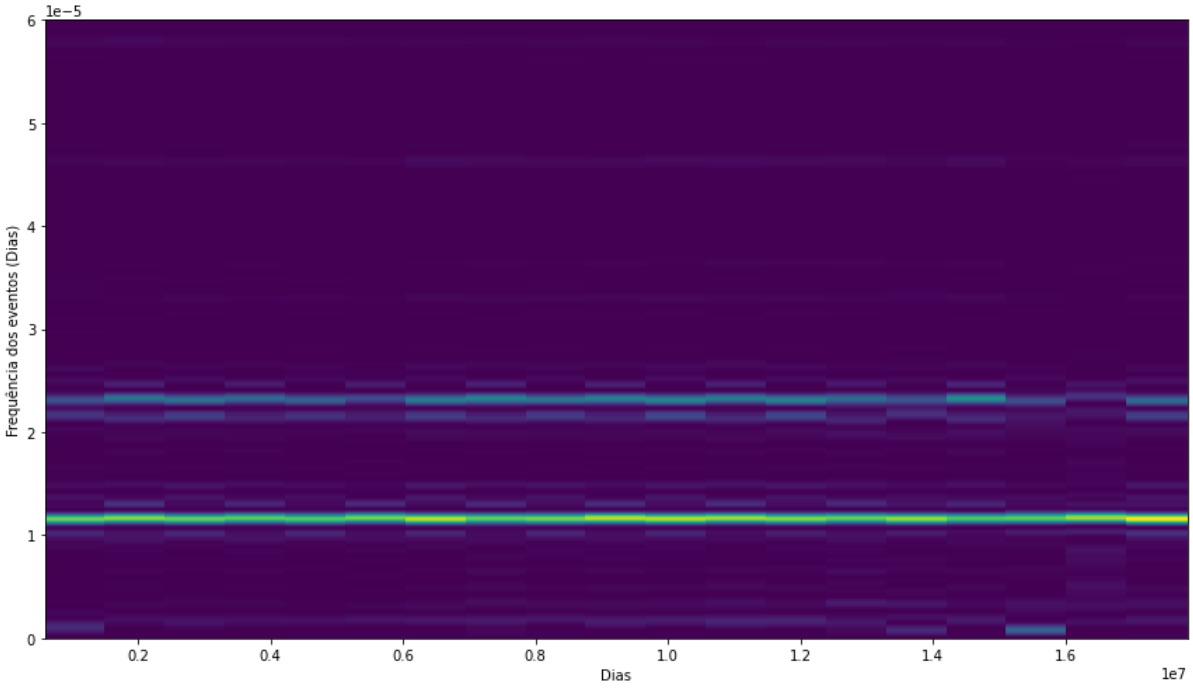


Figure 3.7: Dataset spectogram results for the complete NYC taxi domain.

3.5 Preliminary Technique Comparison

Four major out-of-the-box classifiers, as provided in the Numenta benchmark, have been selected to set a baseline for the following work. Furthermore, the null detector provided in the same benchmark yields an anomaly score of 0.5 for every data point, achieving a final score of 0. The remaining three chosen detectors were HTM (Hierarchical Temporal Memory), ContexOSE (Contextual anomaly detector Open Source Edition), and KNN_CAD (K-Nearest Neighbors Contextual Anomaly Detector) that will now be seen in further detail.

Mentioned NAB scores correspond to the Standard profile, where both false positives and false negatives contribute with 50% each towards the final classification.

3.5.1 Detectors

3.5.1.A Random/Null Detector

The null detector is the most straightforward of all chosen detectors. Yielding a value of 0.5 for every datapoint seen, it achieves a final score of 0 never detecting any point as anomalous. It should be used in order to compare how a classifier compares to a null hypothesis.

Equally important is the random detector which yields a uniformly distributed probabilistic score between 0 and 1 for each seen instance, as we may want to know how our model compares to a random one.

3.5.1.B HTM

Hierarchical Temporal Memory is a machine learning algorithm, which attempts to mimic the relational and inference mechanisms present in the neocortex of the brain. It contrasts with other machine learning techniques by not having a formal mathematical representation but rather attempting to simulate mechanisms in the brain, thus relying on ”regions” which are wired together. This process can be summarized in three steps namely learning, inference, and prediction. First, a model is trained via exposure to a stream of sensory data presented to HTM ”neurons” arranged in columns, layers, regions, and hierarchies which will learn sparse distributed representations upon which inference mechanisms will predict the outcome, or rather a set of outcomes, associated with a seen instance. Time relations are automatically encoded since the model relies on a hierarchy, which implicitly encodes time. Another benefit of this hierarchic representation is efficiency since patterns learned at any given level are then reused at higher steps of the hierarchy, thus shared representations lead to generalization of expected behavior. Furthermore, and as layers are laid out as 3d meshes of neurons, spatial dependencies are also captured when the sparse representations are created.

By matching learned sequences with the current input, regions predict several next inputs due to the sparse representations, keeping context of what might have been seen in previous steps and its influence in the current prediction.

To sum up, each observation is fed as input which is then represented sparsely, upon which, and with previous occurrences in mind, a value for the next observation is inferred. This inference can be used not only to predict the values themselves but also levels of confidence in said answers. Detailed information on the topic and process can be found in the Numenta HTM whitepaper [47].

3.5.1.C CAD OSE

The conformalized anomaly detector works by keeping contexts the algorithm predicts an input. If the prediction is correct no change is made in the ”contexts” however, if such is not true then a new context is generated in such a way that the outcome coincides with the instance. Base on the number of context changes this algorithm is able to issue a score of anomaly.

This algorithm was proposed independently for the NAB 2016 competition achieving thus the lack of information on such. For the avid reader, more information can be found by contacting the creator as only a ”blueprints” and a very crude version of the work is available.

3.5.1.D KNN_CAD

Comprised of a mix of distance, density, and probabilistic methods KNN conformal anomaly detection is a non-parametric method for one-dimensional time-series data processing with a probabilistic interpretation of the anomaly score [19].

Starting by creating a multi-dimensional matrix from an original univariate time-series using the ”Caterpillar” method [48], time-dependencies are encoded.

After this, the dataset is split into training and calibration for which respective nonconformity measure values are calculated.

Then using a training set (x_1,...,x_n) for a new observation x_{n+1} a probability that in the training set we can find an observation with a more extreme value of a non-conformity measure is calculated. This parameter can be interpreted as a probability of data normality.

Due to this, the main drawbacks of this model are high computational complexity, high sensitivity to the k parameter in some implementations, and high sensitivity to cluster densities in others.

3.5.2 Technique’s results over the datasets

NumentaHTM a hierarchical temporal memory model, *KNN_CAD* a nearest neighbor conformal anomaly detector, CAD OSE which is also a conformal anomaly detector, and a *null* detector which outputs a

continuous anomaly score of 0.5 were chosen.

Furthermore, execution times for 1 run of the classification step were taken in order to verify how long (on average) it took to classify each instance so that we can get a better perception of its utility in production environments, i.e. how well the classifier can handle the stream's input velocity. In addition, the standard profile was used were both false positive and false negative classifications are weighted equally.

Below follows a detailed description of the achieved scores and classification times.

3.5.2.A Taxi Domain

Detector	Total Time (ms)	# Records	Time per Record (ms)	Standard Score
numentaHTM	94931	10320	9,198	74,38
KNN_CAD	86367	10320	8,369	52,65
CAD_OSE	20748	10320	2,010	66,67
Random	1268	10320	0,123	6,31

4

Anomaly detection via discord management

Contents

The following chapter addresses the details of the adapting matrix profile method to data streams, more specifically, the means of keeping a model in memory, the management of reported discords from the profiling method, and the issuing of scores.

4.1 Proposal

In this work we propose to address the problem of anomaly detection over data streams through the use of the matrix profiles, and the all pair similarity search, introduced by Eammon Keogh [40]. In this manner, we aim for using matrix profiles for computing and storing the distances between the arriving data and the stored, while making use of a distance-based similarity measure to identify anomalies. As previously noted, we face several challenges, inherent to the streaming nature of our problem:

- Guarantee the time and space needed by our method is kept almost constant;
- Find a set of parameters as close as possible to the optimal solution of the problem;

What we propose to do is to make use of, and adapt the Eammon Keogh matrix profile and the all pair similarity search (APSS), in order to work with data streams and detect anomalies in an online manner.

One way to approach the first problem is by maintaining a compact enough representation of the last seen instances, as it will guarantee the constant requirement. Furthermore, by setting a maximum size for the number of discords/motifs to kept, we can further ensure the previous statement. q The main disadvantage of matrix profiles is being a visual method. Consequently, there is no automatic strategy for the identification of the the best parameters, namely the window size, number of top discords and size of exclusion zone. Therefore, we will try to address this issue by testing different dataset sizes (kept as memory of a near past) to infer where or not they impact the final results achieved.

These datasets are in fact queues where points are inserted by order, up to a limit, and dropped by appearance order in case the queue is full. On top of this, the matrix profiles will be calculated on every new received point. By extracting the longest mismatching sequences and maintaining them, we expect to be able to state something about the seen data point every cycle. Due to this continued inference of the maintained state, it now becomes an online method for data streams. The necessity of maintaining the entire data is overcome by keeping a resume of anomalous data as discords of the past instances, as well as the most recent portion of it.

However, other problems have to be addressed such as the size of the window to represent the discovered patterns, the length of each considered temporal context (for how long is data considered current data, and after how much time we refresh the motifs/discord storage, for example).

Despite any mechanism, or automation we might further explore, the operation cycle of the algorithm can be resumed in figure 4.1.

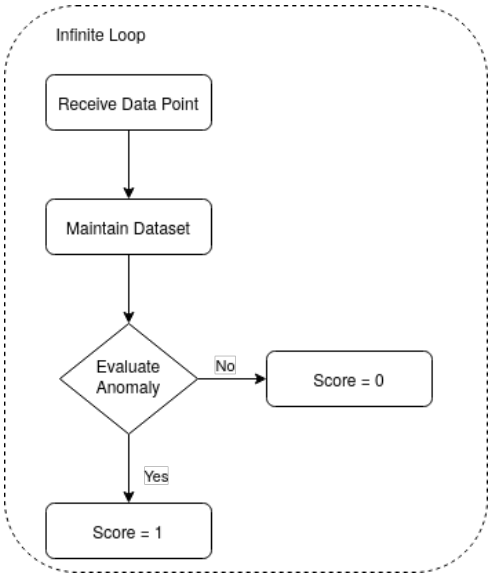


Figure 4.1: Proposed solution flowchart. For every time point received, maintain a data representation, evaluate whether there are new anomalies or not, and return the appropriate score.

4.2 Algorithm cycle overview

In order to tackle any numeric data stream, we propose to follow the previously introduced flowchart in figure 4.1, which will work in an infinite loop.

Prior to feeding any data, the algorithm must be initialized with a set of parameters, namely the dataset size to keep (*S*), window size to scan the dataset (*W*), number of discords to extract (*top_K_discords*), number of discords to keep (*n_discords_keep*), time to live of the discords (*TTL*), the threshold for the similarity function to trigger an insertion (*discords_sim_threshold*), decay rate of kept anomalous subsequences (*decay*), a number of points after which the issuing of scores begins (*start_evaluation*), and an exclusion zone of the retrieved anomalous subsequences (*ex_zone*). Only then our loop is ready to work.

After initialization, for each single data point received, ***P***, the learned model is kept via an handle function that receives each data tuple. Said tuples are comprised of a timestamp and its respective value. Multiple calls on the handle function cause the data points to build up an ordered set, which is the basis of our profiling analysis. The size ***S*** of this set can be seen as an event horizon, as exceeding this size will cause the loss of information (for that point in time) due to being dropped from the dataset.

On top of this dataset, we will use the matrix profiles with a sliding window of size ***W*** with which we scan the previously kept entities, and extract the corresponding matrix profiles. Consequently, this window corresponds to the granularity at which we are comparing patterns, more specifically their length. Upon extraction of the matrix profile from the dataset, we can compute the ***top K*** discording sequences. On the iteration previous to the beginning of the evaluation we will also set (if specified) the minimum

distance required for analysis, corresponding to the first reported discord distance profile.

Then, and with the extracted *top k discords*, we start keeping an anomaly frame of size *n_discords_keep*, which will accommodate the found anomalies.

The anomaly score issued by the proposed method is tightly connected to this frame since the anomaly score will be 1, thus indicating anomaly, whenever a new anomalous pattern is inserted or replaced in case the database is full. The insertion of these patterns is controlled by a threshold cut on the resulting value returned by a similarity function of two sequences, where the first sequence is a retrieved anomaly sequence from the *top k discords* method, and the other is a sequence already within the anomaly database. This choice is motivated by the fact that we want a low, but precise warning rate that only notifies the users for new abnormal behaviours.

As mentioned previously, this process will run indefinitely and can be summarized as shown ahead in algorithm 4.1.

Algorithm 4.1: Adapted Matrix Profile Overview

begin
$receiveDatapoint()$
$updateDataset(received\_point)$
$profile \longleftarrow MatrixProfile.scrimp(dataset)$
$score \longleftarrow manageDiscords(profile)$
return score

4.3 Dataset maintenance

The function that maintains the dataset bound to S elements, `updateDataset(data point)`, works by gatekeeping a data window of size **S**. Until this window is full points are trivially added. Upon being full the oldest point is deleted from the end of queue, followed by appending the new one at the beginning. The indexes are then reset in order to reflect the change in time, with all points moving down one position in the index of the queue. Therefore, we can state that this dataset is in fact a buffer similar to a first in last out (FILO) queue based on it's behavior. Figure 4.2 resumes the maintenance cycle leading to evaluation.

Further ahead we will be analysing the impact of the size of this dataset on the results obtained by the method.

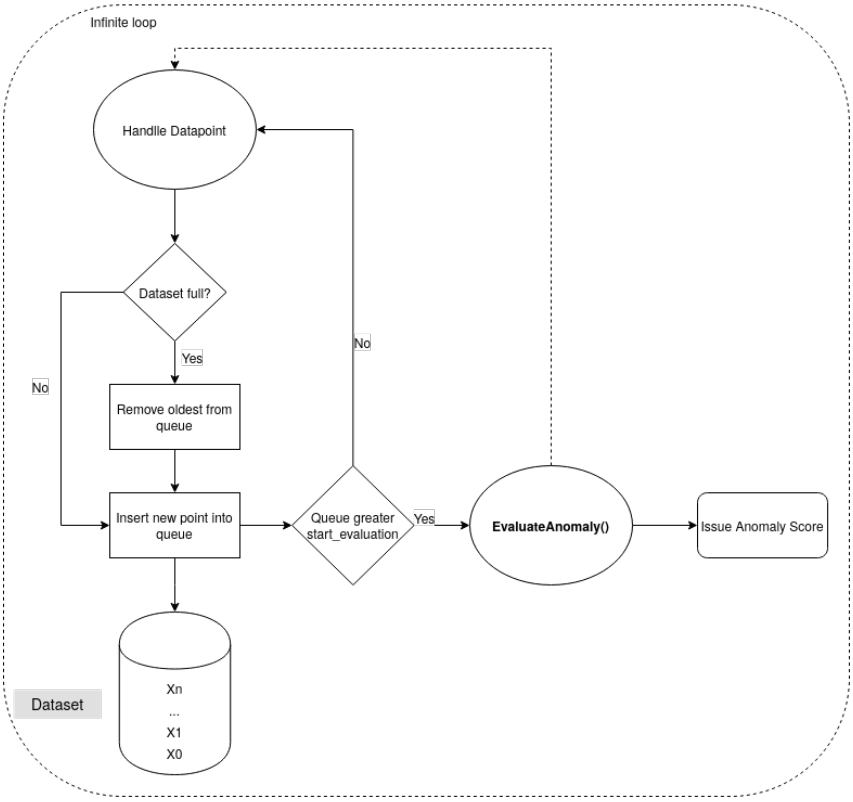


Figure 4.2: Overview of the dataset maintenance cycle. First, accumulate enough points to start an analysis determined by the `start evaluation` parameter. Only after this minimum number of points is met we can start evaluating anomalies.

The following pseudo-code in algorithm 4.2 summarizes the maintenance function of the dataset.

Algorithm 4.2: Dataset maintenance function

```
begin
    point P ⟵ receiveDatapoint()
    if len(dataset) < S then
        dataset ⟵ dataset[len(dataset) + 1] = P
    else
        dataset ⟵ shiftLeft(dataset)
        dataset ⟵ dataset[len(dataset)] = P
    dataset ⟵ reIndex(dataset)
```

4.4 Matrix Profile Computation

The Matrix Profile is calculated as usual. By providing a dataset of size **S**, and sliding a window of size **W**, we retrieve the corresponding data profiles, as introduced in chapter 2 and recalled in figure 4.3. The profiles correspond to the minima of the distances of the subsequence under analysis to all other subsequences. Then, with these profiles we will extract the highest profile distances, corresponding to anomalous sequence beginnings. With these partial results we can later reconstruct the sequences and check their degree of abnormality. However, this process will only happen for discords with a distance greater than the minimum.

