Vasconcelos et al. Journal of Internet Services and Applications (2017) 8:13
DOI 10.1186/s13174-017-0065-0

Journal of Internet Services and Applications

RESEARCH Open Access

Smartphone-based outlier detection: a complex event processing approach for driving behavior detection

Igor Vasconcelos[1,2*], Rafael Oliveira Vasconcelos[1,2], Bruno Olivieri[1], Marcos Roriz[1], Markus Endler[1] and Methanias Colaço Junior[3]

Abstract

The majority of fatal car crashes are caused by reckless driving. With the sophistication of vehicle instrumentation, reckless maneuvers, such as abrupt turns, acceleration, and deceleration, can now be accurately detected by analyzing data related to the driver-vehicle interactions. Such analysis usually requires very specific in-vehicle hardware and infrastructure sensors (e.g. loop detectors and radars), which can be costly. Hence, in this paper, we investigated if off-the-shelf smartphones can be used to online detect and classify the driver's behavior in near real-time. To do so, we first modeled and performed an intrinsic evaluation to assess the performance of three outlier detection algorithms formulated as a data stream processing network which receives as input and processes data streams of smartphone and vehicle sensors. Next, we implemented a novel scoring mechanism based on online outlier detection to quantitatively evaluate drivers' maneuvers as either cautious or reckless. Thus, we adapted a data mining mechanism which takes into account a sensor's data rates and power to determine driver behavior in the scoring process. Finally, as the intrinsic evaluation does not necessarily reveal how well an algorithm will perform in a real-world scenario, we evaluated the algorithm that achieved the best result in a real-world case study to assess drivers' driving behavior. Our results indicate that the algorithm performs quickly and accurately; the algorithm classifies driver behavior with 95.45% accuracy. Moreover, such results are obtained within 100 milliseconds of processing time on average.

Keywords: Online driving behavior detection, Online outlier detection, In-Vehicle sensing, Smartphone

1 Introduction

Driving is an everyday task that has become a necessity for modern society, primarily in large cities. According to Owsley [1], in some cases driving is associated with quality of life. However, reckless driving has caused a growing number of traffic accidents. Reckless driving is defined as driving behavior defined by Tasca [2] as behavior that "deliberately increases the risk of collision and is motivated by impatience, annoyance, hostility or an attempt to save time." According to a global safety report traffic by the World Health Organization [3], 1.24 million people die each year in traffic deaths and an estimated 20–50 million are involved in non-fatal accidents. In addition, it is estimated that $518 billion dollars is spent on the consequences of accidents [4]. However, studies indicate that drivers tend to be relatively safer when monitored or when feedback on their maneuvers is provided [5, 6].

Nevertheless, current intelligent transportation systems (ITS) continue to rely on an infrastructure composed of static sensors and cameras installed on roads, making it difficult to collect, aggregate, and analyze data, especially in real-time [7–9]. Moreover, due to the high cost of installation and maintenance, ITS are often restricted to particular roads or neighborhoods [9–11]. By contrast, the Internet of Things (IoT) aims to pervasively connect billions [12] of things or smart objects such as vehicles, sensors, actuators, and smartphones. The IoT poses a more complicated challenge in multi-stream environments where multiple data streams compete for available memory and processing resources, especially in

* Correspondence: ivasconcelos@inf.puc-rio.br
[1]Department of Informatics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rua Marques de São Vicente 225, Gávea, Office 503, Rio de Janeiro-RJ, Brazil
[2]Department of Informatics, University Tiradentes (UNIT), Aracaju, Brazil
Full list of author information is available at the end of the article

resource-constrained systems such as sensors and mobile devices [13]. For this reason, several IoT solutions—including tools for safe driving analysis—have been designed and developed to perform data processing in a cloud environment, due to a cloud's virtually unlimited capabilities and resources in terms of storage and processing power. For instance, Quintero, Lopez, and Cuervo [14] proposed an approach to classify driving behavior by collecting data onboard the vehicle and forwarding them for processing in the cloud. Leng and Zhao [15] and He, Yan and Xu [16] proposed a cloud computing middleware for the so-called *Internet of Vehicles*. However, due to the large volume of data generated by some mobile smart device (i.e., sensors in vehicles), it has become impractical and costly to transmit all data to a cloud [17]. Among the many challenges of the IoT, such as heterogeneity and interoperability, the authors of [18] also highlighted the following: (i) middleware for communication with the cloud [19]; (ii) technologies supporting dynamic configuration [20]; and (iii) robust, real-time mechanisms for data filtering and data mining to cope with the large amount of raw data provided by smart devices and reduce the amount of data transmitted to the cloud.

Therefore, mining and processing mobile data streams are a key technique for real-time data analysis [21]. In this scenario, a mobile device is used to receive and analyze a vehicle's sensor's data stream such as speed and rotation readings rather than sending these data streams for analysis in the cloud. Furthermore, with this approach, the mobile device can also use its own sensor's data streams—accelerometer and gyroscope readings—to enrich the vehicle's data stream and analyze driver behavior. We argue that analysis of driving behavior using the received data streams can be mapped to the outlier detection problem which refers to the problem of finding data patterns that do not conform to (or deviate sufficiently from) expected behaviors [22, 23], e.g., sudden lane changes and hard breaks. Although outlier detection has been widely studied, little research has been done on detecting outliers in dynamic data streams on mobile platforms. Given that *dynamic* means a non-stationary context, the pattern discovery algorithm must adapt to the available data streams. For instance, approaches for modeling and recognizing driving behavior assume a fixed set of data extracted from onboard vehicle sensors. However, in a real-world scenario, modeled sensors may not be available in a specific type/make of vehicle. For instance, most automobile manufacturers have introduced, in addition to the original, standard onboard diagnostics (OBD-II) [24], another set of sensor data such as steering wheel angle, breaks, airbag triggers, [24, 25] and stability control. Thus, the format of vehicular sensor reading and the available data depends on either the manufacturer or the vehicle [26]. The outlier detection algorithm must precisely perform within the limited computational resources of mobile smart devices, in contrast to the virtually unlimited cloud environment. Moreover, the outlier detection algorithm must operate online, continually processing data items as they are delivered in the input buffer. In contrast, offline algorithms require the complete and finite set of input data for processing. Offline algorithms usually buffer all the input data as a batch before processing them. Consequently, in offline algorithms the detection of a pattern is delayed until the buffer is filled [27].

With the goal of enabling online detection of reckless driving behavior, this paper investigates online outlier detection algorithms for dynamic data streams on mobile devices with limited computational resources. It is

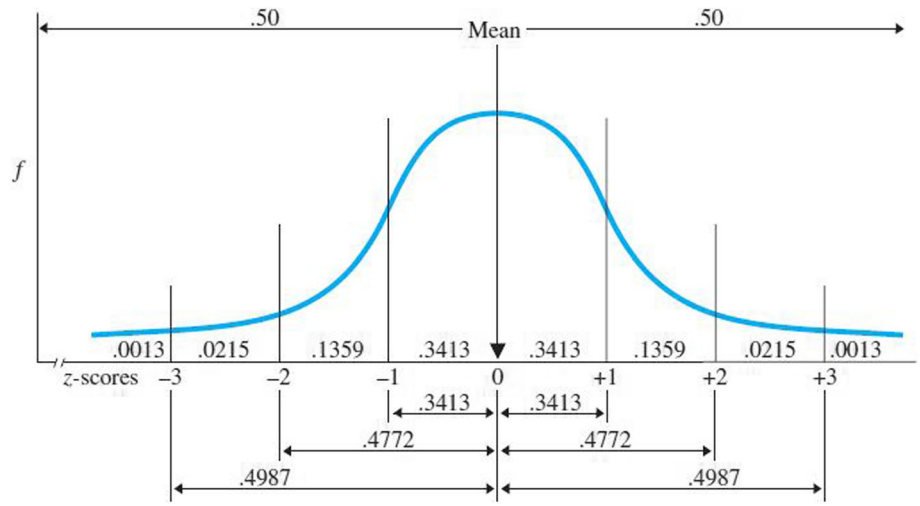

Fig. 1 Z-score frequency [65]

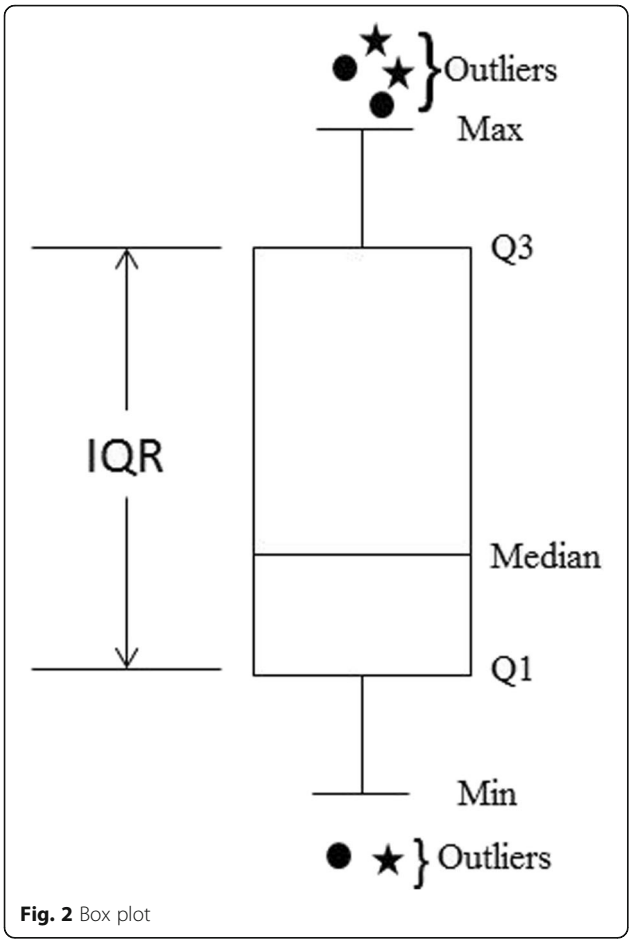

Fig. 2 Box plot

important to highlight that these algorithms need to adapt to the available vehicle's sensors or mobile device's sensors. Our study adapts and compares three classical offline outlier detection algorithms to perform online data stream processing using the Complex Event Processing (CEP) [28, 29] paradigm. Thus, we propose and evaluate a lightweight approach for detecting outliers through CEP in dynamic data streams generated from mobile devices' sensors and the vehicle's onboard sensors. Such an approach should (i) perform online data stream mining to identify outliers while respecting the intrinsic computational and storage limitations of mobile devices, and (ii) be able to adapt to the available input data (i.e., sensors) streams. Specifically, the main contributions of this research include a mechanism to perform online outlier detection over multiple data streams in a resource-constrained device, and a prototype application that implements these requirements to classify driving behavior. A case study was carried out in a real-world scenario in Brazil with the aim of validating the prototype. The results indicate a fast (i.e. ~100 milliseconds of processing time) and accurate (i.e., 95.45% accurate) performance.

1.1 Problem statement

Outlier detection refers to finding patterns in data that do not conform to expected behavior [23]. An outlier commonly contains useful information about abnormal characteristics of the system or entity [30]. Outlier detection is a multidisciplinary field of study that investigates how to extract patterns from large datasets covering a broad spectrum of techniques, such as statistical inference, machine learning, and data mining [23, 31, 32]. Moreover, it has been extensively applied in a variety of applications, for instance, detection of financial fraud, network intrusion, failures in critical systems, sensor faults in sensor networks, speech recognition, and traffic monitoring [23, 32].

Despite extensive research on outlier detection, most existing methods require the entire dataset (or at least a large portion of it) to detect outliers [23, 33], and are designed to perform offline analysis [22] for a large volume of data. These algorithms have no or only restricted

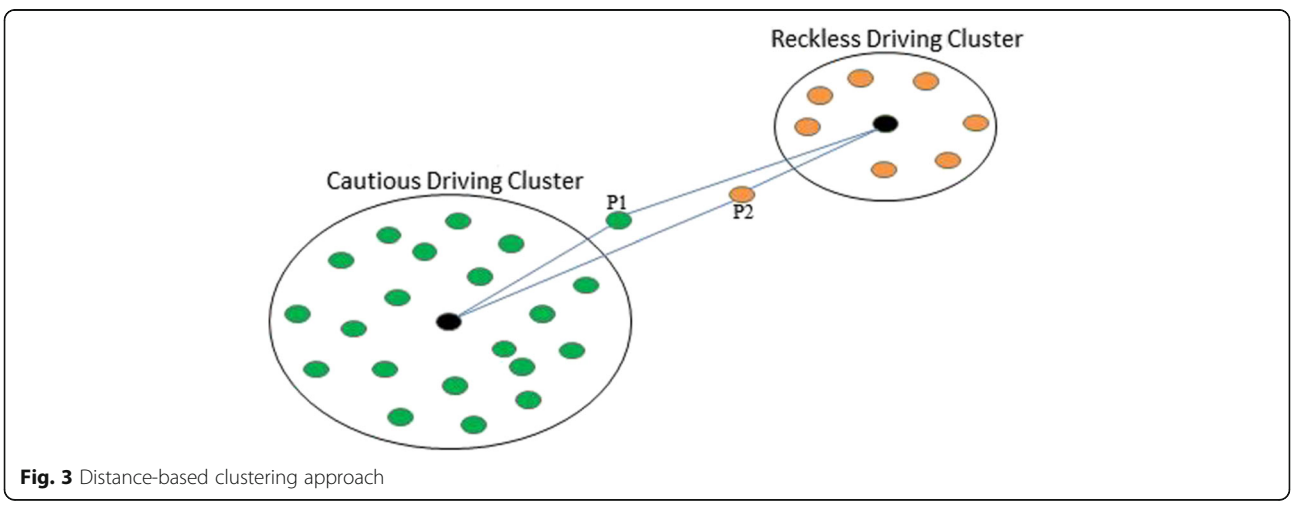

Fig. 3 Distance-based clustering approach

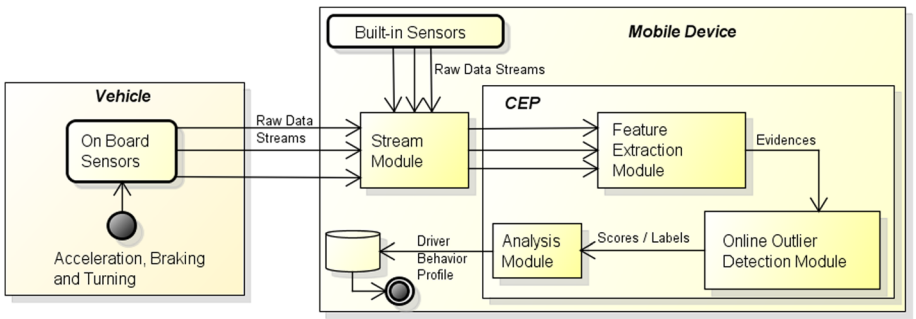

Fig. 4 An overview of application architecture

support for real-time data analysis requirements - such as meeting timing constraints - and have difficulty adapting to continuous non-stationary data [34]. Online data analytics is particularly significant for applications that need real-time analysis of continuous data streams. This analysis needs to be performed in a manner enabling it to run with partial data and with the limited computational resources of mobile devices. It is challenging to adapt existing outlier detection solutions to mobile data streams since they were designed and developed for cloud environments with abundant available resources, in which they normally compute with the complete data input [35]. Furthermore, such solutions are treated as a "black boxes" [34], wherein changes can scarcely be made to internal algorithms. The data mining community has conducted studies addressing outlier detection in data streams; however, these proposals mainly solve difficulties that are not the focus of the current paper, such as clustering [36, 37], mining frequent patterns [38, 39], data analysis [40, 41], and query processing [42] in the cloud environment. The aim of this paper is to investigate online outlier detection over multiple and dynamic data streams. Moreover, the outlier detection is performed in a smart object with limited processing and storage capabilities (unlike cloud environments) within a mobile scenario in which there is no guarantee that all data will always be available. Based on a systematic review of approaches conducted in this paper, we claim that, to the best of our knowledge, the literature offers no solutions to this problem.

A data stream is a continuous and online sequence of unbounded items for which it is not possible to control the order of the produced and processed data [43]. One characteristic of the data stream is its dynamic nature [44], meaning the properties of data instances may evolve or change over time. Additionally, context changes in a mobile scenario. For instance, it may modify the available data stream's inputs, and therefore, it is necessary for an algorithm to adapt to the stream's evolution [45]. Recently, online outlier detection within data streams has attracted attention in many constrained emerging applications, such as mobile crowd sensing, mobile activity recognition, ITS, and mobile healthcare [21]. In these applications, multiple and continuous streams are generated by mobile sensors and these streams need to be analyzed in real time. Based on this scenario, it can be seen that the adaptation of strategies for classical outlier detection algorithms to operate with mobile data streams, thus enabling their operation on mobile devices to be efficient, is a challenging research task [21, 35, 46]. This is because (i) outlier detection for data streams is restricted to the partial set of events within a time window; (ii) random access on the set is not possible; (iii) the algorithm must adapt to hardware resources and available sensor data; and (iv) patterns must be discovered within a single pass over the data stream. Moreover, Chandola, Banerjee and Kumar [23] highlight additional factors that make the outlier detection problem more difficult in such a situation:

- Defining a region encompassing all possible normal driving behavior is difficult. Furthermore, the threshold between normal and abnormal driving behavior is not often precise. Thus, an outlier observation lying close to the boundary may be normal or abnormal.

