Exploring unsupervised anomaly detection for vehicle predictive maintenance with partial information

Apostolos Giannoulidis
Aristotle University of Thessaloniki
Thessaloniki, Greece
agiannous@csd.auth.gr

Anastasios Gounaris
Aristotle University of Thessaloniki
Thessaloniki, Greece
gounaria@csd.auth.gr

Ioannis Constantinou
Istognosis Ltd.
Nicosia, Cyprus
ioannis@istognosis.com

ABSTRACT

Predicting the need for maintenance in vehicle fleets enhances safety and lessens the downtime. While vehicle manufacturers provide built-in alert systems, these often fail to alert the driver when something goes wrong. However, harnessing the power of data analytics and real-time signals can solve this problem. In this work, we describe a challenging real-world setting with scarce and partial data of failures. We propose a non-supervised approach that detects behavioral changes related to failures avoiding using the raw signals directly to cope with driving behavior and weather volatility. Our solution calculates the differences in the correlations of collected signals between two periods and dynamically creates reference profiles of normal operational conditions tolerating noise. The initial experiments are particularly promising, e.g., we achieve 78% precision detecting nearly half of the failures outperforming the behavior of a state-of-the-art deep learning technique. More importantly, we consider our solution as a specific instantiation of a broader framework, for which we thoroughly evaluate a broad range of alternatives.

1 INTRODUCTION

Predictive maintenance (PdM) is an essential tool for accomplishing the Industry 4.0 vision. Its application to vehicles aims to ensure operation and reduce their downtime [7, 17]. Moreover, success in an optimal schedule of maintenance and service leads to safe usage and cost reduction. One of the most crucial components of a PdM solution in vehicles is the prediction of faults, i.e., the need for non-standard non-periodic service and repair.

Most commonly, the prediction and detection of faults in vehicles concerns the original equipment manufacturers (OEM). Diagnostic Trouble Codes (DTCs)[1] form the basis of most solutions from all manufacturers regarding the detection of vehicle faults in practice. Essentially, these are codes related to a faulty behavior of the vehicle and are produced by the engine control unit (ECU), which is programmed by the OEM to produce a code based on rules upon collected sensor values. There are two types of DTCs: the *pending* and the *stored* ones. The *pending* codes are related to a malfunction observed once and do not repeat, while the *stored* codes indicate a repeating malfunction.

Motivation. Although the main tool used by mechanics to detect failures in a vehicle is the examination of DTCs, there are several cases where DTCs do not manage to capture failures. In other words, there are failures where no DTC was produced beforehand. A representation of such cases is depicted in Figure 1 using a sample of the vehicle fleet dataset that we analyze in

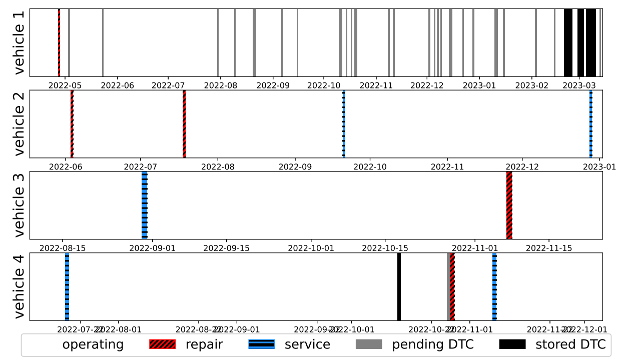


Figure 1: Produced DTCs codes along with repair and service events on 4 vehicles.

this work. In the figure, the events of DTCs, repairs, and services corresponding to 4 vehicles are represented with different colors. As we can see, DTCs are produced before the fault only in one case (vehicle 4) while, for vehicle 1, many DTC codes were produced for a long time after repair without actually needing a repair. Finally, for vehicles 2 and 3, there are no DTC codes before and after the faults.

This sample is representative for the whole fleet and the example indicates that we cannot rely on DTCs for predicting that repairs are needed in the near future. Motivated by this observation, the goal of this work is to devise a practical PdM solution for vehicles without relying on DTCs. However, our setting is challenging, as explained below.

Our setting. The case concerns a fleet of vehicles, which operate daily in both urban and regional areas. The operation of vehicles is monitored with the help of a fleet management system (FMS) platform, where users can see their routes and active hours along with collected data related to the vehicle state, such as DTCs and six Parameter IDs (PIDs) signals communicated using On-Board Diagnostic II (OBDII)[2]. More specifically, the collected PID signals include (i) engine speed in revolutions per minute (rpm), (ii) speed in kilometers per hour, (iii) engine coolant temperature (coolantTemp), (iv) the temperature of the air in the intake manifold (intakeTemp), (v) pressure in the intake manifold or Manifold Absolute Pressure (map-Intake), and (vi) airflow rate (MAFairFlowRate). These signals are a portion of the engine management system-generated signals and are collected at a frequency of one measurement per minute when the vehicle operates.

The vehicles monitored belong to a leasing company that signs a contract with the operator company. The operator company signs a contract with the FMS provider, who is responsible for the PdM solution.

[1] https://github.com/mytrile/obd-trouble-codes/blob/master/obd-trouble-codes.csv

[2] https://en.wikipedia.org/wiki/OBD-IIPIDs

Challenges. Due to the fact that the vehicle operator does not own the vehicles, several events of interest, such as standard services and repairs of vehicles are partially collected. The partial collection concerns the fact that the information regarding those events is available for a subset of vehicles only, while several events of interest take place but they are not recorded. The partial collection of such events makes the problem more challenging, since knowledge is absent regarding (i) the state of vehicles and (ii) failures detected during standard/periodic services that is an obligation of the leasing company and their reports are not communicated to the operator.

