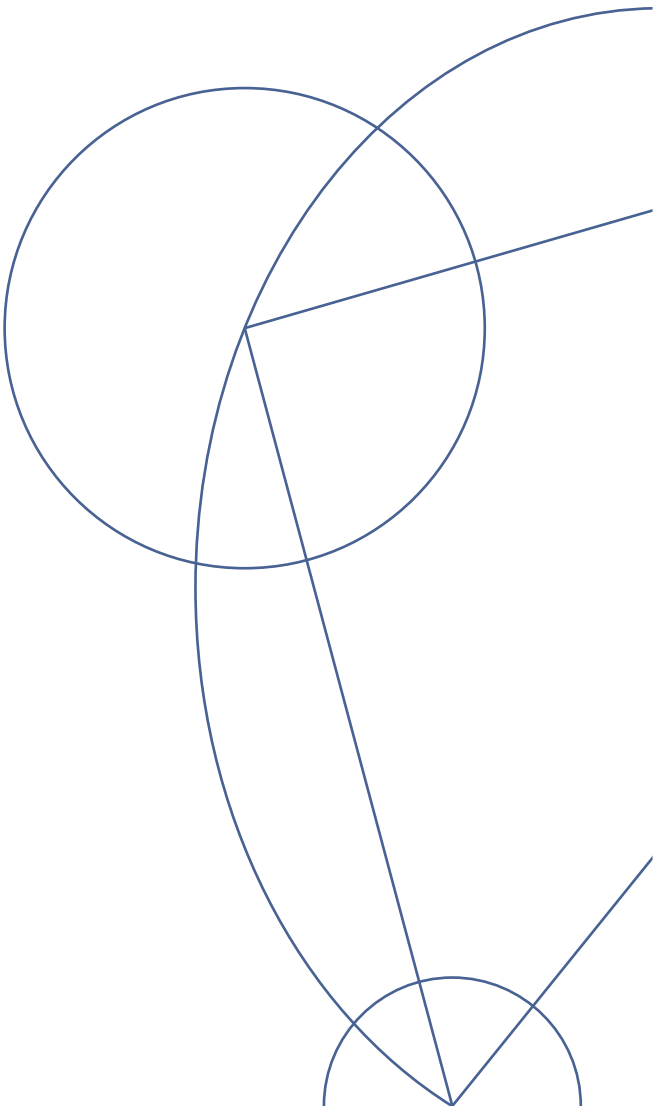
National Technical University of Athens
School of Electrical and Computer Engineering
Inter-Faculty Postgraduate Studies Programme
Data Science and Machine Learning

Extreme Driving Detection

The fourth assignment written in partial fulfillment of the requirements for the completion of the Data Driven Models in Engineering elective course of the Data Science & Machine Learning post-graduate studies programme.

Instructor
Prof. E. I. Vlahogianni

Written by
[redacted]



June 03, 2022

1 Problem Overview

Despite its somewhat more general title, the purpose of the present assignment is the detection of extreme turning movements for moving vehicles. The question of what constitutes an extreme turning event is in itself a significant question, the answer to which requires setting a baseline corresponding to regular driving movements and identifying deviations from this baseline as extreme events. Even then, quantifying such deviations, i.e. setting specific thresholds between regular and irregular events, is nontrivial. In addition, general rules cannot be easily extracted, since the conditions of each driving experiment cannot be reproduced even in a highly controlled environment, due to dependencies on factors such as geography or time period. For these reasons, the problem needs to be approached in the context of outlier/anomaly detection problems.

To this end, a dataset containing 103891 entries in a time series format is used. The records correspond to data acquired from smartphone sensors and more specifically the smartphone's accelerometer, gyroscope and GPS, for an unknown number of trips. Table 1 depicts the dataset's features, as well as some basic statistics in order to provide additional intuition with regards to the data.

	mean	median	standard deviation	minimum	maximum
`NewAccelX`	0.000	0.000	0.058	-1.322	1.224
`NewAccelY`	0.001	0.001	0.061	-2.163	1.033
`NewAccelZ`	0.010	0.011	0.063	-1.415	1.185
`NewRotRateX`	0.000	0.000	0.135	-9.102	7.147
`NewRotRateY`	0.000	0.000	0.129	-12.628	7.212
`NewRotRateZ`	0.000	0.000	0.147	-13.615	6.953
`locationSpeed`	16.333	13.080	12.691	0.000	47.660

Table 1: The dataset's features and basic statistics thereof.

