Local Outlier Detection Methods for Energy Consumption Data

Prepared By:

Matthew Sahota

Enerva Energy Solutions Inc.

Toronto, ON

August, 2024

Abstract

Multiple local outlier detection strategies were tested to find anomalous energy consumption readings from power metres. Twitter Anomaly Detection's (TAD) local outlier detection method was used in various ways to detect unusual energy readings; more specifically, TAD's "anomaly_detect_ts" function was used on time series data to examine outliers. A time-of-week method (TOW) was implemented on the data in select iterations to check for outliers at specific periods of the week, and a Savitzky-Golay filter was also applied in select iterations to smooth the energy readings.

Four outlier detection methods were developed to examine the data, all of which involved using TAD's "anomaly_detect_ts" function to collect outliers: the first method implemented neither the TOW method nor the Savitzky-Golay filter, the second method implemented only the TOW method, the third method implemented only the Savitzky-Golay filter, and the fourth and final method implemented both methods. One hundred random readings were increased by a random multiple greater than two, and the outlier detection methods would be tested against the artificially increased data to see if they could detect the peaks.

The first method was found to have been the most effective at detecting outliers. The second and third methods were less effective in outlier detection, but the second method was more effective in avoiding unintentional outliers through the TOW method, while the third method had a higher accuracy at the cost of unintentional outlier detection around the peaks due to the Savitzky-Golay filter. Finally, the fourth method was found to be the least accurate method, but was effective for avoiding unintentional outliers.

Table of Contents

I: Description of Local Outlier Detection Methods

In this report, several local outlier detection methods for power meter readings will be detailed, as well as some of the techniques used in conjunction with the outlier detection methods. This report will also discuss the accuracy of each of the methods, along with special methods used in outlier detection, comparing the detected outliers from each of the methods with the expected outliers.

The main method used for outlier detection comes from Twitter's Anomaly Detection (TAD) scheme. TAD is a package written in Python that detects local outliers in data, where the response variable (in this case, energy readings) is used for outlier detection. More specifically, we used the “anomaly_detect_ts” function from the TAD package, which detects outliers on time-based data. Using this function, we can also alter some of the parameters; the main ones that we changed include “max_anoms” and “alpha”. The “max_anoms” parameter denotes the maximum proportion of anomalies that are detected by TAD’s “anomaly_detect_ts” function, and the “alpha” parameter acts similar to p-values in statistics; basically, “alpha” acts as a threshold to be compared with the test statistic for whether or not a singular datapoint is a local outlier, and if the test statistic is lower than “alpha”, then the point is deemed an outlier.

We also applied other techniques to check for local outliers. One such technique we applied was a time-of-the-week variable (TOW). The “TOW” variable assigns a value from 0 to 167 depending on the hour and day of the week. For example a value of 0 represents the zeroth hour of the first day of the week (Sunday at 12:00AM), a value of 2 represents the second hour

of the first day of the week (Sunday at 2:00AM), and a value of 24 represents the zeroth hour of the second day of the week (Monday at 12:00AM). Using this method, the data would be filtered from 0 to 167 and “anomaly_detect_ts” would be run on each filtered dataset. All the outliers from each iteration would be compiled in a dataset.

Another technique we applied was scaling the energy per day. With this method, we would account for days that had low or high overall recordings, such as holidays or special events, so we would be able to focus on any outliers that may arise in the specific day instead of incorrectly writing off most of, if not the entire day as an outlier due to the low/high overall recordings. The energy scaling method is applied by dividing each hourly energy recording by the total daily energy and multiplying the result by 24 (the number of hours per day) for scaling, and we would apply this method for every recorded day.

The final technique that we use on the data is a Savitzky-Golay filter for smoothing. With the Savitzky-Golay filter, unwanted outliers would be smoothed out, while intended outliers would not be unintentionally damped. Additional parameters would be required for this method, mainly the “frame” and “degree” variables. The “frame” variable denotes the size of the frame that will be used for smoothing the data, and the “degree” variable denotes the degree of polynomial used for smoothing. Compared to the moving average method, the Savitzky-Golay method retains peaks and troughs in the data while removing any minor artifacts in the data, whereas the moving average method runs the risk of diluting major features in the data, thus making local outlier detection more inaccurate or difficult.