In our case, if the dataset analysis for the last seen instance produced a very different anomaly list than the ones kept, then that point must be anomalous due to its impact on the analysis.

Furthermore, and due to the trivial match limiting zone introduced in Eammon's Matrix Profile implementation [40], we can only start evaluating once the dataset is, at least, twice as large as the window.

Working under this assumption, we will see how we can set a triggering anomaly mechanism around the similarities of the retrieved discording sequences.

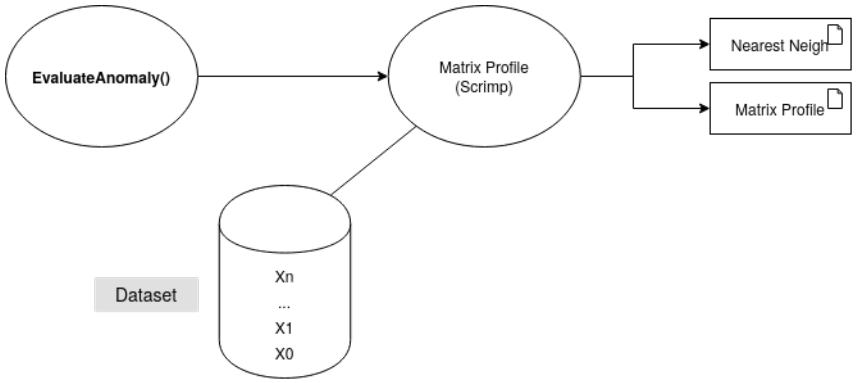


Figure 4.3: Anomaly Detection initial step. The method starts by computing the matrix profile for the current dataset kept, generating a set of starting subsequence profile, and respective nearest neighbours. These measures are recorded for every subsequence pair.

4.5 Discord Management

After computing the matrix profile for the data arriving, we can compute the **top_K_discords** by selecting the K largest distance profile values. If a point is contained in a subsequence that cannot have length *W*, namely sequences starting in indexes higher than the dataset size minus the window size (S-W), are said to be non valid, and consequently no retrieved.

Given that a value contained in a non-valid sequence will have a score of INF, we start by iterating the distance profile searching for the highest valued index that is non-INF. Then, we extract that entry into the *top_K_discords* list and set the value at the matrix profile to INF so that it does not get reported again. Moreover, we also set all consecutive values before and after it to INF, from the reported position forward and backward, up to a quarter of the window length *W*, corresponding to the exclusion zone. After doing this process k times we end up with k different anomalous points. If not enough points can be found (due to the length of the matrix profile being too short for a given k number) we simply report a sentinel value that is later filtered.

Another criteria used to filter out points is their distance profile. A minimum distance, corresponding to the maximum distance for the highest distance profile in the first evaluation iteration, is further used to filter out results. If any given subsequence start position distance profile is greater than the first one found, the resulting subsequence is then deemed as valid for comparison, otherwise it is discarded. A resume of the aforementioned mechanism can be found in figure 4.4.

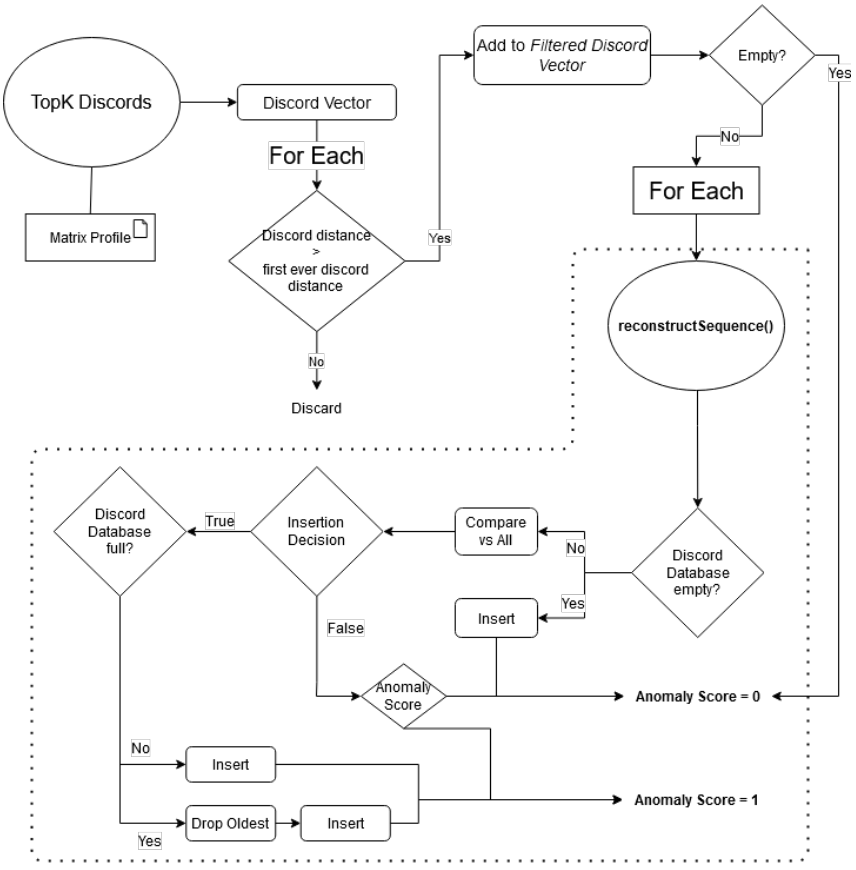


Figure 4.4: Discord Management cycle overview. The final step of this computation corresponds to issuing a normality score associated to the received time point. During the process, a set of discording subsequences are compared to determine their insertion. The insertion of the sequences automatically sets the anomaly score to one.

4.5.1 Managing Discording Sequences

With the previously found anomalous points we can reconstruct the valid sequences by going to the respective indexes in the dataset kept, retrieving their values and appending the W succeeding values.

The pseudo-code in algorithm 4.3 shows how full sequences can be easily retrieved after any anomaly index is provided.

Algorithm 4.3: Sequence reconstruction from index

```
1: SeqA ⟵ Array(W)
2: i ⟵ anomaly_index
3: for j = 0 ; j < W ; j++ do
       SeqA[j] = dataset[i + j]
4: return SeqA
```

The aforementioned sequence is then directly inserted in the discord database if the latter is empty. Otherwise, the sequence is compared for similarity with previously existing sequences present in the discord database, in order to assert whether the anomalous sequence is inserted or not, as resumed in algorithm 4.4.

Algorithm 4.4: Discord Database maintenance

```
begin
    anomaly_score = 0
    for each detected anomaly A1 do
        discordRefresh = False
        for each stored anomaly A2 do
            if cosine_sim(A1, A2) > discord_threshold then
                refresh(A2)
                discordRefresh = True
    if discordRefresh == False then
        insertDiscord(A1)
        anomaly = 1
```

Upon insertion, we register not only the found sequence but also the time of occurrence, a counter of occurrences starting at 1, and a **time to live** (TTL). The *TTL* of the anomaly works as a forgetting mechanism, as keeping the discord forever would cause it to never be detected again, even if it showed up in a different context. In each iteration of the proposed algorithm all discords *TTL* are decreased by a decay factor. If the *TTL* of any discording sequence recorded ever reaches zero the anomaly is removed from the anomaly database. In contrast, if the discord list ever reaches a size of **n_discords_keep**, the one with the lowest ttl is discarded and the new discord is inserted.

As a consequence at least 1 anomalous sequence will always be kept, with the trivial insertions not counting towards the anomaly score as the database is empty at that moment. Moreover, this mecha-

nism ensures that multiple consecutive sequences are only issued once, as they will cause multiple high similarity scores among themselves, thus refreshing the *TTL* and ensure they stay in the database.

4.5.1.A Sequence Similarity

One way to tackle the problem is via comparing the discording points complete sequence. For this task the cosine similarity [49–51] was chosen as a similarity metric. This similarity function choice was motivated by the bounded nature [0,1] of the method, that results in a simple and intuitive threshold selection parameter representing percentage of similarity. Other measures, mainly unbound ones, might suffer from the problem of the interpretation of the result. The euclidean distance was also considered as a similarity metric. However, it was not chosen as it is a distance metric rather than a similarity one, which would result in a more difficult interpretation and calibration of the resulting value, namely the similarity threshold as a distance. The cosine similarity is generally used as a metric for measuring distance when the magnitude of the vectors does not matter, something useful when abstracting the domain under analysis.

Nonetheless, it is important to stress that many other distance/similarity measures could have been chosen such as the Kullback-Leibler, Chebyshev distance or even Dynamic Time Warping (DTW) [52].

After finding their similarity (via the cosine similarity), if it is higher than a given threshold, **discord_sim_threshold**, it will cause a refresh on the anomaly database. If no refresh is detected the anomaly is due to be inserted, triggering a maximum anomaly score.

The similarity of two sequences is found via the following formula:

$$\cos(\mathbf{S1}, \mathbf{S2}) = \frac{\mathbf{S1S2}}{\|\mathbf{S1}\|\|\mathbf{S2}\|} = \frac{\sum_{i=1}^{n} \mathbf{S1}_i \mathbf{S2}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{S1}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{S2}_i)^2}} \tag{4.1}$$

If the nature of the input were to be symbolic, different measures such as the Jaro [53,54] or Jaccard [55] similarities would be more appropriate.

4.5.1.B Scoring Function

If the result of the comparison between a sequence and all the discords in the discord database never exceeds a similarity threshold then the sequence is considered to be a new anomaly and inserted. Upon insertion, the return value of the anomaly score for that time step is set to one and later returned, with multiple insertions having the same impact in the score as a single insertion.

Otherwise, if any sequence comparison is equal or higher than the **anomaly_threshold** it will cause the anomaly being compared to, in the discord database, to refresh it's TTL. In such case, the anomaly score will remain the same as it was before that given iteration. The anomaly score is then returned after all sequence comparisons are made.

After completion of the discord database maintenance task an anomaly result for the current data point will have been found.

4.5.2 Probabilistic Profile Distance Management

Another way to tackle the aforementioned problem is by using the raw distance value. The most straightforward test we can run on the raw distance profile is a mean confidence interval [56]. The objective here is to check, whether a value is within a given confidence interval of a known distribution. As detailed in figure 4.5, we see yet again a resume of the operation cycle, except know with a probabilistic approach for the discord management. For this method, once we filter out all the resulting discording distances, we can start accumulating them. Once we have a universe of distances greater than 3 samples, we can start performing a mean confidence interval test. Then, we iterate through all the kepts discording, and compute the mean, lower bound, and upper bound values.

4.5.2.A Scoring Function

Similarly to the sequence scoring function, we will also need to score our results. One way to approach this is via the mean confidence interval over the distance profiles, so that we can use the already computed distances (mean distance, lower bound and upper bound).

If our distance value is within the mean confidence interval at a given percentage, then the resulting anomaly score is zero. Otherwise, the anomaly score for that timepoint is the greatest value of the calculation of the distance profile value minus the mean of all seen distances.

The resulting score is normalized by the max seen distance within the kept distance vector. Finally, all distance points that passed filtering are simply added to the seen distance vector, and the anomaly score returned. In figure 4.5 we can find the scoring function detailed, within the dashed line.

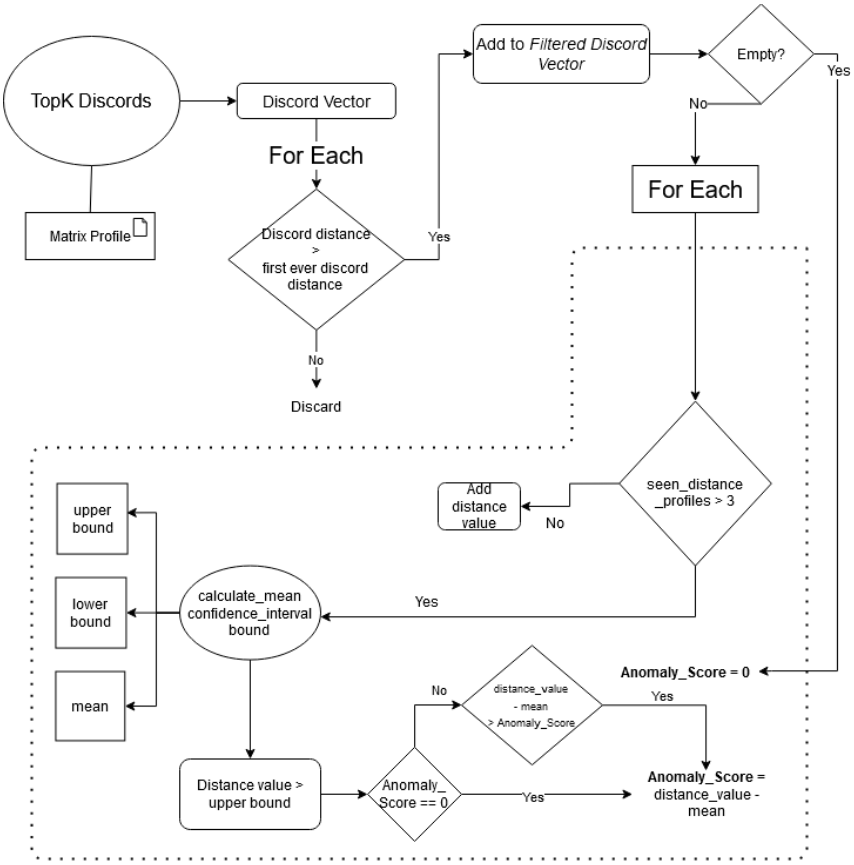


Figure 4.5: Discord Management cycle overview. The final step of this computation corresponds to issuing a normality score associated to the received time point. During the process, a set of discording distance values are compared to determine whether or not they are within a mean confidence interval.

4.6 Toy Problem via Sequence Comparison (Taxi domain)

In order to illustrate the functioning of our algorithm we start off by creating a 961 point dataset for analysis. In figure 4.6 we can see the dataset plotted, where yellow diamonds correspond to an anomaly. This dataset is extracted from the taxi domain in section 3.4.1, between the 2014-10-20 00:00:00 and the 2014-11-08 00:00:00, corresponding to approximately 19 days, containing half hour aggregations for the NYC yellow cab passenger count.

Figure 4.6: Toy problem. Two week period between the 2014-10-20 00:00:00 and 2014-11-08 00:00:00. Anomaly annotated as yellow diamond

Then, we start by setting up our algorithm with the parameters dataset size (S=200), window size (W=20), similarity threshold ($\theta = 0.97$), time to live (TTL=48), n_discords_extract=3. n_discords_keep=3, using a minimum distance, and an exclusion zone of W/4 as shown in table 4.1.

Table 4.1: Toy problem algorithm setup.

Dataset Size	200
Window Size	20
Similarity Threshold	0.97
TTL	48
Decay	1
Discords to Extract	3
Discords to Keep	3
Exclusion zone	Window Size / 4
Start Evaluation	199
Use minimum distance	1
Probabilistic version	0

Only after gathering 200 points we can start the evaluation process. We then calculate the matrix profile for the current values. The three, top k, resulting sequence start position are [15, 110, 161].

The sequence starting at 15 is trivially inserted, with the sequences starting at 110 and 161 being dropped for being too similar, despite having the minimum required distance profile value needed. Let us call this inserted sequence *seq1*.

Further on, at timestamps 2014-11-02 19:00:00 and 2014-11-03 02:30:00 two futher sequences, namely *seq2* and *seq3*, with a distance greater than the minimum distance required, and a similarity lower than `disc_sim_threshold` get inserted. These two anomalies cause the database to be full, and correspond to the two first red diamonds in figure 4.10, corresponding to an anomaly score of one. The contents of the discord database, at this time, can be found in figure 4.7.

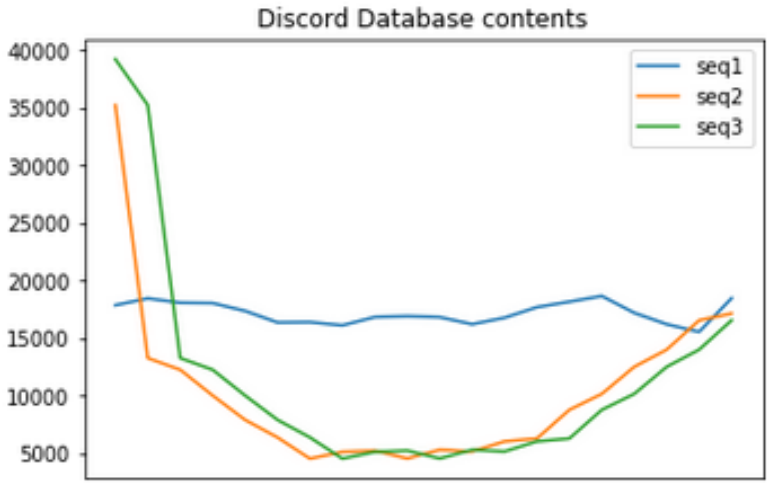


Figure 4.7: Discord database contents . Sequences of 20 points represented in the X axis. Y axis corresponds to the passenger count.