Our PdM solution aims to predict the need for urgent maintenance events after an occurred behavior change regarding a vehicle component. Although these events are significant to avoid, they are very sparse. Specifically, the monitored fleet that is examined in this work consists of 40 vehicles, where, for one year of operation, there are 121 events of interest collected concerning only 26 out of 40 vehicles. Out of these 121 events, only 9 of the events are failures. The rareness of failure data essentially rules out supervised learning solutions. The operational data of the 40 vehicles consist of 1.5 million records, while the failure states of the vehicles, constructed using a 30 or 15 days period before each failure, correspond to 3.6% and 1.9% of the dataset, respectively.

Additionally, the use of a particular vehicle in the fleet may vary compared to other vehicles in the same fleet, or its past usage (e.g. urban or cross-regional rides), which makes the comparison of the vehicles' data tricky, as a high difference in the data may be attributed to a different usage rather than an upcoming failure. We further discuss this in Section 2. Overall, our solution needs to be robust both regarding data changes (due to data drifts that, in general, render simple distance-based techniques problematic) and human errors or indifference (i.e., there are a lot of missing event data because the vehicle owner does not care to provide full reports to the operator so that the FMS can be aware of them).

Problem Formulation and Contribution. Having a fleet with a variety of vehicles operating in different conditions, we seek to answer which vehicles should perform maintenance to avoid a serious damage, based on their operational state derived solely from sensor data and (partial) maintenance event recordings. The absence of labels and clear characterizations of the vehicle operation has led us to use a non-supervised solution upon PID data and be robust to the fact that ground truth data is not clean. Finally, we need to ensure that we do not lose the trust of drivers and mechanics, which highlights the need for increased precision in our results.

To solve the problem above, we devise a framework that does not rely on active involvement of a domain expert and revolves around the dynamic building of a reference dataset that corresponds to normal behavior, despite the fact that there is no guarantee that this dataset is free of noise. We investigate a series of alternative design choices and we derive a final solution that achieves up to 78% precision and 44% recall, i.e., a behavior that indicates that the solution is applicable in practice. Interestingly, a self-similarity-based solution yields better results that other techniques, including a state-of-the-art transformer-based unsupervised anomaly detector.[3]

Paper Structure. Section 2 further discusses the dataset and its associated challenges. The proposed framework and the design choices are presented in Section 3. In Section 4, we comparatively evaluate the alternatives and we describe an instantiation of the proposed framework. Next, we discuss the related work and we conclude in Section 6.

[3]https://github.com/agiannoul/NavarchosPdM

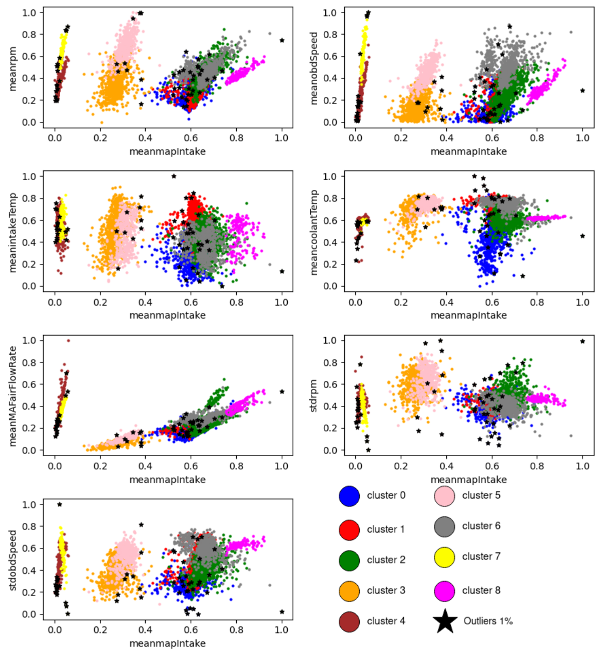


Figure 2: The 9 clusters derived from Agglomerative clustering plotted for different pairs of features along with 1% outliers, notated with star.

2 DATA EXPLORATION

To further understand the nature of our data and the challenges of our case, we perform clustering on the raw sencor data. In our case where data comes from multiple vehicles, it would be interesting to find out if there are any well-separated or meaningful clusters among them indicating (upcoming) failures. First, an aggregation is performed using an one-day timespan, and calculating the mean and standard deviation of each of the PID measurements mentioned above. As a clustering method, we have employed average linkage agglomerative hierarchical clustering [22], which allows for examining the resulting clusters at various levels of granularity. The distance metric is the Euclidean one. In Figure 2, we present 9 clusters plotted using different pairs of measurement types. Moreover, in the figure we can observe the top 1% of outliers, using star notation with black color. We have chosen this number of clusters because we can provide a real-world description to each of these clusters.

More specifically, we can relate each cluster with fleet metadata, like vehicle-model and ride lengths. By examining i) the participation of each vehicle in clusters, ii) the relative location of clusters in the feature pair plot, and iii) the length of the rides in kilometers performed in a day, we result with the following explanations for each cluster: (0) regular rides, (1) extremely small rides, (2) data of a single vehicle, (3) data of a single vehicle, (4) high speed/rpm involving long rides, (5) data of a single vehicle, (6) short rides, (7) data of a single vehicle, and (8) long rides. In summary, from the interpretation of the clusters, we can argue that the difference in the data monitored by different vehicles is highly possible to be attributed to a different type of vehicle (since some vehicles form their own cluster) and usage (since the

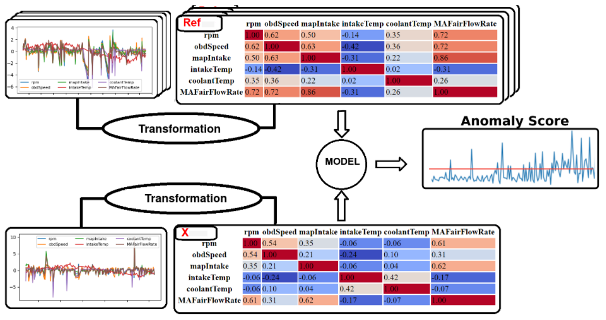


Figure 3: Framework for behavioral change detection.

different types of rides form different clusters), while there is no cluster referring to faulty behavior, i.e., behavior just before an unscheduled repair event.