II: Implementation of Outlier Detection and Visualization Methods

For our outlier detection scheme, we created two custom-made functions called “TAD_reg” and “TAD_sg”; the “TAD_sg” function applies a Savitzky-Golay filter on the energy data, whereas the “TAD_reg” data does not. In both functions, an option to analyze either the scaled energy or raw energy recordings and an option whether or not to use the “TOW” variable for analysis, and they implement TAD’s “anomaly_detect_ts” on the energy data for local outlier detection.

To test our outlier detection schemes, we would use two datasets: one dataset is metering data from the Peel District School Board (PDSB), and another is building data from Close Property Inc. (CPI). Most of our findings will be based on the CPI data. For our dataset to be accepted by the program, it will require certain special columns; the dataset will need a column for dates and times labelled “Date”, two energy columns either labelled “kWh” (for regular energy) or “Scaled Energy” (uses the energy scaling method described in the prior section), and a column for the time of week labelled “TOW” (uses the time-of-week labelling method described in the prior section). The dataset it returns retains all of the aforementioned columns, but adds a column called “Outlier” which determines whether or not TAD has identified the datapoint in question as an outlier. The “Outlier” column either returns “True” or “False”.

The “TAD_reg” and “TAD_sg” functions can also return .csv files that contain the returned dataset and save it onto the user’s local files. There is a field that can be filled called “file_name” that creates a .csv file with the inserted name attached to the file without the .csv

suffix. In short, the .csv suffix is added onto the name entered in “file_name” and saved onto the local computer.

To visualize our outliers, we use another custom-made function called “anom_plot”. The “anom_plot” function uses Plotly, a data-visualization package that is available in Python, to create an interactive plot. This interactive plot has the ability to display outliers, zoom into and scroll between specific time periods, inspect the datapoints one-by-one, and can also be exported as either an .html file or .png file. The file saving method for the .html and .png files is the same as the method used to save the .csv files.

IIIA: Verification of Outlier Detection Schemes: Visualizations

The dataset that we used to verify our methods was the CPI building emissions dataset. The emissions data was collected from May 4th, 2022 to April 30th, 2023 with all holidays being removed from the dataset. The relevant columns in the original dataset included the date and energy in kWh. The “TOW” and “Scaled Energy” columns were subsequently added into the dataset. With our edited dataset, we ran both the “TAD_reg” and “TAD_sg” functions on the data.

For both functions, we would run two iterations: the first iteration would not use the “TOW” column and the second iteration would use the “TOW” column. We would also analyze the dataset by “Scaled Energy”, the maximum proportion of anomalies at 0.05, and the alpha value for our outlier detection to 0.025. For the Savitzky-Golay versions of outlier detection, we would set the frame size to 10 and the degree of polynomial used for smoothing to 6.

For simplicity, we will allocate names to each one of our iterations. The iteration where neither the Savitzky-Golay filter nor the “TOW” column are applied will be called “Method 1”. The iteration where only TOW is applied will be called “Method 2”. The iteration where only the Savitzky-Golay filter is applied will be called “Method 3”, and finally, the iteration where both the Savitzky-Golay filter and TOW are applied will be called “Method 4”.

Next, we will compare the outliers detected from each iteration of our schemes with our expected outliers. To create our expected outliers, we will randomly increase 100 energy

recordings in our CPI dataset by a random multiplier from 2 to 5. All or most of the increased values should be identified as outliers when using each scheme. To randomly set our values, we would use a predefined randomization seed for both our randomized indices for multiplication and their respective multipliers. For a better picture of the accuracy of the outlier detection schemes, we will run two iterations using two randomization seeds for each method. Methods 1 and 2 will have two plots: one for the energy readings themselves (purple line), and another which indicates all the detected outliers (red points). For methods 3 and 4, we would also add the smoothed energy curve from applying the Savitzky-Golay filter in a separate plot, which will be indicated in green.

First, we will take a look at our iteration using Method 1. When we run this outlier detection scheme with the first seed, we can see that all 100 multiplied values are detected, along with local peaks and troughs that were not multiplied. The same result was reported for the second seed that we used. There are few outliers detected near the multiplied values, but this is mainly due to the values being lower than average.

