HEALTH

Hazard Estimation and Analysis of Lifelong Truck Histories

A project within FFI: Machine Learning

Authors (alphabetically ordered):
Peter Berck, Nils Roger Carlsson, Kunru Chen, Reza Khoshkangini, Magnus Löwenadler, Peyman Mashadi, Sławomir Nowaczyk, Sepideh Pashami, Parivash Pirasteh, Mahmoud Rahat, Henrik Ydreskog

Date: Friday 29th November, 2019

FFI Fordonsstrategisk Forskning och Innovation
VINNOVA Energimyndigheten TRAFIKVERKET FKG VOLVO SCANIA VOLVO

Contents

1 Summary

In modern industry, the recognition and analysis of failures ensure product quality, safety, and optimize manufacturing costs. The HEALTH project focuses on building sequence models capable of collecting and exploiting the complete truck history to reveal the correlation between various failures and predict the potential risk of components in the trucks.
Nowadays, trucks are equipped with a number of sensors and telematic equipment to continuously monitor and capture their state over time. The Logged Vehicle Data (LVD), repair information and Diagnostic Trouble Code (DTC) are the useful data sources and tool to exploit at taking multiple aspects of vehicles' population to identify risk and underlying cause of failures and correlation between features on a specific component. The decisions from the models identify which vehicles are likely to fail, and which corrective actions to suggest based on the most probable failure causes.
In the HEALTH project, the collaboration between Halmstad University and Volvo Trucks Aftermarkets led to the deployment of the classifiers for prediction of failures for several components such as turbocharger and air bellow. In addition, advanced machine learning systems are developed for data preparation and aggregation including; a system using Generative Adversarial Neural Network to handle the data imbalance; clustering solutions to detect high risk vehicles and faulty vehicles. Further, sequential modeling, which was the focus of this research, implemented using stochastic Markov process to represent a different aspect of the truck history as well as designing various recurrent neural network systems to predict different component's failures and remaining useful life using different sources of data. Finally, causal relations between the features has been analysed.

2 Sammanfattning (in Swedish)

nom modern industriverksamhet används igenkännandet och analyserandet av fel till att säkerställa produktkvalitet och säkerhet, samt till att optimera tillverkningskostnader. HEALTH-projektet har fokuserat på utvecklandet av sekvensmodeller som är kapabla att samla in och utnyttja lastbilars hela historik för att finna sambandet mellan olika typer av fel och prediktera risken för kommande fel på komponenter på lastbilarna.
Nuförtiden är lastbilar utrustade med många sensorer och telematik för att kontinuerligt övervaka dess status över tid. Loggade fordonsdata (LVD), reparationsdata och diagnostiska felkoder (DTC) är användbara datakällor att utnyttja som verktyg för att ta in åtskilliga aspekter av fordonspopulationen i syfte att upptäcka risker, underliggande orsaker till fel samt korrelationer mellan egenskaper för specifika komponenter. Besluten från modellerna identifierar vilka lastbilar som kan vara på väg att få ett fel, samt kan föreslå vilka de bästa reparationsåtgärderna kan vara, baserat på de mest sannolika felorsakerna.
I HEALTH-projektet har samarbetet mellan Halmstad Högskola och Volvo Lastvagnars eftermarknadsorganisation lett till användande av klassificerare i syfte att prediktera fel på olika komponenter som t.ex. turboaggregat och luftbälgar. Utöver detta är flera system för avancerad maskininlärning utvecklade för förberedande databehandling och aggregering. Detta inkluderar: Ett system som använder generativa motverkande nätverk (GAN) för att hantera obalanserade data; Lösningar för klustring i syfte att detektera riskgrupper av fordon samt fordon med fel; Sekventiell modellering, vilket har varit i fokus för denna forskning, som har implementerats genom användande av stokastiska Markov-processer för att kunna representera olika aspekter av fordonens historik. Dessutom har olika neurala nätverkslösningar utvecklats för att prediktera komponentfel och återstående komponentlivslängd. Slutligen har även orsaksrelationer mellan egenskaperna analyserats.

3 Background

Maintenance is the main part of the total operation plan and costs in modern industries. This is significantly important for customers since it provides and ensures a level of reliability, availability and safety requirements.
In addition to the above criteria, customers expect continued improvement in

the quality of products together with increased functionality, so that vehicles are becoming more personalized and specialized. Such product diversity is desired by manufacturers' sale divisions as means of attracting customers' attention. However, the diversity in product brings out new challenges for maintenance strategy in the longer term when trying to understand the failures in components, weak spots and flawed designs sourced by different reasons. Growth in the number of failures for certain components is often an indicator of a quality issue. This will also translate into an increase in costs that a manufacturer has to pay on warranty claims and a decrease in customers' trust and satisfaction. Therefore, it is important to detect the imminent increase of the failures as quickly as possible, or even to predict it before it happens. Analysing multiple available resources with the aim of deriving useful knowledge from products during their operations enables the manufacturers to increase awareness of the quality problems. It also supports Original Equipment Manufacturers (OEMs) in making decisions to initiate corrective actions as soon as possible. Original Equipment Manufacturers (OEMs) of commercial transport vehicles basically outline maintenance plans taking into consideration different parameters such as age and millage. Within these plans, machine learning approaches have been widely applied to monitoring of equipment condition of vehicles. It has been shown that machine learning is exceptionally beneficial in failure prediction and discovering patterns of interest in data. For example, Lu [Lu and Meeker, 1993] provided a general nonlinear regression technique based on condition monitoring data that is capable of approximating the remaining life distributions of degrading components. In [Chinnam, 1999] Chinna introduces a neural network-based model for online estimation of component reliability. Similarly, Shao et al. [Shao and Nezu, 2000] model the root mean square vibration value as a time series and evolves neural network models to predict the health of a roller bearing. Later, Maillart et al. [Maillart and Pollock, 2002] introduced a policy to determine the required frequency of condition monitoring. They exploited the method of stochastic dynamic programming to build a schedule that reduce the cost of maintenance. All of those methods, however, are based on analysing current sensory measurements.
Today's availability of large amounts of data give us the opportunity to make predictions based not only on the currently measured state of the truck, but also on all the important events and conditions that affected them in the past. New machine learning techniques will lead to novel solutions for predictive maintenance, and many, even unanticipated, problems can be

avoided [Nowaczyk et al., 2013, Prytz et al., 2015, Khoshkangini et al., 2019]. In the HEALTH project, we used Adversarial Neural Network [Ganin et al., 2016] to handle the imbalancy of the huge amount of logged vehicle data, which are captured during the vehicle operation. We exploited sequence models such as Markov Models (MM) [Eddy, 1998] and Recurrent Neural Networks (RNNs) [Graves et al., 2013] to address the limitations of existing approaches. Hidden Markov models (HMMs) are an extension of the Markov chains in which an observed sequence is based on an unknown sequence of unobserved states [Stratonovich, 1965].
The main challenges to be dealt with using these methods was capturing the features from data which are more likely to interpret the hidden pattern behind failure. In the HEALTH project we also took advantage of the abilities of RNNs to handle sequential dependencies and better capture contextual information.