We now turn our attention to outlier objects, e.g., objects in the figures that are far away from their center. The hypothesis that needs to be tested, provided that vehicle measurements close to failures do not correspond to any cluster, is whether there exist anomalous data that are related to upcoming failures. So, except looking at cluster level interpretation, we have to consider the outliers in our data and check if they are related to failures. Using the well known Lof [3] algorithm, we collect the top 1% of outliers in our data. These are observed with a star notation and black color in Figure 2. To measure how these outliers are related to failures, we measure their time difference from the next failure occurrence. More precisely, each dot in the plots derives from the data of a specific vehicle on a specific date. Thus, for each outlier we gather the failure of its corresponding vehicle and examine a) if the timestamp of outlier is at most 30 days before a failure (and so, we consider that is related to it), b) if there is no failure occurrence after the outlier timestamp at all, or finally, c) if the timestamp of the outlier is at least 31 days before the next occurrence of failure after its timestamp. The two latter categories are considered as non related to failures. Surprisingly, no outlier exists in the (a) category, while outliers of categories (b) and (c) comprise 11% and 89% of the total outliers, respectively.

The main lesson learnt from the data exploration activity is that simple distance-based solutions directly applied to the raw measurements using Euclidean distance do not manage to reveal behavioral changes in vehicle data that can be interpreted as warning signs for upcoming failures.

3 DETECTING BEHAVIORAL CHANGE

3.1 Framework Outline

In this work, we propose a framework that is capable to quantify behavioral changes that correspond to a failure state rather than different usage of the vehicle. The proposed framework can be instantiated in several different manners and the purpose of this work is to investigate the effectiveness of main alternatives. The framework consists of three parts: 1) transformation of data in a form that highlights behavioral changes, 2) construction of a normal reference state of a vehicle, and 3) use of a non-supervised model to produce an anomaly score, as depicted also in Figure 3.

Regarding the data transformation, the goal is to transform the data in a space, where behavioral changes are highlighted. Key alternatives include delta transformation, correlation between signals, frequency-domain transformation, histograms, and others.

The second step is the construction of a reference healthy state of a vehicle, *Ref*. Having a relatively normal state of the vehicle, i.e., a state representation that mostly reflects normal operation conditions while tolerating some noise existence, we can detect deviations in real time based on the difference of new data with the reference state. For example, in case of correlation transformation, comparing how the collantTemp is changing as the speed rises in the reference data and how it is changing in the new data may reveal forthcoming failures. A big difference between the correlation of two signals in the reference data and in the current data denotes a behavior change in the vehicle, which can be due to a fault or some other outer factor. To define a reference state, we use a period of vehicle operation after maintenance (or standard service), assuming that the vehicle operates normally after such events without seeking more guarantees; building the reference state based on such events yields a dynamic solution.

Considering the third step, to quantify the deviation between a reference state and current monitoring data, there are different unsupervised and semi-supervised techniques. Three well-known representative types are similarity-based, reconstruction (deep-learning) models, and regression (or forecasting) ones. In our problem, we investigate four different techniques, two similarity-based, one reconstruction, and one regression-based.

In the framework, we keep the second step constant and we explore multiple solutions regarding the first and the third step. As such, the key rationale behind the framework is to capture the degradation or a failure of a vehicle by comparing its current behavior against an assumed healthy reference state that is updated after each service. Note that this idea has been proven to be particularly promising in several other anomaly detection settings, such as [2] and [8]. Below, we elaborate on our main choices for the first and the third step.

3.2 Data Transformations

Our exploratory analysis investigates four different options regarding data transformation, namely (i) correlation transformation, (ii) mean aggregation, (iii) delta transformation, and (iv) keeping the raw data measurements emitted by the sensors placed on vehicles. For all four options, before we transform the data, we first filter out records that correspond to the stationary state of the vehicle and sensor faulty data. Always, the objective is to reveal changes corresponding to the near-failure operation of vehicles given the partial information that is available.

Correlation transformation refers to the calculation of the cross-correlation between the different available features, i.e., measurement types. Using a sliding window over raw data, we calculate the correlation between signal data. If we consider that the initial data features are f_n, after cross correlation we result in a symmetric $f_n \times f_n$ matrix, which can be considered as a vector with $\frac{f_n*(f_n-1)}{2}$ features. The intuition behind this transformation is that different usages of vehicles may produce similar correlations. For example, we expect that speed and rpm are positively correlated regardless of whether the vehicle performs urban or regional rides, while a difference in the correlation of two signals may refer to a failure state. For the second option, we use a mean aggregation of raw data. Using the same sliding window as in the correlation transformation, we simply take the mean over each feature in the window as representatives of the corresponding period. Although in Section 2 we showed that such transformation has led to differentiation of the data based on the vehicle usage and type rather than their operational state, it is essential to further test it in practice.Finally, delta transformation refers to the difference between preceding and proceeding measurements. To transform the data, for each measurement, we subtract its previous values, which is similar to calculating a derivative of each measurement. This transformation has been used in vehicular data [9] providing good results, and so we consider it as a possible choice.

3.3 Closest pair-based anomaly detection

This technique and the following ones refer to the instantiation of step 3.