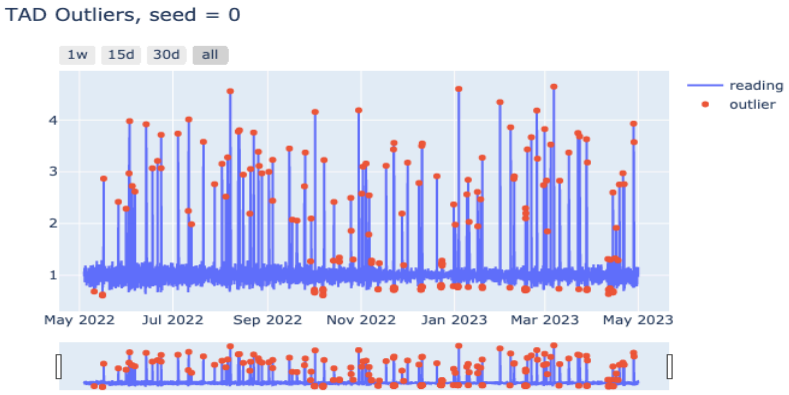

Figure 1: Outliers from Method 1, first seed

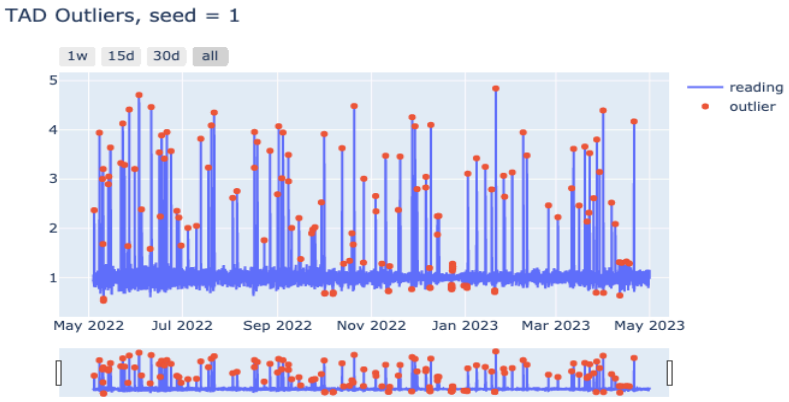

Figure 2: Outliers from Method 1, second seed

Next, we will look at Method 2. When we run this outlier detection scheme with the first seed, we can see that 95 out of 100 of the multiplied values are detected. With the second seed, we had the same detection rate. Similarly, some of the peaks and troughs are detected. We believe that the reason that some of the peaks were not detected was due to the TOW sorting method having peaks or higher values next to each other, meaning that TAD's anomaly detection method would ignore the peaks that were surrounded by the higher values. Fortunately, this was only limited to the lower peaks.