4 Purpose, research questions and method

In HEALTH project multiple solutions in the form of several technical sections have been developed to explain the relations between failures, as well as predicting the components' failures in trucks. The HEALTH project used the available resources such as DTC and Logged Vehicle Data at building a sequential modeling in a novel fashion to early detection of failures in the vehicles, which are operating. The following work packages were included in the project:

- Work package 1 (WP1) was about extraction, pre-processing and aggregation of the available data, in order to define the states needed for sequential modelling. Sequences of observed or partially observed states, associated with events, were needed for modelling the complete lifelong truck history.

- Work package 2 (WP2) was about the intention to represent different aspects of truck histories under the assumption that the states of interest have been identified (see WP1) and can be sufficiently well identified from the available data.

- Work package 3 (WP3) was about the investigation on methods that can model vehicle histories as sequences of unobserved or partially observed states. This work package was developed to extend work package 2 assuming that all the parameters affecting the condition of the truck can be observed.

- Work package 4 (WP4) was about exploiting the knowledge to identify causal relationships. For example, to present how different processes affect different faults, and to support aftermarket experts in designing improved maintenance schedules.
- Work package 5 (WP5) was about the demonstrator that constructed based on the algorithms, which were developed in work packages 1-4 over the project, showcasing the benefits coming from predictive maintenance in heavy duty vehicles. These results either integrated with current platform or presented at Volvo for identifying promising methods for future integration.

5 Objectives

HEALTH project achieved the following objectives which are well aligned with the stated objectives in the application.

- Optimising supervised classifiers for predicting failures of various vehicles' components,
- Developing sequential models to capture the complete truck history,
- Integrating and pre-processing of different sources of data,
- Deploying and integrating machine learning solutions in Volvo trucks production environments,
- Contributing on more than 10 scientific publications and master theses.
- Transferring knowledge and developing machine learning competence.

The results in the following sections address each of these sated goals.

6 Results and deliverables

The results obtained in HEALTH project divided into two main parts of Scientific and Industrial aspects. The first aspect associated to the development of novel methods for representing lifelong histories of trucks and using them for machine learning. While the second aspect focused more on implementation, deployment and evaluation of the Machine Learning methods and approaches in the real business setting. These separate focus areas have led to presentations, software and several conference and journal publications. In this section, we will present an overview of the insights we have obtained.

6.1 Contribution to FFI and ML program goals

We have contributed to the following FFI goals:

- More sustainable society
  A more sustainable society can be achieved through prolonged vehicle life. It is quite clear that a well-maintained vehicle, one that will keep running for as long as possible, leads to less pollution, better utilisation of materials, etc. In addition, the environmental impact of a vehicle in bad condition is often substantially higher than the eco-friendly, well-maintained one.

- New predictive maintenance strategy
  Quality is crucial for the competitiveness of Swedish automotive cluster. Manufacturing companies and dealers continuously track re-occurring and trending problems by monitoring predictive maintenance in a large scale. Volvo Trucks is one of the first OEMs to introduce uptime promise as a service. New predictive maintenance solutions introduced in this project support the manufacturers to increase customer satisfaction.

- Improving the knowledge on data analytics
  HEALTH contributes to an increase in data analytics competence within Volvo Trucks. In addition, predictive maintenance services could promote workshop workers' competence by communicating not only which component is potentially unhealhty, but also the reason behind this assessment.

- Timely and cheaper transport solutions
  Fleet management and logistics can be improved by taking into account the predicted failure of the vehicles for timely scheduling of maintenance occasions.

- Safer traffic
  HEALTH enhances traffic safety by providing solutions to avoid breakdowns on the road, which often cause dangerous situation resulting in accidents or a stranded situation.

We have contributed to the following ML program goals:

- Robust modeling
  The HEALTH project contributes to development of methods for robust modelling, optimization and decision making processes based on available data.

- Technology and development
  Considering the modern vehicles as complex systems, new machine learning developments in HEALTH allow the manufacturers to be able to capture, describe, and predict their condition, behaviour and future operation in a sufficiently accurate way.

- Personalized maintenance functionality
  HEALTH project develops a service offering that can provide personalized maintenance functionality, based on actual usage, conditions and specific needs of individual vehicles. This level of personalization is not possible without Machine Learning, however, currently available solutions are insufficient in dealing with the complexities of real business settings.

6.2 WP1: Data processing, aggregation, and clustering

The following experiments and results target Ob1 under WP1.

6.2.1 Handling data imbalance using Adversarial Neural Networks

In this section we study effects on learning models due to class imbalance on data and algorithmic level. Class imbalance refers to binary or multiple class data where the number of instances from one class, called the majority class, is significantly higher than the other class(es), called the minority class(es) [Japkowicz, 2000]–. In this scenario, standard learning methods perform poorly as they develop a bias in favour of majority class [Liu et al., 2008]. A solution for this problem is to use undersampling or oversampling methods at the data level. Drawback of undersampling is information loss, while oversampling might lead to exact replicas of existing samples [Liu et al., 2008]. To learn the true underlying distribution of the data, we will explore several variants of Generative Adversarial Networks (GANs), as they have the capability to capture the key characteristics of the data they are trained on [Ganganwar, 2012].

Data

In this experiment, we have used 67038 histogram samples of Volvo data from 3376 trucks, which were logged over a three year period. The data consists of two variables, *engine RPM* and *engine torque*, where the information expresses the time spent in each configuration defined by the combination of the two variables. The axes typically contain twenty divisions, and each histogram contains four hundred values. Plotted with one variable on each axis, the histograms resemble images consisting of coloured squares. Figure **??** shows an example of a histogram.
The values in the histograms are cumulative, and are typically downloaded from the trucks every second week. For each truck, we had the complete history captured in bi-weekly histograms. From this data, we created a labelled data set in the following manner. For each truck, the histogram data was labelled *healthy* by default. If there had been a repair at a certain date, the histogram data from that date to three months *before* that date were labelled *unhealthy*. The reasoning behind this is that we expect the degradation of engine components to manifest itself in the data before it breaks down. Note that we are equating repair date with break down date. This gave us a data set containing 87% healthy histograms and 13% unhealthy histograms. In order to generate samples using the oversampling methods we used different proportion of data with different dimensions, namely 400, 400, 268 and 15. The oversampling experiments were divided into four cases, referred to as case 1, case 2, case 3 and case 4).

Oversampling Methodology

In this section we explain how we investigated to mitigate the class imbalance using several different varieties of general adversarial networks (GANs) including; Vanilla GANs (VGANs) [Goodfellow et al., 2014], Wasserstein GANs (WGANs) [Arjovsky et al., 2017], conditional GANs (CGANs) [Mirza and Osindero, 2014] and boundary equilibrium GANs (BEGANs) [Berthelot et al., 2017]. The resulting balanced data sets were evaluated on the failure prediction problem using stochastic gradient descent (SGD) and random forests (RF) models. Additionally, we compared the GAN balanced data with data balanced by traditional oversampling methods, namely with random oversampling (ROS) and synthetic minority oversampling technique (SMOTE). Finally, we tried one new variant where we trained a VGAN on data augmented by SMOTE. To illustrate the similarity between the generated data and the original data, the Frechet inception distance (FID), and T-SNE plots are shown.