The `NewAccelX/Y/Z` features are the smartphone's accelerometer's measurements of the vehicle's acceleration (measured in units of g) along each spatial axis. As far as the spatial axes x, y and z are concerned, they are defined as shown in Fig. 1, following the convention used in [1].

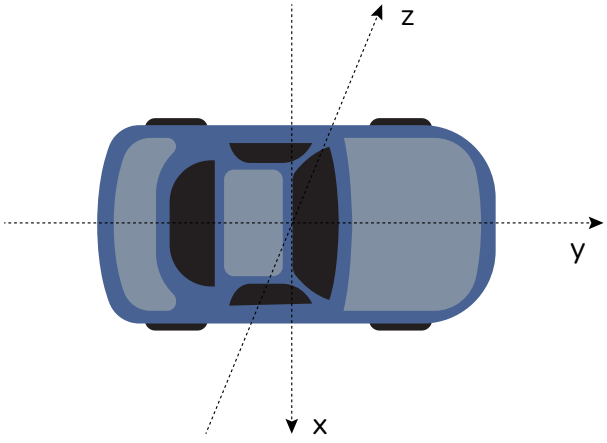


Figure 1: The spatial axes in relation to the car's orientation.

This means that the vehicle moves forward (backward) along the positive (negative) y-axis, turns right

(left) along the positive (negative) x-axis and travels uphill (downhill) along the positive (negative) z-axis. The smartphone's gyroscope measures the `NewRotRateX/Y/Z` features, which correspond to the angular velocity (measured in units of 1/s) along each axis[1]. Finally, using the smartphone's GPS system, the vehicle's speed is extrapolated and its values are given by the `locationSpeed` feature (measured in units of m/s).

An interesting as well as important thing to note is the reason why the prefix "`New`" is added before the `Accel` and `RotRate` features. As mentioned, all of the driving data are acquired from the sensors of a smartphone which is located inside the moving vehicle. However, the smartphone's orientation is not in general necessarily identical to the vehicle's orientation. Moreover, the smartphone is not static in the vehicle's reference frame, as it may be moving inside the vehicle (for example, if the driver picks it up to use it for some reason). Therefore, a dynamic calibration process is performed in 10-second windows, so that the smartphone's axes are reoriented to be in agreement with the vehicle's and thus the data acquired from the smartphone's sensors are transformed to correspond to the reoriented axes.

2 Feature Selection

The time series data provided are in general crucial for analytics tasks and the correlation between the various features can be used to extract physical information about the vehicle's movements, as well as to perform sanity checks on whether the data are erroneous or not. One such example of sanity check is shown in Fig. 2, where the vehicle's acceleration along the y-axis, as well as its speed, are shown for a 300-second time window of the given time series.

