Mini-Project (ML for Time Series) - MVA 2023/2024

Adib Habbou adibhabboupro@gmail.com
Teddy Alexandre teddyalexandre1@gmail.com

December 12, 2023

1 Introduction

Anomaly detection in time series has been a long-standing and significant research topic. It uses a range of techniques, from statistical to advanced deep learning methods, aiming to identify outliers or deviations indicating potential issues or noteworthy events. These methods find application in diverse fields such as healthcare, finance, cybersecurity, and industrial operations, playing a crucial role in early detection, risk mitigation, and system reliability maintenance.

The paper [1] we are going to review focuses on time series discords, which highlight unusual subsequences within time series, the paper addresses the challenge of selecting the optimal subsequence length in order to detect anomalies of various length without relying on specific user-defined parameters. Introducing MERLIN, an innovative algorithm based on DRAG [2], the study aims to significantly enhance the precision of anomaly detection methods.

For all our experiments, no available source code has been used, we re-implemented everything from scratch. We both implemented in a pair-programming session the main core of the code (DRAG and MERLIN) then we tested them on some toy examples we generated manually. We started by making experiments already done in the paper: Adib did the analysis and applied MERLIN on the New York City Taxi dataset, meanwhile Teddy did the analysis and applied MERLIN on the Yahoo dataset. But we went further, by studying the evolution of MERLIN when varying it's two parameters MinL and MaxL. Finally, we tried to see how good can MERLIN be when applied to "hard" problems: Adib generated time series with ultra subtle anomalies, on the other hand Teddy generated time series with twin freak anomalies, we both tested MERLIN on them.

2 Method

First of all, we introduce a few mathematical definitions related to the domain of time series anomaly detection, and in particular to **discord** detection.

Time series (TS): A time series $T = t_1, ..., t_n$ is a sequence of n real values.

Subsequence of a TS: A subsequence $T_{i,L}$ is a contiguous subset of values with length L starting from position i in time series T. Denoted $T_{i,L} = t_i, t_{i+1}, ..., t_{i+L-1}$, where $(1 \leq i \leq |T| - L + 1)$ and L is a user-defined subsequence length with value in range of $3 \leq L \leq |T|$.

Non-self match: Given a time series T containing a subsequence $C = T_{p,L}$ and a matching subsequence $M = T_{q,L}$, we say that M is a non-self match to C at distance of $Dist(M,C)$ if $|p - q| \geq L$.

Z-normalized euclidean distance: $Dist$ is the euclidean distance between two normalized time series. If x has mean μ and standard deviation σ, the z-normalized x is $\hat{x} = \frac{x-\mu}{\sigma}$.

Provided with these four definitions, we can now define a time series discord :

Time series discord: Given a time series T, the subsequence $D = T_{i,L}$ is said to be the discord of T if D has the largest distance to its nearest non-self match. That is, $\forall$ subsequences C of T, non-selfmatch M_D of D, and non-self match M_C of C, $min(Dist(D, M_D)) > min(Dist(C, M_C))$.

With the previous definitions provided, the method employed in the article is discussed. The discord detection algorithm, **MERLIN**, relies a lot on another discord detection algorithm from another article [2], called **DRAG** (Discord Range Aware Gathering). The DRAG algorithm is split in two main phases: candidate selection and discords refinement.

The first phase of the DRAG algorithm aims at finding the possible candidates for the discords. It takes as argument a time series T, a subsequence length L, a range of discords r and returns the set of discord candidates C. It parses the time series, analyzing each subsequence. If this subsequence is greater than r, it may be a potential discord, so this subsequence is added to the list of candidates. However, if this subsequence is lower than r, it is not considered as a discord and is rejected. Thus, the parameter r acts as a threshold for admitting possible discords. At the end, the set of candidates C should contains the true discords, among a list of potential false positives discords. If C is not empty, it is used in the second phase, otherwise the algorithm fails.

The second phase of the algorithm proceeds to a refinement of the discord candidates. This phase takes as argument the same parameters as above, this time including the previous set of candidates C. The iterations are quite similar to the first phase, except that we consider here the true set of discords D. If the subsequence is lower than r, it is not a discord, otherwise we save the discord in D. The algorithm guarantees that there is at least one discord found. At the end, we identify the discords, provided with their starting index and the distance to their nearest non-self match.

The main difficulty of the DRAG algorithm relies on the choice of the parameter r. Indeed, the authors asked themselves about the optimal way to find r, which is the critical point of the DRAG algorithm. After running DRAG on a simple example many times with increasing values of r, they observed that a value of r greater than 10.27 ended in "failure" systematically. On the other hand, the algorithm worked for smaller values than 2 but took a longer time to run. The observations concluded in an estimation of r that was "a little less" than discord distance.