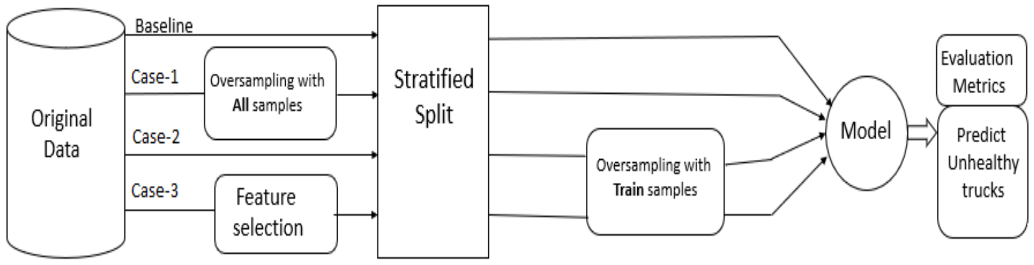


Figure 1: Flowchart of the different methods explored to balance the data.

Our baseline experiment consisted of running the SGD and RF models on the original data, without feature selection and balancing.
We apply a cross-validation setup prone to overoptimism where exact or similar replicas of a given pattern exist in both the training and test sets [Fergus et al., 2013]. In our case the entire data set is oversampled first and then a stratified random split is applied to split the data into a train, validation and test sets.
We designed four different cases to test our data balancing. In case 1, the

oversampling is run on the whole data set, before we divide it into train, test and validation sets. In case 2, we divide the data into train, test and validation sets first, and run the oversampling only on the train set. In cases 3 and 4, we perform a feature selection algorithm before splitting the data into train, test and validation sets. In case 3 we use Wilcoxon, and in case 4 PCA. In both case 3 and 4, the balancing algorithms are only run on the training data, similar to case 2. Figure 1 shows a summary of the different methods.

Model Evaluation and Comparison

Validation is an important phase in analysing the performance of the model. To explain the proper usage of cross validation, we have further compared the results from cross validation before and after oversampling.

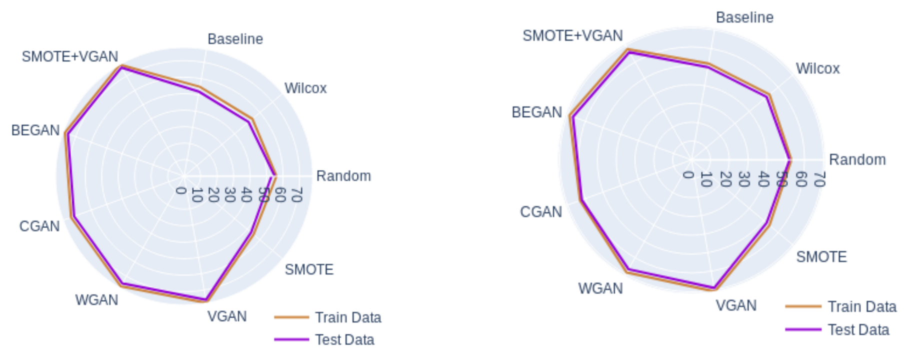


Figure 2: AUC Plot: (left) Case-1, (right) Case-2.

In Figure 2 we show that for case 1 the AUC values in all oversampling methods are higher than in case 2. We take the fact that we also oversampled the test set in case 1 to be the reason for this. In case 2, we only oversampled the training set, and not the test sets. We conclude from the slight difference in AUC values between the train and test sets that the oversampling methods do not suffer from overfitting effects.
Table 1 gives the results of the SGD and RF classifiers in case 1 and case 2. From the table, we see that the performance of case 1 is better than the performance of case 2. The table shows that for case 2, for the SGD classifier, the GAN models perform better on average than the baseline and

		Case-1				Case-2			
		Pre	Rec	F1	AUC±SD	Pre	Rec	F1	AUC±SD
SGD	Baseline	0.79	0.54	0.61	0.51±0.03	0.77	0.56	0.64	0.52±0.02
	ROS	0.78	0.70	0.74	0.52±0.02	0.76	0.71	0.72	0.53±0.01
	SMOTE	0.79	0.47	0.54	0.52±0.02	0.75	0.52	0.61	0.52±0.01
	VGAN	0.82	0.83	0.82	0.75±0.01	0.81	0.75	0.76	0.64±0.02
	WGAN	**0.86**	**0.88**	**0.87**	**0.76**±0.011	0.83	0.76	0.77	0.62±0.01
	CGAN	**0.82**	**0.84**	**0.84**	**0.75**±0	**0.81**	**0.77**	**0.79**	**0.66**±0
	BEGAN	0.81	0.82	0.82	0.74±0.01	**0.84**	**0.72**	**0.73**	**0.68**±0.01
	SMT-VGAN	0.80	0.81	0.81	0.75±0.04	0.82	0.75	0.74	0.64±0.01
RF	Baseline	0.94	0.93	0.93	0.93±0.03	0.93	0.93	0.93	0.94±0.02
	ROS	0.95	0.94	0.94	0.94±0.01	**0.94**	**0.93**	**0.94**	**0.94**±0.01
	SMOTE	0.96	0.94	0.95	0.94±0.02	**0.94**	**0.94**	**0.94**	**0.94**±0.01
	VGAN	0.95	0.94	0.94	0.96±0.02	**0.95**	**0.92**	**0.95**	**0.93**±0.01
	WGAN	**0.94**	**0.96**	**0.95**	**0.96**±0.01	0.94	0.93	0.93	0.93±0.015
	CGAN	**0.95**	**0.94**	**0.94**	**0.96**±0	0.96	0.94	0.95	0.92±0.02
	BEGAN	0.96	0.95	0.95	0.95±0.011	0.94	**0.94**	**0.94**	**0.93**±0.01
	SMT-VGAN	**0.95**	**0.94**	**0.94**	**0.96**±0.01	0.95	0.95	0.95	0.93±0.04

Table 1: Mean ranking of oversampling methods using case 1 and case 2

the other oversampling methods, with CGAN and BEGAN performing best. The reason for ROS and SMOTE not being able to generalise is intuitive: ROS creates exact replicas of existing data, and SMOTE creates synthetic examples by inflating the clusters defined by a k-means algorithm. This might reduce the data variability to existing training patterns, but leads to decreased generalisation. Using the RF classifier, this difference is less obvious, but we still see a stable performance from the GAN models.
Table 2 shows the results of the SGD and RF classifiers on data where the number of features was reduced using Wilcoxon and PCA respectively. For case 3, the results of baseline system are better than the other oversampling methods and for case 4. Eventhough there is a slight improvement using CGAN, it is not enough to comclude this is the best model. Overall there is an evident decrease in model interpretability using the feature selection methods.