As time passes, sequences *seq2* and *seq3* do not get enough refreshes (similar discording sequences showing up) and get eventually dropped out of the database as their TTL expires.

Later two further sequences (*seq4* and *seq5*) get inserted, at timesteps 2014-11-05 21:00:00 and 2014-11-05 21:30:00 respectively, causing the discord database to be full again. The discord database at this moment can be resumed by figure 4.8.

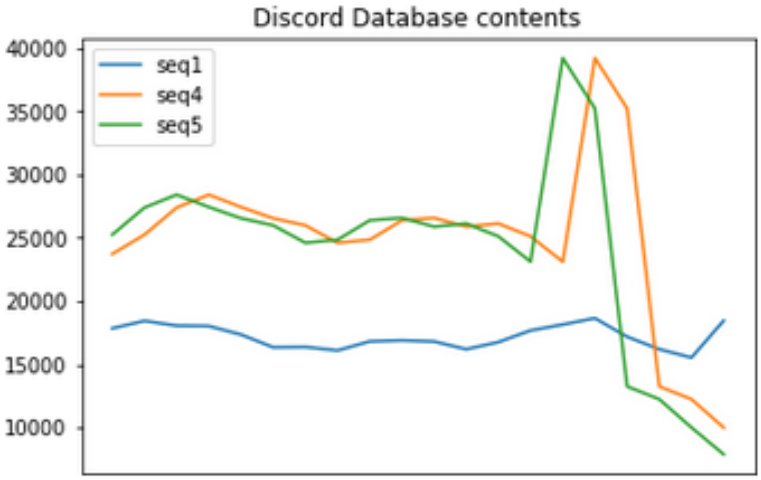


Figure 4.8: Discord database content at 2014-11-05 21:30:00 corresponding to the third red diamond (anomaly).

At timestep 2014-11-05 22:00:00 another discord gets selected to be inserted, however the database is full. Consequently, *seq1* gets dropped for being the oldest seen, with the new anomaly (*seq6*) being inserted. The resulting database at this point can be found in figure 4.9.

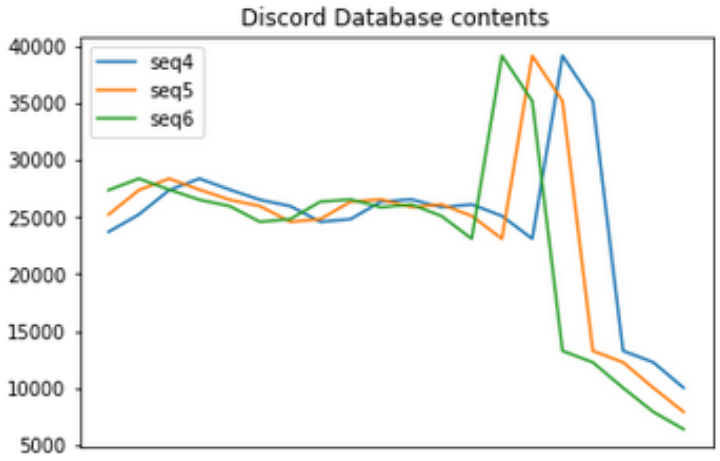


Figure 4.9: Discord database contents after replacement (fifth red diamond).

The classification process continues until all points are fed. The final result for the complete classification, plotted against the ground truth, is shown in figure 4.10.

Figure 4.10: Toy problem classification, complete results. Yellow diamonds represent ground truth whereas red diamonds represent points deemed anomalous.

4.7 Toy Problem via statistical approach (Taxi domain)

For the demonstration of the statistical method we will be using the same dataset as introduced in the previous section, and show in figure 4.6. Using the insertion mechanism and scoring function detailed in subsection 4.5.2, we set up our algorithm with the parameters *probabilistic_version = 1*, *window_size = 48*, *dataset_size_to_keep = 110*, *ttl = 48*, *decay_rate = 1*, *start_evaluation = dataset_size_to_keep-1*, *ex_zone = int(window_size/4)*, *fourier_on = 0* , *mean_confidence_interval = 0.95*, and *min_dist_on = 1*. The experimental setup can be found in table 4.2

Table 4.2: Toy problem algorithm setup.

Dataset Size	110
Window Size	48
Similarity Threshold	0.95
TTL	48
Decay	1
Discords to Extract	10
Discords to Keep	3
Exclusion zone	Window Size / 4
Start Evaluation	Dataset Size - 1
Mean Confidence Interval	0.95
Use minimum distance	1
Probabilistic version	0

Similarly to what we have done before, we start by accumulating data. At timepoint 2014-10-22 06:30:00 we start out evaluation. Given the the distance vector that will hold the distances of the filtered anomalies is empty, once we have 3 points with a distance greater than the minimum distance we can start using our mean confidence interval. These three point may be added extremely fast (if in a step

3 distance profiles with a value greater than the minimum distance show up), or take a while. For our example, these three points (corresponding to three discording distances) are met within two time steps.

However, only at timestamp 2014-10-25 19:30:00 we get our first discording value outside the mean confidence interval we set. At that time, the distances within the used distance vector to calculate the mean confidence interval is a constant vector of size six, represented by the following vector: [10.716486315396866, 10.716486315396866, 10.716486315396866, 10.716486315396866, 10.716486315396866, 10.716486315396866].

The mean, lower bound, and upper bound values for an interval of confidence of 0.95 are all the same, namely 10.71. Then a distance profile of 10.78 shows up, thus being outside the confidence interval. At this moment, the resulting score for this timestamp will be the distance profile minus the upper bound value of the confidence interval, consequently yielding a result of 10.78-10.71 = 0.07.

Finally we add the distance value of 10.78 to the seen distances, repeating the bound calculation and verification for all other filtered distances for that given data-point, and for all remaining data-points.

This operation process results in the classification procedure show in figure 4.11, where the score is scaled by the maximum value within the dataset for representation purposes. However, given that it is normalized by the maximum distance, it will always be in the [0,1] range.

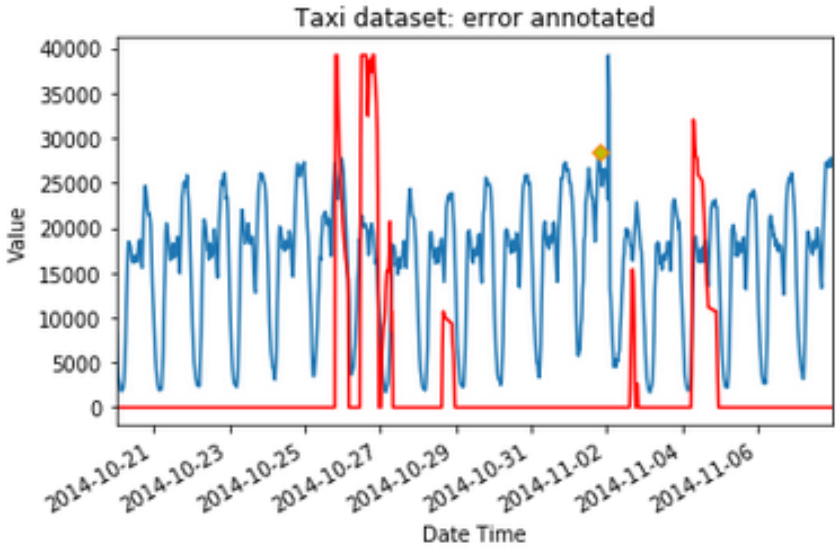


Figure 4.11: Statistical approach over the toy domain. Classification results displayed in red against the data in blue. Yellow diamond represents the annotated anomaly. Test performed for a mean confidence interval of 95% using the setup introduced in section 4.6

4.8 Differentiated Toy Problem via Sequence Comparison

If we grab the toy problem dataset, and for each time-point from the second onward we subtract the previous value, we end up with the differentiated dataset, more specifically the lagged dataset. The result of this operation is resumed in figure 4.13. This corresponds to a common operation used in forecasting, and time-series statistical methods, due to helping in reducing or removing trend and seasonality, thus stabilizing the mean. The anomaly maintains itself, represented by an great spike the value around the second of November.

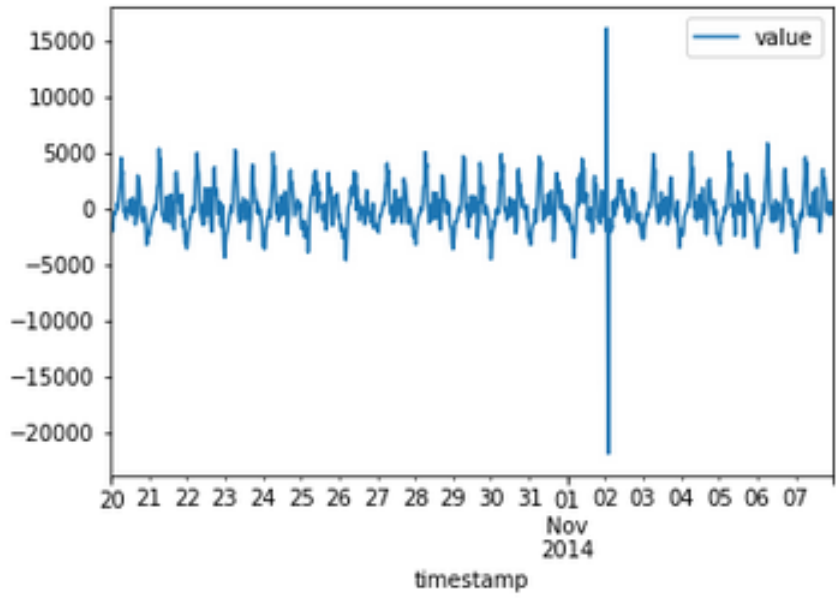


Figure 4.12: Toy problem differentiation process visualization. Time represented on the X axis, with the difference to the previous value represented in the Y.

Next, we setup the algorithm, as introduced in sections 4.5.1 and 4.6, with the same parameters as in the previous section, in table 4.2. The results for the classification procedure can be found in figure 4.13.

For this experiment, the results are similar to the original dataset, with the method experiencing a spike in anomalous detections right after the anomaly. However, we also now seem to have a spike in the score just after the 1 week mark. If wee look back at the same period in figure 4.13 we can visually see a slight disturbance in the wave. However, since this is not annotated it does count as a mistake. In addition, we now have a series (or rather zone) of anomalous classifications, whereas in the original domain we have just singular detections instead of bursts.

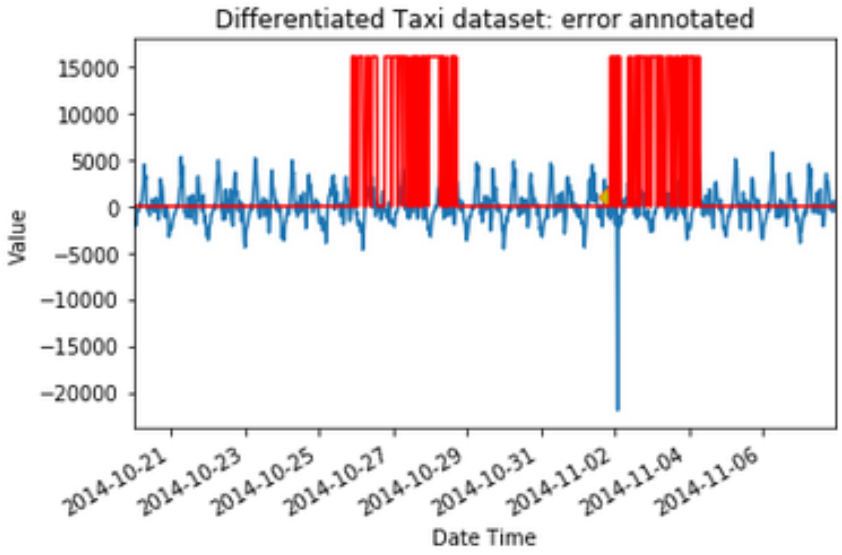


Figure 4.13: Toy problem differentiation process visualization. Time represented on the X axis, with the difference to the previous value represented in the Y.

4.9 Differentiated Toy Problem via statistical approach

Without changing the settings chosen in the previous section we will now test the probabilistic approach, by setting the probabilistic version to 1.

The result of applying the probabilistic solution over the toy problem can be seen in figure 4.14. The results are very similar with the proposed classifier, indicating anomalies within the same approximate regions.

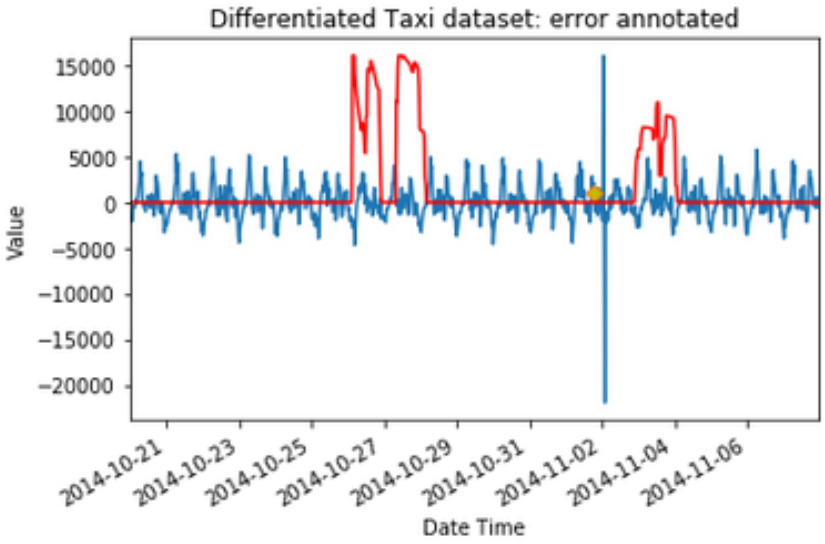


Figure 4.14: Statistical approach over the toy domain. Classification results displayed in red against the data. Yellow diamond represents the annotated anomaly.

4.10 Automatic parameter selection

One of the main problems identified in the literature is the selection of the window size, and how it affects the detection process [9, 57–60]. Furthermore, a large set of parameters must be manually adjusted, thus making parameter selection a challenging task. Consequently, we would like to reduce these via the Fourier Transform [61, 62], approached in section 3.4, in order to find the most important period within the kept data, which will be reflected as the size of our matrix profile window, and subsequence size. This will result in removing the necessity for choosing a window size, and a start evaluation parameter, maintaining only the need to test different thresholds and dataset sizes to keep.

Furthermore, and as a consequence of the application of the Fourier Transform, and once we find the window size, evaluation should start immediately, and the dataset size must stop increasing. If we allowed these parameters to change, the results of the Fourier Transform would no longer be valid as the data that generated it had changed by increasing.

The basis for the new method will be to automatically extract the most common sequence size within the kept data, such that it explains the most frequent working regime, as previously introduced in figure 3.7, in chapter 3.4.1.

First we start by accumulating a maximum number of points defined by the dataset size parameter (**S**). Once the desired number of points is reached we can compute the real Fast Fourier Transform. This computation yields n real and imaginary parts, one for each inserted data points. Then, by calculating the modulus of each coefficient, we are able to translate them into natural frequencies with comparable magnitudes. These frequencies are then further translated into time periods, as they cannot be directly

interpreted. This is done by feeding the results of the Fourier transform, as a modulus vector, and inserting them into a new appropriate scale. This new scale has the X axis in appropriate time units instead of the usual frequency units (rad/s). Once we have the fundamental time period of our analysis in data points, we can move on to computing the distance profiles and resume our evaluation procedure.

The main difference from the previous solution is that now we will only have to test different similarity thresholds for different dataset sizes instead of the previous window, dataset size, threshold selection.

4.10.1 Toy problem spectral analysis

If we further trim the toy problem to one week, we manage to avoid the anomaly, that could heavily influence the upcoming analysis. The resulting week can be visualized in figure 4.15. There we can see that a recurring event is present, with an aparent frequency of a day, or rather 24 hours.

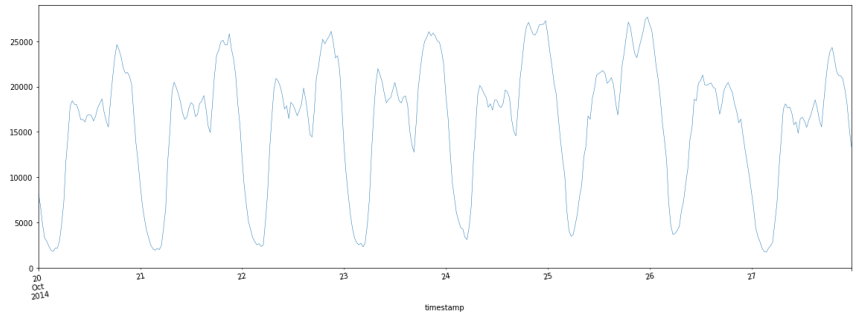


Figure 4.15: First week from toy problem.