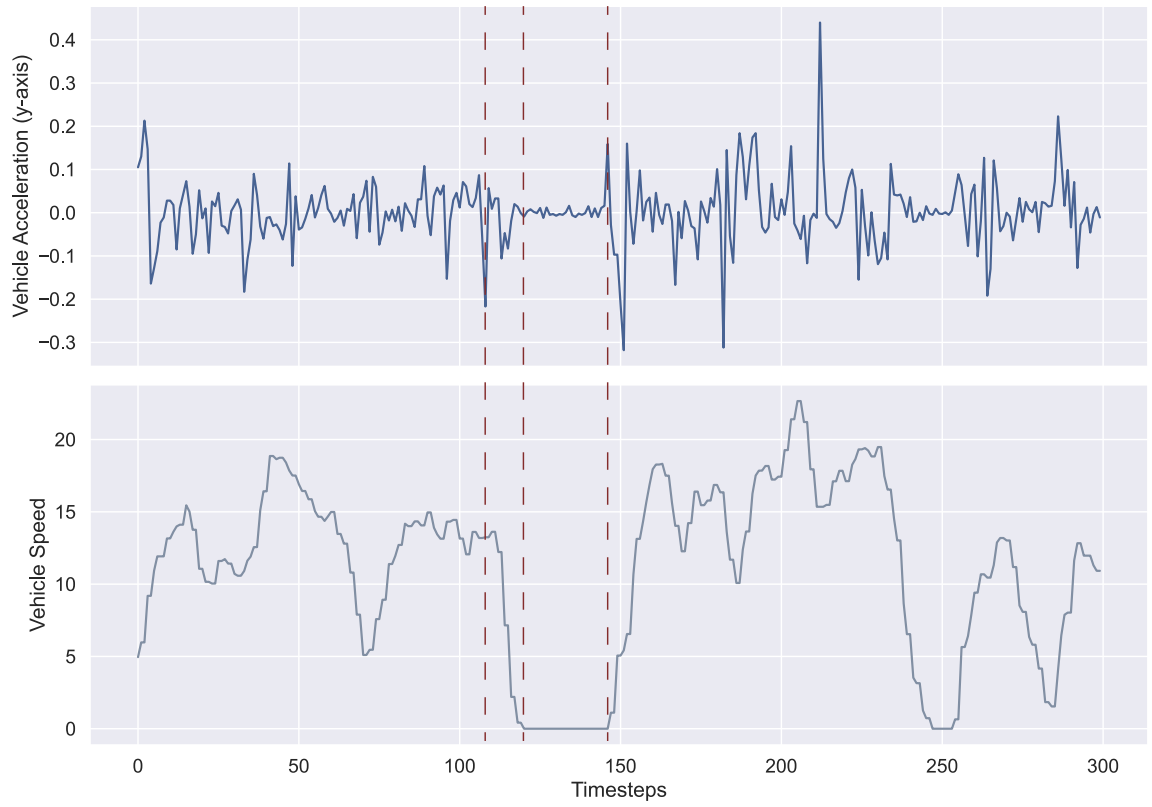


Figure 2: Vehicle acceleration along the y-axis and vehicle speed for a time window of 300 seconds.

[1] Be reminded that the angular velocity's direction is determined using the right hand rule. This means that, for example, a left/right turning movement of the car is due to an angular velocity along the z-axis.

The theoretically expected consistency is present and one can verify this by focusing on the red dashed vertical lines in Fig. 2. The first line corresponds to $t = 108$, when the y-axis acceleration reaches a local minimum, corresponding to a braking move. Right after this, the vehicle's speed seems to be decreasing. The second line corresponds to $t = 120$ and from this point on, for a window of ≈ 20 seconds, the vehicle's y-axis acceleration is approximately zero. Therefore, the vehicle's speed remains constant during this time window. Finally, the third line corresponds to $t = 146$, when the y-axis acceleration reaches a local maximum, corresponding to a speeding move. Right after this, the vehicle's speed seems to be increasing.

Despite the usage all of these features have in gaining physical insight into the problem and analyzing its various aspects or extracting information regarding extreme events in general, this assignment focuses solely on extreme turning movements. If the labels of each event were available, the entire dataset could be analyzed in order to quantify each feature's impact on the classification of a turning movement as "extreme", as well as to determine how different features are correlated during these extreme events. In fact, even a black-box approach would be able to yield satisfactory results in extreme turning movements identification, due to the huge success that deep learning models have demonstrated during the past few years. However, each record's label is not available, which means that the problem has to be approached in an unsupervised manner. For the same reason, it is preferred to work only with the features that seem to be the most relevant with the problem at hand, from a physical standpoint.

The only features that are relevant to turning movements are the x-axis acceleration, `NewAccelX`, and the z-axis angular velocity, `NewRotRateZ` (recall that the vehicle's turning movements correspond to rotations about the z-axis). Nonetheless, it's not the velocity itself that determines the extremity of a turning movement, but its temporal derivative, which is the quantity that shows how rapidly it tends to change[2]. For example, when the driver wants to make a turn, he is expected to increase the vehicle's acceleration on the x-axis and keep it at a steady value while making the turn. In this duration, the angular velocity's magnitude is expected to increase, yet this is a normal and not an extreme turning event. On the other hand, if for some reason the driver momentarily rotates the wheel (for example to bypass a small object), the z-axis angular velocity will not reach a high value, however a small peak will appear in the x-axis acceleration's time series. That being said, all features apart from `NewAccelX` are omitted from the analysis performed for the purposes of the present assignment and the problem of outlier detection is focused on x-axis acceleration outliers.

3 Outlier Detection

Outlier values of the x-axis acceleration are detected as follows: A counter is initialized at 0 for each record. Then, three different unsupervised outlier detection algorithms are employed: a traditional Extrema Detection algorithm, the Elliptic Envelope algorithm and the cutting-edge Empirical Cumulative Distribution Functions (ECOD) algorithm, published in 2022 [2]. Each record that is deemed to be an outlier has its counter increased by 1 every time an algorithm identifies it as such. In the end, the outliers are chosen by a majority voting scheme, where records with a counter equal to 2 or higher are considered to be anomalies. In the following, the principles of each algorithm as well as its results are discussed in separate sub-sections. A comparison between the algorithms and the final predictions are given at the end of the present report.