		Case-3				Case-4			
		Pre	Rec	F1	AUC±SD	Pre	Rec	F1	AUC±SD
SGD	Baseline	0.98	0.84	0.85	0.62±0.01	0.88	0.87	0.84	0.51±0.01
	Wilcox/PCA	0.77	0.82	0.75	0.51±0.01	0.89	0.86	0.85	0.50±0
	ROS	0.78	0.81	0.88	0.53±0.02	0.93	0.90	0.90	0.51±0.01
	SMOTE	0.79	0.79	0.86	0.55±0	0.91	0.92	0.91	0.5±0.02
	VGAN	0.79	0.81	0.85	0.53±0.01	0.92	0.89	0.90	0.51±0.01
	WGAN	0.78	0.81	0.84	0.55±0.01	**0.91**	**0.88**	**0.89**	**0.51**±0.02
	CGAN	0.77	0.81	0.86	0.54±0.02	**0.92**	**0.87**	**0.90**	**0.53**±0.02
	BEGAN	0.78	0.85	0.87	0.52±0.02	0.91	0.88	0.90	0.51±0.01
	S-VGAN	0.78	0.84	089	0.54±0.02	0.90	0.89	0.91	0.51±0.01
RF	Baseline	0.94	0.92	0.93	0.94±0.01	0.88	0.87	0.84	0.91±0.02
	Wilcox/PCA	0.96	0.93	0.93	0.94±0.01	0.89	0.86	0.85	0.92±0.01
	ROS	0.95	0.94	0.95	0.9±0.01	0.93	0.90	0.90	0.94±0.01
	SMOTE	0.95	0.95	0.94	0.94±0	**0.91**	**0.92**	**0.91**	**0.93**±0.02
	VGAN	0.96	0.93	0.93	0.93±0	**0.92**	**0.87**	**0.9**	**0.93**±0.011
	WGAN	0.95	0.93	0.93	0.93±0.01	0.91	0.89	0.89	0.93±0.01
	CGAN	0.94	0.92	0.92	0.93±0.01	0.91	0.89	0.89	0.93±0.01
	BEGAN	0.95	0.94	0.94	0.93±0	0.92	0.88	0.9	0.93±0
	S-VGAN	0.95	0.93	0.94	0.94±0	0.9	0.89	0.91	0.91±0.01

Table 2: Mean ranking of oversampling, cases 3 and 4.

Visualisation of Generated Samples using t-SNE Plots

To visualise the data generated from GANs, a t-SNE plot (Figure 3) was generated for each model. The 5000 samples of healthy data are represented by the colour cyan, 5000 samples of unhealthy data are shown in green and the generated unhealthy samples are represented by a light green colour.
In Figure 3 (a) and (c), VGAN and CGAN have slightly learnt the upper mode but the majority of the resampled minority data produced by these models appears to have lie in same region, with limited diversity. In Figure 3 (b) and (d) both WGAN and BEGAN have learnt the upper mode of the minority data better than the WGAN and BEGAN have.

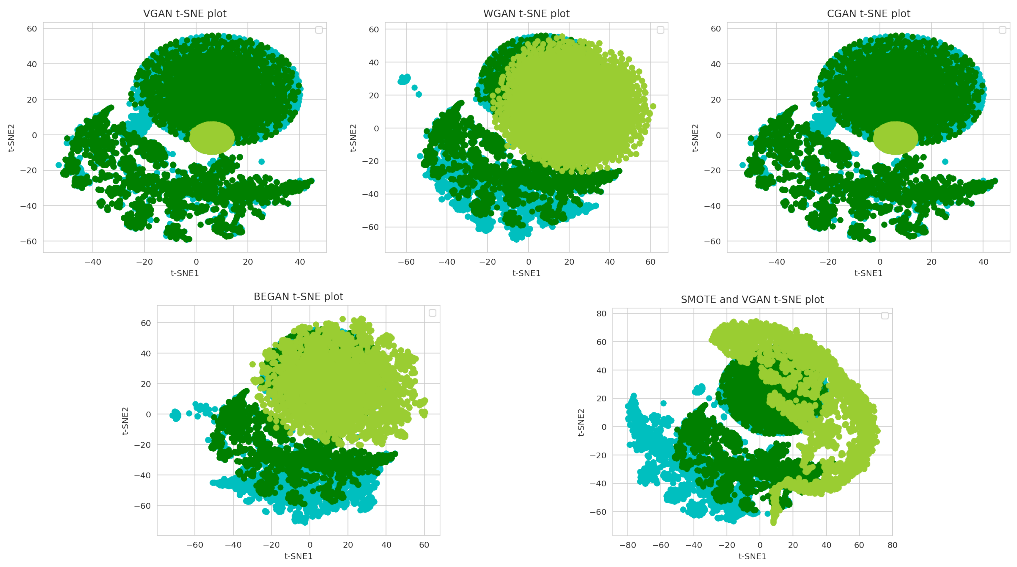


Figure 3: T-distributed Stochastic Neighbor Embedding plots: (a) VGAN (b) WGAN (c) CGAN (d) BEGAN (e) SMOTE+VGAN

FID	Case-1	Case-2	Case-3	Case-4
VGAN	1059.6	1079.7	1107.7	1126.9
WGAN	1061.7	1062.3	1110.4	1136.2
CGAN	1060.3	1063.1	1121.2	1028.4
BEGAN	1064.1	1061.1	1114.9	1157.4
S-VGAN	1058.6	1078.3	1108.1	1197.9

Table 3: Frechet Inception Distance Measure

Unsupervised Evaluation of Generated Samples Using the Frechet Inception Distance Measure

Table 3 represents the Frechet Inception Distance (FID) measure where we compare the difference between 5000 generated samples from all oversampling methods with 5000 original minority samples used for training the GANs. The FID is a metric for evaluating the quality of generated data and specifically developed to evaluate the performance of GANs [Heusel et al., 2017]. For this experiment the FID measure has been used to calculate the distance between the original data and regenerated data from the different GANs. The aim is

to have lowest FID score indicating the generated data is closest to original data. In Table 3 the difference in FID scores between case 1 and case 2 two is least even though the sample range used to train the GAN in case 1 was higher. When we compare all methods, the FID scores in case 3 and 4 seems to have higher value. This difference may be due to lack of diversity but also to loss of information after the feature selection method were applied in these two cases. Considering all the metrics mentioned, in case 2 we were able to resolve the class imbalance by oversampling the method. This can also be observed from the table which gives a lower FID scores in case 1 than in cases 3 or 4. Finally, WGAN and BEGAN were able to learn the underlying distribution of original minority data better than the other oversampling methods. Also, the computational cost of training a BEGAN is less than the costs of the other GANs.
Class imbalanced learning on complex data has always been a challenge for data mining and machine learning research. This research presents deep neural networks for solving the imbalance problems, and also provides insights on how to deal with a skewed data distribution by using different variants of GANs. An insight on the correct usage of cross validation when oversampling methods are used is also provided. We have also implemented traditional oversampling methods such as random oversampling and SMOTE to gain understanding about the oversampling methods in this context.