Closest Pair Detection leverages a healthy period of operation *Ref* to calculate anomaly scores from upcoming data. This technique monitors each feature of the input separately. When a new sample arrives, the technique assigns an anomaly score for each feature, using its distance from its closest neighbor in *Ref*. This results in f_n or $\frac{f_n*(f_n-1)}{2}$ if correlation is used as data transformation anomaly scores per sample. Alarms are produced from violation of the threshold in any of f_n anomaly scores and are accompanied by a description with the the feature that triggered it.

To decide a threshold value, we could use an arbitrary value or a thresholding technique. We decided to use the self-tuning thresholding from [8], since it matches our case and can calculate a different threshold for each different vehicle, using the same parametrization. This method calculates a threshold based on anomaly scores in supposed normal data using a factor. Specifically, the threshold is equal to the mean of anomaly scores plus the standard deviation of them multiplied by a factor. For the calculation of the threshold, we use a small portion of healthy data each time, while using the same factor for all vehicles.

3.4 Grand

Grand is a distance-based solution combined with statistical tests to predict failures in vehicles [17]. The proposed solution models based on the wisdom of the crowd. Essentially, to measure the deviation score of a vehicle, it measures the dissimilarity of samples using distance-based techniques, such Lof and k-nearest neighbours (Knn) to quantify the strangeness of a sample in relation to the data of other vehicles in normal data.

In our case, vehicles differ from each other, and so, we follow another strategy proposed by the same authors. This strategy differs only in the calculation of normality data, which is now formed using an operation period of the same vehicle, as the sample we examine. Then, the same methods are used to define the strangeness and deviation score of the sample [4].

In more detail, to detect deviations, Grand calculates the outlierness of a sample x_t in relation to the rest of the samples in the normal Reference data *Ref*. This calculations is performed using three non-conformity measures: (i) *Median*, which is the distance of a sample from the median of *Ref* (i.e., its most central pattern), (ii) *Knn*, which is the average distance to the k-nearest neighbors of a sample within *Ref*, and (iii) *Lof*, which is the local outlier factor [3]. Leveraging the the score derived from non-conformity measures, the outlierness (*z_score*) of the examined sample and for all samples in the *Ref* is computed. After that, the p-value p_t of the examined sample x_t is computed using [6].

[4]https://github.com/caisr-hh/group-anomaly-detection

3.5 Reconstruction-based (deep learning) anomaly detection

Except similarity-based techniques, reconstruction-based anomaly detection could be used to predict failures in the PdM context. These methods usually involve non-supervised Deep Learning (DL) architectures, trained on data considered normal or healthy, to reconstruct their input.

Specifically, the model is trained to produce output equal to its input, meaning that having as input a sample x and output y, the training objective is x and y to be equal ($x = y$). The y is referred to as a reconstruction of x, while the distance between x and y (i.e. the loss of the model) is referred to as a reconstruction error. So, providing the model only with normal data in the training phase, it learns to reconstruct correctly data that are similar to normal ones. Then, in the online phase, when the model tries to reconstruct a new upcoming sample x_i resulting in output y_i, we measure its reconstruction error (distance between y_i and x_i) and use it as an anomaly score to produce alarms.

In the proposed framework, TranAD (Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data) model [20] could be used to instantiate the model choice in the third step. This model consists of a state-of-the-art transformer DL model and is endowed with valuable properties, such as the capability to operate with fewer training data than competitors and to reconstruct data after a small number of epochs. By training the TranAD model to *Ref* data (provided by the framework), the model can be used as an anomaly detector, since it learns to reconstruct correctly data similar to trained ones and deviates in unknown data.

3.6 Forecasting/Regression methods

Another way to detect anomalies that act as warnings of near future failures is using forecasting or regression models. The intuition here is to train a model using healthy data (*Ref*), to regress future values based on historical data, or to regress a target feature from a data sample when provided with the rest of the features in the input. By training only using healthy data, we expect that, when the model is asked to perform such a task on close-to-failure data, it will produce higher loss, since it is not trained to model data dissimilar to healthy ones. So, during inference, we calculate the anomaly score as the distance between the prediction and actual value. This methodology can be followed both using a forecasting model, to forecast all or one of the input features, and using a regression model to regress one of the features given the remaining ones as input.

In our case, we use the XGBoost [5] regressor as a model. Specifically, we use as many XGBoost models as the number of features in our data, each one trained to predict the value of one of the features, having as input the remaining feature values. So, we train f_n models using the *Ref* data, each time having a different feature as the target one. After that, in inference time, when a new sample arrives, we perform the same procedure on data as in training, this time at a record level, and pass the the data to the models. Each model tries to predict the value of its target feature, where we use the loss of the prediction (i.e. the distance between the predicted and actual value) as the anomaly score.

Similarly to closest-pair detection, when an alarm occurs from one of the regressors, this is accompanied by a description that refers to the target feature that the model tried to predict. I.e., these two alternatives are inherently more amenable to explanations.

4 EVALUATION

Our main evaluation comprises a real-world dataset provided by Istognosis FMS, named as Navarchos (meaning Admiral in Greek). The experiments were conducted in two different settings. The first is *setting40* that uses all 40 vehicles data. However, in this dataset there are 14 vehicles with no recorded events. Therefore, we also experiment with a cleaner dataset that is a subset of the former one using using the 26 vehicles out of 40, for which there is at least one event of service or repair during its lifetime (even if it is at the end). We refer to this second setting as *setting26*. As explained in Section 2, both settings are characterized by partial information, although events in the first setting are significantly less informative than in the second. Overall, in only 9 of the 26 vehicles, for which significant events are recorded, there are repair events recorded; remember repair events imply the need for non-periodic maintenance action to take place and PdM's goal is to alleviate this need.

The main metric we employ is $F_{0.5}$, which, compared to F_1 does not use the harmonic mean of *precision* and *recall*, but equals to $(1+\beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$ with $\beta = 0.5$. This results in precision having more weight than recall, and reflects the real-world requirements that FMS should not raise alarms unnecessarily.