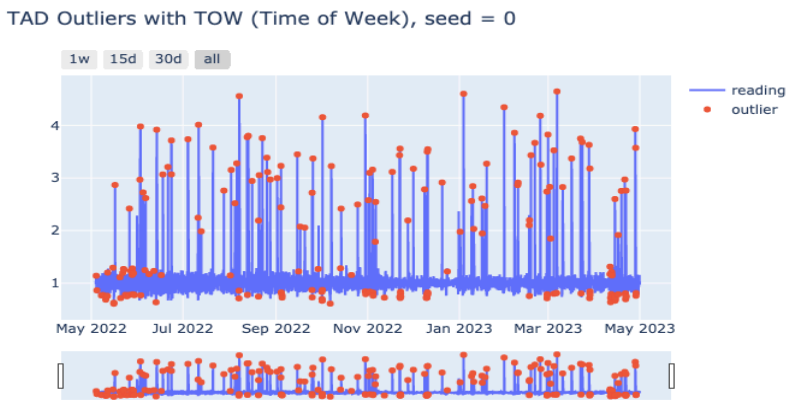

Figure 3: Outliers from Method 2, first seed

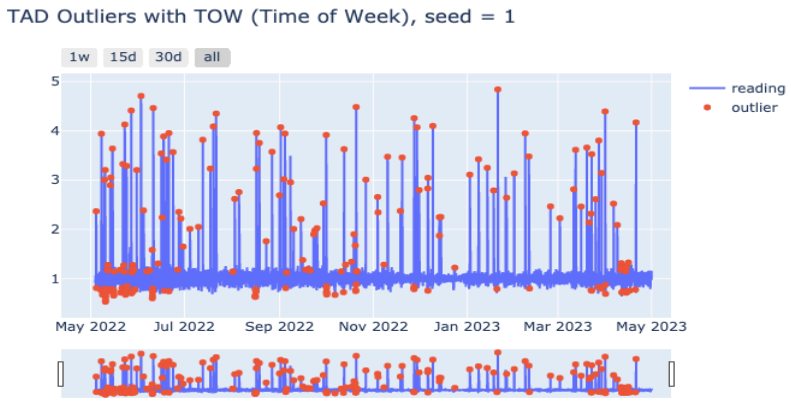

Figure 4: Outliers from Method 2, second seed

With the application of Method 3, we were able to identify all 100 multiplied values with our scheme using the first seed, but with the second seed, we could only identify 93 of the 100 multiplied values. Many of the values around the peaks were falsely identified as outliers due to the degree of polynomial used for the filter; the polynomial would overfit and create unwanted oscillations around the peaks, leading to misidentified outliers. The smoothed peaks would also be lower, but would have two similar values that make up the peak, which could explain some of the misidentification. However, both of these phenomena were only limited to the higher peaks.

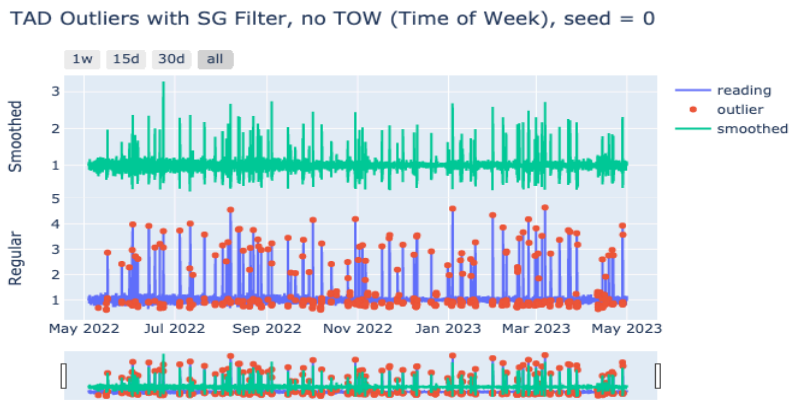

Figure 5: Outliers and smoothed energy curve from Method 3, first seed

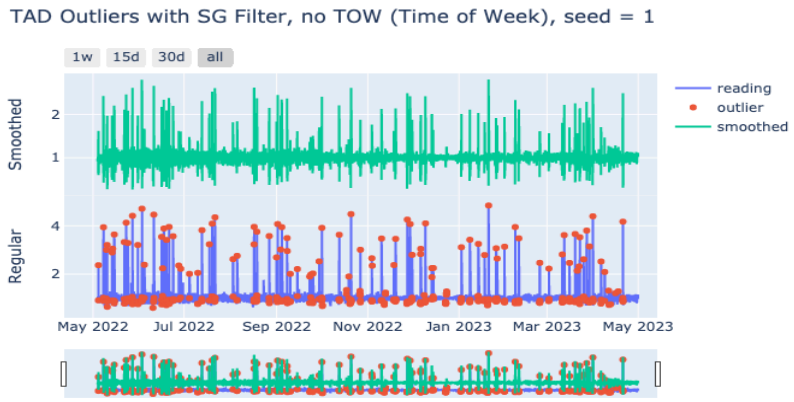

Figure 6: Outliers and smoothed energy curve from Method 3, second seed

Finally, using Method 4, our outlier detection scheme would only find 76 of the 100 multiplied values using the first seed. The scheme using the second seed found 78 out of the 100 multiplied values. While the misidentification of outliers around the peak was avoided, some of the peaks would not register as outliers due to the TOW method being implemented.