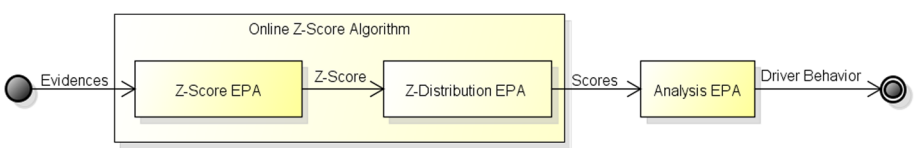

Fig. 5 Online Z-score event processing network (EPN)

```
1. INSERT INTO Z_SCORE_EVENT
2. SELECT (rawValue - avg(rawValue))/stddev(rawValue) AS z_score
3. FROM Evidence.win:time(windowLength sec)
4. GROUP BY dimension
```

Fig. 6 Z-score computation expressed in event processing language (EPL)

- The lack of availability of datasets for training and validation is often a major problem. The exact notion of an outlier differs depending on the application domain. An outlier detection formulation is generated by both the nature of data and the availability of labeled data.

1.2 Motivating scenario

Intelligent transportation systems have received increasing attention from academia, industry, and governments, and have been considered the next technological change in individuals' daily lives [47]. Automobile manufacturers, in an attempt to overcome the aforementioned ITS limitations, have developed products that help drivers, called Advanced Driver Assistance Systems (ADAS). These systems [48, 49] obtain vehicle data from sensors or embedded devices (e.g., cameras and stability control sensors) for the prevention and detection of collisions (e.g., crash sensors can activate airbags), assisted driving, and the generation of offline driving reports. The advantages of ADAS include the rare occurrence of false positives [50] when accessing sensors and devices that are embedded in the vehicle. However, the key impediment of ADAS lies in the fact that they are typically available only in new and high-standard vehicles that have prohibitive prices for most drivers [50–52], even in developed countries. Furthermore, the installation of ADAS in older car models is either impossible or inordinately expensive. Finally, when ADAS become obsolete, upgrading or changing to a newer, more efficient system is a difficult task [50], and exorbitant for most drivers.

By contrast, studies have proposed the use of smartphones to understand and evaluate a driver's behavior [5, 26, 49, 53–57]. The choice of using a smartphone is made due to its affordability and wide adoption, sufficient storage capacity and processing power, as well as its equipment with a variety of sensors. Moreover, a smartphone can act as a processing hub that receives and analyzes data from different vehicle sensors. For instance, with Bluetooth, a smartphone is able to connect and receive data from multiple in-vehicle sensors using the OBD-II standard, simultaneously receiving and processing speed and accelerometer data streams. Furthermore, in-vehicle sensors' data streams can be combined with smartphone-embedded sensors (such as direction and location) to further enrich the analysis. Finally, smartphones allow for the development of ubiquitous and loosely connected systems that provide rich data for the analysis of driving behavior.

Currently, approaches that evaluate driving behavior in general use models and techniques (e.g., Neural Networks, Fuzzy Theory, and Hidden Markov models) with good accuracy [58]. However, they were not designed for data stream processing [40], and according to Lin et al. and Wang, Xi, and Chen [58, 59], have low processing performances, require a long training phase, artificial assumptions, or prior knowledge to formulate rules. Moreover, since these approaches are statics (i.e., non-adaptable), they have difficulty quickly and accurately recognizing parameters [58], for instance, neural networks have subjective methods for adjusting the topology (number of layers and neurons) and require a fixed number of input parameters. A final drawback, highlighted by Wang [59], is that these approaches are

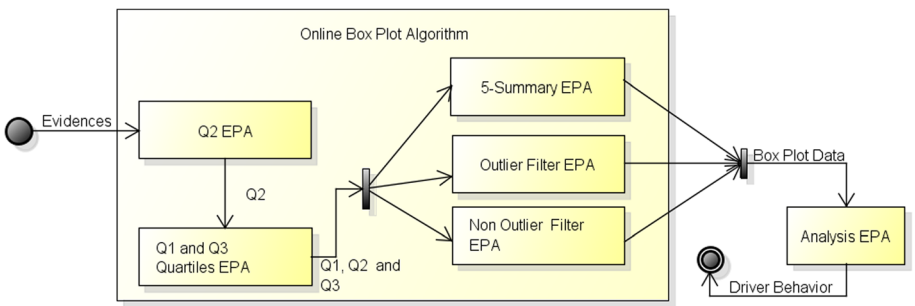

Fig. 7 Online box plot event processing network (EPN)

```
INSERT INTO BOX_PLOT_Q2_EVENT
SELECT median(rawValue) AS q2,
       rawValue  AS rawValue
FROM  Evidence.win:time(windowLength sec)
GROUP BY dimension
```

Fig. 8 Q2 calculation

"black boxes" with little ability to identify causal relationships making it impossible to understand physical behaviors. However, driving conditions (which are influenced by the state of the driver, traffic, and weather conditions) are dynamic and as such, all information that a technique or algorithm needs as an input will not always be available onboard. Thus, we believe that an assessment of a driver's driving behavior would benefit from an online outlier detection approach in dynamic mobile data streams.

1.3 Assumptions

This paper considers the following assumptions.

- Most data instances in data stream are normal. Only a small portion of the data consists of outliers [60, 61].
- Outliers are statistically different from normal data [2, 62].
- Battery power consumption is not a critical requirement because in a vehicle a smartphone can be charged easily when necessary.

However, it is important to note that the first two assumptions complement themselves. Considering only the first assumption, some outliers may have behaviors similar to normal data. The second assumption, however, states that outliers are a set of data with behaviors that differ from normal data.

The remainder of the paper is organized as follows. Section 2 presents an overview of the key concepts and system modeling used throughout this work. Section 3 details the proposed approach to online outlier detection for mobile, dynamic data stream. Section 4 highlights definitions and planning of the case study. Section 5 summarizes the main results of the assessment conducted to evaluate the proposal. Section 6 discusses related work. Finally, Section 7 reviews and discusses the central ideas presented in this paper and proposes paths for future work on the subject.

2 Fundamentals

This section presents the main concepts of complex event processing, as well as outlier detection algorithms.

2.1 Complex event processing

Complex event processing (CEP) is a set of techniques and tools that provides an in-memory processing model for an asynchronous data stream in real time (i.e., minimum delay) for online detection of situations of interest [28]. Complex event processing offers [28]: (i) *situation awareness* through the use of continuous queries that correlate data from different sensors data streams; (ii) *context awareness* by subdividing data streams into different views, such as temporal windows or key partitions; and (iii) *flexibility*, since it can specify events at any time, that is, the specification of events can be dynamically changed while a system is running (i.e., on-the-fly).

The CEP central concept is a declarative event processing language (EPL) to express event processing rules (continuous queries and patterns). These rules are based on the event-condition-action triad, and use operators (e.g., logic, counting, temporal, causal, and spatial) on input events, searching for correlations, exceptional conditions, and the occurrence of patterns. The central task of CEP is to provide mechanisms for *event pattern matching*, i.e., from hundreds or even thousands of events, to identify significant patterns in the application domain [63]. Event processing and pattern detection are made by so-called event processing agents (EPAs) that process an event's stream. Essentially, an EPA filter separates, aggregates, transforms, and synthesizes new complex events from simple events. A reckless maneuver (e.g., rapidly turning at a high speed) is an example of a complex event, in so far as it is based on the composition of primitive events, such as acceleration, speed, and wheel direction. To perform the detection of such complex events, it is necessary to collect and analyze the data stream generated by various primitive sensors, looking for patterns and correlations. To detect the pattern of a maneuver, it is necessary to use an important

```
INSERT INTO BOX_PLOT_Q1_Q3_EVENT
SELECT rawValue, q2,
 (SELECT median(rawValue) FROM BOX_PLOT_Q2_EVENT WHERE rawValue < q2) AS q1,
 (SELECT median(rawValue) FROM BOX_PLOT_Q2_EVENT WHERE rawValue > q2) AS q3
FROM  BOX_PLOT_Q2_EVENT.win:time(windowLength sec)
GROUP BY dimension
```

Fig. 9 Q1 and Q3 computation

```
INSERT INTO BOX_PLOT_EVENT
SELECT rawValue, q1, q2, q3, (q3 - q1) AS IQR,
       fmin(rawValue, non_outlier_expression) AS min_non_outlier,
       fmax(rawValue, non_outlier_expression) AS max_non_outlier
FROM  BOX_PLOT_Q1_Q3_EVENT.win:time(windowLength sec)
GROUP BY dimension
```

Fig. 10 5-summary box plot data

concept of CEP called the *time window* (or just window). A window is a temporal context that defines which portions of the input data stream are considered during the execution of an EPL rule [64], i.e., events in the last 30 s, or a snapshot of such recent events [63]. The most common time window models are the batch and sliding window [64]. The former have a fixed lower bound while the upper bound advances every time a new information item enters the system, that is, the CEP engine buffers and processes all events in a time interval. The latter has a fixed size, however, both lower and upper bounds advance when new items enter the system. In others words, it is a moving batch window. An event processing network (EPN) is a network of interconnected EPAs that implement the global processing logic for pattern detection through event processing [29]. In an EPN, EPAs are conceptually connected to each other—output events from one EPA are forwarded and further processed by other EPAs—without regard to the particular kind of underlying communication mechanism for event dissemination.

2.2 Outlier detection

Outlier detection techniques typically assume that outliers in data are rare compared to normal instances. A variety of outlier detection techniques have been developed in several research communities. Many of these techniques have been specifically developed for specific application domains, while others are more generic. The techniques explained in this paper are used widely in several research areas for identifying outliers in data. The earliest algorithms used for outlier detection were statistical approaches which assume that normal instances occur in high probability regions, while anomalies occur in low probability regions. The standard score (more commonly referred to as the Z-score) is a simple statistical technique that enables one-pass computation over a data stream to identify outliers, making different kinds of data comparable and easier to interpret [65]. The Z-score describes a raw score's location in terms of how far above or below the mean it is when measured in standard deviations [65]. A z-score of 0 means that the raw data instance is equal to the mean. The Z-score is calculated as shown in eq. (1), where Z is the Z-score of a data instance, X stands for the sample value, μ stands for the mean of the sampling, and σx stands for the standard deviation of the mean. This computation creates a unitless score that is no longer relates to the original units (e.g., km/h and m/s^2) as it measures the number of standard deviation units and therefore can more readily be used for comparisons [66].

$$Z = \frac{X - \mu}{\sigma x} \tag{1}$$

According to Heiman [65], a Z-score basically has two components: (1) a sign, positive or negative, indicating

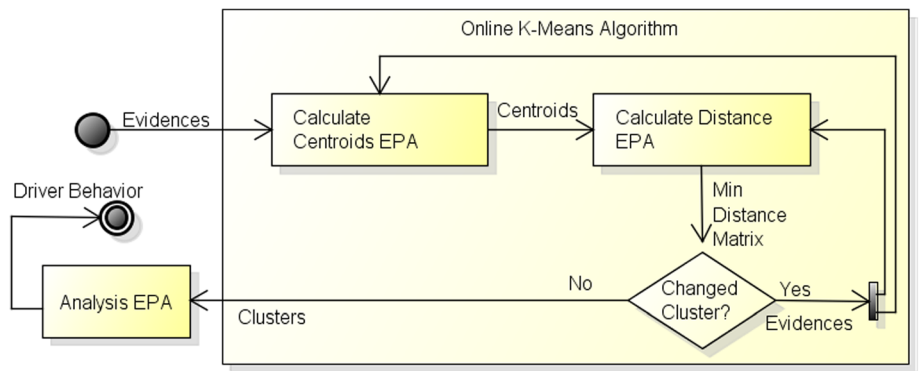

Fig. 11 Online K-means event processing network (EPN)

```
INSERT INTO MIN_DISTANCE_MATRIX
SELECT SQRT(SUM(POW(e.rawValue - c.centroid),2)) AS distance,
       e.cluster_id AS current_cluster_id,
       c.id AS nearest_cluster_id
FROM  EvidenceStreamEvent.win:time_batch(timeWindowLength sec) AS e,
      ClusterStreamEvent.win:time_batch(timeWindowLength sec) AS c
HAVING MIN(SQRT(SUM(POW(e.rawValue - c.centroid),2)))) =
           SQRT(SUM(POW(e.rawValue - c.centroid),2)))
```

Fig. 12 Computing distance to nearest cluster

whether the raw score is above or below the mean; and (2) the absolute Z-score value, indicating the score's distance from the mean when measured in standard deviations. According to Chandola, Banerjee and Kumar [23], all data instances whose Z-score module is greater than 3 are declared an outlier. After computing the Z-score for each data instance, the algorithm calculates the Z-distribution (i.e., the relative frequency of the raw Z-scores of a population or sample). Figure 1 shows a perfect normal Z-distribution (a.k.a., a standard normal curve). It should be noted that 50% of the scores fall below the mean, 50% fall above the mean, approximately 68% of the distribution is between ±1 σx from the mean, and Z-scores higher than +3 and lower than −3 occur less than 1% of the time. If these Z-scores were obtained from driving data, this would imply, for instance, that most of the time a driver maintained a driving behavior without abrupt changes in speed or direction. In cases where outliers are detected, the driver may have conducted evasive maneuvers to avoid accidents or indeed behaved recklessly, but the number of outliers would still be insufficient to consider the driver reckless. The strength of the Z-score arises from the fact that this technique does not require user parameters and outliers are discovered with a single pass over the data stream. However, it is susceptible to the number of data instances in the dataset and has a unidimensional nature [32].

The box plot is likely the simplest statistical technique to detect outliers in both univariate and multivariate data sets that makes no assumptions about the data distribution model [23, 32]. The box plot has become a standard technique for presenting a simple display of a *5-number summary*, which consists of the smallest non-anomaly observation (*min*), lower quartile (*Q1*), median (*Q2*), upper quartile (*Q3*), largest non-anomaly observation (*max*), and interquartile range (IQR)—the difference between *Q3* and *Q1*. This means that 25% of observations are smaller than the first quartile, 50% are smaller than the second quartile, and 75% are smaller than the third quartile. Outliers are points beyond the upper and lower values of the box plot [32]. Laurikkala, Juhola and Kentala [67] suggest a heuristic of (1.5 x IQR) beyond the higher and lower values for outliers; however,

```
INSERT INTO LoopEvent
SELECT TRUE AS loop
FROM MIN_DISTANCE_MATRIX.win:time_batch(timeWindowLength sec) d
WHERE d.AnyOf( i=> i.current_associate_cluster_id != i.cluster_id)
HAVING COUNT(*) > 0
LIMIT 1

INSERT INTO ClusterStreamEvent
SELECT mdm.cluster_id AS id,
       avg( mdm.rawValue) AS centroid
FROM pattern[every mdm=MIN_DISTANCE_MATRIX -> (timer:interval(timeWindowLength sec) AND
     LoopEvent(loop=TRUE))].win:time_batch(timeWindowLength sec)
GROUP BY mdm.cluster_id
HAVING  COUNT(*) > 0

INSERT INTO EvidenceStreamEvent
SELECT  *
FROM pattern[every mdm=MIN_DISTANCE_MATRIX -> (timer:interval(timeWindowLength sec) AND
     LoopEvent(loop=TRUE))]
```

Fig. 13 K-means loop statements

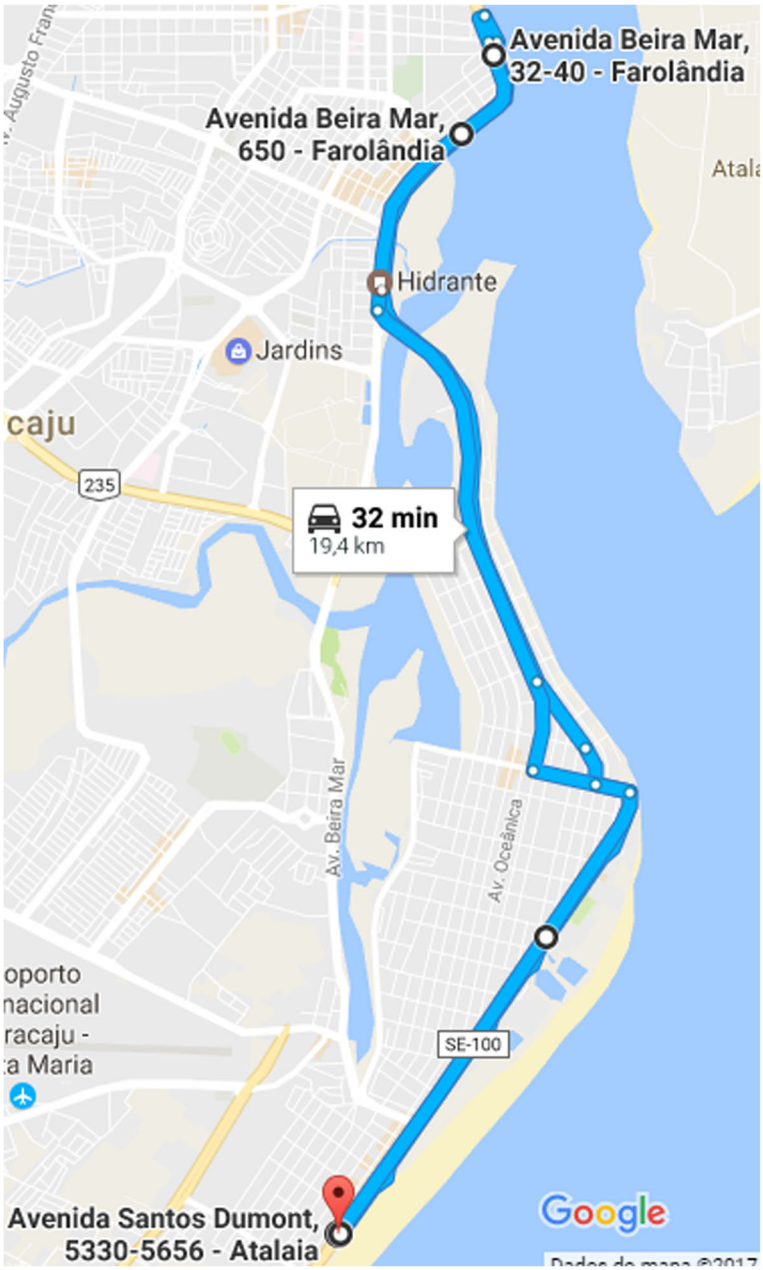

Fig. 14 The study route chosen for data collection (map © 2017 Google)

according to [32], such a heuristic would need to vary across different datasets. A typical box plot can be seen in Fig. 2. Different from the Z-score, box plots make no assumptions about the data distribution model, however, for multivariate datasets, it is possible to perform a pair-wise distance measure. This technique can have quadratic complexity (i.e., in the worst case) since it is founded on the calculation of distances between all data instances [32].