If we now apply the fourier transform to this week, it is easy to see that, once again, we see a peak frequency corresponding to the 24 hours period. There also seems to be another frequency that repeats every 12 hours.

The optimal way to approach the detection problem in this situation would be to learn 2 classifiers over the two peaking frequencies, as image 4.16 suggest, and combine their results. This is due to the fact that the main frequency is also accompanied by another non-deniable important frequency (the second frequency has a very high magnitude compared to all others, and to the main resonating frequency).

Nonetheless, it is important to stress that instead of a 12 hours frequency we now see a 2 day frequency. This is due to the fact that when computing a spectrogram an iterative Fast Fourier Transform is calculated, whereas in figure 4.17 we do a one shot Fourier computation.

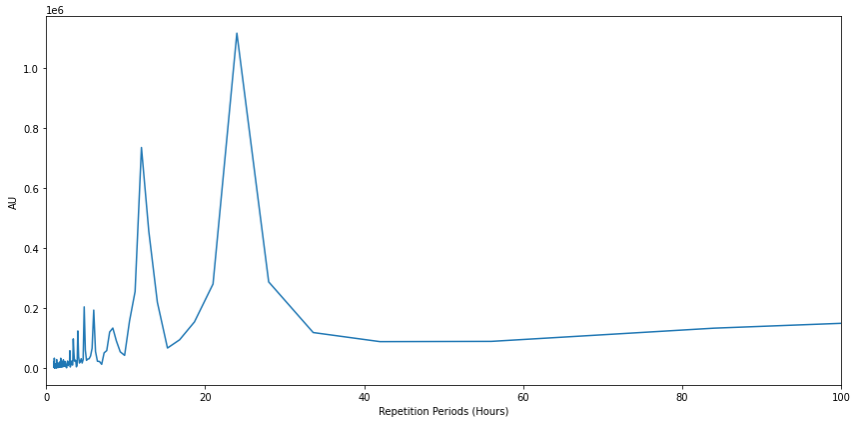


Figure 4.16: First week from toy problem Fourier Transform. Repetition period in hours represented in the X axis. Y axis represents the frequency magnitude, thus importance. Two main frequencies stand out for the weekly analysis, for the 12 and 24 hours respectively.

The spectrogram over the week dataset further confirms our assumption of the 24 hour peaking frequency (corresponding to 48 datapoints) as shown in figure 4.17. Furthermore, the second band in the spectrogram, at the two day mark, reinforces that the taxi domain follows a circadian cycle.

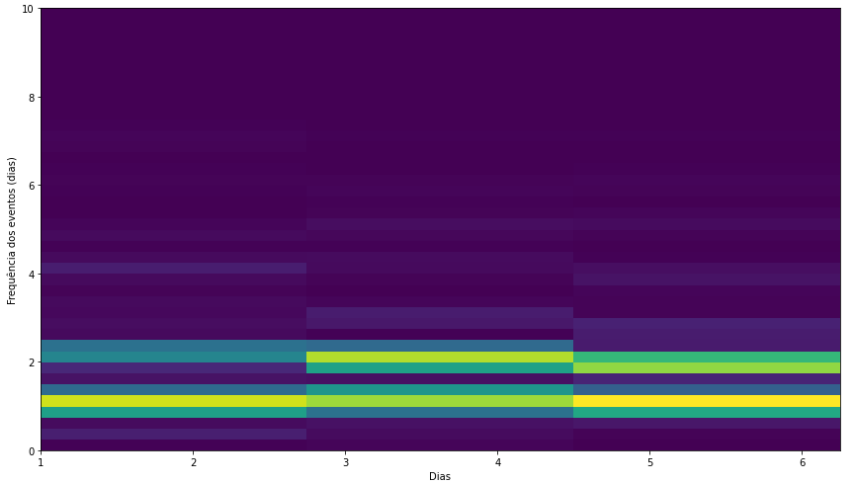


Figure 4.17: Spectrogram over the first week from the toy problem. Results further confirm the ones achieved via the Fourier Transform.

4.11 Methodology Appreciation

Overall, the designed method was able do detect not only point anomalies (as the one present), but also concept drift, and novelties over the temporal data. As shown in subsection 4.6 figure 4.6, via the discord management database we are able to model the usual behaviour of concept drift (represented by the 2 similar, yet drifted patterns). Furthermore, we can also see by the two completely different patterns in

the database that we can also accommodate novelty detection in our method.

Nonetheless, further analysis should be performed on these phenomena, with datasets that reflect concept drift and novelty in particular, and that are properly annotated for the purpose.

5

Experimental Results

Contents

The following chapter demonstrates the potential of the matrix profiles when applied to univariate time series anomaly detection. A total of 3 parameters are tested and the reasoning behind the value ranges is detailed. By simulating each set of parameters (over the dataset), we will be learning models that help characterise and explain the seen behaviour for a given domain.

At the end of this chapter, the final model, and prior experiments are used to derive conclusions on the ability of the designed mechanism to detect anomalies, resulting in a series of pros and cons for the method.

First, we start by defining a simple, and intuitive, baseline behaviour based on the statistical version of the algorithm introduced in section 4.5.2. Then, we will test the anomaly detection method based on sequence similarity, as detailed in section 4.5.1, and compare it against the probabilistic one. Finally, we will test whether the automatic window extraction mechanism is working or not, and if it is able to properly analyse the time series.

All these tests will be performed over the taxi domain, introduced in section 3.4.1. The dataset comprises 10320 data points with 30 minute spacing each. In addition, the dataset contains 5 anomalies (NYC Marathon. Thanksgiving, Christmas, New Years Eve, and a Snow Storm). The dominant label within the dataset is negative (0) in terms of abnormality, with the reported values not following any known distribution.

5.1 Parameter Variation

The goal of the experimental procedure is to achieve the best possible classifier with minimal overhead in variable adjustment.

Note that the method made available by the matrix profile foundation [63, 64] will end in error due to the subsequence length (definition 2) being too large if, at the moment of analysis, the dataset is lower than two times the window size.

A total of three parameters, namely dataset size (S), matrix profile window size (W), discord similarity threshold (`disc_sim_threshold`) are fully tested.

In addition, the parameter that controls the start of evaluation is permanently set to be the dataset_size length minus one, unless stated otherwise. The decay rate parameter will be fixed to one, with the time to live of the discord (in cycles) equal to the dataset size (TTL = dataset_size). In addition, the exclusion zone of the discord calculation is fixed to a quarter of the window size. This parameter will influence the number of records to exclude on the left, and right, of each retrieved discord (grey area in figure 3.1).

The choice for the window sizes is between 1 hour (namely two datapoints) and 48 hours (96 data points), that are tested for increasing dataset sizes between 4 times and 9 times the window. In table 5.1 we can see a resume of the intervals of variation, given that the remaining parameters are set up as

stated above.

Variable	Start Value	Value Step	End Value
dataset size (**S**)	4x window size	1x window size	9x window size
window size (**W**)	4 data points	2 data points	96 data points
`disc_sim_threshold`	0.95	0.01	0.99
use_minimum_distance	1	-	-
fourier_on	0	-	-
probabilistic version	0	-	-

Table 5.1: Parameter range variations

After varying each parameter, we will select the best value found for that run, and then proceed to vary another. Starting with the dataset size per window size, we will move on to the discord similarity threshold.

Another set of experiments will be done over the differentiated version of the dataset (where each point corresponds to the difference between itself and the previous observation), following the same strategy. This set of experiments will hereon be called as the sloped or differentiated version of the dataset, as it corresponds to the momentums of the original taxi domain trivially represented as slopes.

5.2 Statistical approaches over the taxi dataset

For comparison, a set of statistical classifiers, corresponding to an interval of confidence test, is also tested. The workflow for this evaluation mode is as previously introduced in section 4.5.2, and can be resumed as follows:

Working with similar parameters and behaviours as the developed solution introduced in section 4.4, the model starts by accumulating data and running the matrix profiles while filtering points based on the minimum distance. If these points meet the minimum distance requirement, a statistical test corresponding to the mean interval of confidence for the 99th percentile (over the accumulated anomalous distances that met the distance requirement) is run. If the distance is outside the interval, the value for the anomaly score for that point corresponds to (the largest) normalized difference between the discording distance profile value to the mean of the kept distance distribution (all anomalous distances and slopes respectively).

With this in mind, we will be running this detection mode for both the original taxi domain, and the differentiated version. The parameters for this experiment are resumed in table 5.2.

Variable	Start Value	Value Step	End Value
dataset size (**S**)	4x window size	1x window size	9x window size
window size (**W**)	4 data points	2 data points	96 data points
`mean_confidence_interval`	0.99	-	-
use_minimum_distance	1	-	-
fourier_on	0	-	-
probabilistic version	1	-	-

Table 5.2: Parameter range variations

5.2.1 T student confidence interval of 99%, over the raw distance profile (original dataset)

The first thing that stands out in the analysis results for this experiment is the extremely oscillating specificity. Without a mechanism to control the maintenance of sequences, it is clear that using solely the distance profile of the discords probabilistically is not enough establish a proper baseline behaviour. The NAB score, on the other hand, shows that despite the low precision, the learned models managed to detect some anomalous data-points.

However, taking into account that the precision values are also associated with the very low specificity values, we have indication that the detections were merely coincidental (as a collateral result of the low specificity). In addition, the detection labels are binary, but the detection score is not, this creates a problem for basic metrics. For these experiments, corresponding to a statistical classifier over the original dataset, the results can be seen in figure 5.1.

Given the aforementioned aspects, we should always look at the NAB score for indication of performance, and only then analyse the remaining metrics. Traditional metrics will fail here given that the anomaly score is given by the normalized distance to the mean of previously seen distance values, and this [0-1] value cannot be translated as precision/specificity or sensitivity straight forward. The Numenta benchmark takes care of this via an optimizer, that reflects the score for the best threshold found (given all detection values).

The NAB score, however, managed to be higher than zero due to said threshold optimizer run by Numenta that optimizes the resulting (0-1 interval) scores to find the best NAB score. This is a further step present in the NAB score, and thoroughly explained in the benchmark white-paper [21]. Further details on the scoring mechanism can be recalled in section 3.3.

In addition, the size of the dataset does not seem to influence the learning process. If we take a closer look figures 5.1(a), 5.1(b) and 5.1(c), and we compare them against 5.1(d),5.1(e),5.1(f) we may even be lead to think that the higher the dataset size, the worse results it achieved.

Therefore, we can state that running a simple statistical classification mechanism over the discording distance profiles is not enough. Despite this, we will run the same test over the differentiated taxi dataset, a common operation when working with time series.

Accordingly, the best classifier identified corresponds to figure 5.1(a), when the window size is equal to 8.

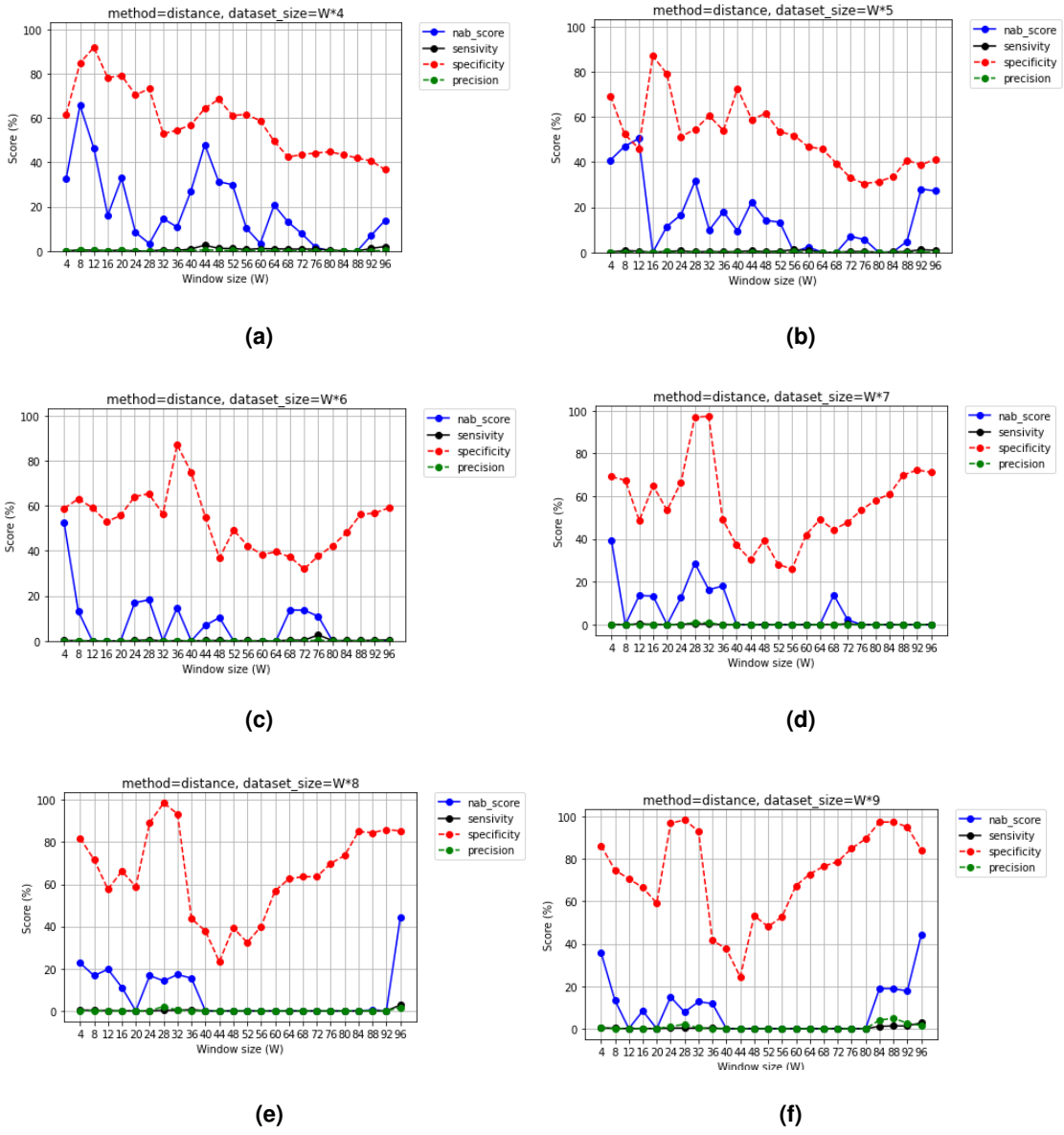


Figure 5.1: Results for dataset size variation by window size. Window sizes are represented in the X axis, nab score, sensitivity, specificity, and precision in the Y. Each vertical line is a simulation with its respective results. Classifier issued scores correspond to the normalized distance to the expected value if the distance under observation is outside the 99% confidence interval. (a) Results for multiple runs with varying window sizes, for a dataset size of 6 times window size, at a fixed confidence interval of 99%. Best classifier is in this batch, when window size = 8 and dataset size = 4 x window size; Final best NAB score achieved of 65.66

5.2.2 T student confidence interval of 99%, over raw distance profiles (differentiated dataset)

For this set of experiments we run the same statistical method, without changing any parameters, except now over the differentiated dataset. The confidence interval for these experiments remained the same, with the only difference being the underlying dataset.

The first thing that stands out, is that for the higher half of the window sizes tested the specificity remained high. This is a clue that for window sizes greater than the main underlying working regime, we can properly establish a base line behaviour. We can further state that, once the fundamental frequency is contained within the window, the matrix profiles method benefits greatly. This positive change is reflected in a steep (average) increase in some indicators, and decrease in others, namely specificity and best NAB score.

In figure 5.2 we can see clearly that, when applying the same statistical method over the sloped dataset we manage to maintain a low alert rate, reflected as a high specificity, contrary to the experiments in the previous section. In addition, the NAB score indicates a minority of anomalies detected throughout all runs, despite the varying dataset size. These indicators together show that we managed to achieve a relatively accurate detection process, despite not finding most anomalies.

We can further state that the lack of a baseline behaviour, shown by this algorithm, seems to be overcome through the differentiation process, when compared to the previous. This is reflected by a very low precision and sensitivity, followed by a high specificity and average NAB score. It is also of equal importance to stress that for window sizes greater than the underlying 24 hour fundamental regime (48 points) we can properly define the normal behaviour, shown as an increase in both NAB score and specificity.

Furthermore, associated with the increase in the dataset size, we also see an improvement (increase) in the specificity rate for the first half of the window sizes, namely sizes lower than 48 points. Consequently, this indicates that the larger the dataset, the more discrepant the matrix profile anomaly distance values will be for windows smaller than the fundamental frequencies.