6.2.2 Clustering for detecting the high risk vehicles

This section describes our intention to analyze the trucks based on two features such as *Horse Power*, which relates to the configuration of the trucks, and *Mileage* that shows the vehicle usage logged over time. These two features are important since they can bring out the knowledge of what style of usage pattern w.r.t the millage in different types of engines can have more possibilities to fail, however previous investigations also suggested that these two features have shown a possible relations to the failure ratio. In this experiment we concentrated on a component connected to power train.
To find out such relations, we have implemented the clustering algorithm on the trucks population with three different engine types including; A, B and C connected to horse powers.
In order to achieve the goal mentioned above, we have designed three different experiments as follows:

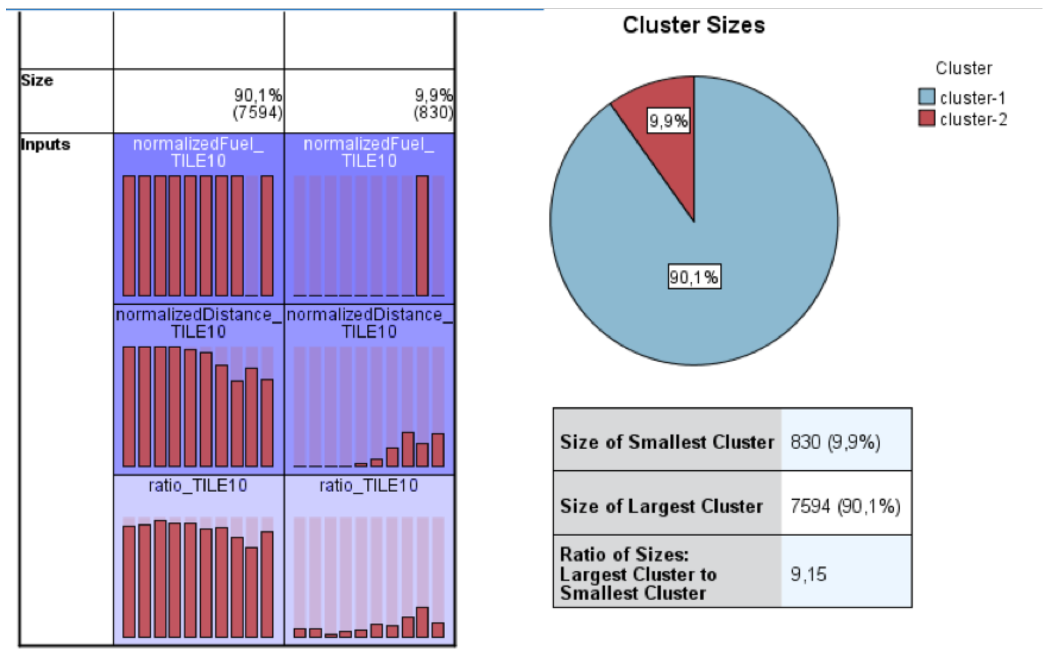
- Engine type A with 10 bins discretisation on the mileage data,

	Failure Ratio and Sample Size in Cluster 1		Failure Ratio and Sample Size in Cluster 2	
Engine Type	Failure Ratio	Cluster Size (sample size)	Failure Ratio	Cluster Size (sample size)
A	lower	90.1%	higher	9.9%
B	similar	82.6%	similar	17.4$
C	higher	73.7	lower	26.3

Table 4: Clustering results with 10 bins discretisation: Failure ratio, which is connected to power train, in different engine types.

- Engine type B with 10 bins discretisation on the mileage data,
- Engine type C with 10 bins discretisation on the mileage data.

Data (related to mileage which are logged over time) collected from heavy duty trucks are normalized and discretised into 10 bins, this decision has been made under the hypothesis that discretisation supports the algorithm for better clustering the trucks. Then, we have implemented K-means [Wagstaff et al., 2001] clustering algorithm on the data, in which the algorithm is set to produce two clusters as the result. We kept this setting for all three experiments.
Table 4 shows the results of the K-means algorithm, which are conducted on the data –with 10 bins–. It can be observed that failures ratio in the B engine type in both clusters are similar (with a very low difference by 0.007), however these two clusters contain different population of the vehicles such as 82.6% and 17.4% in cluster 1 and 2, respectively. We have obtained lower failure rates on the vehicles with A type engine with a highly imbalanced vehicle population in cluster 1 with 90% and cluster 2 by 0.99% shown in Figure 4. The results in terms of C type engine show rather more balanced population distribution in the clusters, having a bit more failure proportion in cluster 1, and lower in the second cluster w.r.t the other engine types.
All these results and figures illustrate that vehicles in cluster 1 with C and A engine types have the highest and lowest failure rates during their operation life. This also indicates those vehicles –B engine type in the second cluster and C engine type in the first cluster– have high risk usage pattern, which led to have higher failure rates in that specific component.



Size of Smallest Cluster	830 (9,9%)
Size of Largest Cluster	7594 (90,1%)
Ratio of Sizes: Largest Cluster to Smallest Cluster	9,15

Figure 4: The result of clustering in low engine size vehicles.

6.2.3 Clustering vehicles dynamically based on logged vehicle data

Taking into consideration the above clustering implementation, which have done based on only two features Millage and Engine size, in this stage of the experiment we intend to use all LVD data, however this data includes many uninformative and redundant features that should be eliminated from the data set. Thus, the main focus of this experiment is to select the most informative features using a supervised classification method and cluster the readouts into a set of representative states that capture different aspects of lifelong history of the vehicle. To do so, we first need to estimate feature importance on readouts. It is known that each readout consists of a large number of attributes, which could affect the quality of the clustering process. One way to tackle this issue is to identify and remove unimportant or unrelated features. This step is essential since the clustering algorithm considers the same weight for all the features. Therefore, it is a common practice to apply feature selection prior to clustering in case of having a high dimensional feature set [Dash and Liu, 2000]. We know that LVD features have different scales and types, so it is important to normalize the data before clustering.

There are different approaches available for estimating feature importance. Since the problem is a two-class classification task (i.e. healthy or unhealhty), we decided to use a supervised method to assign relative importance scores to the features and filter them based on a threshold applied on the number of features. Then, we splitted the data into train, development, and test sets. A random forest classifier is employed on the training set to calculate feature importance. Table 5 represents the fifteen most important features along with their relative importance scores. The scores are scaled up for better readability.

ID	Feature name	Score
1	PMAIRDRYERCARTRIDGERESETCALEND	12.212
2	TOTAL_ENGINE_TIME	9.008
3	GEARBOX_MOVEMENT_GEAR1R_NEUTRA	8.485
4	ENGINE_SPEED_TORQUE_H_Y_INDEX_2	8.093
5	AGE_YEARS	7.223
6	GEARBOX_MOVEMENT_GEAR23_GEAR2	7.108
7	GEARBOX_MOVEMENT_RANGE_HIGH	7.035
8	AGE_MONTHS	7.002
9	MAIN_LOG_KEY_ON_TIME	6.784
10	GEARBOX_MOVEMENT_GEAR1R_GEAR1	6.722
11	GEARBOX_MOVEMENT_RANGE_LOW	6.703
12	GEAR_LEVER_AUTOMATIC_TIME	6.449
13	ENGINE_SPEED_TORQUE_H_Y_INDEX_15	6.283
14	OBDCOND	6.267
15	PMAIRDRYERCARTRIDGERESETENGINE	6.21

Table 5: List of fifteen most important features and their relative importance scores.