The clustering [68] approach is an exploratory data analysis technique in which a set of input objects, normally multidimensional, are classified into groups (i.e., clusters) of similar objects. Furthermore, it is essentially an unsupervised technique which is preceded by a short and semi-supervised testing and training phase

Table 1 Confusion matrix

Actual class	Predicted class	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

Table 2 Performance metrics [83]

Metric	Equation
Accuracy is the percentage of instances (evidence) correctly classified.	$\frac{TP+TN}{TP+TN+FP+FN}$
Recall is the percentage of instances that were correctly classified as *positive*.	$\frac{TP}{TP+FN}$
Precision is the percentage of instances classified as positive (evidence) that are actually *positive*.	$\frac{TP}{TP+FP}$
F Measure is the harmonic mean of precision and recall, meaning it combines the precision and recall.	$\frac{2 \times precision \times recall}{precision+recall}$
Error Rate is the proportion of instances that are incorrectly classified.	$\frac{FP+FN}{TP+FN+FP+TN}$

[30, 69] used to identify outliers. Distance-based clustering approaches use a particular clustering measure, such as Euclidian distance. As in the box plot technique, distance-based clustering can have quadratic complexity. Such an approach is based on the following hypothesis according to Chandola, Banerjee and Kumar [23]: normal data instances lie close to their closest cluster centroid, while outliers are far away from their closest cluster centroid.

Following the aforementioned hypothesis considering two clusters, as shown in Fig. 3, points *P1* and *P2* are considered outliers since they are far away from the clusters' centroids. However, as outliers form clusters by themselves, this technique is not able to detect such outliers because data instances that lie close to a cluster centroid are considered normal data. To overcome this limitation, a second category of clustering relies on the following hypothesis [23]: normal data instances lie close to their closest cluster centroid, while outliers are far away from their closest cluster centroid. Based on this hypothesis, as *P1* is closer to the cautious cluster centroid, it is considered normal data, while *P2* is closer to the reckless cluster centroid and thus considered an outlier. However, it can be extremely costly to collect and label abnormal data [32]. For instance, collecting data that represents reckless driving behavior can even be dangerous, as it may cause traffic accidents. Thus, a clustering algorithm should be capable of identifying outliers with a few data instances that represent reckless driving behavior. For more details regarding outlier detection, we encourage readers to refer to [23, 31, 32].

The K-means algorithm is likely the most popular and the widely used unsupervised clustering algorithm [68] which can classify multidimensional data into different groups on the basis of certain dissimilarity measures. The classical K-means algorithm initially chooses random cluster prototypes according to a user-defined selection process. Next, the input data is applied iteratively and the algorithm identifies the best matching cluster, updating the cluster centroid to reflect the new exemplar and minimize the sum-of-squares clustering function given by eq. (2), where μ is the mean of the points (x^n) in cluster S_j. However, other distance measurements can be used, such as Euclidean distance [23].

$$\sum_{j=1}^{K} \sum_{n \in Sj} \left\| x^n - \mu_j \right\|^2 \quad (2)$$

Through the combination of EPL rules, it is possible to write algorithms to classify drivers' driving behaviors. Thus, we adapted three outlier detection algorithms to EPL rules to perform online processing of a data stream generated by sensors onboard a vehicle. Sections 3.1, 3.2, and 3.3 explains the algorithms.

3 Related work

Kontaki et al. [70] propose four distance-based algorithms for continuous outlier monitoring in data streams. The primary concerns are improving efficiency and reducing memory consumption. To do this, the algorithms use the concept of *outliers* and *inliers.* A data instance x is considered an outlier if there are less than k data instances at a distance, at most D, from x, excluding x itself. On the other hand, if the number of data instances in the D-neighborhood of x is enough (i.e., more than k), then x is characterized as an *inlier*. To improve

Table 3 Confusion Matrix by Algorithm and Driver (TP and TN) for h = 100 and Δ = 10

Confusion Matrix						
Driver	Cautious (True Positive - TP)			Reckless (True Negative – TN)		
	Cluster	Box Plot	Z-score	Cluster	Box Plot	Z-score
D1	5990	5004	5143	17	49	40
D2	5642	5439	5593	4	52	28
D3	5876	5667	5817	26	75	39
D4	13,465	5618	5716	123	191	45
D5	6114	5024	5201	5	58	41

Table 4 Confusion matrix by algorithm and driver (FP and FN) for h = 100 and Δ = 10

Confusion Matrix						
Driver	Cautious (False Positive - FP)			Reckless (False Negative – FN)		
	Cluster	Box Plot	Z-score	Cluster	Box Plot	Z-score
D1	3	19	28	110	175	36
D2	0	34	58	83	201	47
D3	14	40	76	61	191	41
D4	12	14	160	288	131	33
D5	10	27	44	70	206	29

Table 5 Algorithm accuracy comparison

Driver	h = 100 and Δ = 10			h = 50 and Δ = 20		
	K-means	Box Plot	Z-score	K-means	Box Plot	Z-score
D1	98.15%	96.30%	98.78%	98.05%	93.71%	98.58%
D2	98.55%	95.90%	98.17%	98.16%	96.30%	98.80%
D3	98.75%	96.13%	98.04%	98.40%	96.61%	98.59%
D4	97.84%	97.59%	96.76%	96.12%	97.80%	98.94%
D5	98.71%	95.62%	98.63%	97.86%	97.40%	98.57%
Average	98.40%	96.30%	98.07%	97.72%	96.36%	98.70%

Table 7 Algorithm recall comparison

Driver	h = 100 and Δ = 10			h = 50 and Δ = 20		
	K-means	Box Plot	Z-score	K-means	Box Plot	Z-score
D1	98.20%	96,62%	99.30%	98.10%	96.35%	98.91%
D2	98.55%	96.44%	99.17%	98.52%	96.30%	98.80%
D3	98.97%	96.74%	99.30%	98.77%	98.86%	98.60%
D4	97.91%	97.72%	99.43%	96.19%	97.80%	98.94%
D5	98.87%	96.06%	99.45%	98.12%	97.40%	98.63%
Average	98.50%	96.72%	99.33%	97.94%	97.34%	98.78%

efficiency, the concept of micro-clusters is used to reduce the number of distance computations. The window size determines the memory size and the number of data instances considered in the time window approach, that is, all data instances in a time window are stored in the main memory and processed by an algorithm. However, the authors control the arrivals and departures of instances in the time window. In these events, if the number of neighbors of a given data instance of x is greater than k then x will never be an outlier and is called a *safe inlier*. Thus, safe inliers are not stored for further processing and consequently reduce computation and memory use. Each algorithm has a few variations of this process, however, none have been designed to run on devices with memory and processing constraints.

An online outlier exploration platform, or in short, ONION [71], is proposed for modeling and exploring outliers in large datasets based on a distance-based approach. An ONION employs an offline preprocessing phase followed by an online exploration phase, enabling users to establish connections among outliers. As it is difficult to set appropriate D and k values [70, 71], the offline phase is a preprocessing three-dimensional phase that computes all possible combinations of D, k, and entire dataset instances. In fact, k can take in the universe of natural numbers and the user must specify lower and upper bounds for k. This phase outputs all outlier candidates. Then, the online phase, with some rules, determines which candidates are actually outliers.

Zhao et al. [55] propose a driver behavior evaluation scoring mechanism named Join Driving (based on the ISO 2631 standard [72]). This mechanism analyzes passengers' comfort level based on their exposure to vibrations to classify drivers as cautious or reckless. As human's feelings in response to vibration depend on the level, frequency, and duration of acceleration, the mechanism analyzes three-axis accelerometer data. However, because a smartphone is likely in an arbitrary position inside a vehicle, the authors also have developed a novel algorithm for reorientation using GPS and orientation sensor data. The evaluation shows that the mechanism can accurately score driving behaviors in high and mid-value smartphones. The main difference between this approach and those of the current paper is that in Zhao et al. [55] the analysis of the data is offline—performed when a driver reports their arrival arrived at a destination—while our approach is based on online processing, able to help a driver while driving.

To understand and model reckless driving behavior, Hong, Margines and Dey [56] implemented a low-cost, in-vehicle sensing platform. Unlike Zhao et al. [55], which only uses a smartphone's sensors, this platform added an OBD-II diagnostic device to collect data from the vehicle, such as speed, rpm, speed, and throttle position. Furthermore, to detect steering wheel movement, a device called an inertial measurement unit (IMU) was added. Both devices communicated via Bluetooth with a smartphone. To characterize driving behavior, a machine learning-based model analyzed data from acceleration

Table 6 Algorithm precision comparison

Driver	h = 100 and Δ = 10			h = 50 and Δ = 20		
	K-means	Box Plot	Z-score	K-means	Box Plot	Z-score
D1	99.95%	96.22%	99.46%	99.93%	100.00%	99.52%
D2	100.00%	99.38%	98.97%	99.60%	100.00%	100.00%
D3	99.76%	99.30%	98.71%	99.61%	100.00%	100.00%
D4	99.76%	99.75%	97.28%	99.89%	100.00%	100.00%
D5	99.84%	99.47%	99.16%	99.71%	100.00%	100.00%
Average	99.89%	98.82%	98.72%	99.75%	100.00%	99.90%

Table 8 Algorithm F measure comparison

Driver	h = 100 and Δ = 10			h = 50 and Δ = 20		
	K-means	Box Plot	Z-score	K-means	Box Plot	Z-score
D1	99.07%	98.10%	99.38%	99.00%	98.17%	99.45%
D2	99.27%	97.89%	99.07%	99.06%	98.06%	99.30%
D3	99.37%	98.00%	99.00%	99.19%	98.06%	99.20%
D4	98.90%	98.73%	98.34%	98.00%	99.19%	99.42%
D5	99.35%	97.73%	99.30%	98.91%	98.68%	99.30%
Average	99.19%	98.09%	99.02%	98.83%	98.43%	99.33%

Table 9 Algorithm execution time comparison

	$h = 100$ and $\Delta = 10$			$h = 50$ and $\Delta = 20$		
Driver	K-means	Box Plot	Z-score	K-means	Box Plot	Z-score
D1	769.6 ms	132.0 ms	100.2 ms	712.0 ms	129.8 ms	98.3 ms
D2	700.0 ms	187.0 ms	103.1 ms	709.4 ms	186.4 ms	99.0 ms
D3	706.4 ms	189.9 ms	100.9 ms	705.5 ms	190.2 ms	99.7 ms
D4	705.3 ms	186.0 ms	101.5 ms	702.3 ms	186.8 ms	100.6 ms
D5	787.0 ms	188.0 ms	102.7 ms	780.4 ms	190.1 ms	100.3 ms
Average	733.6 ms	176.4 ms	101.6 ms	721.9 ms	176.6 ms	99.5 ms

(smartphone sensor), OBD-II, and the IMU. To determine driving behavior, a trip profile is constructed by summarizing trip-profiles obtained from the last three weeks, called driver's profiles. Finally, the driver's driving behavior is determined from the driver profile and machine learning. As in the work proposed by Zhao et al. [55], analysis of driver behavior is off-line. However, Hong, Margines and Dey [56], templates must be stored, such as acceleration, deceleration, and curves, which are used for comparison with the driver's maneuvers and a subsequent classification as cautious or reckless. According to Banovic et al. [73], this is a weakness because machine learning algorithms classify and predict only the most frequent behaviors. In this respect, infrequent variations in drivers' behaviors are difficult to detect.

Vehicle data stream mining (VEDAS) [74] aims to identify outliers using a device with low computational power—low processing and storage capacity—and was designed to mine a vehicle's data stream. Data are collected through an OBD-II device and stored in a data stream management system (DSMS) that provides mechanisms to control and access the data through queries. The DSMS provides operators with the ability to compute statistical aggregation such as mean, variance, and covariance. After pre-processing aggregate data, VEDAS constructs a representation of low dimensional data through three techniques: Incremental principal component analysis (PCA), Fourier transformations, or linear online segmentation. Although it is possible to dynamically choose which of the techniques will be used, the authors emphasize that PCA does not work well for online monitoring with limited computational resources. According to the authors, VEDAS implements a collection of techniques and algorithms, including proprietary ones, to perform data stream analysis. The authors discuss techniques based on clustering and statistical tests. First, OBD-II data is grouped by K-means to detect abnormal vehicle health monitoring patterns. The goal of clustering is to identify representations in space that correspond to safe vehicle operation. The detection of unusual driving patterns is performed through acceleration analysis with a linear approximation algorithm, the piecewise linear approximation [75]. In addition, a statistical test is performed on the smoothed data with the algorithm assuming a Gaussian distribution to identify unusual patterns. The data used for validation of the proposal were extracted from *Live For Speed*, a driving simulator. However, no driver behavior classification is performed.

The study of Aljaafreh, Alshabatat, and Najim [76] proposes the use of inference by fuzzy logic for online identification of abnormal driving data and driver behavior classification based on acceleration and speed. Lateral and longitudinal acceleration are categorized in three intervals: low, medium, and high. Speed is categorized into five ranges, from very low to very high. The values of these outputs are used to classify drivers' behavior. The proposal of Quintero, Lopez, and Cuervo [14] also uses fuzzy logic; however, the output variables are inserted in a neural network properly trained to classify driver behavior. However, the neural network is on a remote server, so all fuzzy system outputs must be sent to this server which performs offline analysis and driver behavior classification. The authors used a backpropagation algorithm, and the best performing architecture was a two-layer neural network, with nine neurons in the intermediate layer and 31 inputs.

Table 10 Smartphone resource consumption

Situation	RAM(MB)	CPU (%)
Standby	4.53 MB	1.41%
Collecting	5.18 MB	3.60%

4 Online CEP-based outlier detection algorithms

This section presents the aforementioned outlier detection algorithms expressed as a set of CEP rules for

Table 11 Algorithm resource consumption

Algorithm	$h = 100$ and $\Delta = 10$		$h = 50$ and $\Delta = 20$	
	RAM(MB)	CPU (%)	RAM(MB)	CPU (%)
K-means	6.75 MB	20.20%	7.83 MB	23.35%
Box Plot	6.30 MB	11.40%	7.30 MB	11.45%
Z-score	6.15 MB	6.61%	6.40 MB	7.46%

Table 12 Algorithm error rate

Driver	h = 100 and Δ = 10			h = 50 and Δ = 20		
	K-means	Box Plot	Z-score	K-means	Box Plot	Z-score
D1	1.85%	3.70%	1.22%	1.35%	3.20%	1.00%
D2	1.45%	4.10%	1.83%	1.50%	3.40%	0.93%
D3	1.25%	3.87%	1.96%	1.17%	2.80%	1.80%
D4	2.16%	2.44%	3.24%	1.50%	2.00%	1.20%
D5	1.29%	4.38%	1.37%	1.45%	3.80%	1.10%
Average	1.60%	3.70%	1.93%	1.39%	3.04%	1.21%

online outlier detection to operate over a multiple mobile data stream, enabling their efficient operation on mobile devices. The algorithms are generic, however, as highlighted by Chandola, Banerjee, and Kumar [23], the exact notion of "*outlierness*" differs according to the application domain. Therefore, in our case study, we aim to classify driver behavior based on outlier detection. Our driving behavior characterization algorithms are based on a *pattern-recognition* approach. Although the modeling relies on an idea proposed by Zhang [77], the difference between the proposals is the fact that Zhang's work aims to identify the driver's skill level (e.g., expert or novice) through receiving driver behavior measurements as input. Our research aims to identify driver behavior based on online outlier detection through measurements of the signals from different sensors embedded onboard the vehicle, as well as sensors of the mobile device onboard the vehicle.