This notion of "little less" is modeled by the study of the variance of the last few discord values. That way, r can be adapted iteratively through the algorithm and be more robust. This gives the intuition behind MERLIN algorithm. MERLIN takes as input the time series, a subsequence lower bound $MinL$ and a subsequence upper bound $MaxL$:

1. The first phase aims at finding the first discord by decreasing r until one is found (r is initialized with $r = 2\sqrt{MinL}$ at the beginning of the algorithm).

2. Next, the code locates the subsequent for discords by iteratively adjusting r based on the previous distance until each discord is identified.

3. Finally, it identifies the remaining discords by estimating an optimal r using local statistical measures (mean and standard deviation) and iteratively adjusting r until discord is found.

All in all, the MERLIN algorithm consists in adjusting the parameter value r to reach the optimal way of finding discords using DRAG, but to do so he introduces $MinL$ and $MaxL$.

3 Data

3.1 Toy Examples

The first data we used to test if our implementation of DRAG and MERLIN were correct was:

- a constant time serie of length 1000 with one anomaly of size 50 (see Figure 4)
- a constant time serie of length 1000 with two anomalies of size 20 and 50 (see Figure 5)
- a sinusoidal time serie of length 1000 with one anomaly of size 50 (see Figure 6)

3.2 New York Taxi Dataset

The dataset comes from the Numenta Anomaly Benchmark [3] and more precisely from NYC Taxi and Limousine Commission which provides on her website a dataset listing the traffic at different time stamps. The initial dataset contains more than 10 000 samples. To have a doable time execution we limited ourselves to the 1400 samples from November 2018 where we have two anomalies: one caused by the Daylight Saving Time and one caused by Thanksgiving. By looking at Figure 7 we clearly see that the time series exhibits a periodicity, repeating weekly. However, the periodogram in Figure 8 reveals a lack of a dominant frequency, suggesting the absence of seasonality. Additionally, the DFT Figure 9 shows peaks at the beginning and the end indicating potential anomalies or significant shifts in traffic volume.

3.3 Yahoo Dataset

The dataset comes from the Yahoo database [4] and contains real and synthetic time-series with tagged anomaly points. The synthetic dataset consists of time-series with varying trend, noise and seasonality, whereas the real dataset consists of time-series representing the metrics of various Yahoo services. The Figure 11 shows one of the 367 Yahoo time series from the database. Additionally, by looking at the boxplot Figure 14, we clearly see that there are some high extreme values really far from the median indicating the potential presence of anomalies.

3.4 Ultra Subtle Anomalies

The next type of data we tested are time series with ultra subtle anomalies, as they showed two examples of these type of data in the paper without investigating further. In order to generate this type of data we used a sinusoidal time serie of length 500 and we modified slightly a part of the signal. As shown in Figures 15 (from position 208 to 210) and 19 (from position 200 to 205).

3.5 Twin Freak

To provide further analysis on the performance of MERLIN, we tested it on a time series with two anomalies that are quite similar and have the same shape. This is called "twin freak". The goal here is to reproduce the fail of the algorithm on this type of time series. To generate this, we generated a noise signal, and introduced some polynomial of degree 2 in two intervals ([300, 350] and [700, 750]) with similar coefficients. The Figure 22 shows one type of "twin freak" time series.

4 Results

4.1 Toy Examples

Merlin demonstrated good anomaly prediction across the three toy examples (as illustrated on Figures 16, 17 and 18) successfully detecting both anomalies in the scenario featuring two distinct anomalies of varying lengths. However, occasional false positives were observed as Merlin detected anomalies in instances where they did not exist. But one limitation we want to point out, as it's not investigated at all in the paper [1], is the variance of output regarding the choosed $MinL$ and $MaxL$. Figure 1 illustrates a notable fluctuation in detection accuracy corresponding to the changes in $MinL$ and $MaxL$ values in a range from 3 to 40.

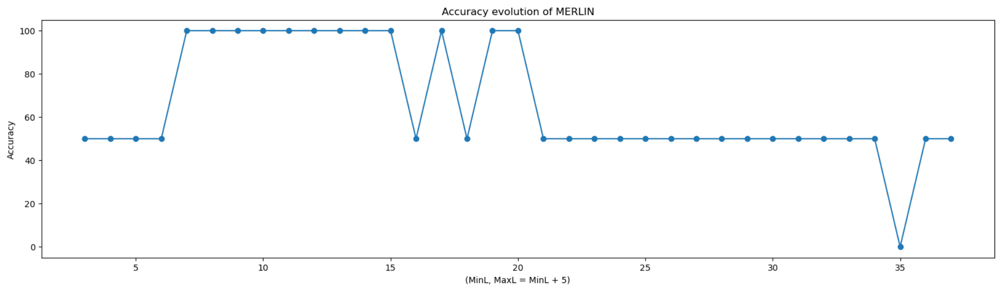

Figure 1: Evolution of prediction accuracy for different values of $MinL$ and $MaxL$ on toy example

Even in toy examples, and as we'll exhibit in other datasets, the choice of $MinL$ and $MaxL$ is crucial for MERLIN to detect anomalies correctly. Which shows the requirement for a framework to determine these parameters, raising doubts about the paper's title asserting "parameter-free".