To calculate the recall and precision, we use a *prediction horizon (PH)* period, which ends with a repair event. where one or more alarms that fall within *PH* are counted as one true positive instance, while each alarm outside of *PH* is counted as a false positive.

To allow for a fair comparison, we use the same thresholding technique for all methods except Grand Inductive (which is the only one that produces a score between 0 and 1), where a constant threshold value is used. To measure the performance of each method, we use multiple factors regarding the thresholding technique, while, in the case of grand inductive, we use several constant values thresholds.

Finally, we clarify that the purpose of this evaluation is to investigate if there is a dominant or at least a preferred choice with regards to the alternatives presented in Section 3 with a view to developing an operation PdM solution for the Navarchos FMS platform. Therefore, at the end of the evaluation section, we present the complete solution that we advocate adopting in our use case.

4.1 Result Discussion

The summarized results can be seen in Figures 4 and 5 for *setting40* and *setting26*, respectively. In each of the figures, we can observe the effectiveness of different techniques for each data transformation choice. The final results of $F_{0.5}$ are depicted with a darker color for *PH*=15 days and with a lighter color for *PH*=30 days on the same bar. In general, results in *setting26* are better than those in *setting40*, which is somehow expected, since the additional vehicles in *setting40* can only contribute to false positives despite the fact that in reality there may exist actually failures unknown to us due to lack of systematic recording. Also, results for *PH*=30 days are either the same as or better than those for *PH*=15 days. The highest $F_{0.5}$ value observed is 0.68, which corresponds to 78% precision and 44% recall, i.e., a behavior that indicates that the solution can be applied in practice.

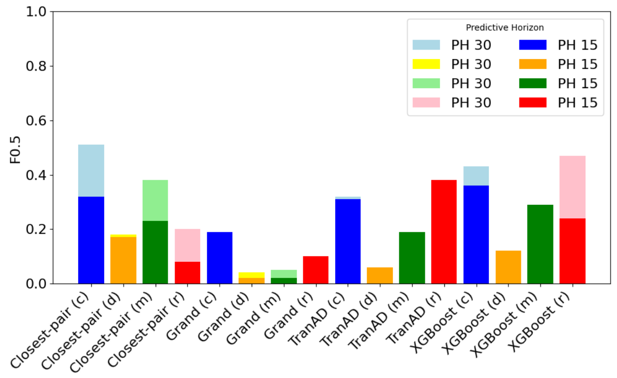


Figure 4: Results for the different data transformation and Predictive horizons of setting40 data.

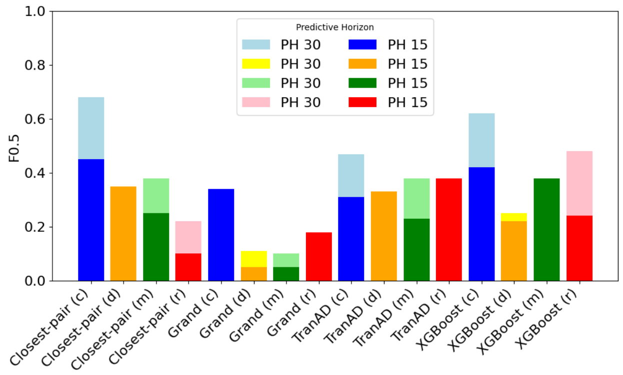
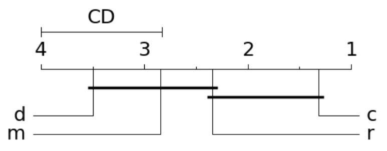
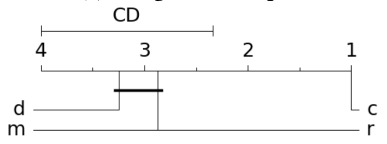


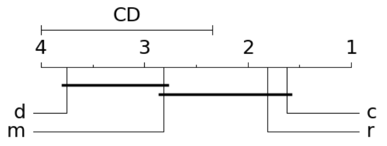
Figure 5: Results for the different data transformation and Predictive horizons of setting26 data.



(a) Using all techniques.



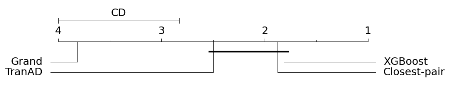
(b) Using similarity-based techniques.



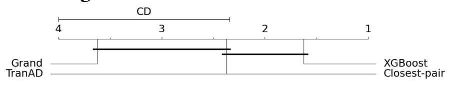
(c) Using XGBoost and TranAD.

Figure 6: Critical diagrams for data transformation choices.

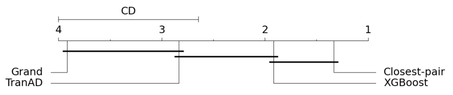
Based on the presented results, we first discuss the choices regarding data transformation. At first glance, we can see that the



(a) Ranking over all data transformation choices.



(b) Ranking over correlation and raw data transformation choices.



(c) Ranking over all data transformations, except raw data.

Figure 7: Critical diagrams for anomaly detection techniques

data transformation options have a different impact on the performance of each technique. Interestingly, *XGBoost* and *TranAD* exhibit their best or close to their best results using raw data, whereas on the other hand, the two similarity-based solutions, *Grand* and *Closest-pair*, could not perform well on raw data. This observation is supported by the ranking of the different transformations, produced by the Friedman test followed by Wilcoxon signed-rank test [21] (using autorank tool [10]) at three levels of granularity, as shown in Figure 6. These levels correspond to considering all techniques, only the similarity-based ones, and only the non-similarity based ones, respectively. Specifically, we observe that the ranking is consistent in all three cases, with the *correlation (c)* being the best, followed by *raw (r)* data, *mean aggregation (m)*, and finally, *delta (d)* transformation.