[2] Strictly speaking, the temporal derivative of the z-axis angular velocity is the z-axis angular acceleration. However, its magnitude is directly connected to the magnitude of the linear acceleration along the x-axis due to geometric arguments.

3.1 Extrema Detection

The first algorithm employed is a traditional extrema (i.e. "peaks" and "valleys") detection algorithm. Similarly to what is done in [1], the x-axis acceleration time series is scanned for peaks and valleys that meet specific criteria as far as their minimum height and distance from neighboring extrema are concerned. Note that extrema detection is a process which is fundamentaly different from a simple threshold check. For a given threshold $t > 0$, the check $|\texttt{NewAccelX}| > t$ would return all records with a "relatively" (depending on the threshold) high acceleration (i.e. with a large magnitude). In this case, a turning maneuver on a long turn, for example, which requires a constant and high value for the angular acceleration, would correspond to a series of outliers, even though it does not actually correspond to an extreme turning event.

An important point for this algorithm is determining the minimum height of a peak or a valley. The studied dataset corresponds to driving data from a human-operated vehicle, which means that the corresponding time series is expected to consist of a lot of small peaks and valleys: even the most experienced driver cannot control the vehicle's acceleration and either maintain it at a constant value or smoothly change it when required. As a result, setting a small height threshold would cause the algorithm to identify the vast majority of the dataset's records as outliers. On the other hand, setting a large height threshold would lead to the algorithm missing a lot of events that were annotated as extreme. Studying the relevant bibliography (Table 1 of [1]), it is concluded that a suitable threshold range in units of g for the x-axis acceleration is 0.16 to 0.26 (and -0.26 to -0.16, for the opposite direction). Fig. 3 shows the x-axis acceleration's time series in a randomly chosen 100-second time window where the peaks and valleys have been identified with a height threshold of 0.16 (left) and 0.26 (right).

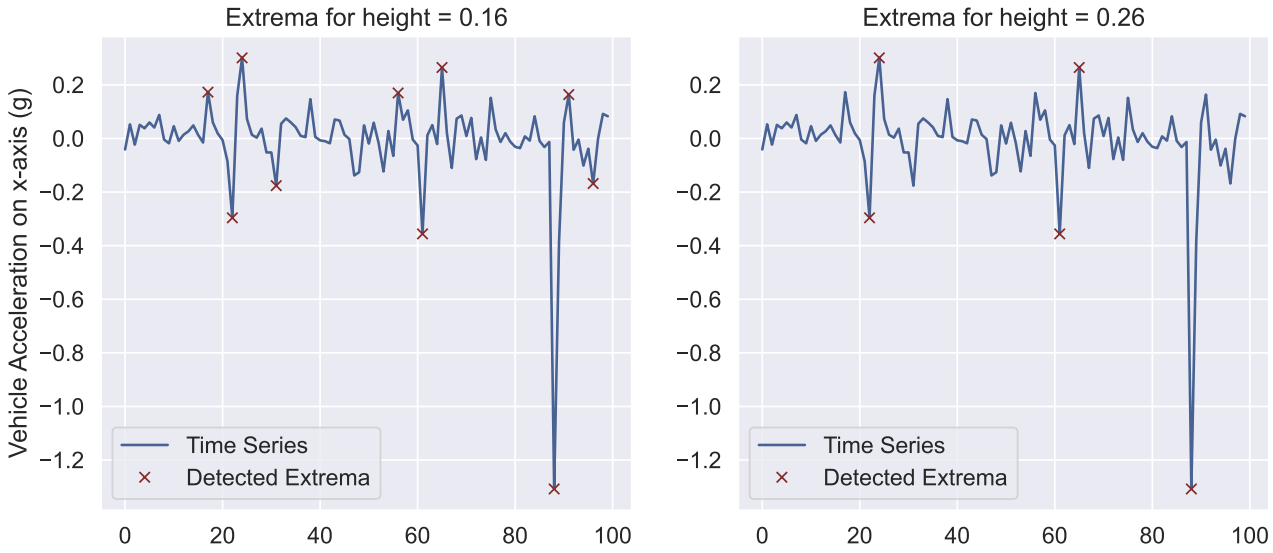


Figure 3: Extrema detection for a randomly chosen 100-second time window for minimum height equal to 0.16 (left) and 0.26 (right).