4.2 New York Taxi

On the New York Taxi data limited to November 2018 we were not able to reproduce the exact output as in the paper where MERLIN detects both anomalies at the same time. In our experiments with various values of $MinL$ and $MaxL$ we either detected only one anomaly (the daylight saving time as it's the easiest one) or either none of them as shown in Figure 2.

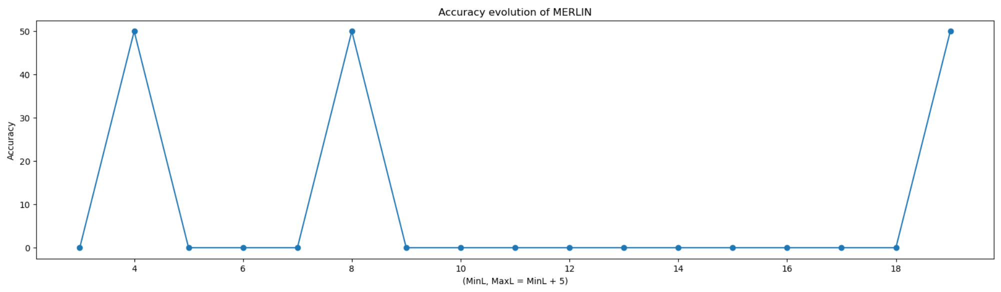

Figure 2: Evolution of prediction accuracy for different values of $MinL$ and $MaxL$ on NYC dataset

Again, we don't doubt of the fact that the results shown in the article [1] are true, but we think that they "cherry-picked" the value of $MinL$ and $MaxL$ to obtain the best model by trying empirically a large range of possible values since there is no specified frame-work for choosing them in the paper [1]. But we weren't able to do the same, since one run of MERLIN on a time serie of 1400 samples takes 10 minutes and testing a range of 20 values takes a little bit more than one hour.

4.3 Yahoo

On the Yahoo! time series, we ran into similar difficulties when trying to obtain a performant MERLIN algorithm. It barely detects any discord on different values of $MinL$ and $MaxL$, as seen in the Figure 3. It's apparent that MERLIN in most cases is not able to detect the anomalies.

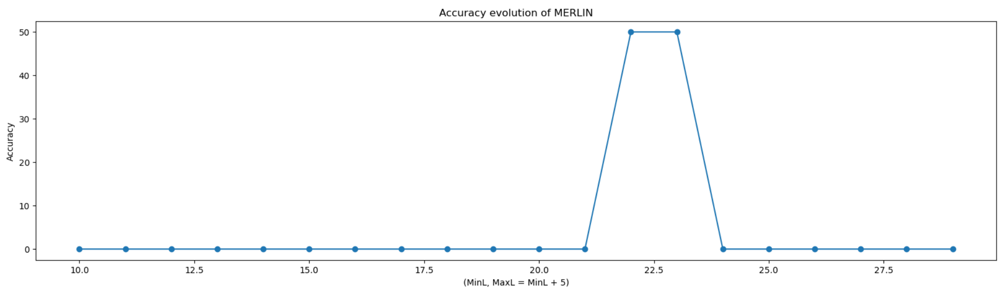

Figure 3: Evolution of prediction accuracy for different values of $MinL$ and $MaxL$ on Yahoo!

We are here again limited by the fact that the choice of values for $MinL$ and $MaxL$ are quite dependent of the time series we are working with. There is no intuition for the set of values $MinL$ and $MaxL$ can take, since it may vary even for two quite similar time series. Moreover, the algorithm takes again a long time to run, which represents another limitation.

4.4 Ultra Subtle Anomalies

Testing on ultra subtle anomalies was also a bit disappointing since MERLIN wasn't able to detect the anomaly we added to the sinusoidal signal even when testing for a large range of $MinL$ and $MaxL$ values. You can find some of the output MERLIN gives on Figures 20 and 21.

4.5 Twin Freak

As evidenced in Figure 23, the MERLIN algorithm has poor performance on the "twin freak" time series, as it does not detect any of the two anomalies, and detects one false positive in between the two anomalies. This is coherent with the observations made by the authors.

4.6 Optimizing MERLIN execution time

MERLIN++ [5] was introduced by the same authors utilizing Orchard's indexing technique to efficiently identifying time series discords of all lengths. PALMAD (Parallel Arbitrary Length MERLIN-based Anomaly Discovery) [6], a GPU-based parallelization scheme enhanced MERLIN's efficiency by eliminating redundant calculations and utilizing advanced data structures.

4.7 Choice of $MinL$ and $MaxL$

In order to improve MERLIN we need to explore ways of choosing $MinL$ and $MaxL$ parameters faster than a classical Grid Search and more generalizable than using specific domain knowledge. The main idea would be coming up with a dynamic parameter adaptation that could be applied during the algorithm. Parameters could adapt based on the length of already discovered discords and also to events occurring in the time series such as sudden spikes, drops, or changes in patterns.