A significant difference between the first (correlation) and second (raw) transformation is observed only in similarity-based techniques. This can be explained that the functionality of the similarity-based techniques is based on the distance between samples, while XGBoost and TranAD try to learn the relationship between the various features. A smaller observation regarding delta transformation is that, although it yields a lower score in most cases of *setting40*, this is not the case in *setting26*. However, the key point is that the fact that correlation's highest performance has a noticeable statistical significance for similarity-based techniques and, as shown already, *Closest-pair* achieves the highest $F_{0.5}$; this observation is pivotal for establishing a dominant solution later.

More specifically, by observing the results at the technique level, the *Closest-pair* detection method produces the best results using the correlation transformation for both settings and predictive horizons (except when *PH*=15days in *setting40*, where TranAD is slightly better).

By ranking the different techniques using the same procedure as with data transformation approaches in Figure 7, considering both settings and all data transformations, we observe that *TranAD*, *Closest-pair*, and *XGBoost* exhibit statistically significant better results than the *Grand* inductive method. Moreover, we can observe that *XGBoost* is first in ranking and very close to the *Closest-pair* detection solution. This can be observed also from the results of the techniques in Figures 4 and 5, where *XGBoost* shows increased robustness regarding the different choices of

	Grand	Closest-pair detection	TranAD	XGBoost
raw	5162	**823**	62350	5386
delta	4989	**761**	62868	4922
correlation	27	**19**	110	139
mean agr.	32	**14**	128	73

Table 1: Execution time in seconds.

Setting	PH	F0.5	F1	Precision	Recall
setting26	15 days	0.38	0.40	0.36	0.44
setting26	30 days	0.68	0.57	0.78	0.44
setting40	15 days	0.30	0.35	0.29	0.44
setting40	30 days	0.50	0.48	0.52	0.44

Table 2: Analytical results of the best configuration (the same method parameters are used for all depicted results).

data transformation except delta transformation. So we could argue that, if someone prefers not to be engaged in fine tuning of techniques regarding data forms and parameters, the *XGBoost* is an attractive solution.

Calculating the ranking of the techniques in two additional granularities, using only the transformations of correlation and raw data in Figure 7b, and using all transformations except raw data in Figure 7c, we can observe that the ranking of the techniques changes. *TranAD* and *XGBoost* are favored when raw data transformation is considered in the experiments, which means that they are capable to perform (relatively) well on these data. Again this can be explained from their more complicated functionality, which allows the extraction of knowledge using raw data (both *TranAD* network and *XGBoost* use learnable parameters).

Finally, to check the applicability in real-time scenarios, the aggregate execution time of the different techniques (using the different data transformations) are presented in Table 1. All experiments are conducted on a single computer using an Intel Core i5-6500 CPU clocked at 3.20GHz with 4 cores and RAM 16 GB, and are implemented using Python 3.8. All techniques can be computed on a daily basis easily, but *Closest-pair* is an order of magnitude faster than its competitors. Below, we present its application in real-time.

4.2 Complete Solution

After exploring the different choices of data transformations in step 1 and different approaches to modeling the deviation of vehicles in step 3, we conclude a complete instantiation of the proposed framework for detecting failures in specific vehicles, as presented in Algorithm 1. Based on our results, we opt for *Closest-pair* over *correlated* data. The summary of the performance data of this solution is provided in Table 2.

The operation of timely fault prediction can take place in a streaming environment. Each time a new data record arrives, we first check if it refers to an event of interest or PID measurements. In case of a new event, we further examine if it consists of an event of interest that triggers the creation of a new normal reference data, e.g., a standard service is performed, therefore the reference profile is updated. In case of a need for new reference *Ref* data, we discard the old data. If the new sample refers to measurements, the first action is to perform the data transformation, which, in our case refers to correlation values. So we pass the data to a data transformer, which is responsible for keeping buffers of raw

Algorithm 1 Detecting failures on vehicles

```
Require: $th_{factor}$, $profile\_length$
  $buffer \leftarrow []$
  $Ref \leftarrow []$
  $tran \leftarrow Correlation()$
  while $Operating$ do
      $sample_{new} \leftarrow new\_data$
      if $sample_{new}$ is event then
          if $sample_{new}$ triggers Reset then
              $Ref \leftarrow []$
      else if $sample_{new}$ is data then
          $transformer.collect(sample_{new})$
          if $tran.ready()$ then          ▷ wait until buffer is full.
              $x \leftarrow tran.transform(sample_{new})$
              if $len(Ref) < profile\_length$ then
                  $Ref \leftarrow Ref + [x]$
              else if $len(Ref) < profile\_length$ then
                  $closest\_pair\_detection.fit(Ref, th_{factor})$
              else
                  $alarm \leftarrow closest\_pair\_detection.predict(x)$
```

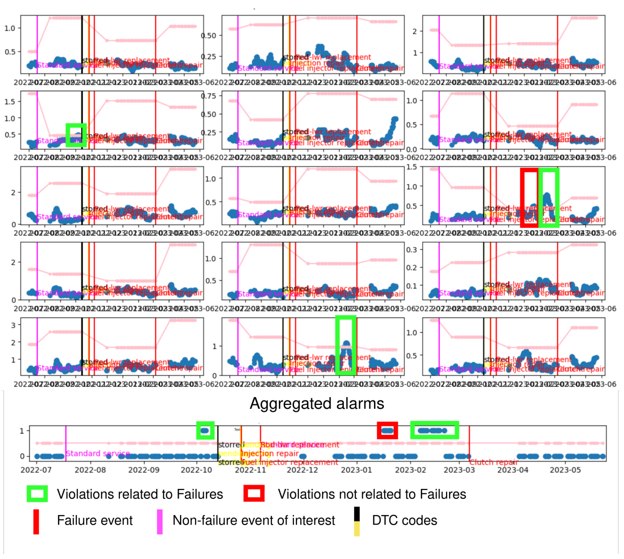


Figure 8: Results of one vehicle using closest pair detection after correlation data transformation.

data, to calculate the wanted transformation. After collecting the transformed data x, we iteratively populate the *Ref* set until it is full. When the *Ref* is ready, we pass it to the anomaly detection method, i.e., the *Closest pair detection*.