The processing workflow begins with the interaction between the driver and the vehicle. Each driver exhibits behaviors that can be divided into two types, namely short-term and long-term driving behaviors. The former concerns drivers' instantaneous behavior that should be taken into account separately, such as pressing the accelerator or the brake. The latter represents larger driving maneuvers, such as making a turn. In this case, it is necessary to consider several issues, namely how the driver accelerates or brakes, the steering wheel angle, and the driver's speed [78]. From these behavior types, it is possible to detect a unique driving pattern for each driver, enabling the formulation of a profile representing the driver's behavior [78].

To sense data from the driving behavior, the *stream module* is responsible for discovering, connecting, and reading both onboard vehicle and built-in mobile device sensors. This module is able to communicate with onboard sensors via short-range wireless communication technologies, such as Bluetooth and Bluetooth Low Energy. More details are available in our previous work [79]. This module acts as a hub and forwards the data stream to the CEP engine for preprocessing. This raw data preprocessing consists in producing higher-level data (referred in this paper as *evidence*) that best represent the driver behavior. This process is known as *feature extraction*. A feature is a measurable property that best represent a phenomenon and feature extraction is the processes of deriving the values of such features [80]. As discussed, to measure long-term driver's behavior, some available features need to be analyzed and correlated over a time period. These features include speed ($S = [s_1, s_2, \ldots, s_n]^T$) and acceleration ($A = [a_1, a_2, \ldots, a_n]^T$). The parameter n denotes the number of instantaneous sampled values and T denotes a specific time period. Additional features, such as mean speed excluding stops, mean acceleration, mean deceleration, both acceleration/deceleration changes, yaw, and a combination of other physical measurements may be used to measure a driver's behavior, as discussed in [80].

The *online outlier detection module* is responsible for finding patterns in the available evidence that deviate sufficiently from expected behavior. The online outlier detection algorithm adapted for CEP rules runs in this module. An important feature of any outlier detection algorithm is the manner in which outliers are reported [23]. On one hand, scoring algorithms, such as Z-scores, assign a score to each evidence estimating the "*outlierness*". On the other hand, label algorithms, such as box plots, assign a label (normal or outlier) to each evidence.

Table 13 Performance metrics comparison

Algorithm	Performance Metrics				
	Accuracy	Recall	Precision	F Measure	Error Rate
K-means	98.06%	98.22%	99.82%	99.01%	1.50%
Box Plot	96.33%	97.03%	99.41%	98.26%	3.37%
Z-score	98.39%	99.06%	99.31%	99.18%	1.57%
VEDAS [74]	97.61%	98.55%	99.02%	98.77%	2.38%
Fuzzy [76]	98.22%	99.84%	98.36%	99.10%	1.78%
Backpropagation [14]	99.34%	100.00%	99.43%	99.72%	0.57%
Join Driving [55]	95.45%	100.00%	92.31%	96.00%	4.55%
Naïve Bayes classifier [56]	81.82%	90.91%	76.92%	83.33%	18.18%

Table 14 Quality metrics comparison

Algorithm	Quality Metrics		
	RAM(MB)	CPU (%)	Time (ms)
K-means	7.29 MB	21.78%	727.75 ms
Box Plot	6.80 MB	11.43%	176.50 ms
Z-score	6.27 MB	7.04%	100.60 ms
VEDAS [74]	6.34 MB	26.39%	501.00 ms
Fuzzy [76]	6.99 MB	26.67%	10.11 ms
Backpropagation [14]	13.72 MB	27.22%	11.21 ms
Join Driving [55]	6.32 MB	8.27%	10.64 ms
Naïve Bayes classifier [56]	12.57 MB	17.38%	15.12 ms

Finally, these scores or labels are analyzed by the *analysis module* to classify the driver behavior (i.e., cautious or reckless) and update the driver profile. In practice, the mobile modules act as an EPN, that is, there is a set of interconnected EPAs in each of them with their respective set of EPL rules. The prototype application architecture is shown in Fig. 4.

To compare different analysis approaches, the three outlier detection algorithms shown in Section 2.2 were adapted for continuous outlier monitoring over data stream, discussed in Sections 3.1, 3.2 and 3.3.

4.1 Online CEP-based Z-score algorithm

Because the online Z-score algorithm receives a stream of data instances and cannot wait until all evidences have been received, it needs to divide the stream into a sequence of *windows*, each of which contains a set of evidence. Therefore, the online Z-score, shown in Fig. 5, is calculated according to eq. (*3*). Unlike the classical Z-score algorithm, the online Z-score sample mean values and standard deviation of the mean are computed over evidences in a specific sliding window *T*. So, *temporal context* rules determine which data instances are admitted into which window. Then, the algorithm calculates the Z-score of the available evidence in each window. Finally, the Z-distribution is analyzed to classify the driver behavior.

$$Z = \left(\frac{X-\mu}{\sigma x}\right)^{T} \tag{3}$$

The EPL statement that implements the Z-score algorithm is illustrated in Fig. 6. The *time* clause in line 3 is a temporal operator that segments the evidence data stream instances into a sliding window of *windowLength*, a time period argument. The statement output is inserted in the stream of Z-score events for further processing, denoted here by *z_score_event*. Then, another EPL statement computes the Z-distribution from the *z_score_event* data stream according to Fig. 1.

4.2 Online CEP-based box plot algorithm

The design of the algorithm for driving behavior detection with a box plot technique is shown in Fig. 7. First, as with the Z-score, a temporal context needs to be performed. Additionally, to avoid computation of pairwise distances for all evidence that can have quadratic complexity [23], we chose to perform the computations for each dimension individually. For the last step, the analyzes EPA just need correlate the outliers. The EPL statement that implements these two steps is illustrated in Fig. 8. As an output, this statement inserts the computed median into a stream of *box_plot_q2_event*. This computation is shown in Fig. 8. Second, Q1 and Q3 are computed. To do so, we designed an EPL that subscribes to *box_plot_q2_event* and uses them as threshold to compute Q1 and Q3. The result is inserted into the *box_plot_q1_q3_event* stream, as shown in Fig. 9.

Third, three computations are performed simultaneously: (i) *min*, *max,* and *IQR* are computed as shown in Fig. 10. The *non_outlier_expression* filters all data instances in the stream that are not outliers and *outlier_expression* filters all instances of outlier in the data flow.

Fig. 15 Smartphone in mounted position

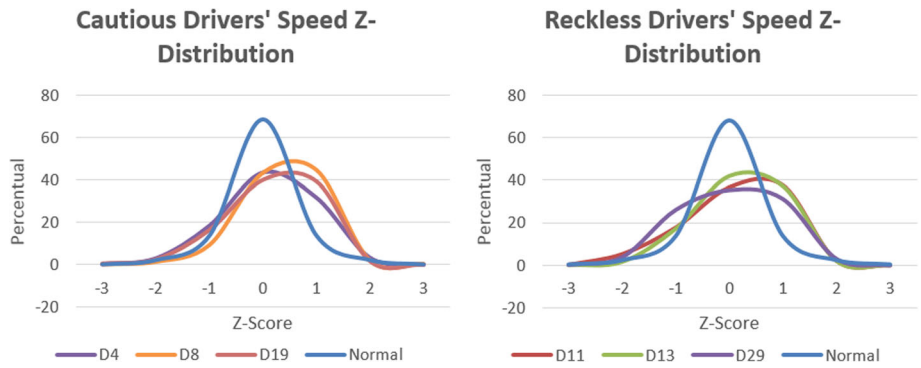

Fig. 16 Speed Z-distribution comparison

These two expressions use the heuristic proposed by Laurikkala, Juhola and Kentala [67], as explained in Section 2.2, and are expressed respectively by eqs. (4) and (5). The *fmin* and *fmax* functions return both the lowest and highest values, respectively, which are not considered outliers in the *box_plot_q1_q3_event* data stream, respecting the restrictions imposed by *non_outlier_expression*. As an output, this statement inserts the computed 5-number summary into a stream of *box_plot_event* streams. (ii) An EPL rule filters all data instances in the flow that are not outliers using the *non_outlier_expression.* (iii) The other EPL rule filters all outliers' instances using the *outlier_expression* in data flow. These outputted data are forwarded to an EPA responsible for the analysis. Finally, similar to Z-score analysis, if most of the time the evidence is close to the median, then the driver is classified as cautious. Otherwise, if most of the time the median is close to Q1 or Q3, or IQR is high, the driver is classified as reckless.

$$rawValue \geq (q1 - (1.5^{*}IQR)) \text{rawValue} \leq (\text{q3} + (1.5^{*}\text{IQR})) \quad (4)$$

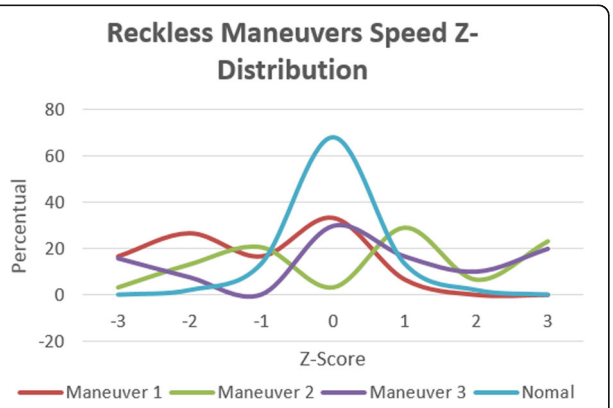

Fig. 17 Speed maneuvers' Z-distribution comparison

$$rawValue < (q1 - (1.5^{*}IQR)) \,||\, rawValue > (q3 + (1.5^{*}IQR)) \quad (5)$$

4.3 Online CEP-based K-means algorithm

An overview of the K-means algorithm work flow for online driving behavior detection is shown in Fig. 11. Unlike the previous modeling approaches, this is an iterative algorithm. Therefore, it is necessary to use a batch window to make iterating possible until the algorithm converges. Thus, the incoming evidence data stream is first separated in different temporal contexts (batch windows). Second, cluster centroids are chosen. The traditional implementation of the K-means algorithm chooses K random instances and defines them as clusters' centroids. The main disadvantage of this method lies in its sensitivity to initial values of the cluster centroids. Our developing tests displayed a poor performance with the traditional random choice of initial centroid's values of the clusters. To overcome this problem, the algorithm starts with previously acquired knowledge in the training phase. More details are given in Section 3.5.

Third, the distances between evidence and clusters centroids are calculated and evidence are assigned to the nearest cluster, as shown in Fig. 12. While some evidence changes from one cluster to another (Fig. 13 from line 1 to 6), the algorithm performs two parallel processing only if there is a *min_distance_matrix* event followed by *loopEvent* equal to true in a specific time window: (i) new cluster centroids are calculated (Fig. 13, line 8–14) and (ii) the evidence is put back into the flow for another calculation of the distances to the cluster centroids (Fig. 13, line 16–19). Otherwise, the clusters are forwarded for driver behavior analysis. At the end of each batch window, if for most of the time the evidence belongs to the cautious cluster, then the driver is classified as cautious. In any other way, the driver is classified as reckless. Thus, if a driver maintains a driving behavior

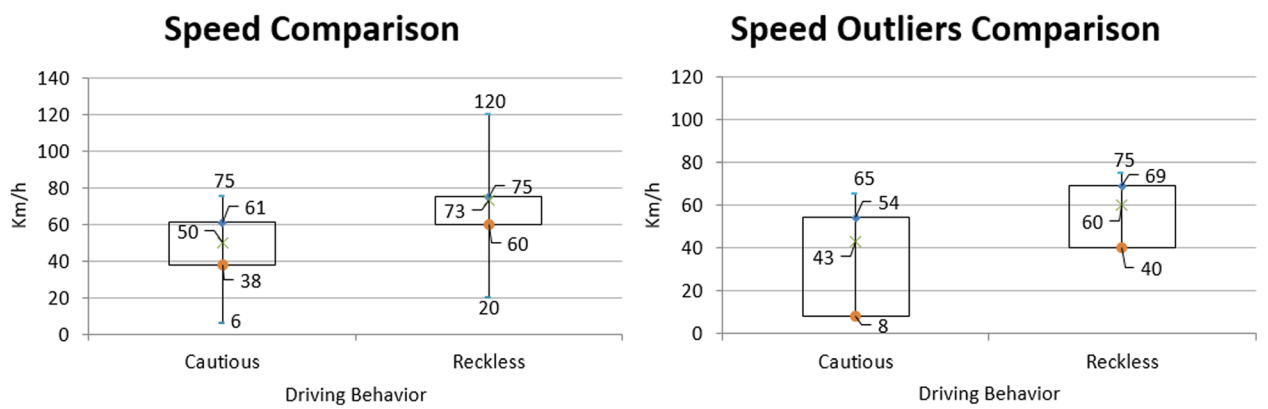

Fig. 18 Speed comparison

without abrupt changes of speed or direction, for instance, the percentage of evidences that belong to the cautious cluster is higher than to reckless cluster and the driver is classified as cautious.

4.4 Limitations

While CEP provides several benefits for data stream processing, such as continuous query, pattern detection, and temporal windows, it is difficult to express iterative controls, e.g., *while* and *for* repetition structures, using its primitives. Typically, CEP-based applications follow a pipeline stage topology, with data flow in a given direction from one stage to one or more stages, but without returning to previous stages. This can be troublesome when describing iterative algorithms, such as K-means, which require iterations to converge. To overcome this problem, we simulated the loop check by using two EPL rules. If the loop check is false, i.e., if evidence has not change its centroid, we push the event to the next processing stages. However, if the loop check is true, i.e., evidence has changed its centroid, we reinsert these events into the initial loop stage, which recomputes the centroid distance. We do so by translating the events to *EvidenceStreamEvents*, the event type that the initial loop phase (distance computation) expects.

Although useful, the independency of each CEP processing stage can difficult to coordinate between them. For instance, the proposed K-means algorithm buffers the received evidence in batches of Δ time period which are sent to the next processing stage, as shown in Fig. 12. This is required so that during interactions, the algorithm analyzes the same set of evidences to partition them into clusters. Thus, even though the events are buffered, the batch events are analyzed one by one by default in the sequential processing stages. To do so, the stages buffer the incoming events in a minimum window so they can be output as a batch. This minimum time window is associated with the mobile device memory and processing power and is usually less than a second. This limitation is shown in the EPL rule at the top of Fig. 13. The *timeWindowLength* parameter is the minimum time for to buffer all outputted streams in the *min_disntance_matrix* event and check if evidence has changed its centroid.

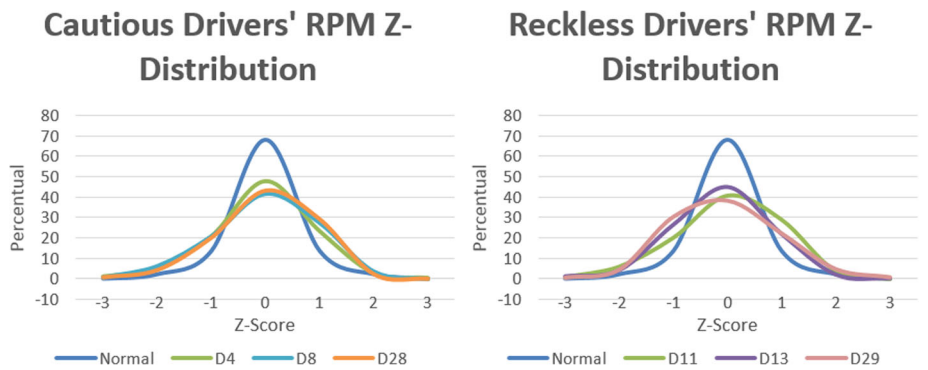

Fig. 19 RPM Z-distribution comparison

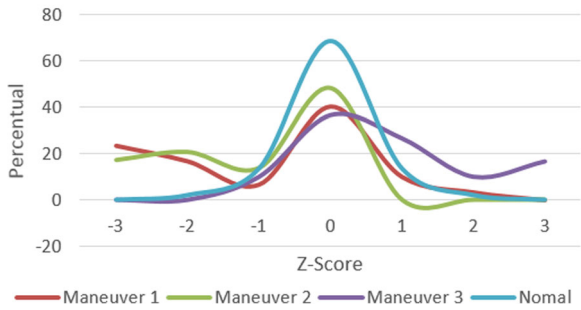

Fig. 20 RPM maneuvers Z-distribution comparison

5 Defining and planning the case study

In this section, the case study is presented with a focus on the definition and planning of the objective.

5.1 Objectives and contributions

The aim of this case study is to evaluate the designed and implemented algorithms, as shown in Section 4, to identify drivers' driving behavior based on outlier detection. More specifically, the objectives are as follows.

- Evaluate the effectiveness of online outlier detection algorithms. That is, evaluate the performance analysis of the pattern recognition algorithms for online outlier detection in the context of limited computational resources (i.e., with a smartphone).
- Perform a case study to assess a driver's driving behavior on driveway sections, such as roundabout, turns, tangent sections, semaphores, intersections (all-way stop), and crosswalks based on online outlier detection.
- Provide an open dataset of driver behavior with a rich set of sensed data, such as speed, rpm, throttle position, accelerometer, and gyroscope.

5.2 Research questions

The research questions that need to be answered through the case study are:

- Which is the best algorithm in terms of accuracy?
- Which is the best algorithm in terms of precision?
- Which is the best algorithm in terms of recall?
- Which is the best algorithm in terms of F-measure?
- Which is the best algorithm in terms of average execution time?
- Which is the best algorithm in terms of resource consumption (memory and CPU)?