References

[1] Ryan Mercer Takaaki Nakamura, Makoto Imamura and Eamonn Keogh. Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives. In *Industrial Conference on Data Mining*, 2020.

[2] E. Keogh D. Yankov and U. Rebbapragada. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. In *ICDM*, pages 381–390, 2007.

[3] Lavin A. Purdy S. Ahmad, S. and Z Agha. Unsupervised real-time anomaly detection for streaming data. In *Neurocomputing*, 2017.

[4] N. Laptev and S. Amizadeh. S5 - a labeled anomaly detection dataset, version 1.0(16m). In *Distributed by Yahoo Research*, 2015.

[5] Ryan Mercer Takaaki Nakamura, Makoto Imamura and Eamonn Keogh. Merlin++: parameter-free discovery of time series anomalies. In *Industrial Conference on Data Mining*, pages 670–709, 2023.

[6] Mikhail L. Zymbler and Yana A. Kraeva. High-performance time series anomaly discovery on graphics processor. In *arXiv:2304.01660v1*, 2023.

Appendix

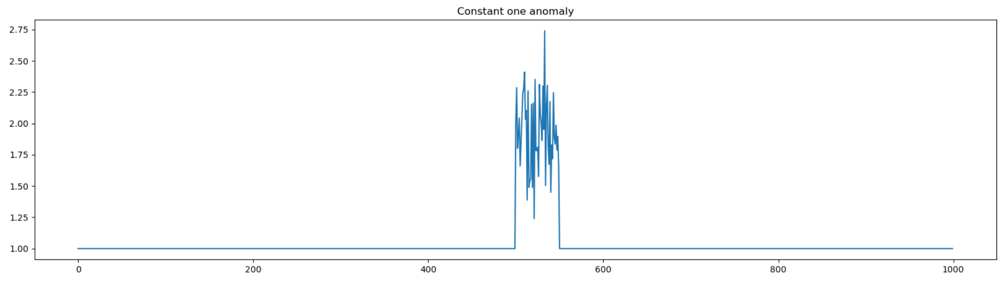

Figure 4: Constant Time Serie with one anomaly

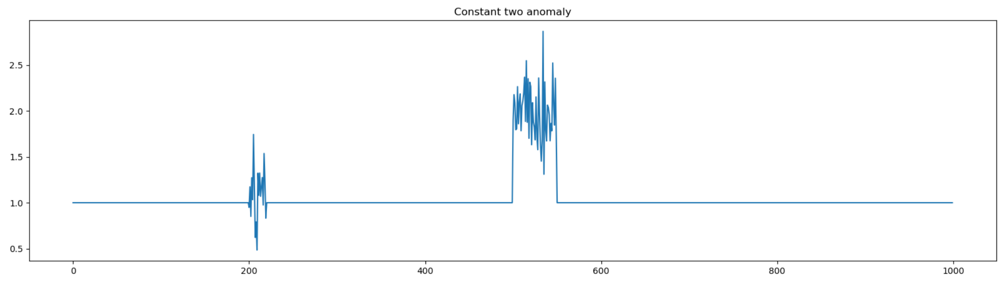

Figure 5: Constant Time Serie with two anomalies

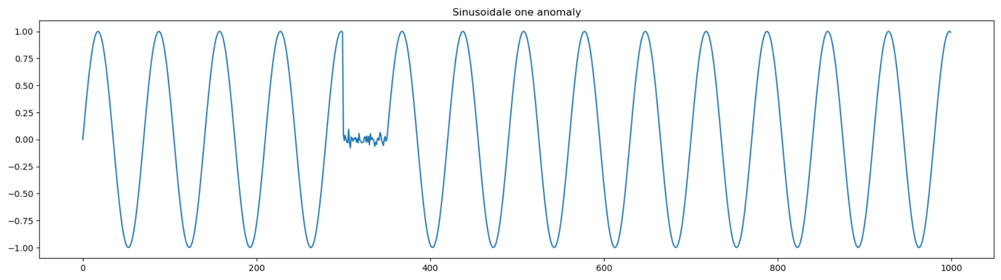

Figure 6: Sinusoidale Time Serie with one anomaly

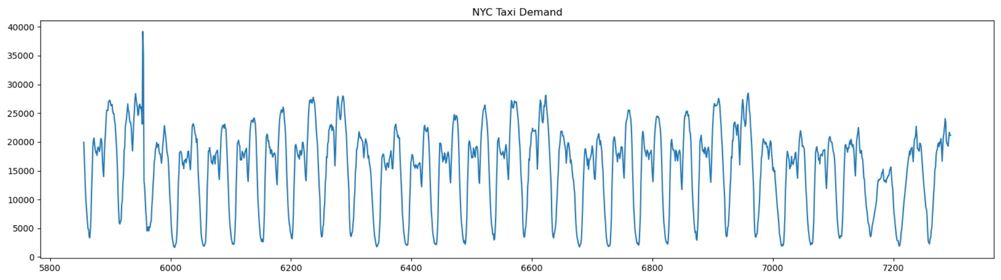

Figure 7: New York City Taxi Traffic on November 2018

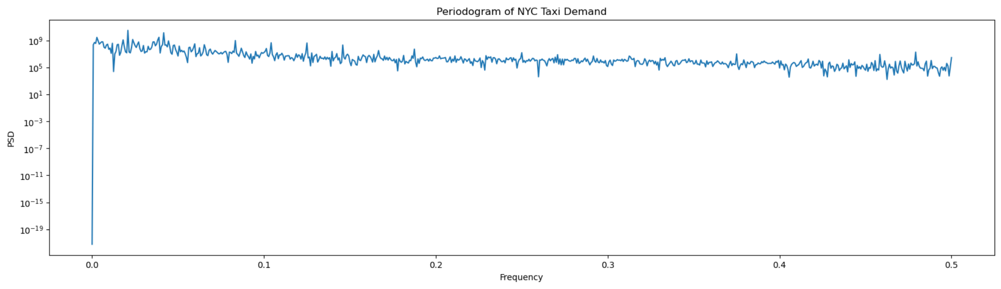

Figure 8: Periodogram of New York City Taxi Traffic on November 2018

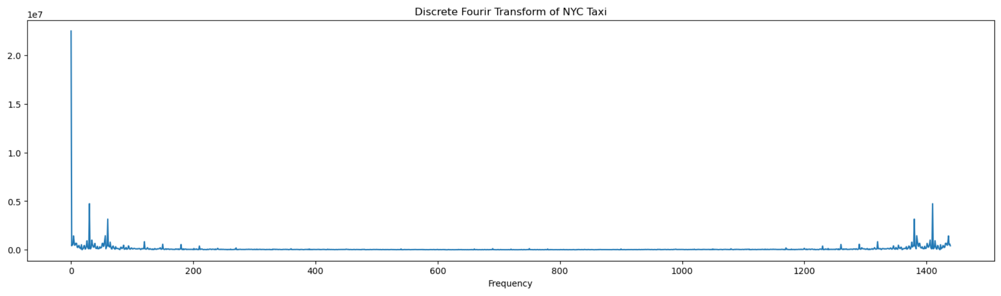

Figure 9: DFT of New York City Taxi Traffic on November 2018

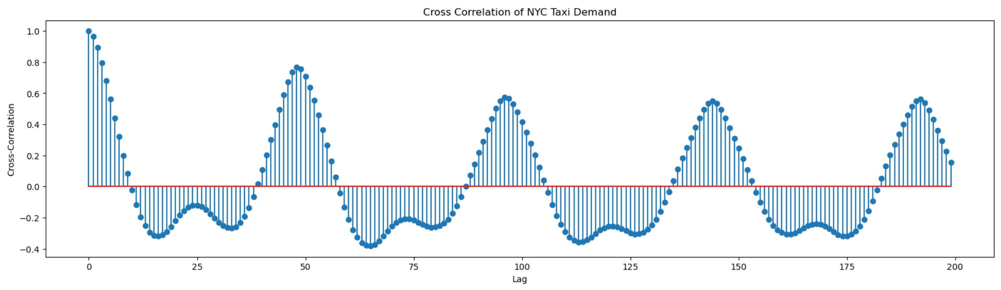

Figure 10: Cross Correlation on New York City Taxi Traffic on November 2018

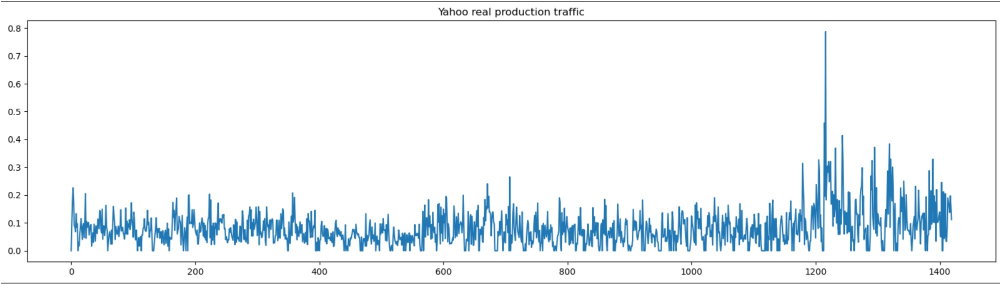

Figure 11: Yahoo Time Serie based on real production traffic to some of the Yahoo! properties