For a better view regarding the results of the closest pair detection using correlation transformation, we present the actual results of the technique (anomaly scores for each feature produced from correlation transformation) for one vehicle in Figure 8. In this example, we observe the different anomaly scores the technique produced for each feature for all data of a particular vehicle. Moreover, in pink color, we observe the threshold, while failures and non-failure events are marked with red and purple vertical lines, respectively. Finally, at the bottom of the image, we observe the aggregated results, where violations have led to alarms.

Setting	PH	F0.5	F1	Precision	Recall
setting26	15 days	0.18	0.23	0.16	0.44
setting26	30 days	0.58	0.36	1.00	0.22
setting40	15 days	0.11	0.14	0.10	0.22
setting40	30 days	0.45	0.32	0.66	0.22

Table 3: Analytical results of *Closest-pair* on correlation data without resetting upon service events (here, in each row the results correspond to different choice of threshold).

In the above example, someone can observe that the threshold value is neither constant nor the same over the different features. The reason is that in Step 2 of our framework (construction of normal reference), each time a service or repair event arrives, we build the reference data *Ref* again, and so the techniques are updated with the new data, along with the threshold calculation. Using the *PH*=30 days, we can characterize whether the alarms are related to the failure (true positives) or non-related (false positives), which are depicted in the figure with green and red rectangles, respectively.

Before closing, we evaluate one of the design choices in the 2nd step of the framework, where we define how the healthy representation is defined through *Ref*. Although this choice is highly dependent on domain knowledge, we also test the choice of ignoring all service events, and resetting the *Ref* solely after repairs of failures. This choice forces the framework to stick to the initial state of the vehicles as *Ref* (when vehicle starts operating for the first time after its monitoring), for all vehicles without repairs events. The results of such choice are depicted in Table 3, where, compared to Table 2 either the precision drops too much to keep the same amount of failures detected, or the recall drops up to on 0.22 (2 out of 9 failures detected). This performance degradation occurs despite the fact that we fine tune each row separately. This final experiment proves the importance of leveraging all information available despite the fact that is known to be incomplete.

5 RELATED WORK

In this section, we make a comparative discussion of state-of-the-art PDM solutions. In our problem, we deal mainly with signal data. Although we collect DTC codes (which can be seen as events), our vehicle fleet is assembled from new vehicles, resulting in a very spare appearance of DTCs. So solutions such as [14] and [18], which perform an event-based analysis to predict failures, are difficult to apply. Nevertheless, discretizing the signal input and creating artificial events is an interesting direction for future research.

Our framework focuses on a non-supervised methodology, where the majority of works in PdM rely on supervised methods requiring labeled data. Although the use of deep learning methods could boost efficiency, these methods need a large amount of data [19].

Turning our attention to PdM solutions tailored to vehicles, the proposal in [15] leverages the prediction error of a Multi-Layer Perceptron (MLP) to detect faults in vehicles. This is achieved by training the MLP for a regression task, where several vehicle signals are used to predict the value of the engine load. The rationale behind this approach is that the MLP will learn to estimate the engine load with low prediction loss when data are similar to training ones, but will produce high loss otherwise. Several works, including [1, 11, 16, 23], have implemented this scheme with more advanced models than a simple MLP. One drawback of this approach, depending on the underlying model, is the need for large amounts of data per vehicle to be tagged as healthy by an expert. When raw data are used, another downside if we attempt to transfer the solution in our use case, is that, in our case, the vehicles perform occasionally different types of rides, and so, for each vehicle, we have to carefully add a balanced amount of such rides to avoid a high loss in the event of rare rides that do not constitute a failure. In any case, we test similar, if not a more advanced methods, regarding step 3 of our framework, namely TranAD and XGBoost.

In [4], the authors predict the time until the next failure in vehicles, by integrating contextual information, such as GIS and weather conditions. They accomplish this by utilizing supervised techniques like MLP, random forest, and gcforest. However, in our particular case, we do not have enough data to train such models as we have a small number of failures compared to the large number of vehicles we have. The authors in [12] try to produce real-time alarms aiming to avoid near-future failures in aerial vehicles. To accomplish that, they leverage the isolation forest method. Such a method could become an option for the third step in our framework, but XGBoost that we tested is expected to behave at least as well as IF.

Finally, the Grand PdM solution for fleets is presented in [17] and extended in [7] and [13]. Although these methods are unsupervised, this solution is applied to similar vehicles (buses) that perform similar urban rides. Moreover the application of such techniques to raw data, in our case, is irrelevant because we do not seek anomalies in terms of the deviation of selected data to their history or other vehicles, given the fact that an instance of raw data may be an anomaly due to the weather, driver, and the usage of a vehicle when, as shown in Section 2. Nevertheless, after our preliminary analysis showed that failures can be spotted by comparing correlations between periods, we can transform periods of raw instances into such features and we have explicitly tested the applicability of the Grand Inductive method from [17].