However, the increase in the dataset does not have any major impact in the remaining window sizes. Nonetheless, it is important to note that the larger the window, the more unreliable the statistical method became in detecting anomalies. This is an indicator that the solution for the problem is not trivially achieved by simply adding points to the universe of samples under consideration.

Furthermore, we can see the NAB score peaking, around the 44 point mark for the window size in all runs. This is congruent with the spectral analysis of the dataset that revealed an underlying fundamental frequency for the 24 hour period (corresponding exactly to the 48 points of the window size).

Finally, we can state that by simply differentiating the dataset, an increase in quality of the simple statistical method is achieved. With this in mind, we will continue testing the differentiated dataset for

the upcoming versions of the classifier.

For this section, the best identified method corresponds to figure 5.2(a) at a window size of 44. The best NAB score achieved by this classifier was of 46.97.

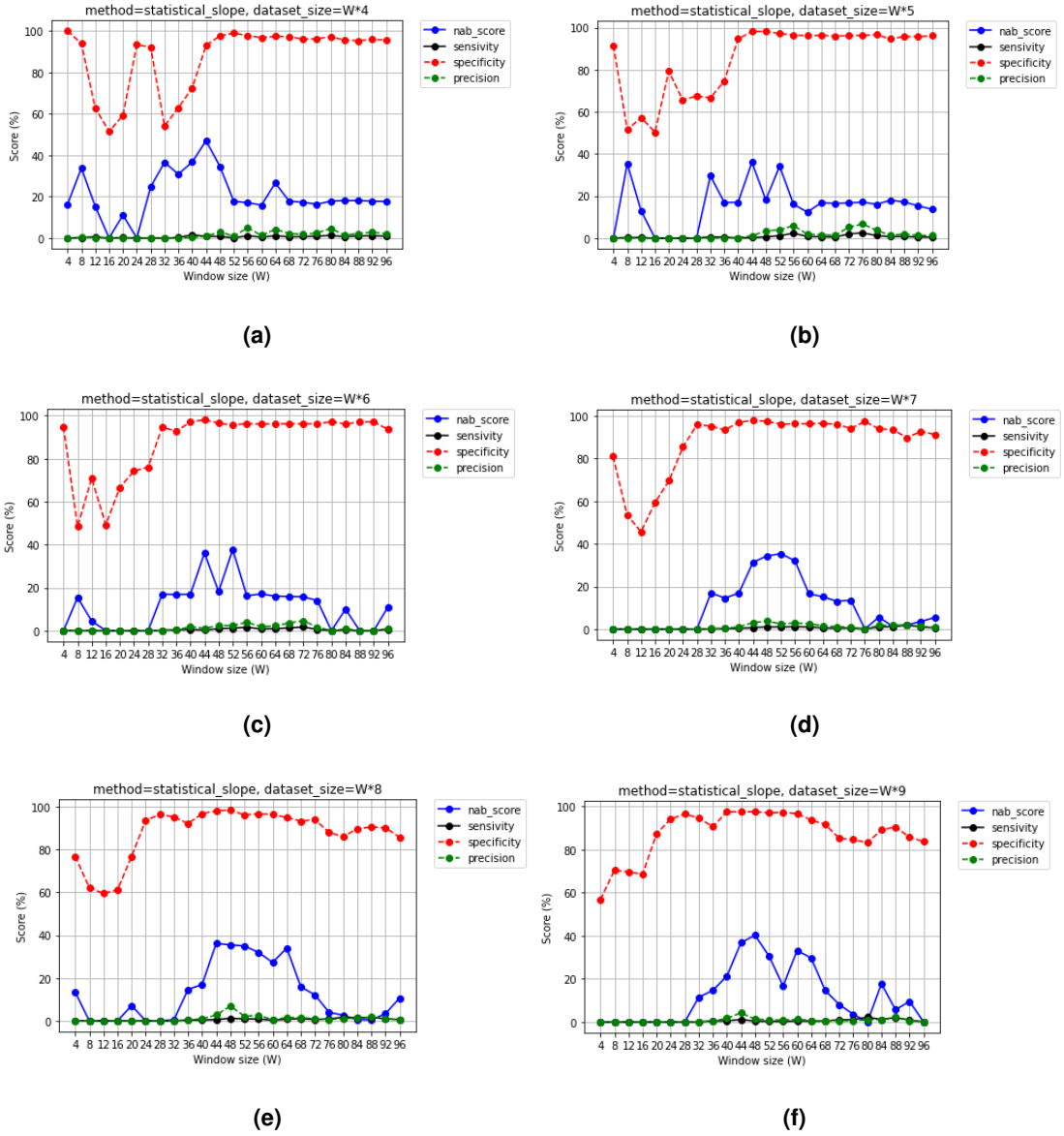


Figure 5.2: Results for dataset size variation by window size. Window sizes represented in the X axis, nab score, sensitivity, specificity, and precision in the Y. Each vertical line is a simulation with its respective results. Scores issued correspond to the normalized distance to the expected value if the distance under observation is outside the 99% interval. (a) The best classifier corresponds to a window size of 44 that got the maximal NAB score found of 46.97;

5.2.3 Distance Based Discord Management over the original dataset

The following subsections concern with the non probabilistic version of distance based method. The algorithm tested in the following subsections is as introduced in section 4.5.1. For the first set of experiments the unmodified taxi domain will be used, whereas for the second part of the experiment the differentiated dataset is used. All paremeter variations remain the same, except now istead of using a mean confidence interval of 99% to test points, we will use a minimum similarity threshold of 99% between sequences.

For the distance based discord management algorithm over the original dataset something recurring in all simulations is that the specificity parameter remains relatively high (over 95%). Despite the window size variation, no major changes can be found within each run.

On the other hand, the sensitivity parameter remained relatively low ($<$10%), despite being higher than all experiments so far. This is due to the fact that when we define the temporal period for which an anomaly detection interval is valid, we inherently define a zone where the true label is extended through time, with this being true for all experiments.

In other words, only the earliest valid detection would be enough to get a high score however, all points around the anomaly would also have be to equally classified in order to achieve a high sensitivity. Consequently, the analysis of this parameter by itself might not be sufficient to explain the quality of the achieved model.

Nonetheless, it provides valuable insight on the behaviour of the model when used with other indicators. All sub-figures in figure 5.3 show this behaviour clearly, where once the sensitivity increases, the NAB score decreases and vice versa. This trade-off is due to the increase in wrong classification, thus it suffers from the same problem as the specificity in the previous section. In order to achieve both high sensitivity and specificity, we would need to make continuous correct predictions for the whole interval around the anomaly.

Consequently, if we compare the sensitivity measure with the precision (true positive rate) we clearly see that an increase in sensitivity came at the cost of more errors, reflecting in a lower precision.

On the other hand, we can also see that high precision models tend to have high NAB scores such as, the beginning, and the end of the window size range (low and high respectively) in figures 5.3(d),5.3(e), and 5.3(f).

Furthermore, we can see that an increase in the dataset size resulted in an increase in the average NAB score for the run. It is however important to stress that some runs may not have a precision score at all (represented by the lack of connection between the two neighbouring points). In these cases no point tested as anomalous, and the run stands out by the lack of a connected line for this parameter. e.g figure 5.3(f), window size = 28.

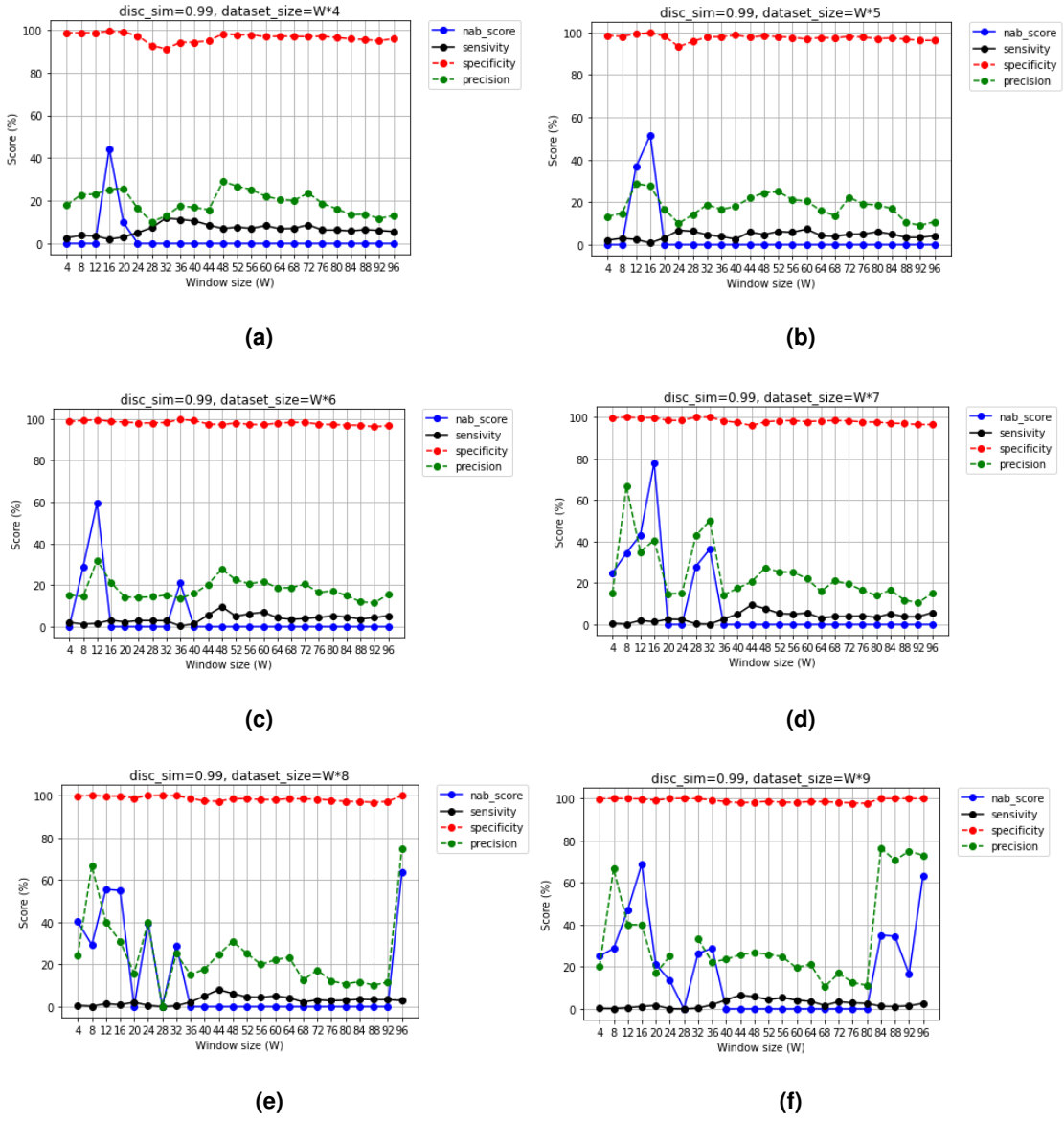


Figure 5.3: Results for window size variation by dataset size, for a given similarity threshold. Window sizes represented in the X axis with nab score, sensitivity, specificity, and precision in the Y axis. Each vertical line is a simulation with its respective results. In sub-figure (a) we can see the results for multiple runs with varying window size, and fixed similarity threshold of 0.99;

If we take a close look, we will find that figure 5.3(f) contains the best solution found in this run due to the high precision rate associated with high NAB score. Thus, in the next set of experiments we will set the dataset size to 9 times, and vary the discord similarity threshold parameter, with results shown in figure 5.4

Likewise, we also see the same lack of precision for some experiments, always associated with a NAB score and sensitivity of zero. As a consequence, the specificity parameter for these points is 100%.

If we further tested the impact of the similarity threshold, represented in figure 5.4, we can see that for thresholds between 0.95 up to 0.98, some of the models with lower window sizes managed to achieve perfect precision scores, implying that the detections made were correct. Yet, the NAB score for these models was relatively low. This is due to the fact that for lower thresholds we are too lax, only allowing big changes in the sequences to be detect. As a result, some abnormal sequences manage to fly under the radar, going by undetected.

With this in mind, we expected the more restricted models to achieve better results. However, a slight relaxation in the similarity of the sequences proved to increase the precision vastly and in some cases, and also the nab score in others. Figures 5.4 5.4(a), 5.4(b), 5.4(c), 5.4(d), and 5.4(e) show this clearly.

These experiments were run for for all similarity thresholds, window, and dataset size combinations, despite not being shown for the sake of simplicity while having similar repeated behaviour.

We conclude that, when compared to the simple statistical method, we have achieved a much better detection with less false positives, less false negatives and more true positives. Furthermore, the best found setup corresponds to the window size of 96 in figure 5.4(b). The NAB was score was respectively 69.99.

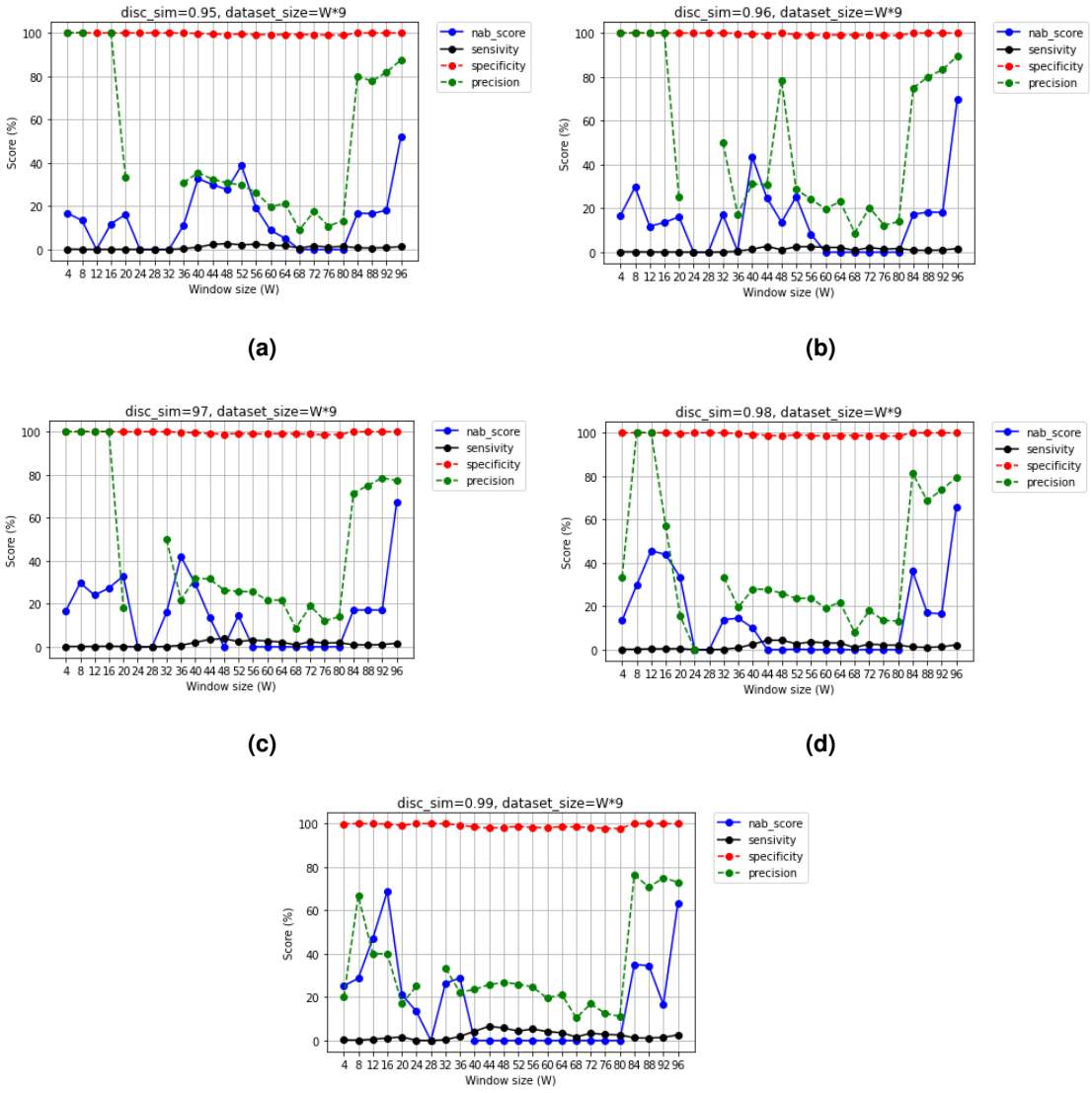


Figure 5.4: Results for window size variation by discord similarity threshold, for a fixed dataset size of 9x the window size. Window sizes represented in the X axis, nab score, sensitivity, specificity, and precision represented in the Y axis. Each vertical line is a simulation with its respective results. In sub-figure (b) are the results for multiple runs with varying window sizes, for a dataset size of 9 times window size, at a fixed similarity threshold of 0.96. In this batch we also locate the best result for our experiment at 69.99 NAB score with 17 true positives and 2 false positives, corresponding to the window size of 96.;

5.2.4 Distance Based Discord Management over the slopped dataset

Similarly to the experiments in the previous section we will repeat the same evaluation process except now, we will be using the differentiated dataset. The results for this new dataset, corresponding to the sloped taxi time series, can be found in figure 5.5.