Using the scores calculated by random forest classifier, we filtered the number of attributes to fifty most important features. This number is selected empirically since there is a drop on the importance scores close to this number. The idea behind applying feature selection is to reduce the dimension of the data by identifying and removing unrelated features in order to improve the clustering performance. Subsequently, we applied k-means clustering technique on LVD readouts based to the selected features. Figure 5 illustrates the distribution of the clusters along with the five most important features.

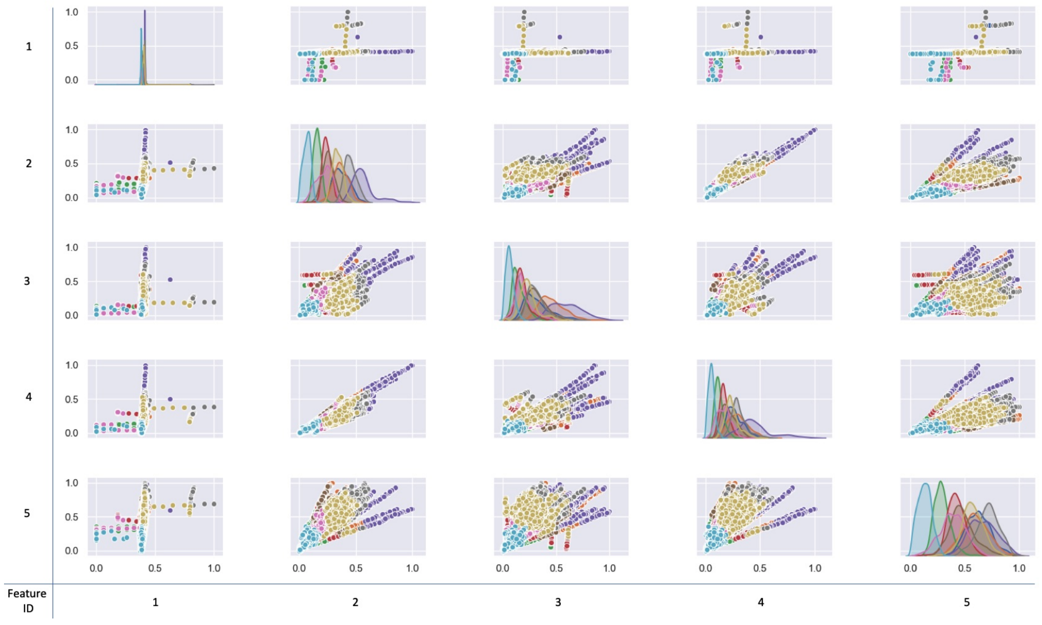


Figure 5: A plot displaying the distribution of cluster members over five most important features.

One important parameter of k-means algorithm is the number of clusters. In this experiment, it is set to a predefined fixed value. The effect of changing this parameter on the model efficiency is further investigated in Section 6.3.2. To evaluate the performance of the clustering results, we need to inspect how it affects the overall performance of a classifier on the final task of failure prediction. To this end, in the next section (Section 6.3.1), we model vehicle behaviour by considering each cluster as a Markov process i.e. members of each cluster represent a state. The clustering is applied on the train set. Then we query closest cluster centroid to each readout in the test (or dev) set to map it to a respective Markov state. It is shown that both feature selection and clustering are beneficial for doing the final task of failure prediction.

6.3 WP2: Modeling components health status using Markov process

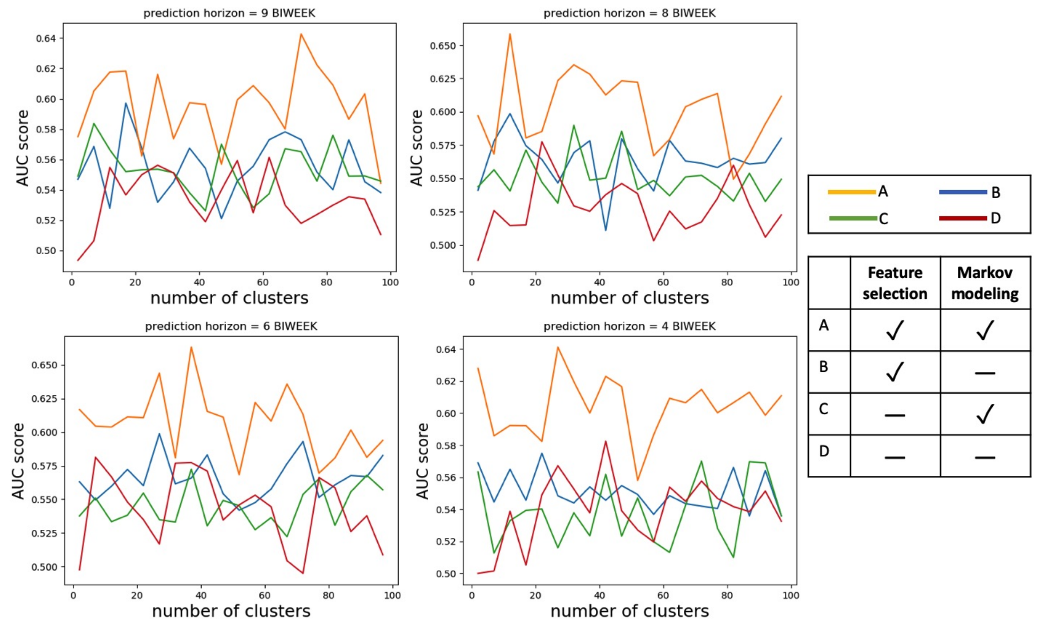
6.3.1 Predicting turbocharger failures using Markov process

The advancements of the telematics and connectivity solutions have provided new opportunities for the field of vehicle predictive maintenance. The number of the sensors installed on a vehicle is increasing over the time and manufactures are looking for new ways to improve up-time of their fleet while at the same time reduce the costs related to unexpected breakdown. The nature of the aggregated data from vehicles is sequential and it is interesting to investigate existing methods for modeling fully observable state sequences to detect common patterns of failure. To do so, in this section, we consider stochastic Markov process to represent various aspects of truck histories. In this representation, we are interested to automatically create some states that could help to signify similarities between readouts from different trucks during their lifetime. To do so, we apply k-means clustering on the logged vehicle data (LVD) (see section 6.2.3). Then, each cluster is used to indicate a Markovian state. Since the clustering process is applied on the data logged and captured from different vehicles, the state of a vehicle could change from one cluster to another during the time.
The sequence produced by each truck is then aligned with repairs information of the truck to further investigate the hidden patterns of failures. We employed the prediction horizon technique introduced in [Prytz et al., 2015] to label the readouts. The prediction horizon indicates a period of time (time-window) ahead of a failure that a replacement recommendation should be made for the part. Based on the prediction horizon parameter, each readout is labeled as zero if that part had no repair in the following time span, otherwise, it is set as one to imply that a replacement for that part is recommended. More formally, each time-window/time span is assigned a binary label according to the following equation.