5.3 Drivers and route selection

Due to the difficulty of recruiting drivers and the costs associated with assessing driving behavior, the process of driver selection was a matter of convenience and sampling was completed by quota. However, we attempted to establish a sample that represented a broad swath of drivers, preserving the same behavioral characteristics. Thus, 25 drivers were chosen for the study. Sixteen were male and nine female, their ages ranged from 20 to 60 years. Another important factor is driver experience. In our sample, driver experience ranged from 2 to 42 years. Finally, all drivers were familiar with local traffic condition and regulations. This is important so that during the assessment their behaviors reflect the daily driving habits.

Regarding route selection, several potential test locations were evaluated. We chose a paved route comprised of streets and avenues ranging from one to three lanes covering approximately 19.4 km in Aracaju-SE Brazil. In addition, the route, shown in Fig. 14, contains traffic lights, pedestrian crossings, and turns (including 45° and 90° turns). The speed limit on the route was 60 km/h. A pilot study was conducted with all the 25 drivers on the chosen route and this pilot study provided insight into drivers' behaviors.

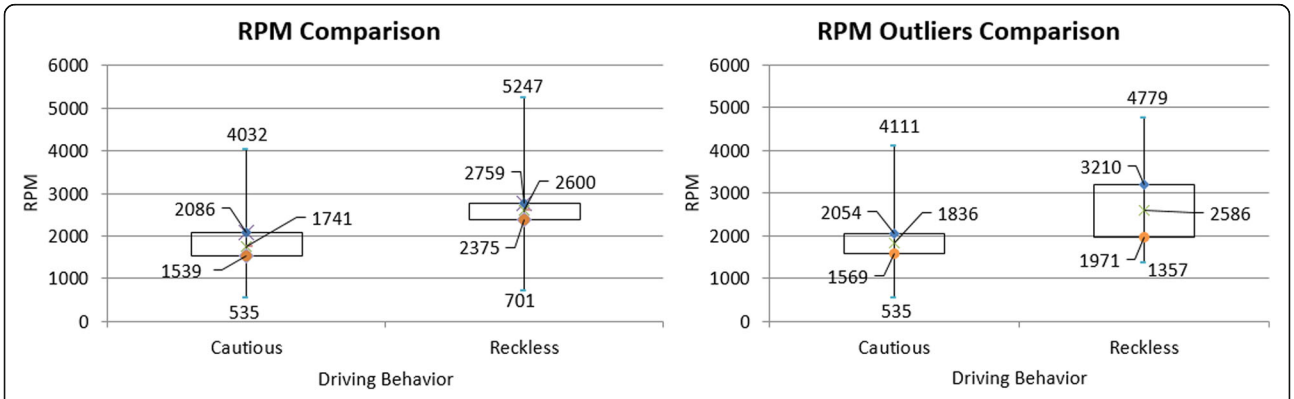

Fig. 21 RPM comparison

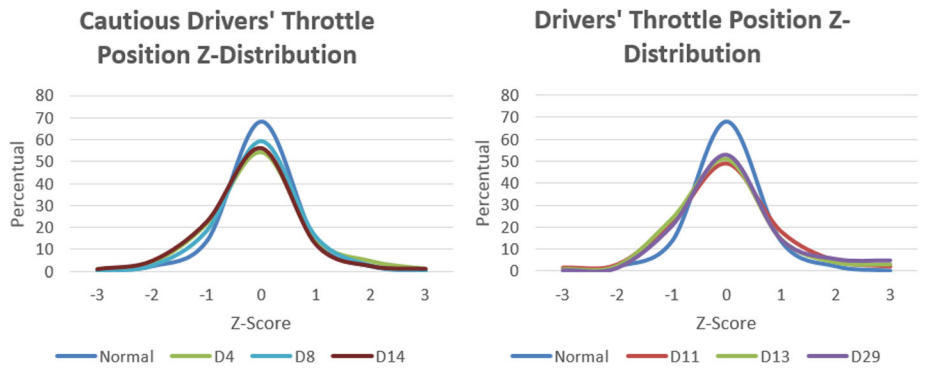

Fig. 22 Throttle position Z-distribution comparison

5.4 Instrumentation

The instrumentation process began with the implementation of the algorithms through CEP rules as described in Section 4. The algorithms were implemented in EPL, an structured query like language (SQL-like) where streams replace tables as the source of data with events replacing rows as the basic unit of data for running in ASPER, a CEP processing engine based on ESPER (an open source CEP engine [81]) and adapted for Android.

A Brazilian version of the Citröen C3 manual transmission was equipped with a Samsung Galaxy SIII 1.4 GHz Quad Core with 1GB of RAM and a Bluetooth OBD-II device. Our prototype was installed in the smartphone running the online Z-score algorithm. Further details regarding the choice of algorithm are given in Section 6.4. The data collected by the prototype and processed by the Z-score were stored in SQLite [82], an embedded and free SQL database engine.

5.5 Measurement metrics

A confusion matrix is a suitable technique to evaluate the predictive ability of an algorithm to classify data instances, [83]. For n classes, the confusion matrix is table of n x n. The *actual class* column corresponds to the correct classifications and the *predicted class* represents the algorithms' classifications. When there are only two classes, one is considered positive (in our case, cautious driving) and the other is considered negative (reckless driving) [83], as shown in Table 1.

Thus, TP means that an instance of positive class is correctly classified as positive (cautious driving evidence), FN means that a positive class instance is misclassified as negative (reckless driving evidence), TN means that a negative class instance is classified correctly as negative, and FP means that a negative class instance is misclassified as positive. Based on a confusion matrix, we can calculate five performance metrics, shown in Table 2, which can be used to evaluate the algorithms. In addition, as a quality metric, we used the *average execution time*, that is, the arithmetic mean of execution times for a given algorithm and the algorithms' *average resources consumption*.

6 Operation of the case study

This section describes the preparation and execution of the real world case study.

6.1 Preparation

To train the algorithms, we used the open dataset provided by Bergasa et al. [57]. The dataset provides three axis accelerometer data labeled as cautious and reckless based on thresholds given by Paefgen et al. [54] for acceleration, braking, and turning. This dataset contains driving data for six different drivers and vehicles, D1 through D6, for two different routes, one is 25 km in a road with 3 lanes in each direction and a speed limit of 120 km/h, the other is ~16 km on a secondary road with one lane on each direction and a 90 km/h speed limit. Each driver drove the same route three times. For each driver's data, a *3-fold cross-validation* was performed, where each driver's data are randomly divided into two pieces, one piece of 35% for training, and one piece of 30% for testing, and subset of data that generated the

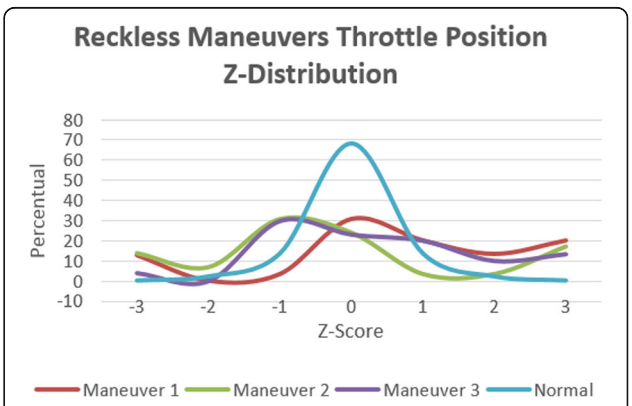

Fig. 23 Throttle position maneuvers Z-distribution comparison

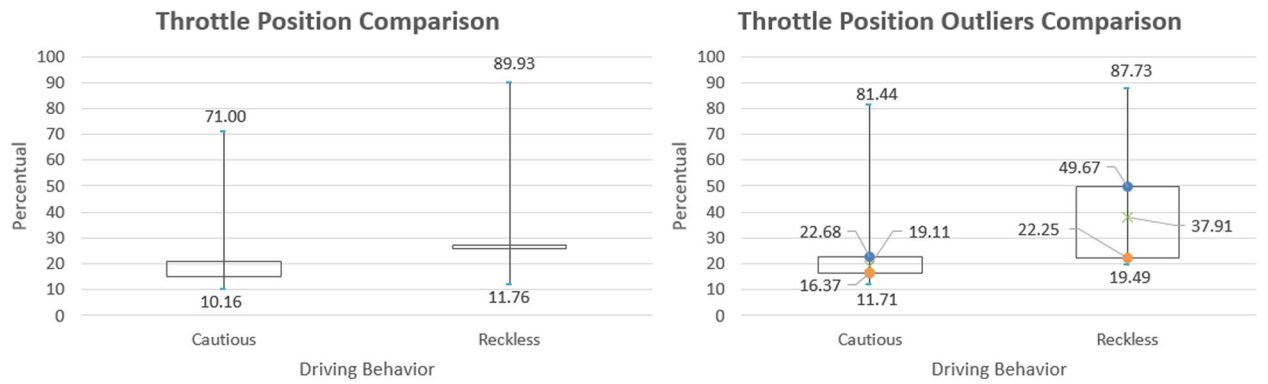

Fig. 24 Throttle position comparison

best results for each algorithm checked. The training and testing hardware was the smartphone discussed in Section 5.4.

Regarding the K-means performance problem highlighted in Section 3.3, we randomly chose a driver and extracted driving data from cautious and reckless behavior from the training dataset and computed the clusters' centroids. Thus, the algorithm begins with previously acquired knowledge instead of choosing random values for the clusters' centroids. With this strategy, the K-means algorithm significantly improved results, as is seen in Section 6.1.1.

To be operational on a mobile device, the applications need to vary the data rate based on the available computation resources. So, the algorithms need to adapt their behavior to perform outlier detection with good accuracy. Based on this scenario, we defined two setups. Firstly, we set the sensor data sample rate to be h = 100 Hz and a time window of Δ = 10 s (setup 1). Second, we set the sensor data sample rate to be h = 50 Hz and a time window Δ = 20 s (setup 2).

6.1.1 Intrinsic evaluation of the knowledge model

In this subsection, we present the results of the training using the open dataset cited in Section 6.1. We repeated each evaluation 5 times and the confidence level for all results is 95%. As shown in Table 3 and 4, the dataset contains more positive instances than negative instances. This data corroborates our first assumption, highlighted in Section 1.3. From the confusion matrix, we calculated the performance metrics defined in Table 2 for the two defined setups. The three algorithms had excellent performance, as the worst result classified 93.71% of evidences correctly and there is no significant (greater than 1%) difference between the average results, as shown in Table 5. Despite the good overall performance, the Z-score and K-means algorithms stood out with the highest average accuracies.

In the first setup, the box plot and Z-score had practically the same precision and K-means had an impressive result with an average precision of 99.89%. This means that K-means correctly classifies 99.89% of cautious evidences. On the other hand, in the second setup, the box plot correctly classified all cautious evidences. Only K-means had worse precision, as shown in Table 6. According Table 7, the Z-score achieved a better recall performance in both setups. This means that, on average, the Z-score reached better true positive rates. The K-means algorithm also achieved excellent results.

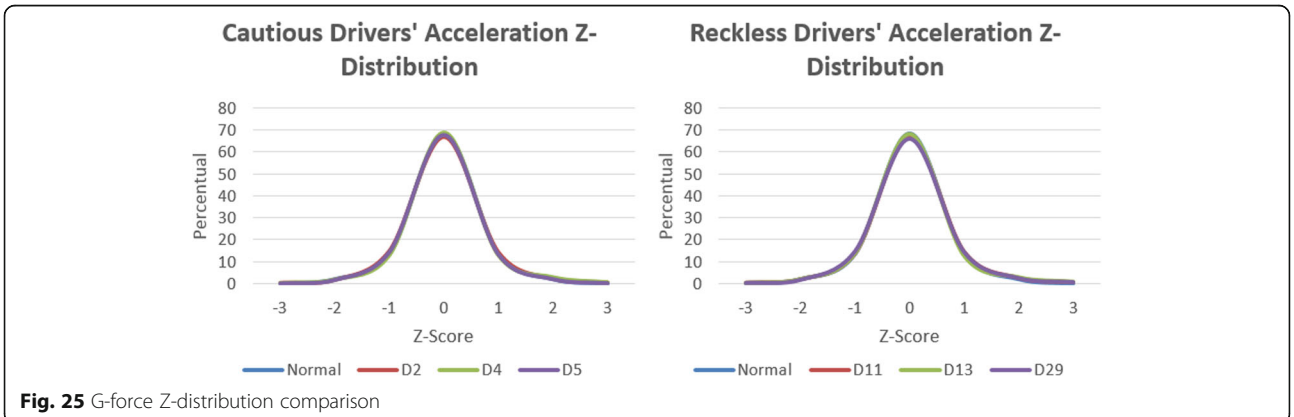

Fig. 25 G-force Z-distribution comparison

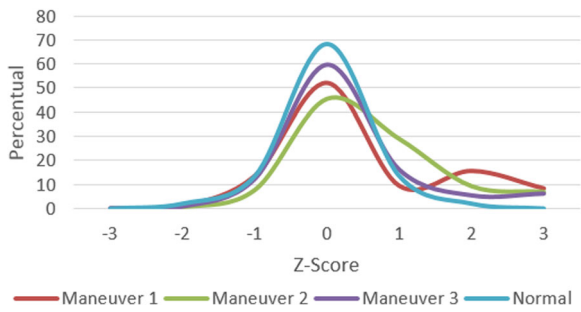

Fig. 26 G-force maneuvers' Z-distribution comparison

Regarding F measure, K-means and Z-score once more obtained similar overall performances and the box plot obtained a slightly lower performance in the first setup. However, in the second setup, the Z-score stood out with the highest average F measure, as shown in Table 8.

Table 9 shows the execution time in milliseconds for each algorithm. It is possible to note that the Z-score requires considerably less processing time than the other algorithms and because the K-means requires computation of pairwise distances for all evidences against cluster centroids, it has quadratic complexity, in contrast to the linear complexity of the box plot and Z-score. This explains the large execution time. Furthermore, while the box plot and Z-score execute a single pass over the entire data stream, K-means requires, on average, three iterations for each time window. The results confirm our intuition. For the chosen setups, there are no significant (greater than 1%) differences in each algorithm's execution time.

To check the algorithms' resources consumption in both setups, we first verified the smartphone's memory and CPU usage in two situations: standby and collecting data from the smartphone's own sensors and OBD-II device without processing them. It is possible to note in Table 10 that merely collecting data increases memory consumption by 12.54% and CPU usage by 60.83%. By analyzing data from Table 11, it becomes clear that the Z-score outperforms the K-means and box plot algorithms in both setups, and K-means stands out negatively in terms of CPU usage. Moreover, a larger time window results in higher memory consumption and processing.

After reviewing these results and calculating the average error rate of the algorithms, K-means and Z-score stand out with the lowest average error rates. Both had similar average rates, while the K-means achieved a better performance in setup 1 and the Z-score achieved a better performance in setup 2. Thus, regarding performance metrics, K-means and Z-score obtained similar results (difference between 0.1 and 1.1%). However, K-means obtained better results in setup 1 and Z-score in setup 2. The only exception was in the recall metric that the Z-score obtains the best result in both setups. Regarding the quality metrics, the Z-score average execution time was roughly seven times smaller than the K-means and two times smaller than the box plot. In addition, the Z-score required an average of 14% less memory consumption than the K-means and 8% less than the box plot. Finally, the Z-score required, on average, a CPU usage of 68% less than K-means and 38.5% less than the box plot (**Table** 12).