If we take a closer look, we can see that, as expected, the normal behaviour is broadly captured. In all experiments 5.5(a) to 5.5(f), the specificity remained relatively high, indicating the majority of points were indeed true negatives. On the other hand, and contrary to the previous results on figure 5.4, we see a lower specificity rate. Despite this, the NAB score and precision turned out to be relatively high in comparison.

We can also see that throughout all figures, when the window size is lower than 48 points, the sensitivity is higher than any other previous experiment. This tells us that any time an anomalous point is identified, most surrounding points were also found to be anomalous. However, this came at the cost of an unstable baseline behaviour that, once again, when we reach the 48 mark, stopped happening, which further confirms that the spectral analysis results turn out to be very good guess for the window size. Once we reach sequences higher than the ones found by the Fourier Transform, we see an abrupt decrease in sensitivity accompanied by a high rise in precision. In addition, the instability in the specificity associated with defining the baseline behaviour stops after said period.

It is curious to find that the method over the differentiated series spikes (similarly to the previous subsection) in all measures around the 48 point mark, corresponding to the 24 hour period in the dataset analysis in chapter 3, more precisely figure 3.7.

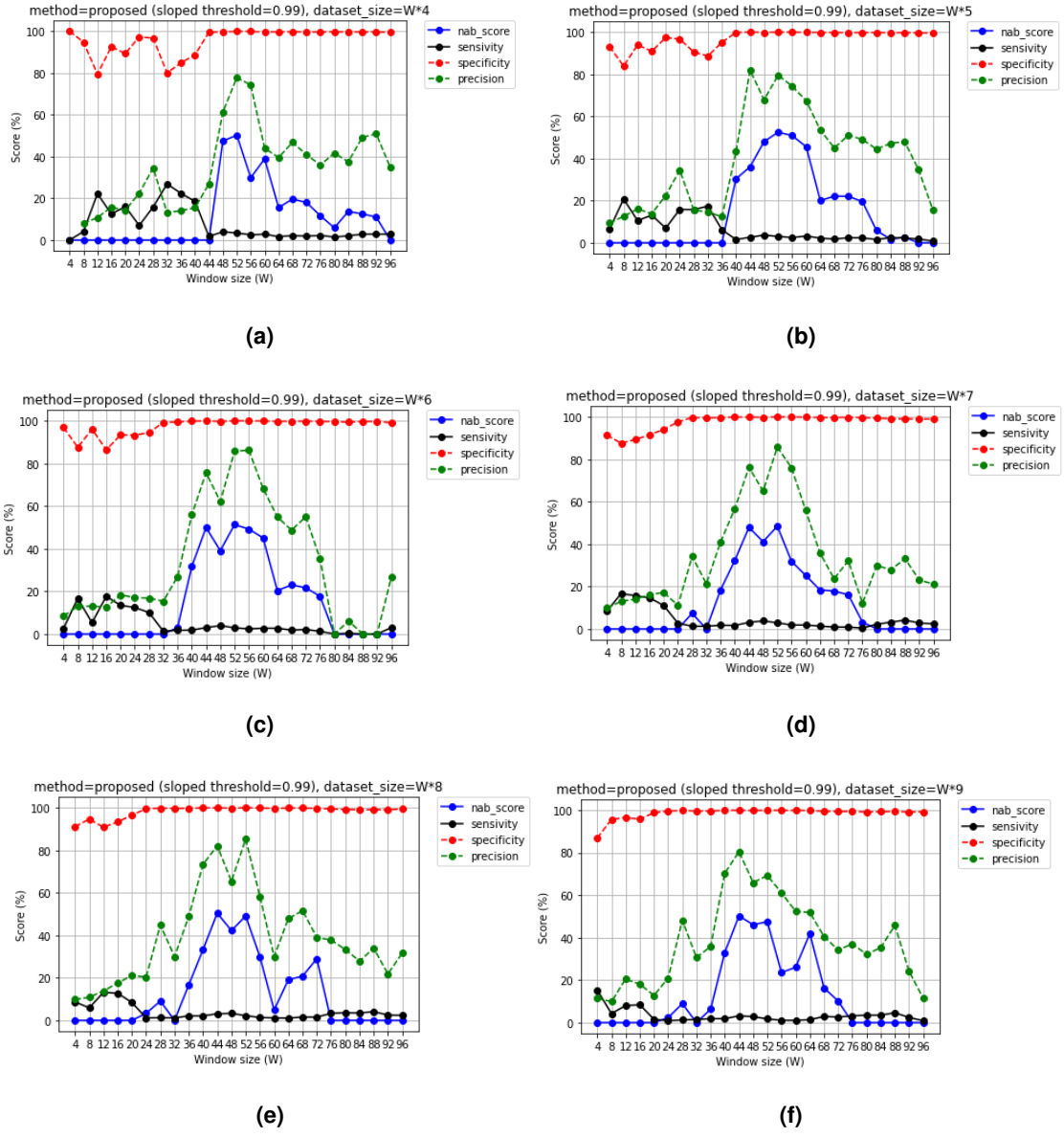


Figure 5.5: Results for window size variation by dataset size, for a given similarity threshold. Window sizes represented in the X axis with nab score, sensitivity, specificity, and precision in the Y axis. Results shown correspond to the sloped (differentiated) version of the taxi domain.

Finally, we can see in figure 5.6 that the variation of the threshold parameter (similar to the previous subsection) did not produce any noticeable change in the algorithm results over the differentiated dataset. Thus, we can say that the differentiated series is far more resilient to variations in the similarity parameter than the original dataset. The choice for this new set of experiments in figure 5.6 was done as in the previous subsection, by identifying the best classifier in the first set, and setting the window size constant while varying the discord similarity threshold.

In sum, the best classifier found for this set of experiments corresponds to figure 5.5(e) where *window size = 52*, *dataset size = window size x 6*, and *discord similarity = 0.99*. The final NAB score for this classifier was of 49.11.

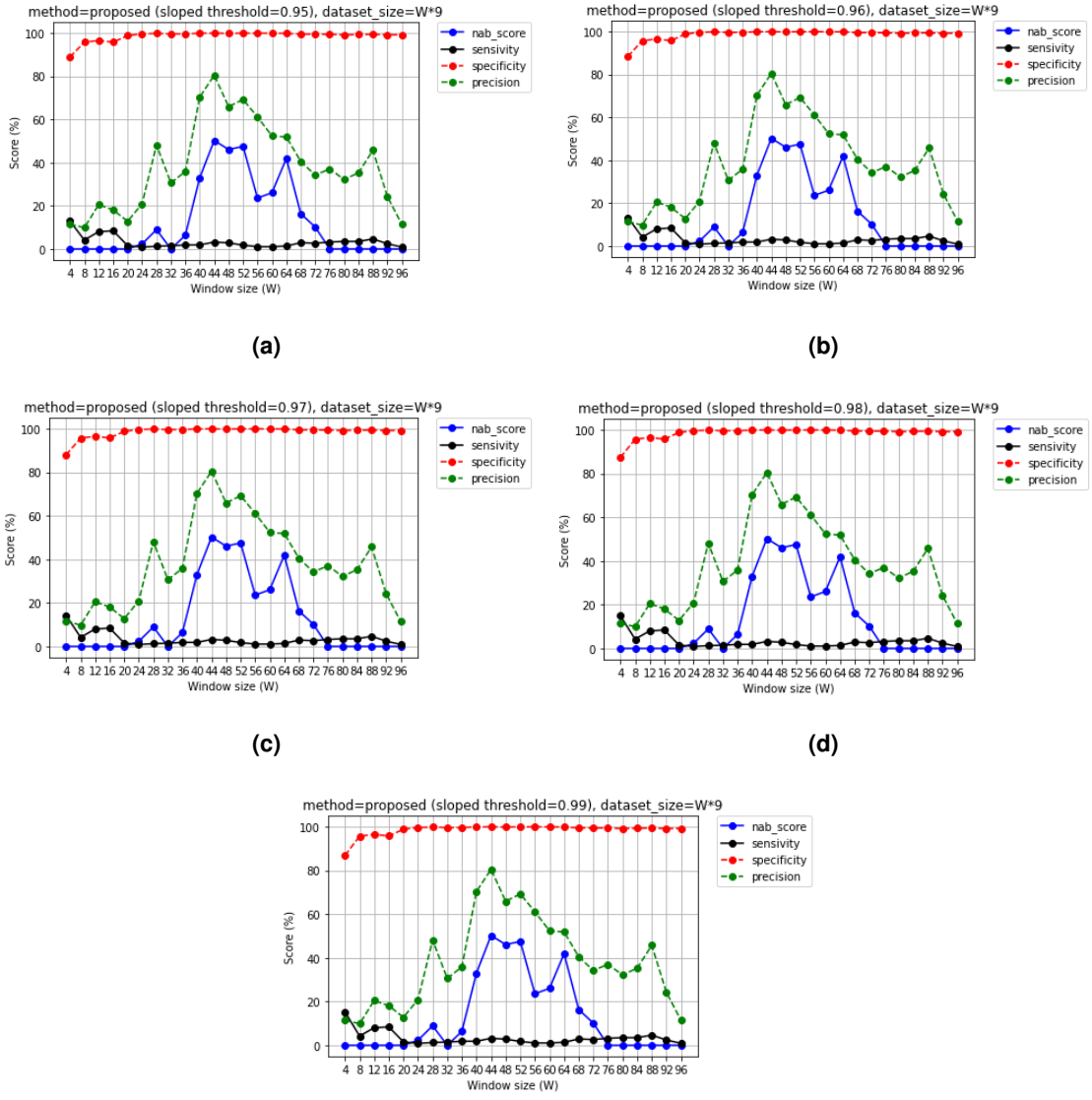


Figure 5.6: Discord similarity threshold variation results for a fixed dataset size of 9 times the window size. Multiple window sizes tested for each run (represented in X axis). Represented set of experiments reflects the discord variation for the best window size found in previous experiments, in this case 9 times.

5.3 Automatic window selection via Fourier Transform

Finally, the automation mechanism will be tested in order to assert whether or not the classifier does produce viable results.

First we will setup the test parameters as shown in table 5.3, varying the dataset size and similarity threshold, as the window parameters will be automatically derived. All other parameter were kept as introduced in section 5.1. The dataset will range from as little as 1 week to to an entire month, with a 3 day step increment.

Table 5.3: Parameter range variations

Variable	Start Value	Value Step	End Value
dataset size (**S**)	48*7	48*3	48*7*4
window size **W**	auto	-	-
`disc_sim_threshold`	0.95	0.01	0.99
use_minimum_distance	1	-	-
fourier_on	1	-	-

The first thing we notice from this set for experiments, represented in figure 5.7 is that the specificity parameter remained almost at 100%, as well as the sensitivity near 0%. This seems to be a behaviour congruent with the ones previously shown for the distance based algorithm in figures 5.3 and 5.4, reflecting the learning of the normal behaviour. Next, we can see that for all runs the precision score was lower for lower dataset sizes and higher for higher dataset sizes. Furthemore, with the increase in the dataset sizem not only did the precision increase, but also the NAB score. Thus we can state that the larger the dataset provided was, the better the results we got (specially true for experiments 5.7(a),5.7(b), and 5.7(c)).

However, something quite remarkable for this set of experiments is that the discord similarity threshold parameter tested had minimal impact in results, as we can see through experiments 5.7(a) to 5.7(e). It did however impact the results for dataset sizes between 480 and 912 points, lowering the results of the NAB for tighter similarity values. This implies that the higher the similarity the later the detection took place, thus the slightly lower NAB score.

In sum, for this batch of experiences the best classifier corresponds to a dataset size of 1200 points, for a discord similarity of 0.96 (located in figure 5.7(e)) that got a total score of 60.71 with 9 true positive classifications, and 3 false positive.

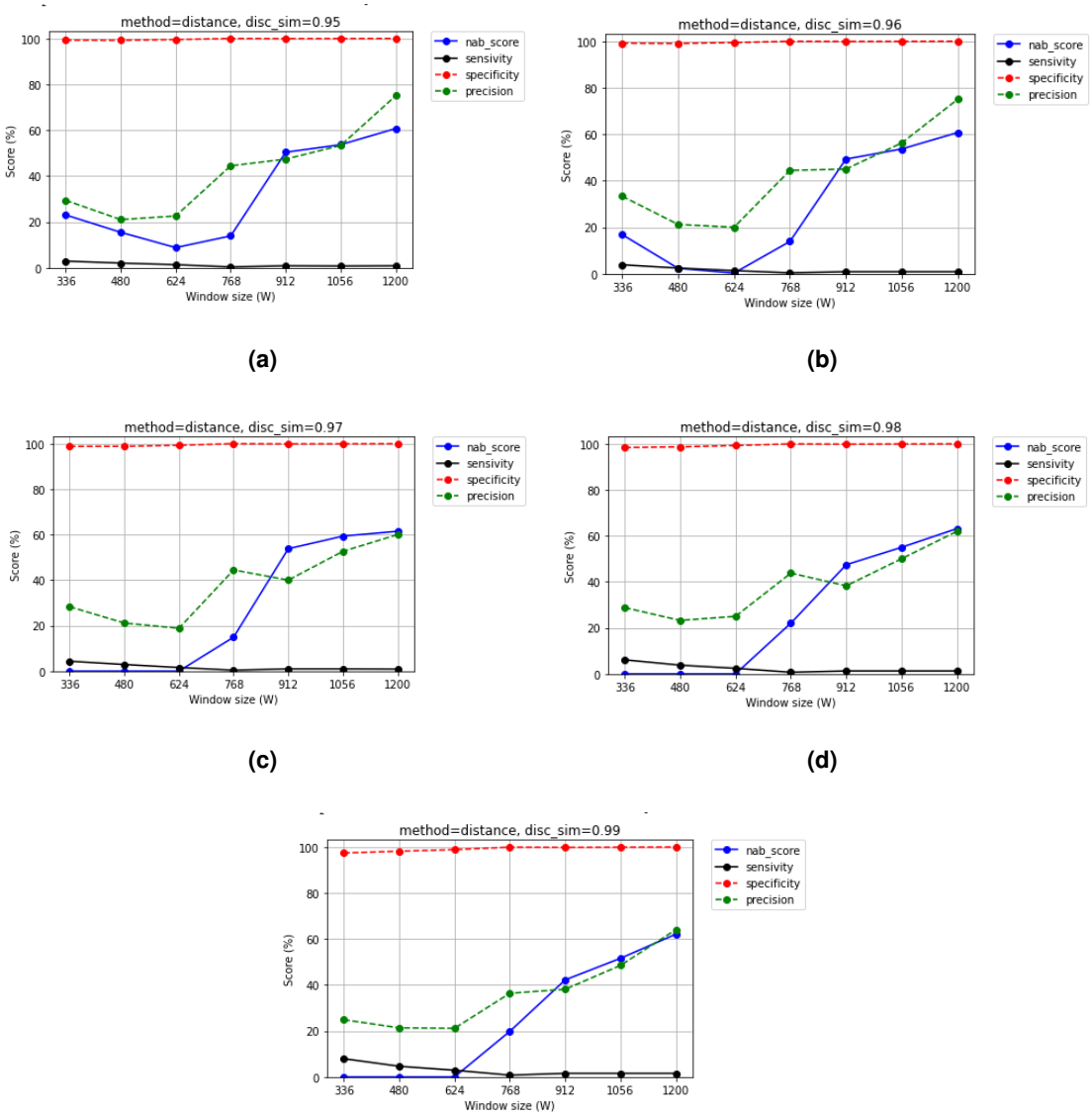


Figure 5.7: Results for the automatic window selection procedure over the original taxi domain. X axis represents the dataset size in data points. Multiple quality metrics shown for the final results of each run.

5.3.1 Automatic Window Distance Based Sequence comparison over the differentiated taxi domain

For the set of experiments over the differentiated taxi domain, shown in figure 5.8, some peculiar behaviour takes place.

Upon closer analysis of all sub-figures, we can see that the results for all similarity thresholds were the same. Thus, the only parameter that affected the classification procedure was the dataset size (since the window size is automatically extracted).

It can be said that, due the lack of one single dominant component within the time-series (previously analysed via the Fourier Transform and spectral analysis) we cannot properly identified anomalous points with 1 single window in the differentiated domain. This may be due to the presence of multiple dominant frequencies, as explored in section 3.4 or due the differentiation process. However, it does suggest some difficulty from the method in working with the results of this type of operation, namely differentiation.