The number of detected peaks and valleys is 10 for the left graph and 5 for the right graph. It becomes evident that choosing the upper bound for the height threshold leads to missing some peaks and valleys which are quite likely to correspond to extreme turning events. Note at this point that one of the main reasons for using the extrema detection algorithm before the other two algorithms is because the other two algorithms require a contamination parameter as input. The contamination parameter, $\mathscr{C}$, is simply the percentage ($0 < \mathscr{C} < 0.5$) of the dataset's records which correspond to outliers. Of course, this information cannot be known beforehand. As a result, the extrema detection algorithm isn't only employed in order to identify outliers, but also to yield a rough estimate for the contamination

parameter, so that it can be used in what follows. For this purpose, it is safer to use the lower bound for the height threshold, thus not limiting the other two algorithms to the detection of only a fraction of the outliers, in the case that their number is underestimated by the extrema detection algorithm.

Taking these into consideration, an even lower height threshold is chosen (0.14 instead of 0.16) for the purposes of the present assignment. The reasoning behind this further reduction of the threshold is the following: the detection of extreme turning events is mostly useful for penalizing artificial intelligence agents or human drivers during their learning process. For this reason, the correct detection of an extreme turning event cannot be equally important to the correct detection of a regular turning event, simply because misidentifying an extreme event as normal is worse than misidentifying a normal event as extreme (the former leads to more reckless, while the latter leads to more careful drivers). This is simply a reiteration of the famous saying "better safe than sorry". Fig. 4 depicts a 300-second time window of the x-axis acceleration's time series, where all peaks and valleys with a minimum height of 0.14 have been detected.

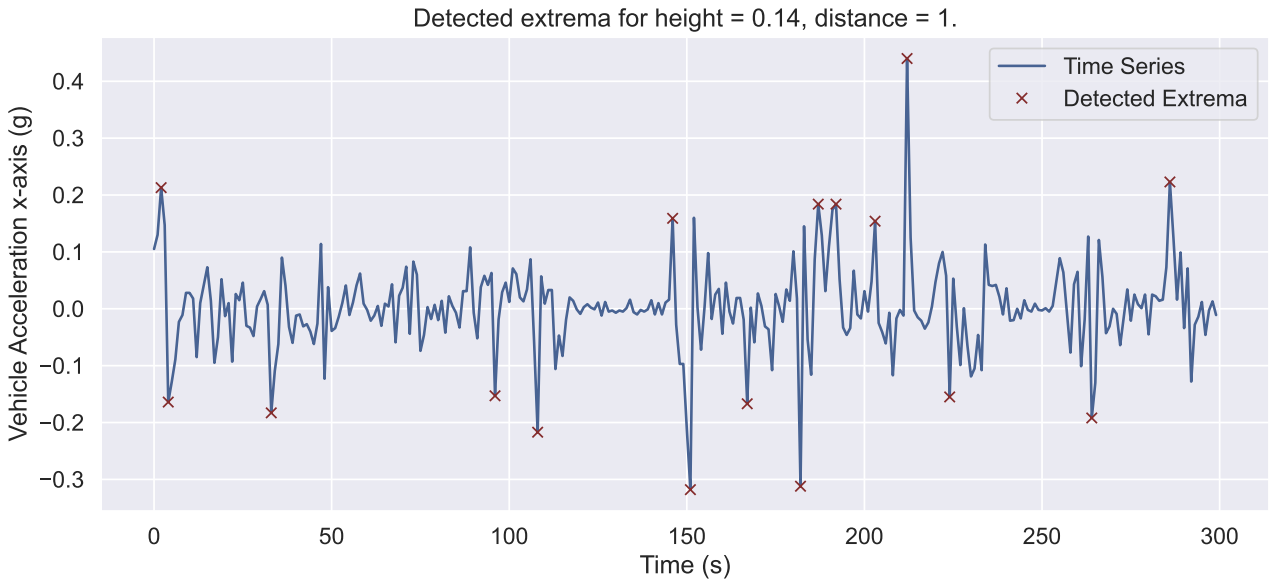


Figure 4: Extrema detection for a 300-second time window with minimum height equal to 0.14.