6.2 Comparison with the state of the art

To compare the approach used in this paper to classify driving behavior and therefore show its benefits, three related work were implemented. A simplified version of VEDAS [74] was developed, however, the piecewise linear approximation algorithm was implemented based on the information provided by Keogh et al. [75] through CEP rules using sliding windows. The work of Aljaafreh, Alshabatat, and Najim [76] that uses fuzzy logic inference for online identification of abnormal driving data was implemented using Fuzzylite [84], a free and open

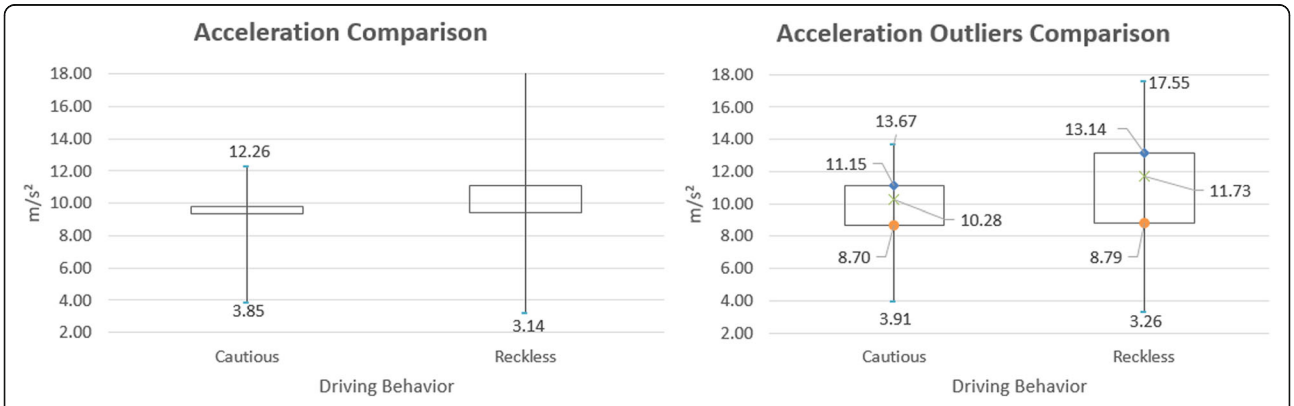

Fig. 27 Acceleration comparison

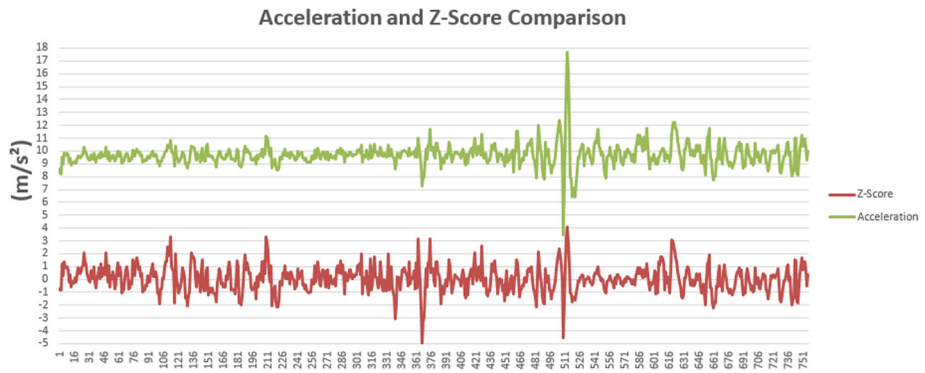

Fig. 28 Acceleration analysis

source fuzzy logic control engine for multiple platforms, including Android. The Quintero, Lopez, and Cuervo [14] proposal was implemented. However, unlike the initial proposal whose processing of fuzzy system variables is done offline by a neural network on a remote server, all processing was done on the smartphone using Neuroph [85]. Nevertheless, we used the same configuration proposed by Quintero, Lopez, and Cuervo [14] – a two-layer neural network, with nine neurons in the intermediate layer, 31 inputs, and trained with a backpropagation algorithm. The Join Driving scoring mechanism was implemented according to Zhao et al. [55]. Finally, a machine learning model (naïve Bayes classifier) [56] using data from smartphone and OBD-II device was implemented. The performance evaluation process of each approach followed the steps described in section 6.1.

Table 13 shows the algorithms' performance metrics. For the K-means, box plot, Z-score and piecewise linear approximation (VEDAS) implemented by CEP rules, Table 13 shows the average obtained in configurations 1 and 2. Through comparison of the results, it can be seen that the approach using neural networks resulted in a superior performance in all the evaluated metrics except precision. In addition, the neural network obtained an extremely low error rate. However, the Z-score performed close to the neural network since, in percentage terms, for each metric the difference was always less than or equal to 1%. Join Driving [55] had a good performance, however, it scored below the previous approaches. The Naïve Bayes classifier approach as presented by Hong, Margines, and Dey [56], clearly underperformed compared to the other approaches. It is worth mentioning that this result is compatible with the work of [86–88] in which a Naïve Bayes classifier did not obtain good results.

Comparing the quality metrics shown in Table 14, it is notable that the Z-score memory consumption was 1.10%, 10.30%, 54.30%, 0.79%, and 50.11% more efficient than the VEDAS, fuzzy logic, back-propagation, Join Driving and Naïve Bayes classifier, respectively. Among the approaches that analyze a set of instances belonging to a time window to determine which are outliers, the Z-score undoubtedly obtained the best result. The fuzzy logic, neural network with backpropagation algorithm,

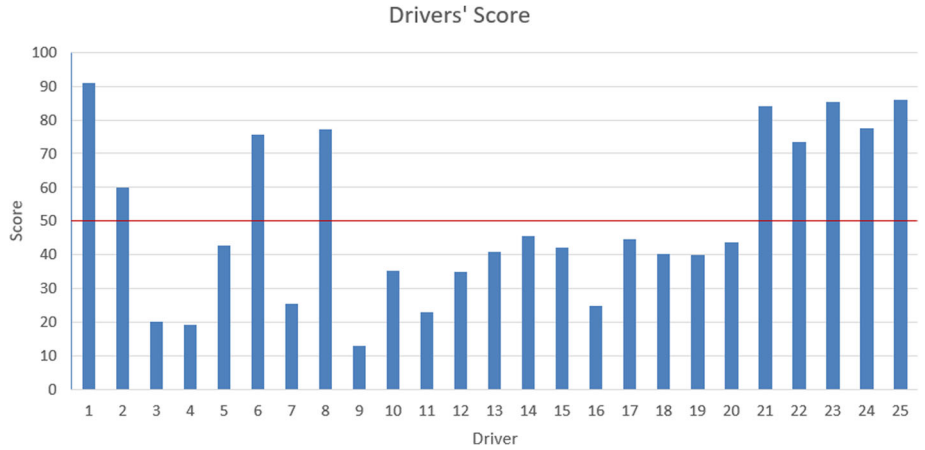

Fig. 29 Drivers' scores within the case study

Table 15 Online Z-score performance

Metric	Value
Accuracy	95.45%
Recall	92.86%
Precision	100.00%
F-Measure	96.30%
Error Rate	4.55%
RAM(MB)	6.35 MB
CPU (%)	7.24%

Join Driving and Naïve Bayes classifier approaches classified an instance as an outlier or normal, respectively, in ~10, ~11, ~10.6 and ~15 milliseconds. However, considering a sensor update rate of 100 Hz and a time window of 20 s, there are 2000 instances to be analyzed. The Z-score takes ~100 milliseconds to analyze all these data and classify each instance as normal or an outlier. This is an average of 0.05 milliseconds per instance analyzed. Through this reasoning, the Z-score takes less time to process an instance. Regarding CPU usage, the box plot and Z-score achieved the best performances. However, the Z-score consumed 73.32%, 73.60%, and 74.13% less CPU than VEDAS, fuzzy logic, and backpropagation, respectively.

6.3 Execution

The smartphone was installed in the center of the vehicle windshield, as shown in Fig. 15. The OBD-II reader device was connected to the OBD-II port of the vehicle and read a variety of data from the vehicle bus. The OBD-II device sent the data streams via Bluetooth to the smartphone. Execution of the case study consisted of performing the process of outlier detection on driving data streams for each volunteer driver.

6.3.1 Data collection

Driver behavior data were collected over seven sunny days and the volunteers drove between 9 AM and 8 PM. Each driver made one trip on the chosen route. Thus, a total of 485 km was covered, comprising 20 h of driving. In addition, before the execution of the case study, the purpose of the study and the absolute confidentiality of personal information was explained to each volunteer.

Then, the chosen route was explained in detail and drivers were asked to drive as they usually would. The volunteer driver was also informed that a driving expert with 15 years' experience would follow him or her during the case study—similar to an expert-based test administered in initial tests to judge driver performance—but we emphasized that our goal was to analyze and classify the driver's behavior as cautious or reckless, and not to approve or disapprove of the actions. This classification served as a ground truth.

The prototype collected data from smartphone sensors (i.e., accelerometer, gyroscope, magnetic compass, and GPS) and vehicle sensors (i.e., speed, RPM, and throttle position percentage) through the OBD-II device. These sensors' data streams were sent to the smartphone via the Bluetooth connection. The connection between the smartphone and OBD-II device was facilitated using generic mobile middleware [79] for short-range communication.

6.4 Extrinsic evaluation of the knowledge model

According to Bramer [83], there is no infallible way of finding the best classifier for a given application. Nonetheless, regarding the quality metrics of CEP-based online outlier detection algorithms, the Z-score obtained an undoubtedly better performance; regarding performance metrics, the Z-score and K-means achieved similar performances, as shown in Section 6.1.1. Thus, for this reason, the online Z-score algorithm was used to analyze

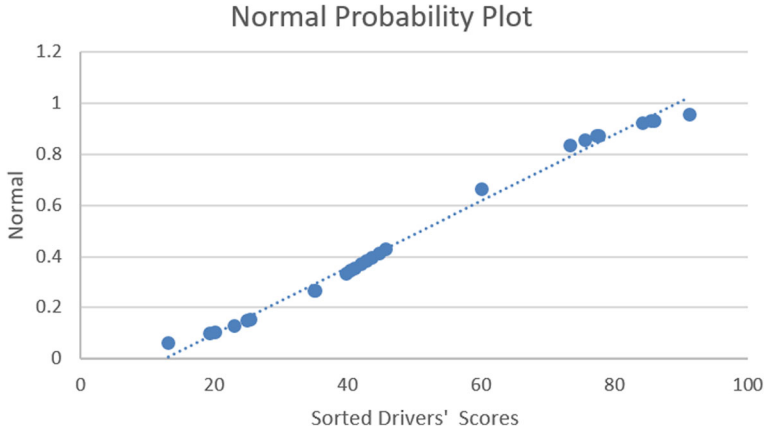

Fig. 30 Normal scores probability

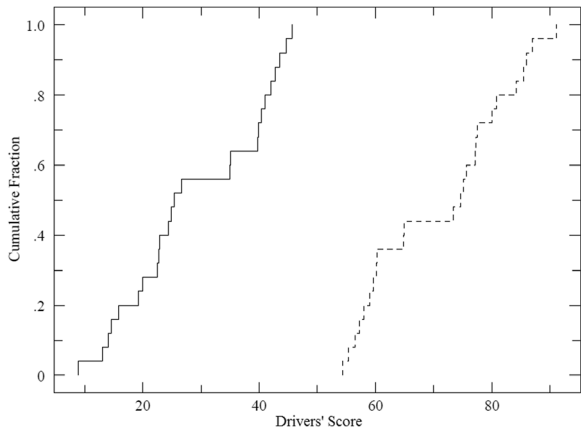

Fig. 31 A Kolmogorov-Smirnov 2 test comparison cumulative fraction

drivers' behavior in the real-world study case. Based on online outlier detection, this section shows and compares the drivers' driving profiles.

Unlike the results obtained by Hong, Margines and Dey [56], both cautious and reckless drivers had substantial differences regarding speed. However, when we analyzed drivers' average Z-distribution, it was not possible to see significant differences, as shown in Fig. 16. This is compatible with results achieved by Hong, Margines and Dey [56]. Nevertheless, through an online analysis, it is possible to identify aggressive maneuvers that result in significant changes in Z-distribution, as shown in Fig. 17. In our case study, by performing offline analysis, as shown in Fig. 18 (left), it is possible to note that more than 75% of speed samples from cautious drivers were lower than 60 km/h (the speed limit) and 50% were lower than 47 km/h. This means that drivers classified as cautious remained below the speed limit most of the time. Excessive speed is a serious violation; however, the driving expert did not realize or did not consider this behavior to be recklessness for some drivers. In contrast, reckless drivers remained above the speed limit most of the time, i.e., only 25% of speed evidence was below 60 km/h. An interesting point lies in the fact that the IQR from cautious drivers was larger than reckless drivers. This means that cautious drivers had a wider variability of speed, and reckless drivers, while remaining above the speed limit in 75% of the evidence collected, maintained little variability in their speed.

Further, as shown in Fig. 18, it is notable that both cautious and reckless drivers produced outliers. On one hand, through an off-line analysis, it is possible to notice an opposing behavior in relation to speed. Cautious drivers remained below the speed limit in 75% of the speed samples while the reckless drivers remained within the speed limit only in 25% of the samples, as shown in Fig. 18 (left). On the other hand, by performing an online speed analysis, in particular, the online Z-score algorithm analysis, we noted that sudden changes identified by outliers in cautious drivers' speed occurred at low speeds, as shown in Fig. 18 (right). Half of the sudden changes occurred in a range between 8 and 43 km/h and 25% of sudden changes occurred in a range between 43 and 54 km/h. It would be valuable to carry out a study to identify the motivation for this behavior. Moreover, reckless drivers exhibited no sudden changes in speeds lower than 40 km/h, as shown in Fig. 18 (right). However, 50% of outliers related to speed occurred above 60 km/h. Another point of note is that even considering driving at extremely high speeds (120 km/h), outliers occurred only up until 75 km/h. These data show that

Fig. 32 Sections with poor quality asphalt

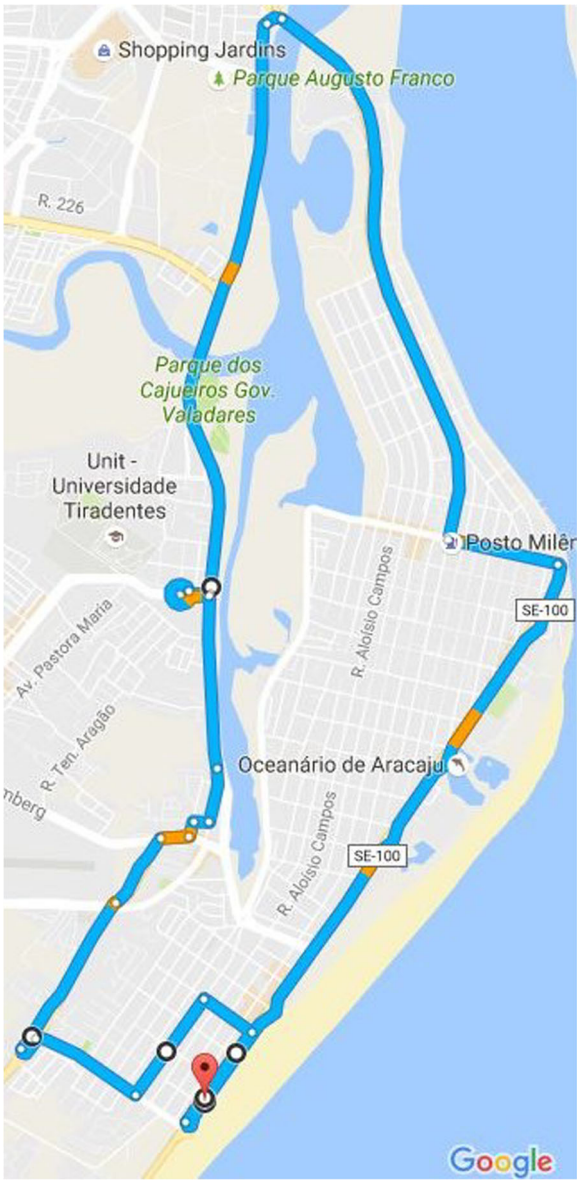

Fig. 33 Route with sections with poor quality asphalt (map © 2016 Google)

speed is appropriate for classifying driver behavior and the online outlier detection approach is a good alternative for classifying drivers as cautious or reckless.

Figure 19 shows the average rpm Z-distribution for four cautious (left) and reckless (right) drivers. This common behavior is repeated for both classes of drivers. Moreover, there are no significant differences between cautious and reckless drivers. However, there is a notable Z-distribution difference in reckless maneuvers, seen while performing the online analysis, as shown in Fig. 20. In maneuvers 1 and 2, the totals of outlier evidence are 17% and 23.5%, respectively. Furthermore, in an offline analysis, as shown in Fig. 21 (left), the rpm from

Table 16 Z-score Online performance *with good asphalt*

Metric	Value
Accuracy	84.00%
Recall	76.47%
Precision	100.00%
F-Measure	86.67%
Error Rate	16.00%
RAM(MB)	6.35 MB
CPU (%)	7.24%

cautious drivers ranged from 1.539–2.086 rpm occurring 50% of the time, while 75% of the time, rpm did not exceed 2.086. It should be noted that reckless drivers had small rpm variation and the reckless Q1 value is 12.16% higher than the cautious Q3 value. Probably in attempts to save time, reckless drivers incurred higher rpms to reach faster speeds as soon as possible, and in doing so, exceeded the speed limit, as shown in Fig. 18. However, Fig. 21 (right) shows that reckless drivers have a broader distribution of outliers, identified by the online Z-score algorithm. Twenty-five percent of reckless drivers' sudden changes in rpm occurred between 3.210–4.779 rpm, and 50% ranged from 1.971–3.210 rpm. These are high values according to the manufacturer's documentation and demonstrate that reckless drivers have an inconsistent driving style.

Figure 22 shows that cautious and reckless drivers' average throttle position Z-distribution is close to the normal distribution. Thus, it is not possible to identify differences between cautious and reckless drivers with an aggregated view of the evidence. However, the online Z-score algorithm identified a different distribution for reckless maneuvers, as shown in Fig. 23. For instance, the maneuvers 1 and 2 had, respectively, 32.67% and 31.14% of evidence classified as outliers. Corroborating with these data, Fig. 24 (left) shows an offline throttle analysis. A consistent throttle usage (percentage usage) is observed, especially for reckless drivers since the IQR value is extremely small. However, regarding identifying throttle position outliers through the online Z-score algorithm, it is noted in Fig. 24 (right) that even smooth throttle changes by cautious drivers have been identified as outliers. However, it is important to highlight that only approximately 0.2% of throttle position samples were higher than 60% of throttle usage. Furthermore, reckless drivers approach the throttle more aggressively. This behavior likely affects fuel consumption and gas emissions.