The best classifier for this set of experiments corresponds to a dataset size of 336 points (3 days), at any discord similarity threshold tested. These, however, fall very short from other classifiers previously explored, implying the automatic window selection mechanism, via the Fourier Transform may not capture the best single window for the detection, but only approximate guesses, this being especially true for the sloped domain.

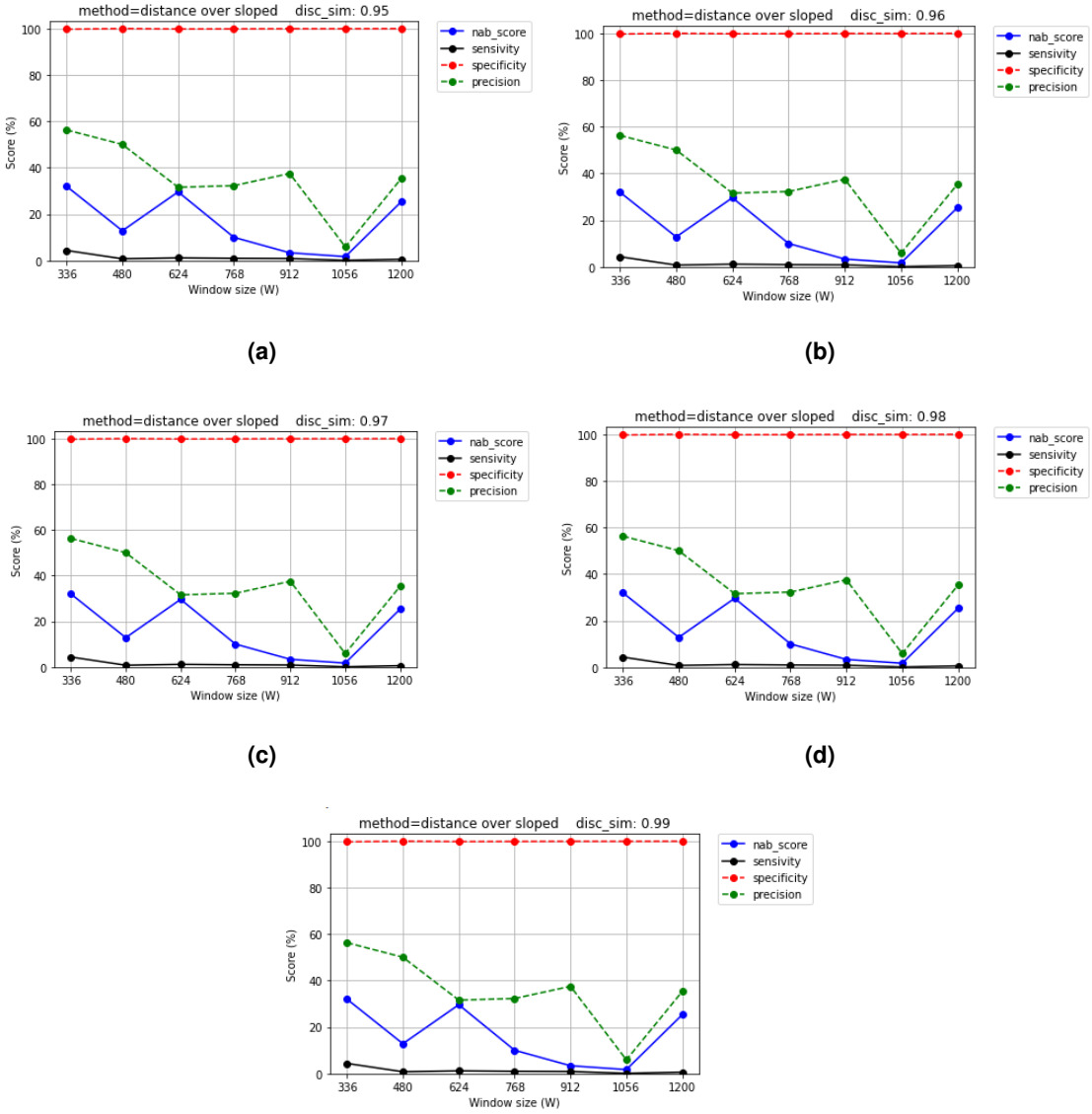


Figure 5.8: Results for the automatic window selection procedure over the differentiated taxi domain. X axis represents the dataset size in days. Multiple quality metrics shown for the final results of each run. No difference can be detected for the varying similarity threshold parameter.

5.4 Best model configuration results

The KNN CAD algorithm that came bundled with the Numenta benchmark proves to have a very erratic behaviour during classification, as show in figure 5.9. Because of this ever oscillating anomaly score, thus making the algorithm inappropriate for human inspection, and given no optimization step by the benchmark optimizer is run, the resulting values are useless for the process by having no clear separation boundary of what is anomalous and what is not. It did however manage to finish with a final NAB score of 52.65 with 3 true positives and 4 false positives, correctly identifying 2 out of 5 anomalies.

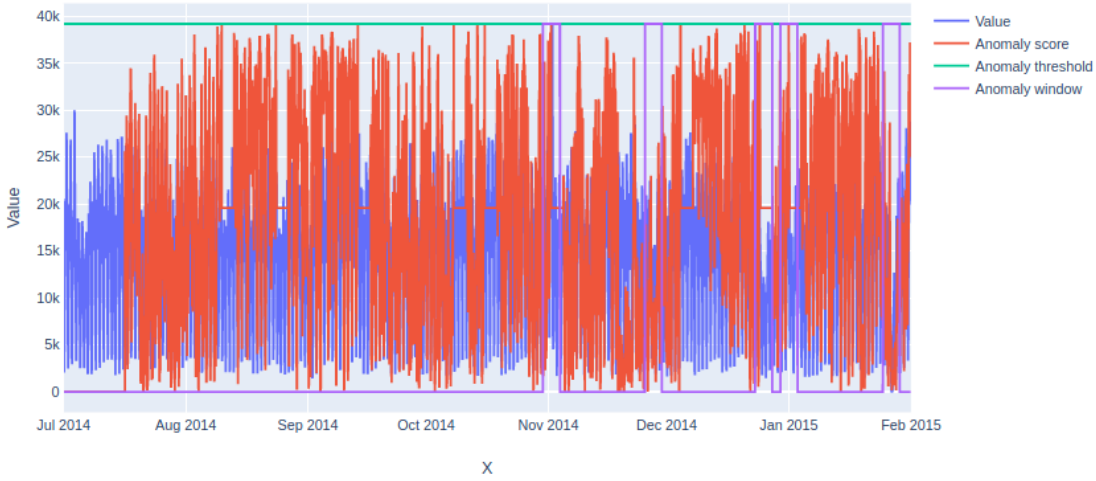


Figure 5.9: Results for the KNN CAD classifier present in the Numenta Benchmark.

The following figure (fig. 5.10) reflects the classification process as achieved when using the Numenta Hierarchical Temporal Memory algorithm. Despite a couple of errors throughout the complete execution, this classifier is capable of identifying correctly 4 out of 5 anomalies with only 1 true error (around the 17th of September), thus making it a very strong candidate for a baseline comparison with a final NAB score of 74.38.

Once again, and given the optimizer that find the best threshold cut for the classification step (in case the label is continuous and not binary) we may be lead to think that the score would not be humanly readable. However, and unlike KNN-CAD, we see see a clear separation between ”normal” score values (below the 15k mark when scaled) and real anomalies (seen as global spikes) with the bare eye. Thus, we can state that this model is much more relevant the the previous KNN-CAD, and we expect to see some improvement over this model by using a binary label while maintaining the detection quality.

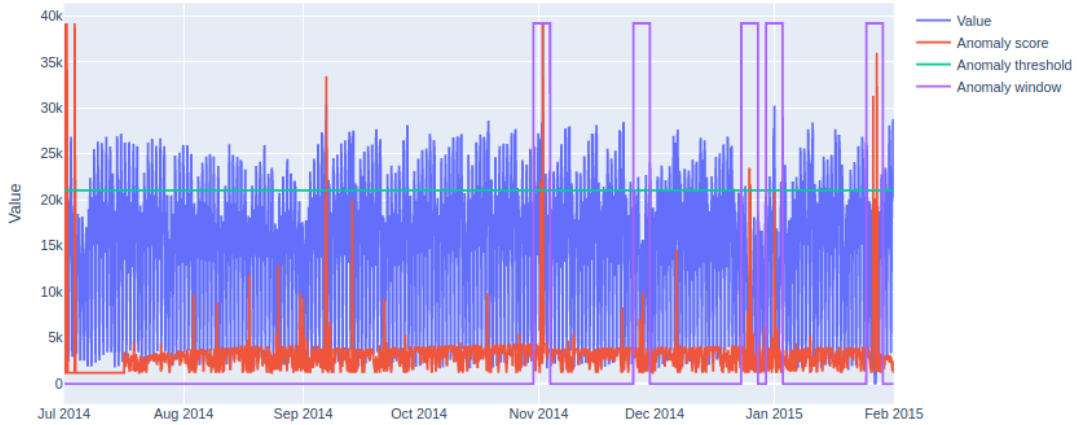


Figure 5.10: Results for the Numenta classifier.

The best model for the probabilistic version over the taxi domain managed to achieved a score of 65.66 for a window size of 8 and dataset size of 4 times window size. Despite achieving a high score it is important to notice that, if not given an optimization step where we look through all anomaly threshold cuts, this classifier would suffer from the same problems as the KNN-CAD, namely difficulty in interpreting the score and finding a suitable threshold cut. Contrary to the latter, our achieved resulting scores are much more simpler to interpret/distinguish. However, a few more mistakes were made than all previous.

Exact results for the entire classification process can be found below in figure 5.11

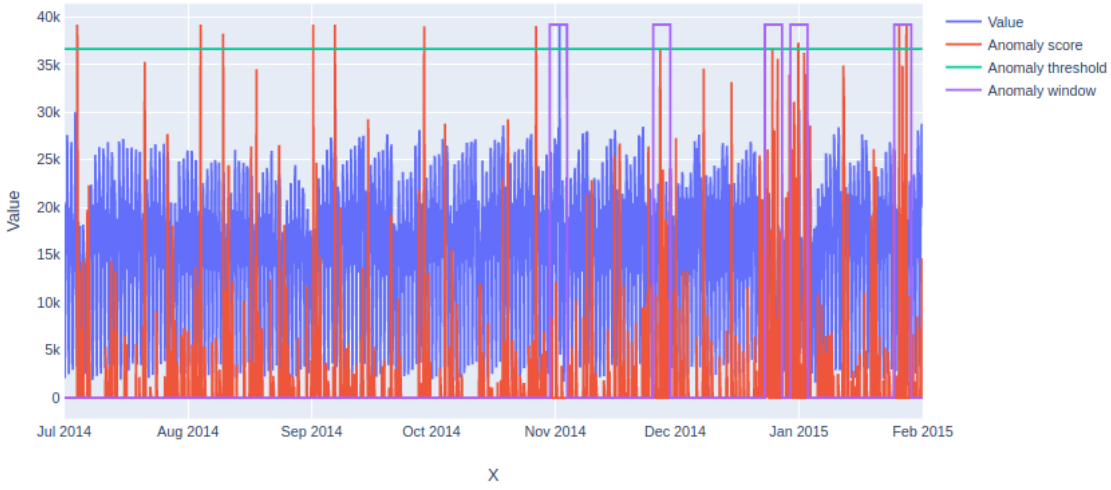


Figure 5.11: Results for the statistical classifier over the original dataset.

The best statistical model over the differentiated domain is shown below in figure 5.12 and did not prove to be an upgrade over the statistical model running on the original dataset. The anomaly score

is not only very hard to interpret, but also when plotted against the time series, it did not prove to spike only within anomaly windows. Furthermore, it spiked at different intensities (namely higher and lower score) independently of having an anomaly or not.

However, it did manage to finish with a final score of 46.97, mainly due the only valid detection being late (corresponding to the November anomaly).

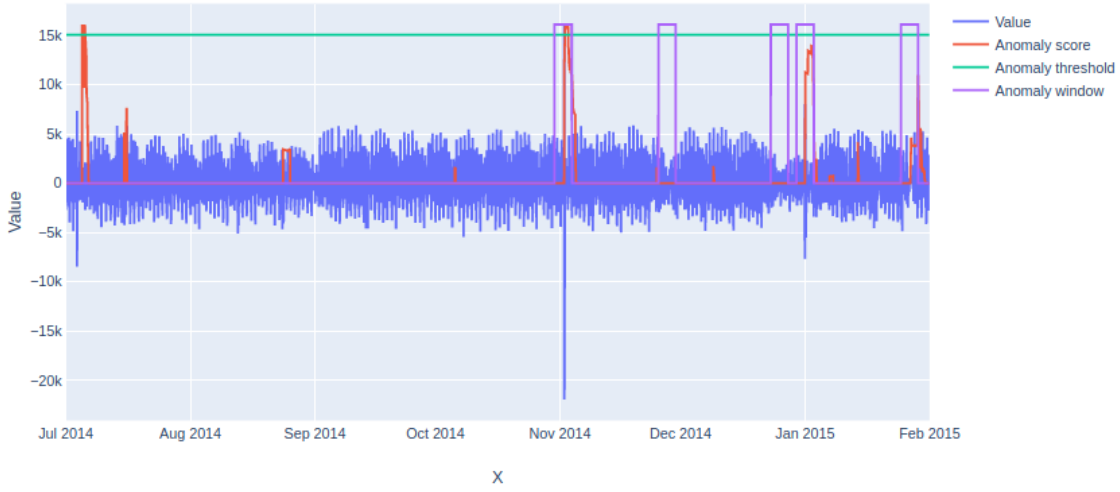


Figure 5.12: Results for the statistical classifier over the sloped dataset. Window size = 44, dataset size = 2*window size, ex zone = window size / 4, start evaluation = dataset size - 1. Resulting NAB score of 46.97. Overall bad performance due to the high false positive rate. 1 out of 5 anomalies are effectively captured.

For the Distances Based Anomaly Detection model based on sequence comparison, introduced in section 4.5, the results for the best classifier can be found in figure 5.13. It is the best achieved model due to the high NAB score of 69.99 combined with a very high precision. The setup used to achieved the results corresponds to a window size of 96, for a similarity threshold of 0.96 at a dataset size of 9 times the window. The binary label also removes are problems associated with score value interpretation and optimization.

With this classifier we have managed to detect 4 out of five anomalies, with minimal mistakes, and when these happened they were located around a real anomaly.

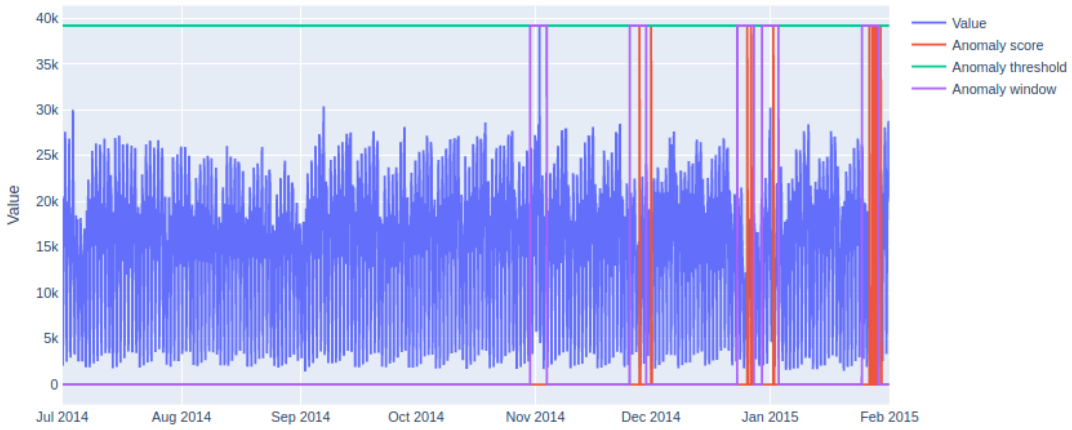


Figure 5.13: Results for the best sequence comparison based classifier over the original taxi dataset, with anomaly value visualisation. Parameters used are *window size = 96*, *dataset size = 9 x window size* for a `disc_sim_threshold` of 0.96. Four out of five anomalies and correctly identified with minor errors. Further inspection reflect a total of 17 true positives, 2 false positives and a final NAB score of 69.99.

The distance based classifier over the taxi dataset managed once again to achieve great results. Detecting 3 out of 5 anomalies, one of which that had previously gone undetected (corresponding to the anomaly in November). It finished with a final NAB score of 49.11 detecting 3 out of nine anomalies with 1 major error, between January and February.

The used parameters were *window size = 52*, *dataset size = window size x 6*, and *discord similarity = 0.99*. The result of the classification can be found in figure 5.14, and similarly to the distance based sequence discord management method, managed to achieve very promising results with 23 true positives, and only 4 false positives. If we gather the results of the classifier under appreciation, for both the sloped, and non sloped domains of the taxi dataset, manages to detect all annotated anomalies, thus a combination of both would make a great ensemble for this specific data stream represented.