$$L_t = \begin{cases} 1 & \text{if failure in } [\text{t,t+}\tau) \\ 0 & \text{if no failure in } [\text{t,t+}\tau) \end{cases} \tag{1}$$

, where t refers to a time window –e.g., one week, two weeks, one months.– that has a highest impact on failures in trucks.
In this experiment, we narrow our attention down to a single component related to turbocharger failures. A turbocharger is a vital and expensive part

of the vehicle, where its' failure usually leads to a total breakdown of the vehicle. Besides, most of the time this part fails unexpectedly and on rare occasions. This fact adds more complexity to the problem.
Another complexity of the problem is the imbalanced classes issue. Considering the entire LVD readouts, only a very small portion of the readouts have a repair associated with them. This indicates that the number of vehicles, which have failures associated by 1 as the target value, are very low compared to the number of healthy vehicles labeled by 0. Thus, in this experiment, to overcome this issue, we focused on a group of vehicles that have had at least one recorded repair in our data set. This somehow decreases the imbalancy of the classes, however, the challenge remains to a high degree.
The Markov model enables us to assign a state to every readout and further convert the history of a truck into a sequence of states. It is also possible to study the relations between the produced sequences of hidden states and variables of interest like previous repairs, which are known to the model.
We conducted an experiment to analyze the efficiency of the feature selection and modeling process for predicting truck failures. In this experiment, we exploit random forest classifier to compare four possible settings regarding whether we apply feature selection and Markov modeling on the data or not. Both steps prove to be beneficial for improving the overall performance of the failure prediction task.
Figure 6 displays the results produced by random forest classifier on four parameter settings with four different prediction horizon values. In two settings (A and B) we apply feature selection, while for the rest we consider all available features. On the other hand, the Markov model has exploited only on A and C. It is worth noting that in A and C, due to the clustering, the number of features has been reduced to one i.e. the associated Markov state to the readout.
As you can see in Figure 6, setting A (that encompass both feature selection and Markov modeling) outperforms other settings. The feature selection process proves to be efficient since the application of clustering on the original features (setting C) improved the results compared to other approached. Each plot in Figure 6 represents results for a specific prediction horizon with different values for the number of clusters. It could be seen that conclusions drawn from the experiments are valid for the most of the values of prediction horizon and number of clusters.



	Feature selection	Markov modeling
A	✓	✓
B	✓	—
C	—	✓
D	—	—

Figure 6: Classification results on four possible settings concerning with application of feature selection and Markov model.

6.3.2 Optimizing number of states

As discussed in Section 6.3.1, building a Markov model based on the clustering of LVD features enhanced the performance of failure prediction in this context. Though, we did not elaborate on an important parameter of the model i.e. optimized number of states (clusters). Perceptibly, using a predefined value for this parameter is not efficient and we need to adjust it based on the data. In this section, we conduct an experiment to estimate an optimal range of values for the number of states. Then, we describe the history of each truck through a sequence of states extracted from the optimized Markov model. We refer to this sequence as the trajectory of the truck. Then we create a supervised hidden Markov model (HMM) classifier to model the failures of the turbocharger. Subsequently, the prediction result of this model is compared with a random forest classifier applied on the original LVD features.
Since we are modeling several different trucks with the same hidden Markov model, the implementation should be able to compute the emission and transition probabilities from not only one sequence of observations but also

many parallel sequences. It is also important to apply the Viterbi algorithm to the trajectory of each truck separately to estimate the output labels. Due to the fact that existing implementations did not fill the requirements of our problem (i.e. accepting multiple parallel input sequences), we applied a customized HMM.
Figure 7 shows the effect of changing the number of Markov states on the performance of the model for predicting the failures. The vertical axis displays the AUC scores gained by HMM classifier, while the horizontal axis represents the number of states (clusters). The number of clusters changes from 1 to 2000, and for each value, we run the entire system (i.e. model generation using clustering and classification using HMM) three times to compute mean and standard deviation of the area under the ROC curve (AUC).
The blue points in the plot display the mean values, while the standard deviations are illustrated using a vertical green line crossing each point. This experiment is conducted using train and development set and the prediction horizon is set to 6 time-steps.

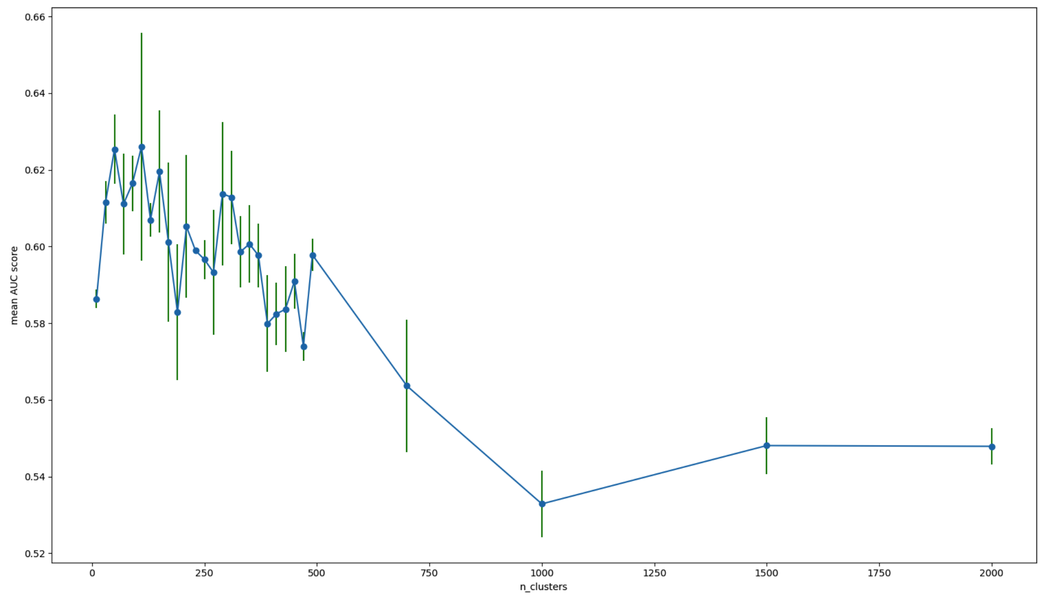


Figure 7: The plot shows how AUC score of HMM classifier changes by altering number of Markov states

As you can see in Figure 7, performance of the system drops as we increase the number of clusters to more than 500. The best performance achieved by

110 clusters. Considering this value as the optimized number of states, we conducted another experiment this time on the test dataset and compared the HMM classifier with two random forest classifiers. Note that HMM is applied on the trajectories produced from sequences of readouts, whereas random forest is applied on a single readouts. Table 6 presents the final results.

Classification method	Feature selection	AUC
Random Forest	No	0.5233
Random Forest	Yes	0.5660
Hidden Markov Model	Yes	0.5847

Table 6: Results of applying random forest and HMM classifiers on the test data.