Analyzing acceleration considering the 3-axis accelerometer, it should be noted that cautious and reckless Z-distribution is practically equal to the normal curve, as shown in Fig. 25. Unlike other studies, such as Hong,

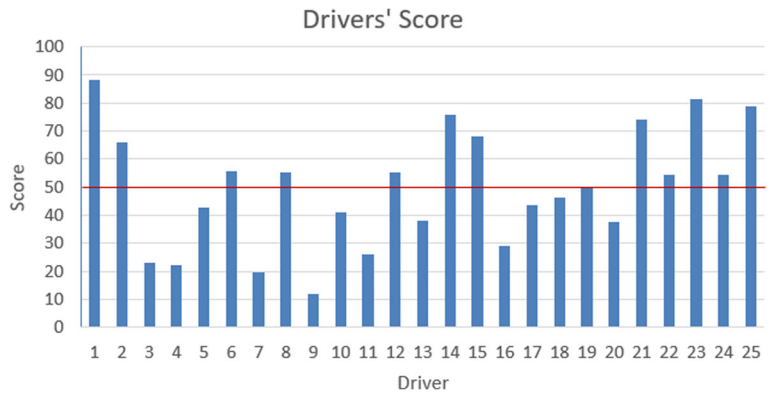

Fig. 34 Drivers' scores within the case study with poor quality asphalt

Margines and Dey [56], that consider only lateral and longitudinal acceleration, we decided to consider 3-axis acceleration because in Brazil many roads are of poor quality. Thus, we believe that an analysis considering 3-axis acceleration more faithfully depicts the circumstances in Brazil. However, to our surprise, and contrary to the results of several studies evaluating driver behavior, such as Hong, Margines and Dey [56], considerable changes in acceleration Z-distribution went unnoticed, even in reckless maneuvers, as shown in Fig. 26. For instance, unlike the aforementioned data, maneuvers 1 and 2 had, respectively, only 9.37% and 7.86% of samples classified as outliers. However, Zhao et al. [55] measured the level of passenger comfort/discomfort based on the effects of human exposure to acceleration according to the threshold defined by International Organization for Standardization [72]. Based on these parameters, and considering that in our case study a stopped vehicle had acceleration equal to 9.8 m/s^2, we assumed that passengers felt comfortable while acceleration was within a range of 8.9–11.2 m/s^2. Figure 26 shows a comparison between offline analysis (left) and online outlier detection (right). In an offline analysis, is possible to see that the drivers were within the comfortable acceleration range in more than 50% of the samples when both IQRs were within a comfortable range for the passengers.

Nevertheless, through the online analysis, shown in Fig. 27, it is possible to note that reckless drivers' events—such as sudden lane changes, abrupt accelerations/decelerations, and jerks—generated more uncomfortable feelings for passengers once more than 75% of the outliers were out of the comfort range. On the other hand, for cautious drivers, approximately 50% of outliers were in the comfortable range. Further, reckless drivers' outliers were, on average, 41.6% higher than cautious drivers' outliers. Figure 28 shows 20 s of acceleration data. It is possible to see nine outlier points (i.e., Z-scores greater than 3 or less than −3). However, only two points were in the aforementioned comfortable range. This demonstrates one of the advantages of our approach: the algorithm analyzes the entire context and not just the individual values. In these two points, variation in the acceleration even within the range considered comfortable was reported as uncomfortable by the

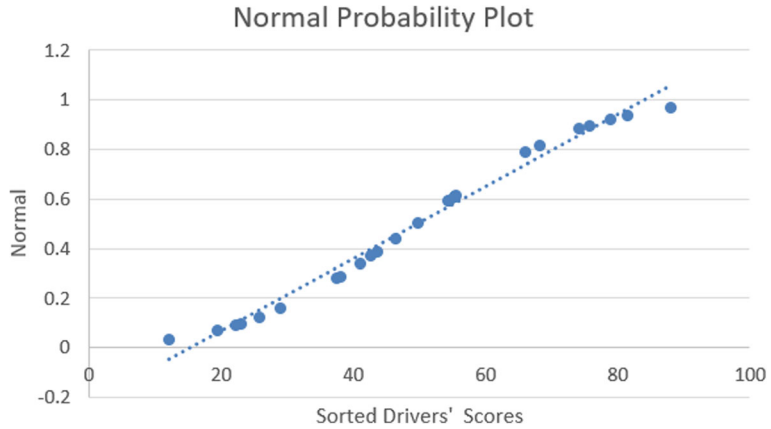

Fig. 35 Normal scores' probability with poor quality asphalt

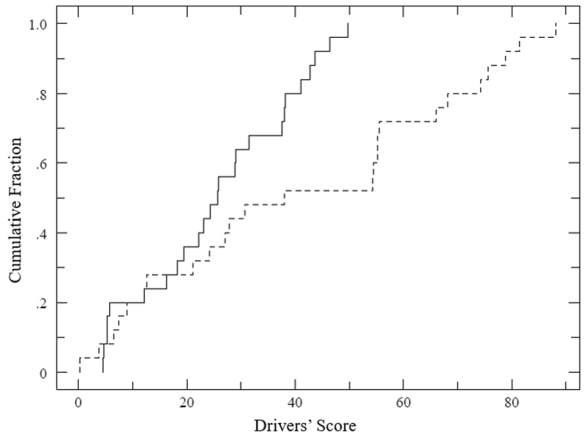

Fig. 36 A Kolmogorov-Smirnov 2 test comparison for drivers' scores in a route with poor quality asphalt

expert driver. This behavior was repeated in other samples.

6.4.1 Scoring driving behaviors

To score the drivers' behaviors, it is necessary to consider that (i) sensors have different acquisition rates. For instance, in this case study, the OBD-II device and smartphone's accelerometer average acquisition rate was 8 Hz and 140 Hz, respectively. Thus, during the data stream processing, we had 17.5 times more evidence of acceleration than speed. And, (ii) certain evidences may have little authority in discriminating driver behavior. To this end, we adapted a statistical mechanism used in document mining to evaluate how important a word is to a document in a collection, called inverse document frequency [89], to identify the importance of an outlier in a data stream.

We defined *outlier frequency* (of_d) as the number of outliers that occurs in a dimension d. Furthermore, we defined the *inverse outlier frequency* (iof_d) of a data instance in dimension d as shown in eq. (6).

$$iof_d = \log\left(\frac{N}{of_d}\right) \tag{6}$$

Thus, the iof_d of a rare outlier evidence is high, whereas the iof_d of a frequent outlier evidence is likely to be low. To weight each outlier evidence in a time window, we combined the definition of outlier frequency and inverse outlier frequency (*ofiof*) as given by eq. (7), where e is the *outlier evidence value* and d is the *dimension*. Therefore, a driver's trip score is given through the weighted average of sum of all *ofiof,* as shown in eq. (8), where t is the number of time windows during the trip.

$$ofiof_{e,d} = of_{e,d} * iof_d \tag{7}$$

$$Score = average\left(\sum_{i=1}^{t}\left(ofiof_{e,d}\right)\right) \tag{8}$$

Figure 29 shows the drivers' score. For this case study, drivers with scores greater than 50 were classified as reckless. This threshold was chosen by analyzing data from six other drivers. These drivers drove on the same route, but three were asked to drive cautiously and the others asked to drive recklessly. The maximum score for cautious drivers was 35 and the minimum score for reckless drivers was 65. Therefore, we consider the threshold of 50 as the upper bound in the classification of cautious drivers and the lower bound in the classification of reckless drivers. Comparing the algorithm classification with the ground truth, an excellent performance can be noted, as shown in Fig. 29. This performance is confirmed in Table 15 and is quite similar to the results showed in Section 6.1.1 in which we used the open dataset provided by Bergasa et al. [57].

In addition to the aforementioned metrics for performance evaluation of the algorithm, we used the Kolmogorov-Smirnov test (KS test) [90]. The KS test is a nonparametric test used to measure the separability of two data distributions from their cumulative distribution functions (CDF). In binary decision systems based on scalar threshold systems, this metric, shown in Fig. 30, is used to measure dissimilarity between distributions [91]. It was accepted that the data are normal with a p-value of 0.27 above the level of significance. A variation of the KS test is used for measuring the classifier discriminating ability, called the KS2 test [91]. The KS2 test evaluates how well a model can distinguish negative (evidence of reckless driving) from positive predictions (evidence of cautious driving) [90]. Thus, the higher the value of the KS2 was, the better the model performed. Based on the threshold and the algorithm score, we calculated the probability of the driver being cautious or reckless. Figure 31 shows the KS2 test plot in which the maximum KS value was 100%.

Finally, although the algorithms presented excellent results, they were not able to identify some reckless behaviors highlighted by Tasca [2], such as tailgating, preventing a vehicle from passing, flashing headlights, horn-honking, improper passing (e.g., cutting too close in front of a vehicle being overtaken), unwillingness to extend cooperation to motorists, unable to merge, changing lanes due to traffic conditions, and running red lights. Furthermore, we observed two additional behaviors: driving with only one hand on the wheel, and/or driving with a hand on the gear. These behaviors are shown in Fig. 15. These behaviors deliberately increase the risk of collision and are thus considered reckless behaviors.

6.4.2 Brazilian road conditions

The chosen route in the aforementioned case study has good quality asphalt. However, as shown in Fig. 32, several Brazilian roads have sections with poor quality asphalt. At these sections, the drivers had to perform both sudden braking and lane changes. Thus, these repeated maneuvers may have occasioned a bias in the analysis of the drivers' driving patterns. For this reason, we performed an additional assessment to verify that even in conditions of poor quality asphalt our approach could accurately classify the drivers. This new route, shown in Fig. 33, is a paved course comprised of streets and avenues ranging from one to three lanes covering approximately 14.5 km in Aracaju-SE Brazil. Thus, a total of 362.5 km was covered, comprising 12.5 h of driving. In addition, the route contains roundabouts, traffic lights, pedestrian crossings, and turns (including 45° and 90° turns). The speed limit on the route was 60 km/h. Furthermore, the drivers described in Section 5.3 and the driver expert were re-invited for this new assessment.

Table 16 shows the Z-score online performance in this new assessment. Compared with results achieved in Table 15 it should be noted that the error rate increased from 4.55% to 16% and all other metrics showed poorer results, excepting RAM and CPU, which remained stable. Although it is still a good result, poor asphalt quality generated noise in the data in such a way that the online Z-score classified it as an outlier. In addition to the error rate that increased 76.56%, the accuracy (i.e., the percentage of correctly classified instances) and recall (i.e., the percentage of instances correctly classified as cautious) stand out for decreasing by 12% and 17.65% respectively.

Analyzing the drivers' scores in Figs. 29 and 34, it can be noted that with good asphalt quality, drivers 12, 14, and 15 were classified as reckless, an increase of 12%. Finally, Fig. 35 and Fig. 36 show the KS and KS2 tests respectively. In the former, as well as in Fig. 30, it was accepted that the data are normal with a p-value of 0.27 above the level of significance; the latter shows the KS2 test plot where the maximum KS value is 48%.

6.4.3 Threat to validity

In this section, we highlight the biases that threaten the validity of the case study. The first possible bias is that drivers did not drive their own cars. This may have led drivers not to drive as they usually would. Additionally, one specific vehicle model was used and the smartphone location inside the vehicle was the same. Other types of vehicle and other smartphone locations may generate different results.

Other conceivable biasing factors are the traffic, which is specific to Aracaju–SE, unique characteristics of the chosen test route, and meteorological conditions. However, we believe that the results of our case study would not change substantially in a setting where these factors were altered. Moreover, a situation highlighted by Paefgen et al. [54] and not considered by us was smartphone use during the case study. For instance, handling a smartphone during a call or the execution of other applications would likely have introduced additional noise. The time that drivers drove may have led to a bias in the behavioral analysis. Other studies should be conducted to assess how the peak time influences drivers' behavior.

We understand that adapting three offline algorithms to perform online processing naturally has a bias. However, the tests show that the algorithms have good efficiency and effectiveness when compared to traditional algorithms. In addition, although there are efficient pattern detection algorithms in the literature such as [70, 74, 92] that were designed to perform continuous data processing, they use spatial data structures, such as R-Trees, S-Trees and Quad-Trees, which provide efficient indexes and query functions for spatial data. Nevertheless, for continuous-mode pattern detection in data streams, these data structures can become troublesome due to their difficulty in accessing and modifying the spatial tree in parallel [93–95]. Further, due to frequent spatial tree modifications, there is an additional cost of frequently balancing the tree in order to reduce its height. For instance, to avoid inconsistencies in the spatial tree index, these algorithms process the data stream sequentially, which can lead to scalability issues when considering a data stream with thousands of data items per second. Other algorithms for online outlier detection in data streams either were not designed for Kontaki et al. [70] or did not have good results [74] in devices with memory and processing constraints.

7 Conclusion and future

In this paper, we introduced an online outlier detection for approaches to driver behavior detection. Unlike many studies that aim to provide faster outlier detection, optimize memory consumption, facilitate parameter settings, or adapt algorithms to perform a distributed processing, our proposal concerned performing an online outlier detection on mobile devices, such as a smartphone with limited computational resources, and did not assume a fixed set of input data. Hence, the main contributions of this paper are as follows. (i) Three classical offline outlier detection algorithms adapted to perform online outlier detection. In addition, these algorithms are operational on mobile devices and able to adapt their behavior based on available computational resources, that is, change sensors' refresh rates and time windows without varying the algorithm accuracy. (ii) A prototype to identify

driver behavior based on online outlier detection. (iii) A dataset, available at http://www.inf.puc-rio.br/~rvasconcelos/DataSet, from real-life drivers that can be used in other experiments. (iv) Experiments that validate and demonstrate the performance of our proposal.

The research on online outlier detection over multiple data streams certainly needs more investigation, but we believe that the findings presented in this paper are an interesting step forward towards reaching online outlier pattern detection in data streams. Our approach delivered an excellent performance, since it can classify the drivers' behavior with an accuracy of 95.45% and 84.00% on good and poor quality roads, respectively. However, considering the encouraging performance evaluation, we are confident that our approach can be used in several other IoT scenarios. The aforementioned results indicate that the Z-score is most appropriate for devices with resource constraints, followed by the box plot algorithm. Although the K-means presents good results, the processing time and CPU usage compromise its use in scenarios with more sensors. For the future, we expect to advance our work along the following lines: (i) adapt the algorithms to scenarios where energy consumption is critical; (ii) perform a distributed online spatial outlier detection; (iii) perform semantic outlier detection over data streams considering data such as weather and traffic conditions; (iv) include historical data in the analysis for identification of emergency situations; (v) combine machine learning and CEP, applying machine learning to the static driving dataset to determine CEP rules; and (vi) regarding driver behavior, identify behavior that precedes accidents and identify the relationship between driving behavior, fuel consumption, and air pollution.

Acknowledgements
The authors would like to thank CNPq and Nucleo de Informação e Coordenação who partly funded this work. In addition, the authors wish to thank the anonymous reviewers for their valuable comments.

Authors' contributions
IOV is the main contributor of this work, undertaken as part of his Ph.D. studies. IOV has participated in the design of this study, designed and implemented the algorithms and the prototype, and conducted the evaluation experiments. ROV, BO, and MR have contributed to the conception of this study. ME, the supervisor of IOV, and MCJ have made substantial contributions to the conception and design of the work and the draft of the manuscript. IOV, ME, and MCJ wrote the manuscript. All authors have read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Publisher's Note

Author details
[1]Department of Informatics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rua Marques de São Vicente 225, Gávea, Office 503, Rio de Janeiro-RJ, Brazil. [2]Department of Informatics, University Tiradentes (UNIT), Aracaju, Brazil. [3]Department of Informatics, Federal University of Sergipe (UFS), Aracaju, Brazil.