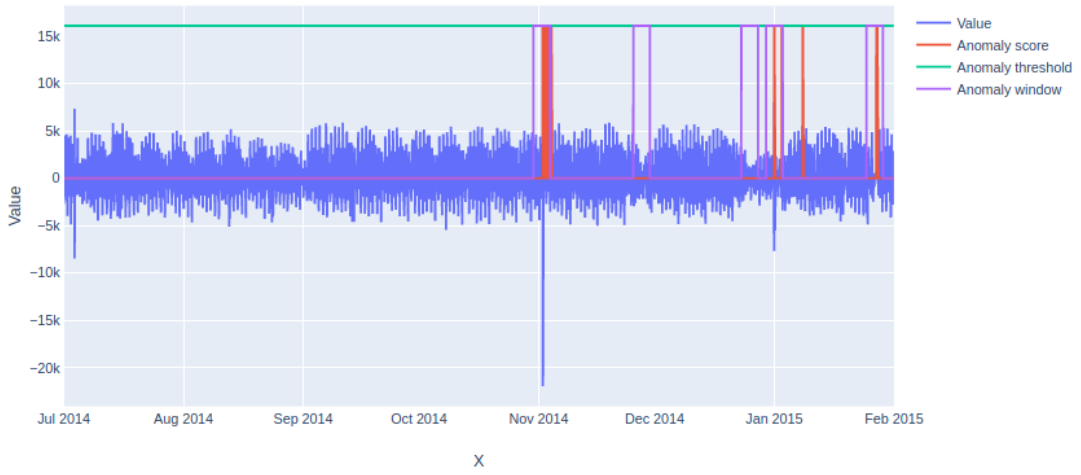


Figure 5.14: Results for the best classifier found (proposed solution) over the sloped dataset. Anomaly value visualisation.

To sum up this section, the best classifier achieved during this work was our distance based discord management over the original dataset. Not only it achieved an approximate NAB score to the Numenta classifier, but also managed to get a very low false positive rate. However, we consider it to have a much more intuitive score interpretation, thus the choice. It finished with a NAB score of 69.99, issuing 17 true positive labels and only 2 false positive. Last but not least, the distance based sequence comparison algorithm managed to detect the November anomaly only on the differentiated dataset.

Consequently, the final, best found configuration procedure is show below in figure 5.15.

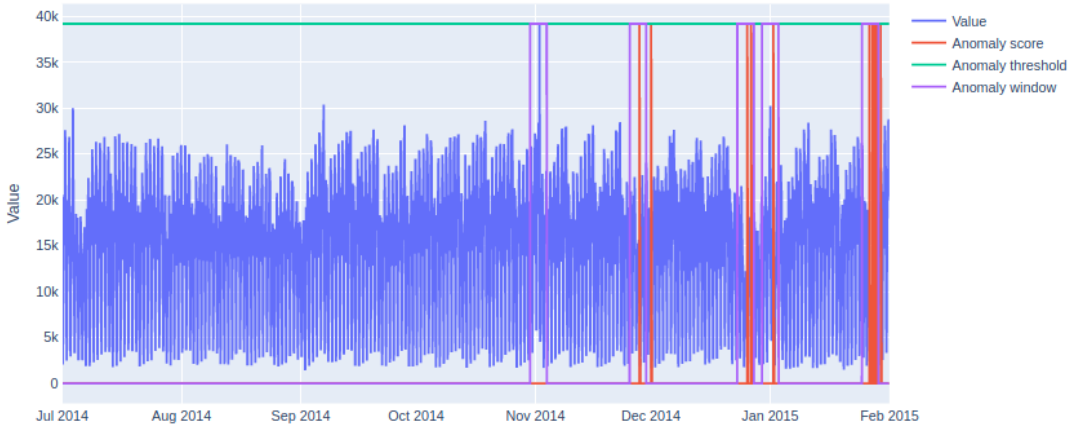


Figure 5.15: Results for the best sequence comparison based classifier over the original taxi dataset, with anomaly value visualisation. Parameters used are *window size = 96*, *dataset size = 9 x window size* for a `disc_sim_threshold` of 0.96. Four out of five anomalies and correctly identified with minor errors.

6

Conclusions

Contents

6.1 Conclusions

Anomaly detection is a prominent area of analysis for finding anomalous and/or unexpected behavior in data. One particular area of interest for anomaly detection is datastreams, possible infinite flows of information, where it is often expected to score one information point in time before the next one arrives. Given that datastreams may be endless, and inherent problem with the volume of data for analysis, space and time arise. One way to do this is to adapt the Matrix Profiles introduced by Eammon Keogh to evaluated univariate time-series and detect anomalous data points.

Our proposal has proven to be a reasonable solution for the problem at hand overcoming the identified problems, in particular, evaluating datastreams with constant overhead while being able to issue anomaly scores at each time point, and naturally dealing with the concept drift phenomena. This work was validated over the known New York City Taxi domain.

We have demonstrated that using the matrix profiles, it is possible to detect anomalies in univariate time series, at par with state of the art anomaly detection algorithms, achieving almost perfect scores. Our best classifier managed to detect 4 out of 5 anomalies from the taxi domain, with a precision of 87%, 17 true positives, 2 false positives, and a final NAB score of 69.99%. Consequently, we can state that all the goals set for this work were fully met, namely detection of anomalies, in an online manner, over datastreams.

The proposed approach on Eammon Keogh Matrix Profiles, turned out to be competitive with state of the art algorithms. despite its very simplistic and intuitive approach to the problem.

6.2 Future Work

In the future, more detail should be taken to the adjustment process of the window via the Fourier Transform, as we have shown that, for a single window, this method may not yield the best results.

In addition, thorough work should also be put in trying to set up an ensemble for the data stream classification. As we have shown, a single classifier with a single window may not detect all anomalies, thus voting between assembled algorithms should take place. One straight forward way to do this is to continuously differentiate the kept dataset and work on both the sloped and normal versions at the same time.

Further impacts of the dataset characteristics on the evaluation procedure should also be studied, with special emphasis on the similarity function that is not able to distinguish collinear vectors.

Bibliography

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009. [Online]. Available: http://doi.acm.org/10.1145/1541880.1541882

[2] G. Piateski and W. Frawley, *Knowledge Discovery in Databases*. Cambridge, MA, USA: MIT Press, 1991.

[3] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth *et al.*, "Knowledge discovery and data mining: Towards a unifying framework." in *KDD*, vol. 96, 1996, pp. 82–88.

[4] C. Shearer, "The crisp-dm model: the new blueprint for data mining," *Journal of data warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[5] O. Marban, G. Mariscal, and J. Segovia, "A data mining & knowledge discovery process model," in *Data Mining and Knowledge Discovery in Real Life Applications*, J. Ponce and A. Karahoca, Eds. Rijeka: IntechOpen, 2009, ch. 1. [Online]. Available: https://doi.org/10.5772/6438

[6] A. Azevedo and M. Filipe Santos, "Kdd, semma and crisp-dm: A parallel overview," 01 2008, pp. 182–185.

[7] S. Angée, S. Lozano, E. N. Montoya-Munera, J. Ospina Arango, and M. Tabares, *Towards an Improved ASUM-DM Process Methodology for Cross-Disciplinary Multi-organization Big Data & Analytics Projects: 13th International Conference, KMO 2018, Žilina, Slovakia, August 6–10, 2018, Proceedings*, 07 2018, pp. 613–624.

[8] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39 – 57, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217302631

[9] F. Giannoni, M. Mancini, and F. Marinelli, "Anomaly detection models for iot time series data," *arXiv preprint arXiv:1812.00890*, 2018.

[10] “Ucr time series website,” https://www.cs.ucr.edu/~eamonn/time_series_data/, accessed: 2020-02-22.

[11] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2523813

[12] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” *Neurocomputing*, vol. 262, pp. 134 – 147, 2017, online Real-Time Learning Strategies for Data Streams. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217309864

[13] K. Nayyar, “Anomaly detection for univariate time-series data,” 2015.

[14] S. Ben-David, E. Kushilevitz, and Y. Mansour, “Online learning versus offline learning,” *Machine Learning*, vol. 29, no. 1, pp. 45–63, Oct 1997. [Online]. Available: https://doi.org/10.1023/A:1007465907571

[15] R. M. Karp, “On-line algorithms versus off-line algorithms: How much is it worth to know the future?” in *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing ’92, Volume 1 - Volume I*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co., 1992, pp. 416–429. [Online]. Available: http://dl.acm.org/citation.cfm?id=645569.659725

[16] Y. Cui, C. Surpur, S. Ahmad, and J. Hawkins, “A comparative study of htm and other neural network models for online sequence learning with streaming data,” 07 2016, pp. 1530–1538.

[17] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, “Real-time big data processing for anomaly detection: A survey,” *International Journal of Information Management*, vol. 45, pp. 289 – 307, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0268401218301658

[18] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection for discrete sequences: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, pp. 823–839, May 2012.

[19] E. Burnaev and V. Ishimtsev, “Conformalized density- and distance-based anomaly detection in time-series data,” 2016.

[20] J. Gama, R. Sebastião, and P. P. Rodrigues, “On evaluating stream learning algorithms,” *Machine Learning*, vol. 90, no. 3, pp. 317–346, Mar 2013. [Online]. Available: https://doi.org/10.1007/s10994-012-5320-9

[21] A. Lavin and S. Ahmad, “Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec 2015, pp. 38–44.

[22] T. Wen and R. Keyes, “Time series anomaly detection using convolutional neural networks and transfer learning,” *CoRR*, vol. abs/1905.13628, 2019. [Online]. Available: http://arxiv.org/abs/1905.13628

[23] R. P. Adams and D. J. C. MacKay, “Bayesian online changepoint detection,” 2007.

[24] A. Y. Liu and C. E. Martin, “Smoothing multinomial naïve bayes in the presence of imbalance,” in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 46–59.

[25] M. Urvoy and F. Autrusseau, “Application of grubbs’ test for outliers to the detection of watermarks,” in *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security*, ser. IHamp;MMSec ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 49–60. [Online]. Available: https://doi.org/10.1145/2600918.2600931

[26] Y. Zhang, X. Yang, and H. Li, “An outlier mining algorithm based on confidence interval,” in *2010 2nd IEEE International Conference on Information Management and Engineering*, 2010, pp. 231–234.

[27] R. Laxhammar, G. Falkman, and E. Sviestins, “Anomaly detection in sea traffic - a comparison of the gaussian mixture model and the kernel density estimator,” in *2009 12th International Conference on Information Fusion*, July 2009, pp. 756–763.

[28] Z. Wang, J. Yang, Z. ShiZe, and C. Li, “Robust regression for anomaly detection,” in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[29] P. Schiilkop, C. Burgest, and V. Vapnik, “Extracting support data for a given task,” in *Proceedings of the 1st international conference on knowledge discovery & data mining*, 1995, pp. 252–257.

[30] J. Ting, E. Theodorou, and S. Schaal, “A kalman filter for robust outlier detection,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 1514–1519.

[31] W.-K. Wong, A. Moore, G. Cooper, and M. Wagner, “Rule-based anomaly pattern detection for detecting disease outbreaks,” in *AAAI/IAAI*, 2002, pp. 217–223.

[32] J. Roy, “Rule-based expert system for maritime anomaly detection,” in *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX*, vol. 7666. International Society for Optics and Photonics, 2010, p. 76662N.

[33] V. Ishimtsev, I. Nazarov, A. Bernstein, and E. Burnaev, “Conformal k-nn anomaly detector for univariate data streams,” *arXiv preprint arXiv:1706.03412*, 2017.

[34] Y. Liao and V. R. Vemuri, “Use of k-nearest neighbor classifier for intrusion detection,” *Computers & security*, vol. 21, no. 5, pp. 439–448, 2002.

[35] M. Çelik, F. Dadaşer-Çelik, and A. Ş. Dokuz, “Anomaly detection in temperature data using dbscan algorithm,” in *2011 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE, 2011, pp. 91–95.

[36] S. Salvador, P. Chan, and J. Brodie, “Learning states and rules for time series anomaly detection.” in *FLAIRS conference*, 2004, pp. 306–311.

[37] L. Portnoy, “Intrusion detection with unlabeled data using clustering,” Ph.D. dissertation, Columbia University, 2000.

[38] K. Leung and C. Leckie, “Unsupervised anomaly detection in network intrusion detection using clusters,” in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*. Australian Computer Society, Inc., 2005, pp. 333–342.

[39] C. D. Truong and D. T. Anh, “An efficient method for motif and anomaly detection in time series based on clustering,” *International Journal of Business Intelligence and Data Mining*, vol. 10, no. 4, pp. 356–377, 2015.

[40] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, A. Dau, D. Silva, A. Mueen, and E. Keogh, “Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets,” 12 2016, pp. 1317–1322.

[41] R. J. Bayardo, Y. Ma, and R. Srikant, “Scaling up all pairs similarity search,” in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 131–140.

[42] H. Lee, R. T. Ng, and K. Shim, “Similarity join size estimation using locality sensitive hashing,” *arXiv preprint arXiv:1104.3212*, 2011.

[43] D. Brauckhoff, K. Salamatian, and M. May, “Applying pca for traffic anomaly detection: Problems and solutions,” in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 2866–2870.

[44] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, “Anomaly detection via online oversampling principal component analysis,” *IEEE transactions on knowledge and data engineering*, vol. 25, no. 7, pp. 1460–1470, 2012.

[45] J. a. Gama, R. Sebastião, and P. P. Rodrigues, "Issues in evaluation of stream learning algorithms," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 329–338. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557060

[46] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[47] K. J. Hole, *The HTM Learning Algorithm*. Cham: Springer International Publishing, 2016, pp. 113–124. [Online]. Available: https://doi.org/10.1007/978-3-319-30070-2_11

[48] D. Danilov and A. Zhigljavsky, "Principal components of time series: the 'caterpillar'method," *St. Petersburg: University of St. Petersburg*, pp. 1–307, 1997.

[49] G. Sidorov, A. Gelbukh, H. Gomez Adorno, and D. Pinto, "Soft similarity and soft cosine measure: Similarity of features in vector space model," *Computación y Sistemas*, vol. 18, 09 2014.

[50] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between euclidean and cosine angle distance for nearest neighbor queries," in *Proceedings of the 2004 ACM Symposium on Applied Computing*, ser. SAC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 1232–1237. [Online]. Available: https://doi.org/10.1145/967900.968151

[51] D. J. Weller-Fahy, B. J. Borghetti, and A. A. Sodemann, "A survey of distance and similarity measures used within network intrusion anomaly detection," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 70–91, 2015.

[52] C.-J. Hsu, K.-S. Huang, C.-B. Yang, and Y.-P. Guo, "Flexible dynamic time warping for time series classification," *Procedia Computer Science*, vol. 51, pp. 2838 – 2842, 2015, international Conference On Computational Science, ICCS 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915012521

[53] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, 1989. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478785

[54] ——, "Probabilistic linkage of large public health data files," *Statistics in Medicine*, vol. 14, no. 5[U+2010]7, pp. 491–498, 1995. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780140510

[55] L. Hamers *et al.*, "Similarity measures in scientometric research: The jaccard index versus salton's cosine formula." *Information Processing and Management*, vol. 25, no. 3, pp. 315–18, 1989.

[56] A. Bellas, C. Bouveyron, M. Cottrell, and J. Lacaille, “Anomaly detection based on confidence intervals using som with an application to health monitoring,” *Advances in Intelligent Systems and Computing*, vol. 295, 07 2014.

[57] Y. Yu, Y. Zhu, S. Li, and D. Wan, “Time series outlier detection based on sliding window prediction,” *Mathematical Problems in Engineering*, vol. 2014, pp. 1–14, 2014.

[58] Z. Ding and M. Fei, “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window,” *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12 – 17, 2013, 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016314999

[59] T. Zoppi, A. Ceccarelli, and A. Bondavalli, “On algorithms selection for unsupervised anomaly detection,” in *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*, 2018, pp. 279–288.

[60] L. Feremans, V. Vercruyssen, B. Cule, W. Meert, and B. Goethals, “Pattern-based anomaly detection in mixed-type time series,” 07 2019.

[61] L. Stanković, T. Alieva, and M. J. Bastiaans, “Time–frequency signal analysis based on the windowed fractional fourier transform,” *Signal Processing*, vol. 83, no. 11, pp. 2459 – 2468, 2003, fractional Signal Processing and Applications. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016516840300197X

[62] A. Palmeri and A. Cicirello, “Physically-based dirac’s delta functions in the static analysis of multi-cracked euler–bernoulli and timoshenko beams,” *International Journal of Solids and Structures*, vol. 48, no. 14, pp. 2184 – 2195, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020768311001260

[63] “Matrix profile page,” https://matrixprofile.org/, accessed: 2020-02-22.

[64] “Matrix profile github,” https://github.com/target/matrixprofile-ts, accessed: 2020-02-22.