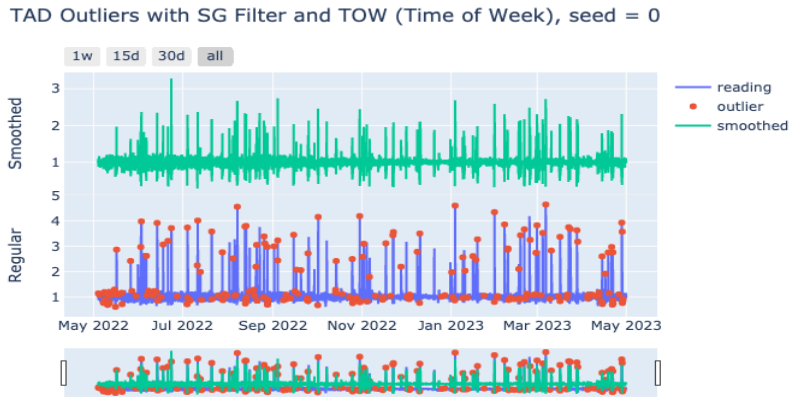

Figure 7: Outliers and smoothed energy curve from Method 4, first seed

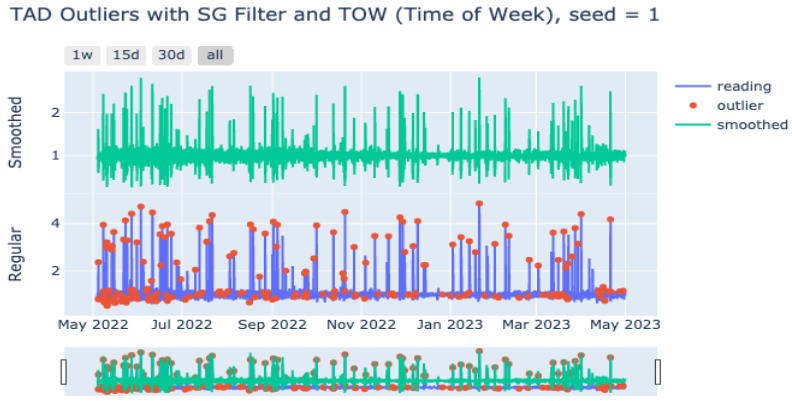

Figure 8: Outliers and smoothed energy curve from Method 4, second seed

IIIB: Verification of Outlier Detection Schemes: Larger Scale

In this section, we will examine the outlier detection methods using the same methods as described in the previous section, but we will run 25 iterations of each scheme with randomized seeds. Like before, 100 random values will be multiplied by a random value between 2 and 5. For convenience, there will be no graphs of the outliers provided. Instead, this section will be dedicated towards examining the accuracy of each scheme for detecting anomalies and how consistent the schemes are with outlier detection. Below is a table with the statistics (average accuracy, standard deviation, maximum, minimum, and median accuracy) of all the outlier schemes and their accuracies of detecting multiplied values as peaks amongst all 25 iterations. For convenience, we will shorten the term “Savitzky-Golay” to SG in our table.

Method Used	**Average Accuracy (%)**	**Standard Deviation (%)**	**Maximum Accuracy (%)**	**Minimum Accuracy (%)**	**Median Accuracy (%)**
No SG filter, no TOW (Method 1)	100	0	100	100	100
No SG filter, TOW (Method 2)	95.72	1.904	99	92	96
SG filter, no TOW (Method 3)	96.96	1.968	99	90	98
SG filter and TOW (Method 4)	77.96	4.286	86	69	79

We can see that the method that Method 1 is the most accurate, being able to detect all 100 peaks in all 25 iterations. With Method 2, we can see that it is slightly less accurate, but quite consistent in detecting peaks. Comparing Method 3 to Method 2, Method 3 is more accurate according to the mean and median measures, but Method 3 runs the risk of being more inaccurate at times as given by its larger range of values when compared to Method 2. Finally, Method 4 is far more inaccurate and inconsistent than all the other methods. Please note that these measurements do not take the detection of non-multiplied outliers or local behaviour into consideration; this is meant to illustrate the accuracy of the schemes in detecting excessively high energy consumption readings.

IV: Conclusion

Out of all the local outlier detection methods, the scheme using TAD's "anomaly_detect_ts" function with neither the TOW method nor Savitzky-Golay filter (Method 1) had the highest accuracy for outlier detection. The Savitzky-Golay filter is an excellent way to smooth the data for outlier detection while preserving times with anomalously high or low energy readings, but depending on the degree of polynomial used for the filter, it can run the risk of creating undesired anomalies in the smoothed data via overfitting. While the TOW method runs the risk of skimming over certain desired outliers, especially when compounded with the Savitzky-Golay filter, it is a good way to counteract accidental outlier detection around the peaks.

To rectify our outlier detection schemes, we could try using different window sizes and lower polynomial degrees for comparison for the Savitzky-Golay method to avoid overfitting and creating accidental outliers around the peaks. We could also use different smoothing methods on our data and subsequently run TAD's "anomaly_detect_ts" function on the smoothed data, along with running an accuracy test on the new smoothing methods. Finally, for the example where multiple iterations were run and analyzed, there should be some elaboration on the behaviour of the outlier detection schemes on the non-multiplied values to provide a better picture of how the schemes respond to more localized outliers.