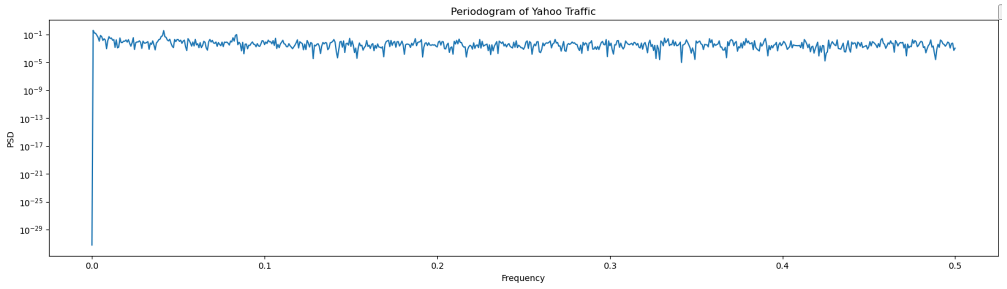

Figure 12: Periodogram of Yahoo time serie based on real production traffic to some of the Yahoo!

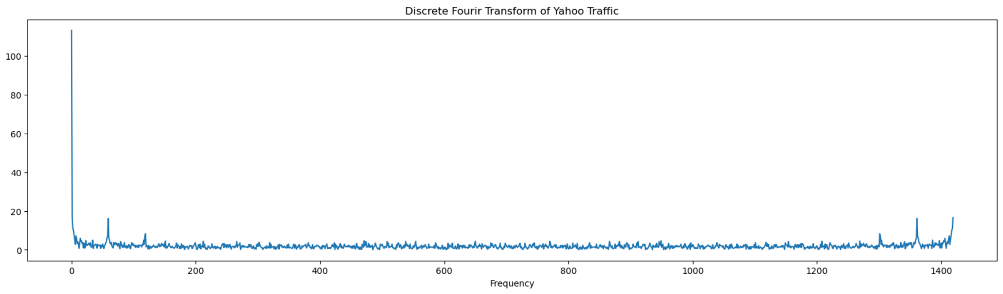

Figure 13: DFT of a Yahoo Time Serie based on real production traffic to some of the Yahoo!

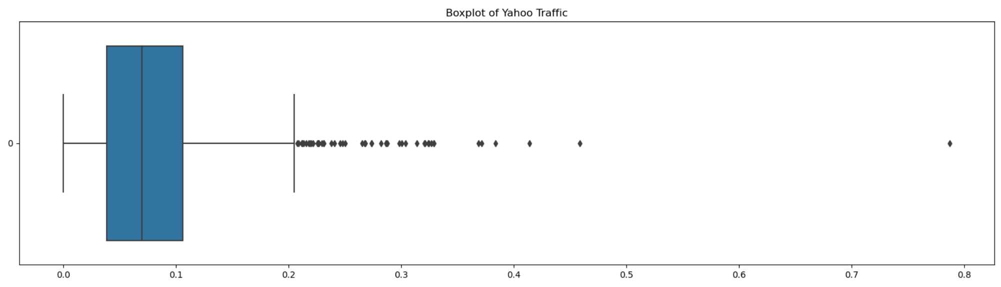

Figure 14: Boxplot of a Yahoo Time Serie based on real production traffic to some of the Yahoo!