Received: 7 December 2016 Accepted: 4 September 2017
Published online: 26 September 2017

References
1. OWSLEY C. Driving Mobility, Older Adults, and Quality of Life. Gerontechnology. 2002;1:220–30.
2. Tasca, L.: A review of the literature on aggressive driving research. Ontario Advisory Group on Safe Driving Secretariat, Road User Safety Branch, Ontario Ministry of Transportation. (2000).
3. Violence, W.H.O., Prevention, I., Organization, W.H. Global status report on road safety 2013: supporting a decade of action. World Health Organization. 2013. http://www.who.int/violence_injury_prevention/road_safety_status/2013/en/.
4. Jacobs G, Aeron-Thomas A, Astrop A, Britain G. Estimating global road fatalities. TRL Report 445. Crowthorne: Transport Research Laboratory; 2000.
5. Johnson DA, Trivedi MM. Driving style recognition using a smartphone as a sensor platform. In: 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC): IEEE; 2011. p. 1609–15. http://ieeexplore.ieee.org/document/6083078/.
6. Hickman JS, Geller ES. Self-Management to Increase Safe Driving Among Short-Haul Truck Drivers. J Organ Behav Manag. 2005;23:1–20.
7. Moosavi V, Hovestadt L. Modeling urban traffic dynamics in coexistence with urban data streams. In: Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing - UrbComp '13. New York, New York, USA: ACM Press; 2013. p. 1.
8. Aslam J, Lim S, Pan X, Rus D. City-scale traffic estimation from a roving sensor network. In: Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems - SenSys '12. New York, New York, USA: ACM Press; 2012. p. 141.
9. Yang Z, Song Y, Wang T, Li Y. Detecting expressway traffic incident by traffic flow and robust statistics. In: 2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet): IEEE; 2012. p. 2537–40. http://ieeexplore.ieee.org/document/6201448/.
10. Terroso-Saenz F, Valdes-Vela M, Sotomayor-Martinez C, Toledo-Moreo R, Gomez-Skarmeta AF. A Cooperative Approach to Traffic Congestion Detection With Complex Event Processing and VANET. IEEE Trans Intell Transp Syst. 2012;13:914–29.
11. Berger CR, Smith E. Intelligent Transportation Systems Provide Operational Benefits for New York Metropolitan Area Roadways: A Systems Engineering Approach. In: 2007 IEEE Long Island Systems, Applications and Technology Conference: IEEE; 2007. p. 1–8. http://ieeexplore.ieee.org/document/4312628/.
12. Perera, C., Zaslavsky, A.B., Christen, P., Georgakopoulos, D.: Sensing as a Service Model for Smart Cities Supported by Internet of Things. CoRR. abs/1307.8, (2013).
13. Xie Q, Zhu J, Sharaf MA, Zhou X, Pang C. Efficient buffer management for piecewise linear representation of multiple data streams. In: Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12. New York, New York, USA: ACM Press; 2012. p. 2114.
14. Quintero MCG, Lopez JO, Cuervo Pinilla AC. Driver behavior classification model based on an intelligent driving diagnosis system. In: 2012 15th International IEEE Conference on Intelligent Transportation Systems: IEEE; 2012. p. 894–9. http://ieeexplore.ieee.org/document/6338727/.
15. Leng Y, Zhao L. Novel design of intelligent internet-of-vehicles management system based on cloud-computing and Internet-of-Things. In: Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology: IEEE; 2011. p. 3190–3. http://ieeexplore.ieee.org/document/6023763/.
16. He, W., Yan, G., Xu, L.: Developing Vehicular Data Cloud Services in the IoT Environment, (2014).
17. Perera, C., Jayaraman, P.P., Zaslavsky, A.B., Christen, P., Georgakopoulos, D.: MOSDEN: An Internet of Things Middleware for Resource Constrained Mobile Devices. CoRR. abs/1310.4, (2013).
18. Vermesan O, Friess P, Furness A. The Internet of Things 2012 New Horizons. Halifax, UK: European Research Cluster; 2012.
19. Vasconcelos RO, Vasconcelos I, Endler M. A middleware for managing dynamic software adaptation. In: Proceedings of the 13th Workshop on

Adaptive and Reflective Middleware - ARM '14. New York, New York, USA: ACM Press; 2014. p. 1–6.
20. Vasconcelos RO, Vasconcelos I, Endler M. Dynamic and coordinated software reconfiguration in distributed data stream systems. Journal of Internet Services and Applications. 2016;7:8.
21. Krishnaswamy S, Gama J, Gaber MM. Mobile Data Stream Mining: From Algorithms to Applications. In: 2012 IEEE 13th International Conference on Mobile Data Management: IEEE; 2012. p. 360–3. http://ieeexplore.ieee.org/document/6341420/.
22. Ferdowsi H, Jagannathan S, Zawodniok M. An Online Outlier Identification and Removal Scheme for Improving Fault Detection Performance. IEEE Transactions on Neural Networks and Learning Systems. 2014;25:908–19.
23. Chandola V, Banerjee A, Kumar V. Anomaly detection. ACM Comput Surv. 2009;41:1–58.
24. Zaldivar J, Calafate CT, Cano JC, Manzoni P. Providing accident detection in vehicular networks through OBD-II devices and android-based smartphones. In: Proceedings - Conference on Local Computer Networks, LCN; 2011. p. 813–9.
25. Quintero MCG, Oñate López JA, Rúa JMP. Intelligent erratic driving diagnosis based on artificial neural networks. In: ANDESCON, 2010 IEEE; 2010. p. 1–6.
26. Li K, Lu M, Lu F, Lv Q, Shang L, Maksimovic D. Personalized Driving Behavior Monitoring and Analysis for Emerging Hybrid Vehicles. In: Proceedings of the 10th International Conference on Pervasive Computing. Newcastle, UK: Springer-Verlag; 2012. p. 1–19.
27. Karp RM. 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, vol. Volume 1 - Volume I. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co.; 1992. p. 416–29.
28. Luckham DC. The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2001.
29. Etzion O, Niblett P. Event processing in action: Manning Publications Co; 2010. http://dl.acm.org/citation.cfm?id=1894960.
30. Agrawal S, Agrawal J. Survey on Anomaly Detection using Data Mining Techniques. Procedia Computer Science. 2015;60:708–13.
31. Gupta M, Gao J, Aggarwal CC, Han J. Outlier Detection for Temporal Data: A Survey. IEEE Trans Knowl Data Eng. 2014;26:2250–67.
32. Hodge VJ, Austin J. A Survey of Outlier Detection Methodologies. Artif Intell Rev. 2004;22:85–126.
33. Gaber M.M, Zaslavsky A, Krishnaswamy S. Mining data streams. ACM SIGMOD Record. 34, 18. 2005.
34. Deng X, Ghanem M, Guo Y. Real-Time Data Mining Methodology and a Supporting Framework. In: Network and System Security, 2009. NSS '09. Third International Conference on; 2009. p. 522–7.
35. CUI, H.: Online Outlier Detection Over Data Streams, (2005).
36. Kianfar J, Edara P. A Data Mining Approach to Creating Fundamental Traffic Flow Diagram. Procedia - Social and Behavioral Sciences. 2013;104:430–9.
37. Heng L. Fast Clustering Optimization Method of Large-Scale Online Data Flow Based on Evolution Incentive. In: 2014 Fifth International Conference on Intelligent Systems Design and Engineering Applications: IEEE; 2014. p. 469–73.
38. Li CW, Jea KF. Using count prediction techniques for mining frequent patterns in transactional data streams. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on; 2012. p. 1155–9.
39. Han C, Sun L, Guo J, Chen X. Mining frequent itemsets in data streams based on genetic algorithm. In: 2013 15th IEEE International Conference on Communication Technology: IEEE; 2013. p. 748–53. http://ieeexplore.ieee.org/document/6820474/.
40. Čermák M, Tovarňák D, Laštovička M, Čeleda P. A performance benchmark for NetFlow data analysis on distributed stream processing systems. In: NOMS 2016–2016 IEEE/IFIP Network Operations and Management Symposium; 2016. p. 919–24.
41. Agarwal S, Prasad BR. High speed streaming data analysis of web generated log streams. In: 2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS); 2015. p. 413–8.
42. Ananthi M, Sreedhevi DK, Sumalatha MR. Dynamic continuous query processing over streaming Data. In: 2016 International Conference on Computation of Power, Energy Information and Commuincation (ICCPEIC); 2016. p. 183–7.
43. Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02. New York, New York, USA: ACM Press; 2002. p. 1.
44. Elahi M, Li K, Nisar W, Lv X, Wang H. Efficient Clustering-Based Outlier Detection Algorithm for Dynamic Data Stream. In: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery: IEEE; 2008. p. 298–304. http://ieeexplore.ieee.org/document/4666541/.
45. Bouchachia A. Incremental learning with multi-level adaptation. Neurocomputing. 2011;74:1785–99.
46. Tang X, Li G, Chen G. Fast Detecting Outliers over Online Data Streams. In: 2009 International Conference on Information Engineering and Computer Science; 2009. p. 1–4.
47. Ma S, Wolfson O, Lin J. A survey on trust management for intelligent transportation system. In: Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science - CTS '11. New York, New York, USA: ACM Press; 2011. p. 18–23.
48. Chan C-Y. On the detection of vehicular crashes-system characteristics and architecture. Vehicular Technology, IEEE Transactions on. 2002;51:180–93.
49. Chaovalit P, Saiprasert C, Pholprasit T. A method for driving event detection using SAX on smartphone sensors. In: 2013 13th International Conference on ITS Telecommunications (ITST): IEEE; 2013. p. 450–5. http://ieeexplore.ieee.org/document/6685587/.
50. White J, Thompson C, Turner H, Dougherty B, Schmidt DC. WreckWatch: Automatic Traffic Accident Detection and Notification with Smartphones. Mobile Networks and Applications. 2011;16:285–303.
51. Singh GR, Dongre SS. Crash Prediction System for Mobile Device on Android by Using Data Stream Minning Techniques. In: 2012 Sixth Asia Modelling Symposium: IEEE; 2012. p. 185–90. http://ieeexplore.ieee.org/document/6243944/.
52. Singh S, Nelakuditi S, Tong Y, Choudhury RR. Your Smartphone Can Watch the Road and You : Mobile Assistant for Inattentive Drivers. In: MobiHoc. New York, NY, USA: ACM; 2012. p. 261–2.
53. Dai J, Teng J, Bai X, Shen Z, Xuan D. Mobile phone based drunk driving detection. In: Proceedings of the 4th International ICST Conference on Pervasive Computing Technologies for Healthcare: IEEE; 2010. p. 1–8. http://ieeexplore.ieee.org/document/5482295/.
54. Paefgen J, Kehr F, Zhai Y, Michahelles F. Driving Behavior Analysis with Smartphones: Insights from a Controlled Field Study. In: Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia. New York, NY, USA: ACM; 2012. p. 36:1–8.
55. Zhao H, Zhou H, Chen C, Chen J. Join driving: A smart phone-based driving behavior evaluation system. In: 2013 IEEE Global Communications Conference (GLOBECOM): IEEE; 2013. p. 48–53. http://ieeexplore.ieee.org/document/6831046/.
56. Hong J-H, Margines B, Dey AK. A smartphone-based sensing platform to model aggressive driving behaviors. In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14. New York, New York, USA: ACM Press; 2014. p. 4047–56.
57. Bergasa LM, Almeria D, Almazan J, Yebes JJ, Arroyo R. DriveSafe: An app for alerting inattentive drivers and scoring driving behaviors. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings: IEEE; 2014. p. 240–5. http://ieeexplore.ieee.org/document/6856461/.
58. Lin N, Zong C, Tomizuka M, Song P, Zhang Z, Li G. An Overview on Study of Identification of Driver Behavior Characteristics for Automotive Control. Math Probl Eng. 2014;2014:1–15.
59. Wang W, Xi J, Chen H. Modeling and Recognizing Driver Behavior Based on Driving Data: A Survey. Math Probl Eng. 2014;2014:1–20.
60. Assent I, Kranen P, Baldauf C, Seidl T. AnyOut: Anytime Outlier Detection on Streaming Data. In: Lee S, Peng Z, Zhou X, Moon Y-S, Unland R, Yoo J, editors. Database Systems for Advanced Applications: 17th International Conference, DASFAA 2012, Busan, South Korea, April 15–19, 2012, Proceedings, Part I. Berlin Heidelberg, Berlin, Heidelberg: Springer; 2012. p. 228–42.
61. Portnoy L, Eskin E, Stolfo S. Intrusion detection with unlabeled data using clustering. In: Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001); 2001. p. 5–8.
62. Javitz, H.S., Valdes, A., Ooie, A.S., Patton, R.D.: The NIDES Statistical Component: Description and Justification. (1994).
63. Dunkel J. On complex event processing for sensor networks. In: 2009 International Symposium on Autonomous Decentralized Systems: IEEE; 2009. p. 1–6. http://ieeexplore.ieee.org/document/5207376/.

64. Cugola G, Margara A. Processing Flows of Information: From Data Stream to Complex Event Processing. ACM Comput. Surv. 2012;44(15:1–15):62.
65. Heiman GW. Basic Statistics for the Behavioral Sciences. Cengage Learning. 2006;(6 ed.):504. https://www.amazon.com/Basic-Statistics-Behavioral-Sciences-6th/dp/0840031432.
66. Cutter GR, Baier ML, Rudick RA, Cookfair DL, Fischer JS, Petkau J, Syndulko K, Weinshenker BG, Antel JP, Confavreux C, et al. Development of a multiple sclerosis functional composite as a clinical trial outcome measure. Brain. 1999;122:871–82.
67. Laurikkala J, Juhola M, Kentala E. Informal Identification of Outliers in Medical Data. In: Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology IDAMAP-2000 Berlin, 22 August. Organized as a workshop of the 14th European Conference on Artificial Intelligence ECAI-2000; 2000.
68. Pakhira MK. A Linear Time-Complexity k-Means Algorithm Using Cluster Shifting. In: 2014 International Conference on Computational Intelligence and Communication Networks: IEEE; 2014. p. 1047–51. http://ieeexplore.ieee.org/document/7065640/.
69. Basu S, Bilenko M, Mooney RJ. A Probabilistic Framework for Semi-supervised Clustering. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2004. p. 59–68.
70. Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y. Efficient and Flexible Algorithms for Monitoring Distance-based Outliers over Data Streams. Inf Syst. 2016;55:37–53.
71. Cao L, Wei M, Yang D, Rundensteiner EA. Online Outlier Exploration Over Large Datasets. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2015. p. 89–98.
72. International Organization for Standardization (ISO): ISO 2631–1-1997: Mechanical vibration and shock-Evaluation of human exposure to whole-body vibration-Part 1: General requirements. (1997).
73. Banovic N, Buzali T, Chevalier F, Mankoff J, Dey AK. Modeling and Understanding Human Routine Behavior. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16. New York, New York, USA: ACM Press; 2016. p. 248–60.
74. Kargupta H, Bhargava R, Liu K, Powers M, Blair P, Bushra S, Dull J, Sarkar K, Klein M, Vasa M, Handy D. VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. In: Proceedings of the 2004 SIAM International Conference on Data Mining. pp. 300–311. Society for Industrial and Applied Mathematics, Philadelphia, PA. 2004.
75. Keogh E, Chu S, Hart D, Pazzani M. An online algorithm for segmenting time series. In: Proceedings 2001 IEEE International Conference on Data Mining: IEEE Comput. Soc. p. 289–96. http://ieeexplore.ieee.org/document/989531/.
76. Aljaafreh A, Alshabatat N, Najim Al-Din MS. Driving style recognition using fuzzy logic. In: 2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012): IEEE; 2012. p. 460–3. http://ieeexplore.ieee.org/document/6294318/.
77. Zhang Y, Lin WC, Chin YKS. A Pattern-Recognition Approach for Driving Skill Characterization. IEEE Trans Intell Transp Syst. 2010;11:905–16.
78. Dangra BS, Bedekar MV, Panicker SS. User Profiling of Automobile Driver and Outlier Detection. International Journal of Innovative Research & Development. 2014;3:49–55.
79. Talavera LE, Endler M, Vasconcelos I, Vasconcelos R, Cunha M, Da Silva e Silva FJ. The Mobile Hub concept: Enabling applications for the Internet of Mobile Things. In: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops). Missouri, USA: IEEE; 2015. p. 123–8.
80. Karthik V. Driver Telematics Analysis: Master's Thesis in Computer Science; 2015.
81. Esper: EsperTech. http://www.espertech.com/. Accessed 11 Dec 2014.
82. Owens, M.: Embedding an SQL Database with SQLite. Linux J. 2003, 2 (2003). http://dl.acm.org/citation.cfm?id=774727.774729
83. Bramer M. Principles of Data Mining. London: Springer London; 2013.
84. Rada-Vilela J. fuzzylite: a fuzzy logic control library. 2014. http://www.fuzzylite.com. Accessed 18 Jan 2017.
85. Sevarac Z, Goloskokovic I, Tait J, Morgan A. Carter-Greaves L Neuroph. http://neuroph.sourceforge.net/. Accessed 19 Jan 2017.
86. Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning - ICML '06. New York, New York, USA: ACM Press; 2006. p. 161–8.
87. Zhang, H., Su, J.: Naive Bayesian Classifiers for Ranking. Presented at the (2004).
88. Kotsiantis SB. Supervised Machine Learning: A Review of Classification Techniques. In: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies. Amsterdam, The Netherlands, The Netherlands: IOS Press; 2007. p. 3–24.
89. Manning CD, Raghavan P, Schütze H. An Introduction to Information Retrieval. Cambridge, England: Cambridge University Press; 2008.
90. Conover WJ. Practical Nonparametric Statistics. , Chap. 6, Third Edition: John Wiley & Sons; 1999. https://www.amazon.com/Practical-Nonparametric-Statistics-3rd-Conover/dp/0471160687.
91. Adeodato PJL, Salazar DSP, Gallindo LS, Sa AG, Souza SM. Continuous variables segmentation and reordering for optimal performance on binary classification tasks. In: 2014 International Joint Conference on Neural Networks (IJCNN): IEEE; 2014. p. 3720–5.
92. Campos MM, Carpenter GA. S-TREE: self-organizing trees for data clustering and online vector quantization. Neural Netw. 2001;14:505–25.
93. Garofalakis M, Gehrke J, Rastogi R. Querying and mining data streams. In: Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD '02. New York, New York, USA: ACM Press; 2002. p. 635.
94. He Y, Tan H, Luo W, Mao H, Ma D, Feng S, Fan J. MR-DBSCAN: An Efficient Parallel Density-Based Clustering Algorithm Using MapReduce. In: 2011 IEEE 17th International Conference on Parallel and Distributed Systems. Washington, DC, USA: IEEE; 2011. p. 473–80.
95. Zheng K, Yu Z, Yuan NJ, Shang S. On discovery of gathering patterns from trajectories. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE): IEEE; 2013. p. 242–53. http://ieeexplore.ieee.org/document/6544829/.