The algorithm is implemented using the Python language and utilizing `SciPy`'s `find_peaks` method. The detailed code can be found in the Notebook or the Script that accompanies this report. The total number of outliers identified by the Extrema Detection algorithm is 2490.

3.2 Elliptic Envelope

Moving on, the Elliptic Envelope is the choice for the second algorithm. In general, one common way of performing outlier detection is to assume that the regular data come from a known distribution. From this assumption, one may try to define the "shape" of the data and thus be able to define outlying observations as observations which stand far enough from the assumed shape. The Elliptic Envelope method assumes that the records follow a Gaussian distribution (hence the name) and therefore estimates the inlier location and covariance of the records in a robust way (i.e. without being influenced by outliers). The Mahalanobis distances obtained from this estimate are used to derive a measure of how close a given record is to being classified as an outlier.

Of course, the assumption that the x-axis acceleration's values follow a Gaussian distribution is a strong one. It is known that the dataset's contents are not generated from Gaussian distributions, however if their actual distribution can be approximated by a Gaussian, then the Elliptic Envelope

algorithm will be able to make valid predictions as far as possible outliers are concerned. A formal way of investigating whether the records' distribution can be approximated by a Gaussian would be to generate a model distribution and calculate its Kullback-Leibler divergence from a Gaussian or perform the Kolmogorov-Smirnov test. Since this goes well beyond the purposes of the present assignment, it will suffice to perform a visual check. Fig. 5 depicts the histogram of the x-axis acceleration's values on the same axes with a Gaussian distribution whose mean and standard deviation are equal to the records' mean and standard deviation (i.e. $7.89 \cdot 10^{-5}$ and 0.058, respectively).

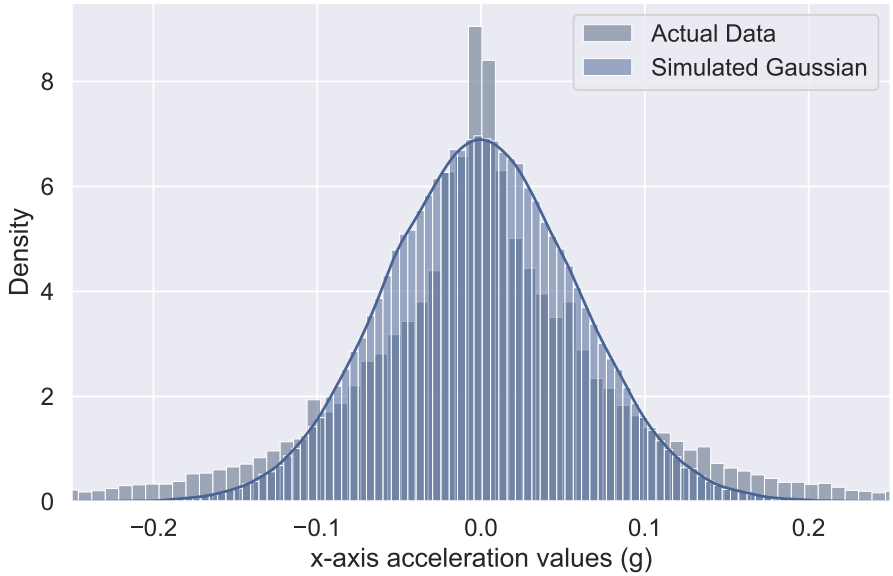


Figure 5: Histograms for the x-axis acceleration's values (grey) and randomly generated values from a Gaussian distribution with mean and standard deviation equal to the mean and standard deviation of the dataset's records (blue).

It becomes evident that, while the two distributions seem to have significant differences (for example, the tails of the actual records' distribution are longer than the Gaussian's), there is also a significant overlap. As a result, one may conclude that the use of the Elliptic Envelope algorithm is at least reasonable, if not well-advised.

As previously discussed, a required input for the Elliptic Envelope algorithm is the dataset's contamination parameter, $\mathscr{C}$. The extrema detection procedure yielded a rough estimate for the total number of outliers present in the dataset; the 2490 outliers found correspond to 2.4% of the dataset's records. Therefore, a good choice for the contamination parameter is $\mathscr{C} = 0.026$, which is 1.1 times the estimated percentage. Once again, the algorithm is implemented using the Python language and this time `scikit-learn`'s `EllipticEnvelope` module is utilized. The detailed code can be found in the Notebook or the Script that accompanies this report. The total number of outliers identified by the Elliptic Envelope algorithm is 2720.