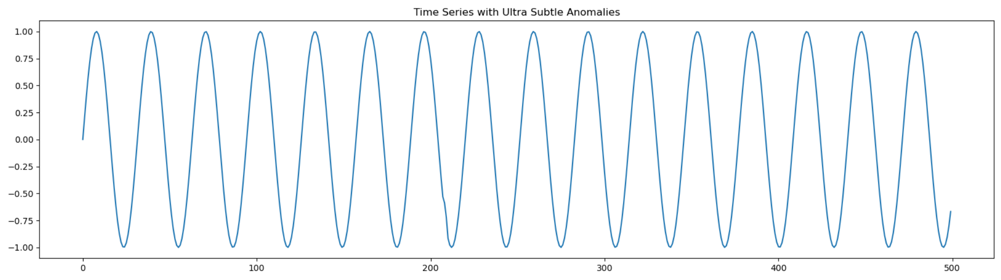

Figure 15: Time Serie with Ultra Subtle anomaly from position 208 to 210

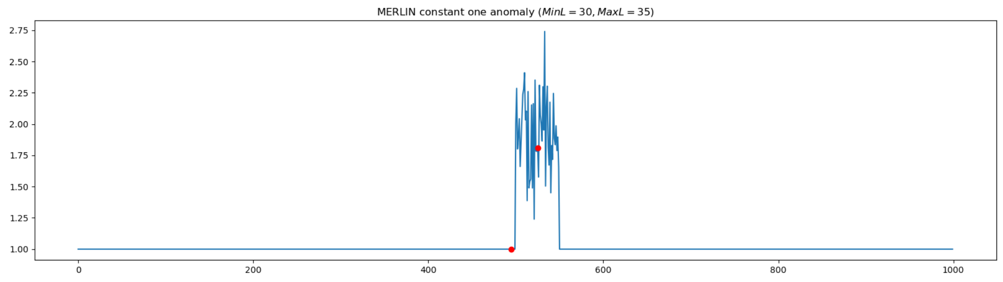

Figure 16: MERLIN output on a Constant Time Serie with one anomaly

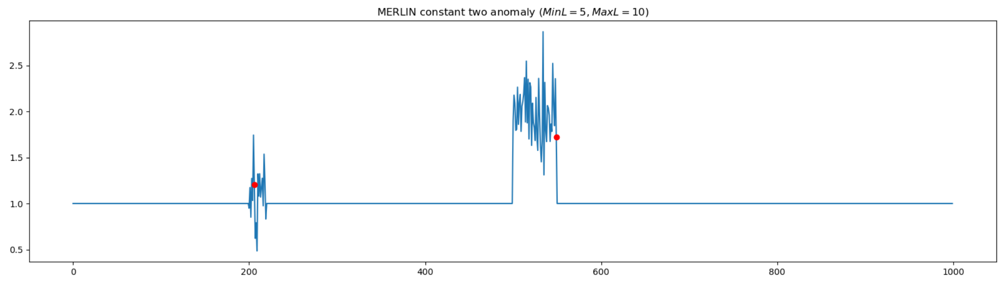

Figure 17: MERLIN output on a Constant Time Serie with two anomalies

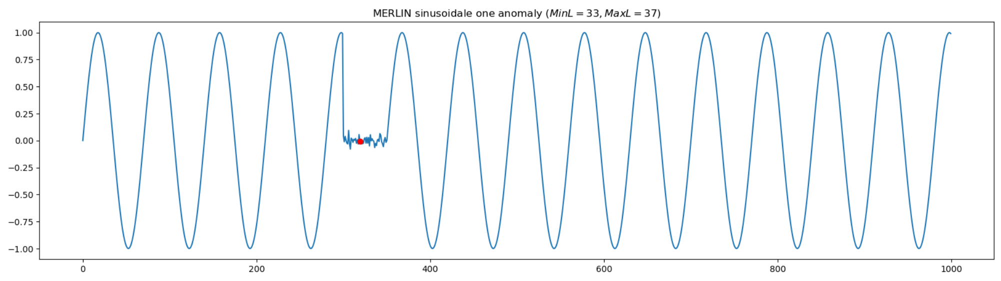

Figure 18: MERLIN output on a Sinusoidale Time Serie with one anomaly

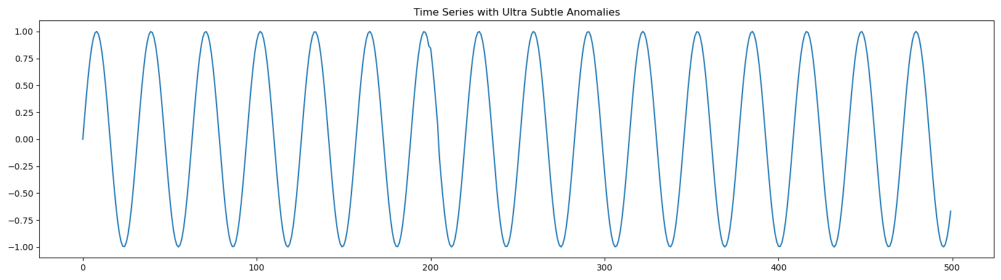

Figure 19: Time Serie with Ultra Subtle anomaly from position 200 to 205

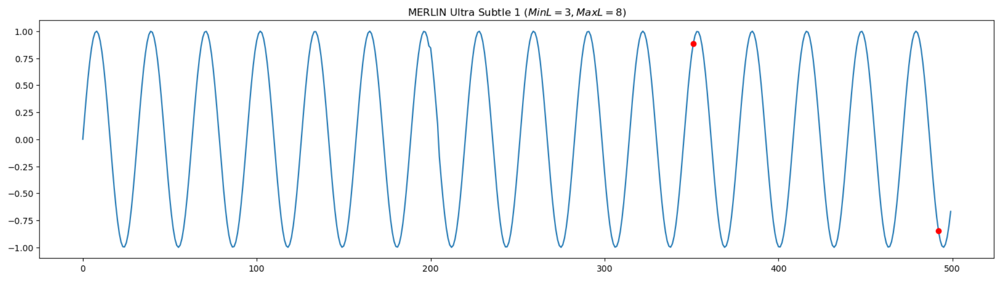

Figure 20: MERLIN output on a Time Serie with Ultra Subtle anomaly from position 208 to 210