6 CONCLUSION AND FUTURE WORK

We deal with a challenging PdM scenario in a vehicle fleet management system, where the failure information is partial and scarce. We suggest a generic framework for predicting serious failures of vehicles. In addition, we perform an exploratory and comparative analysis regarding instantiating the steps of the proposed framework, examining different combinations of anomaly detectors and data transformations. This exercise provides interesting remarks, such as that different techniques are influenced differently from the data transformations, where the best transformation for our case is working on the correlations of the individual signal measurements. To this end, we also propose a dynamic similarity-based approach for detecting failures, which, along with an appropriate transformation of data, achieves the best results, surpassing the performance of other state-of-the-art techniques, including deep learning ones. Notably, even if the manner we finally instantiate the framework is different from the most appropriate one in other cases sharing the fact that no supervised techniques are applicable and the information regarding the operation environment is partial and evolving, the framework is still applicable and the investigation process described in this work can be followed by third parties.

Acknowledgements. The research is funded under the programme of social cohesion "THALIA 2021-2027" co-funded by the European Union, through Research and Innovation Foundation.

REFERENCES

[1] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. Usad: Unsupervised anomaly detection on multivariate time series. In *Proc. of the 26th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*. 3395–3404.

[2] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Automated Anomaly Detection in Large Sequences. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 1834–1837. https://doi.org/10.1109/ICDE48307.2020.00182

[3] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. *SIGMOD Rec.* 29, 2 (may 2000), 93–104. https://doi.org/10.1145/335191.335388

[4] Chong Chen, Ying Liu, Xianfang Sun, Carla Di Cairano-Gilfedder, and Scott Titmus. 2020. Automobile Maintenance Modelling Using gcForest. In *2020 IEEE 16th Int. Conf. on Automation Science and Engineering (CASE)*. IEEE, 600–605.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[6] Liang Dai and Mohamed-Rafik Bouguelia. 2020. Testing exchangeability with martingale for change-point detection. arXiv:1810.04022 [math.ST]

[7] Apostolos Giannoulidis and Anastasios Gounaris. 2023. A context-aware unsupervised predictive maintenance solution for fleet management. *Journal of Intelligent Information Systems* 60, 2 (01 Apr 2023), 521–547. https://doi.org/10.1007/s10844-022-00744-2

[8] Apostolos Giannoulidis, Anastasios Gounaris, Nikodimos Nikolaidis, Athanasios Naskos, and Daniel Caljouw. 2022. Investigating Thresholding Techniques in a Real Predictive Maintenance Scenario. *SIGKDD Explor. Newsl.* 24, 2 (dec 2022), 86–95. https://doi.org/10.1145/3575637.3575651

[9] Flavio Giobergia, Elena Baralis, Maria Camuglia, Tania Cerquitelli, Marco Mellia, Alessandra Neri, Davide Tricarico, and Alessia Tuninetti. 2018. Mining sensor data for predictive maintenance in the automotive industry. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 351–360.

[10] Steffen Herbold. 2020. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software* 5, 48 (2020), 2173. https://doi.org/10.21105/joss.02173

[11] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proc. of the 24th ACM SIGKDD Int. Conf. on knowledge discovery & data mining*. 387–395.

[12] Samir Khan, Chun Fui Liew, Takehisa Yairi, and Richard McWilliam. 2019. Unsupervised anomaly detection in unmanned aerial vehicles. *Applied Soft Computing* 83 (2019), 105650. https://doi.org/10.1016/j.asoc.2019.105650

[13] Patrick Killeen, Bo Ding, Iluju Kiringa, and Tet Yeap. 2019. IoT-based predictive maintenance for fleet management. *Procedia Computer Science* 151 (2019), 607–613. https://doi.org/10.1016/j.procs.2019.04.184 The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

[14] Panagiotis Korvesis, Stephane Besseau, and Michalis Vazirgiannis. 2018. Predictive Maintenance in Aviation: Failure Prediction from Post-Flight Reports. In *2018 IEEE 34th Int. Conf. on Data Engineering (ICDE)*. IEEE, 1414–1422.

[15] Alessandro Massaro, Sergio Selicato, and Angelo Galiano. 2020. Predictive Maintenance of Bus Fleet by Intelligent Smart Electronic Board Implementing Artificial Intelligence. *IoT* 1, 2 (2020), 180–197. https://doi.org/10.3390/iot1020012

[16] Hengyu Meng, Yuxuan Zhang, Yuanxiang Li, and Honghua Zhao. 2020. Spacecraft Anomaly Detection via Transformer Reconstruction Error. In *Proc. of the Int. Conf. on Aerospace System Science and Engineering 2019*, Zhongliang Jing (Ed.). 351–362.

[17] Thorsteinn Rögnvaldsson, Sławomir Nowaczyk, Stefan Byttner, Rune Prytz, and Magnus Svensson. 2018. Self-monitoring for maintenance of vehicle fleets. *Data Mining and Knowledge Discovery* 32, 2 (March 2018), 344–384. https://doi.org/10.1007/s10618-017-0538-6

[18] Uferah Shafi, Asad Safi, Ahmad Raza Shahid, Sheikh Ziauddin, and Muhammad Qaiser Saleem. 2018. Vehicle Remote Health Monitoring and Prognostic Maintenance System. *Journal of Advanced Transportation* 2018 (18 Jan 2018), 8061514. https://doi.org/10.1155/2018/8061514

[19] Andreas Theissler, Judith Pérez-Velázquez, Marcel Kettelgerdes, and Gordon Elger. 2021. Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry. *Reliability engineering & system safety* 215 (2021), 107864.

[20] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.* 15, 6 (jun 2022), 1201–1214. https://doi.org/10.14778/3514061.3514067

[21] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80–83. http://www.jstor.org/stable/3001968

[22] Marie Lisandra Zepeda-Mendoza and Osbaldo Resendis-Antonio. 2013. *Hierarchical Agglomerative Clustering*. Springer New York, New York, NY, 886–887. https://doi.org/10.1007/978-1-4419-9863-7_1371

[23] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proc. of the AAAI Conf. on Artificial Intelligence*, Vol. 33. 1409–1416.