3.3 Empirical Cumulative Distribution Functions

The final algorithm employed to detect outliers is the unsupervised outlier detection algorithm using empirical cumulative distribution functions (ECOD). This algorithm is inspired by the fact that outliers are often the "rare events" that appear in the tails of a distribution. In a nutshell, the first step of ECOD is the estimation of the underlying distribution of the input data in a nonparametric fashion by computing the empirical cumulative distribution per dimension of the data. This differentiates it from the previously studied Elliptic Envelope, as ECOD makes no assumptions regarding the distribution

from which the studied records are generated. ECOD then uses these empirical distributions to estimate tail probabilities per dimension for each data point. Finally, it computes an outlier score for each data point by aggregating estimated tail probabilities across dimensions. This means that ECOD does not only identify outliers within a given dataset, but also generates scores which quantify the certainty of each outlier identification.

Similarly to the Elliptic Envelope, the ECOD algorithm requires the contamination parameter as its input. Again, the value $\mathscr{C} = 0.026$ is chosen. The algorithm is implemented using the Python language and utilizing `PyOD`'s `ECOD` module. The detailed code can be found in the Notebook or the Script that accompanies this report. The total number of outliers identified by the ECOD algorithm is 2725. It is worth noting that the lowest confidence value assigned to an outlier identification by the ECOD algorithm is equal to 4.334, which corresponds to a high confidence level.

4 Final Results

After running all three algorithms using the x-axis acceleration's values in order to identify outliers, it is possible to find the percentage of agreement between different algorithms by comparing their predictions. These "agreement scores" can be seen in the heatmap of Fig. 6. Note that ED and EE correspond to Extrema Detection and Elliptic Envelope, respectively.

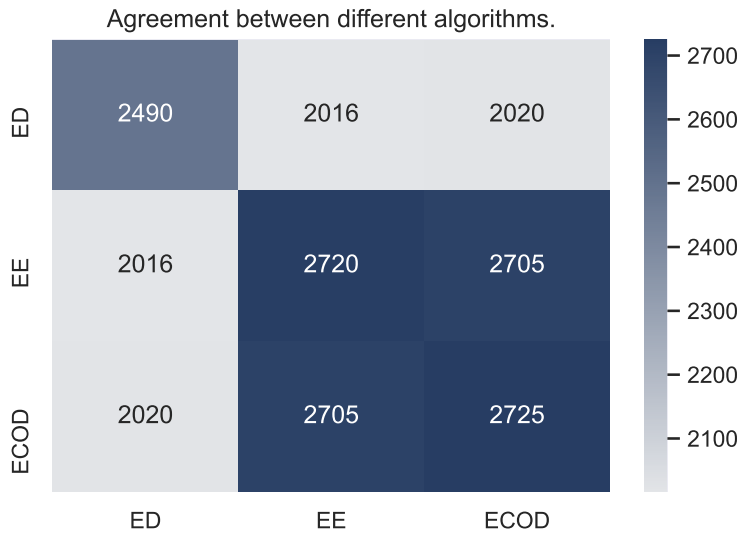


Figure 6: Heatmap of the agreement between different algorithms' predictions.

Apparently, the Elliptic Envelope algorithm's predictions are almost in complete agreement with the corresponding predictions by ECOD. Their agreement score is a stunning 99.45%, once again indicating that the assumption that the data can be approximated by a Gaussian distribution was not far from the truth. The agreement scores between the Extrema Detection and the Elliptic Envelope and ECOD are 80.96% and 81.12%, respectively, which are still rather high.

Taking into account the majority voting scheme implemented in order to make the final predictions for the dataset's outliers, the total number of outliers identified by this ensemble of algorithms is 2735, a number that corresponds to 2.63% of the dataset. The exact predictions can be found in the `outliers.csv` file that accompanies the present report, where the 0 annotation corresponds to a regular record and a 1 annotation corresponds to an outlier.

References

[1] E. I. Vlahogianni and E. N. Barmpounakis, "Driving analytics using smartphones: Algorithms, comparisons and challenges," *Transportation Research Part C*, vol. 79, pp. 196–206, 2017.

[2] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen, "ECOD: Unsupervised outlier detection using empirical cumulative distribution functions," *IEEE Transactions on Knowledge and Data Engineering*, 2022.