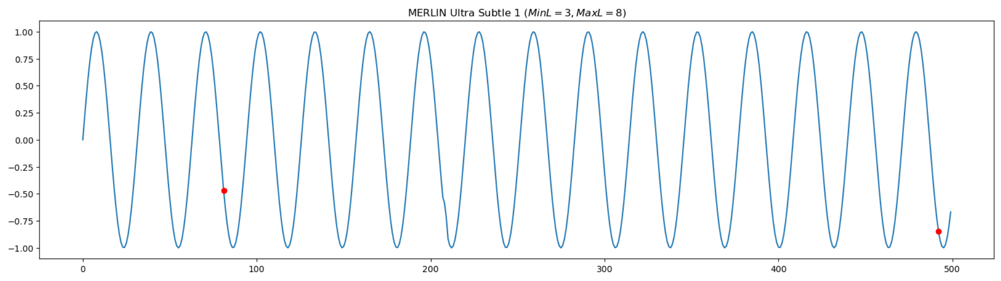

Figure 21: MERLIN output on a Time Serie with Ultra Subtle anomaly from position 200 to 205

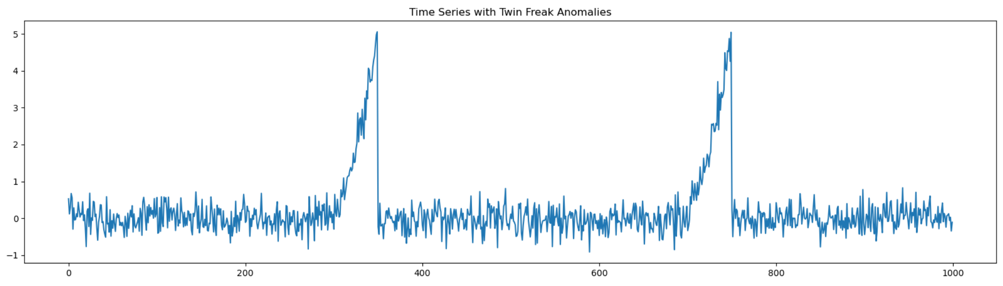

Figure 22: Time Serie with Twin Freak anomalies on [300, 350] and [700, 750] intervals

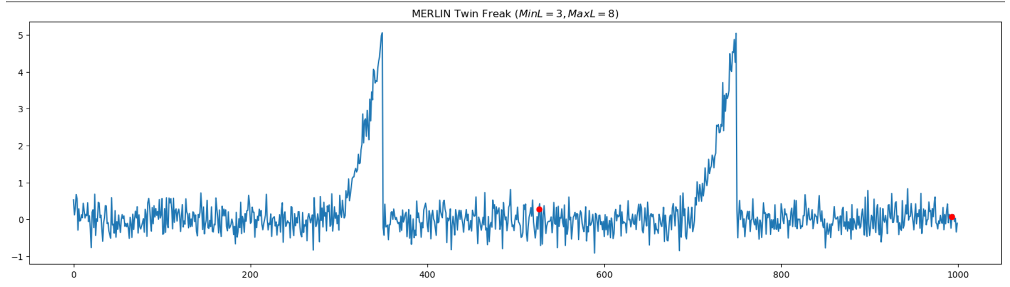

Figure 23: MERLIN output on a Time Serie with Twin Freak anomalies on [300, 350] and [700, 750]

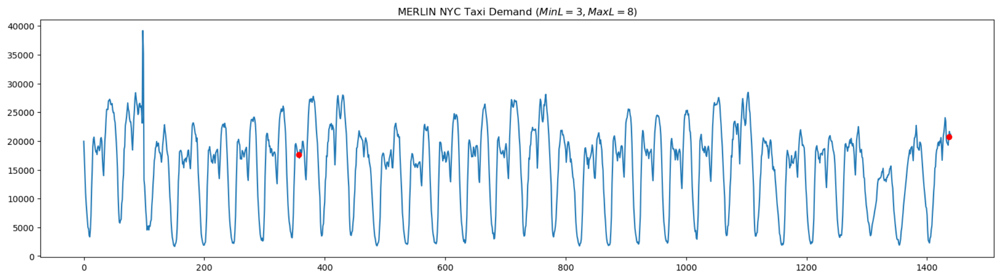

Figure 24: MERLIN output on New York City Taxi Traffic on November 2018

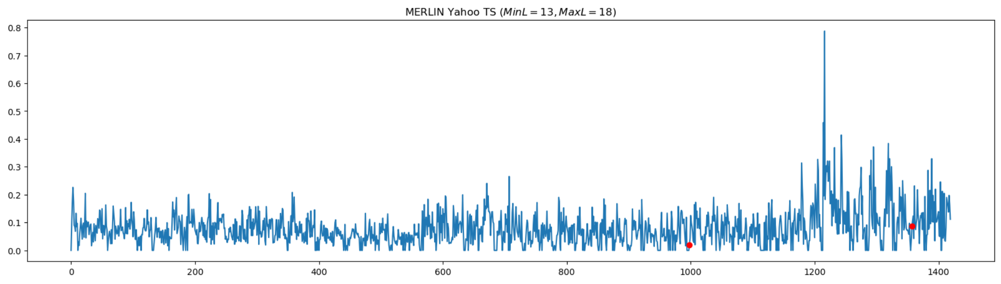

Figure 25: MERLIN output on Yahoo Time Serie based on real production traffic