In Table 6, the feature selection column indicates whether the number of attributes is reduced to the set of selected features or the entire LVD feature set is used. The AUC score column depicted in Table 6, shows the final performance of each approach, which express the HMM classifier outperforms other approaches.

Although we were able to improve the performance of the final system by processing the sequence of trajectories instead of single readouts, the scores are still low. There might be many reasons contributing to this fact. For instance, the difficulty of the problem and having a highly imbalanced data set. The weakness of the Hidden Markov Model to memorize the complex patterns of failures could also be the sourced of the problem. This comes from the fact that HMM applies the Markov assumption which considers only previous state and ignores the rest of the sequence of events that precede it to compute the conditional probability of the current state. Since we produce the trajectories based on clustering, in many cases, the state of a truck stays in the same state for a couple of time steps before changing to another state. Combining this fact with the Markov assumption, the system is unable to do deep inference over the trajectories.

Another important factor is the labeling process. As mentioned earlier, the LVD readouts are not annotated originally, so that we assigned zero and one labels based on the prediction horizon technique. Hence, the final AUC score of the system is highly sensitive to how we assign the labels.

6.4 WP3: Learning partial observable models using neural networks

This section in an attempt to ease the assumption of observability of all parameters that influences the state of the trucks by considering only partially observed data in the form of sequence of data. Section 6.4.1 shows the use of Long Short-Term Memory (LSTM) type of recurrent neural networks to predict whether a failure will occur within next 90 days interval. Section 6.4.2 adopts an ensemble of recurrent neural networks in order to predict time to next failure. In these networks, a specific type of ensemble learning, known as Stacked Ensemble consisting of two layers of models, is used. Finally, section 6.4.3 utilises 2D histograms of engine-speed and torque which is formed by counting the occurrence of combinations of these two components. This 2D histograms are then fed to recurrent neural networks for the task of failure prediction.

6.4.1 Recurrent Neural Networks for Fault Detection using LVD

As mentioned above, many researchers have worked on predictive maintenance in the last decade, and some of their works are nowadays implemented in practical settings. Random Forest is one of the powerful methods due to its ability to handle highly dimensional and highly imbalanced data.
In addition, the method – in overall – is one of the easiest algorithm to use, since it does not require the data science expertise to set the parameters. However, Random Forest assumes that each data sample is independent, which is generally not true in equipment monitoring applications, since equipment wears out over time. Failures on some components, such as air compressor, are caused by many different factors and usually develop gradually, since the current state of the vehicle is influenced by the prior usage. In this experiment, the original available data is collected as a time sequence, and different periods in this sequence are highly correlated. Therefore, it is desirable to investigate models that can describe the dynamic behaviours of the system and capture temporal relationship between samples. One example of such methods is recurrent neural networks, and in particular LSTM.
In this section, we introduce a prediction model based on LSTM to detect whether an air compressor failure will occur within the next 90 days. We used two datasets from Volvo authorized workshops, *Logged Vehicles Data* (LVD) and *Volvo Service Records* (VSR). The LVD dataset contains aggregated

information from various sensors on-board a truck, continuously collected during normal operation. It is downloaded when the truck is in the workshop. This actually poses an interesting challenge, since the length of the interval between two visits (to the workshop) is irregular. Records in VSR dataset are also collected when a truck goes to the workshop. They contain detailed information about the replaced parts and maintenance operations that were performed during the visit.

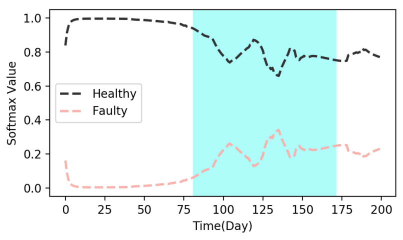
Subsets of both LVD and VSR are separately used in this paper, we called them *reading table* and *repair table*, respectively. In the reading table, there are more than 160,000 records from about 1,000 Volvo heavy duty trucks collected over 2 years. The records in this table contain information about vehicle configurations and sensor readings. Attributes which reflect the amount of wear, such as mileage, battery mode and hydraulic oil level, are available. Additionally, this data contains information on the working environment of each vehicle. On the other hand, since we are focusing on the air compressor, a component that does not fail very often, there are less than 200 entries in the repair table. Each of these entries describes when and which truck has been repaired. Clearly, the vast majority of trucks from the reading table never had compressor issues.

If a truck does not have any record in the repair table, we consider it to be a *healthy* truck, and all its records in the reading table will be categorized as negative class (i.e., not requiring a maintenance intervention). On the other hand, any truck that has a repair record is categorized as *unhealhty* truck. For an unhealthy truck, a subset part of its readout records is categorized as positive class. Clearly, for the classifier to be useful in the predictive maintenance setting, it should recommend maintenance intervention only in close vicinity to the failure time. Therefore, we defined a specific condition for positive class, namely, if the record is collected less than 90 days before air compressor repair. In this way, each unhealhty vehicle has a maximum of 90 unhealhty records, and all the other records from the same truck are labeled as negative class. This prediction horizon of 90 days is determined by both domain experts and previous study in [Prytz et al., 2015]. It is based on the fact that air compressor issues require time to develop and it is a gradual process rather than a sudden incident.

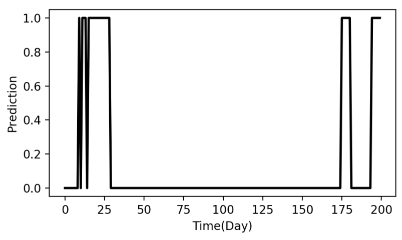
The results from both Random Forest and LSTM predictions are shown in Table 7, which clearly indicate that Random Forest outperforms LSTM models taking into consideration the accuracy and the AUC scores. If that were the only criterion, the preference for this model would be very natural,

Table 7: Results from RF and LSTM in different measurements

	Random Forest	**L(64, 32)D(10, 2)**
TN	0.92±0.15	0.80±0.13
FP	0.08±0.15	0.20±0.13
FN	0.27±0.34	0.37±0.19
TP	0.73±0.34	0.63±0.19
Accuracy	0.86±0.22	0.74±0.06
Precision	0.83±0.31	0.62±0.07
Recall	0.73±0.34	0.64±0.17
F1-Score	0.78±0.32	0.62±0.09
AUC-Score	0.91±0.17	0.82±0.09



(a) LSTM for unhealhty truck.



(b) Random Forest for healthy truck.

Figure 8: Example result visualization.

given also benefits in terms of simplicity and much shorter training time. However, the stability of the results has not been considered.
In this context, we realize that it is interesting to see how our LSTM model gives prediction over time. Therefore, we visualize the sequence of predictions for a single truck in a form shown in Figure 8a. The X-axis corresponds to time and demonstrates the window size of that learning setup. The Y-axis is the outcome of the softmax function: red curve means the probability that the LSTM predicts the data record as an unhealhty record, while the black curve means the same kind of probability for the healthy one. The blue area corresponds to the ground truth and shows the portion and the position of the unhealhty period. The repair always happens at the end (right-hand side) of the blue area. The records in the white belong to the healthy (negative)

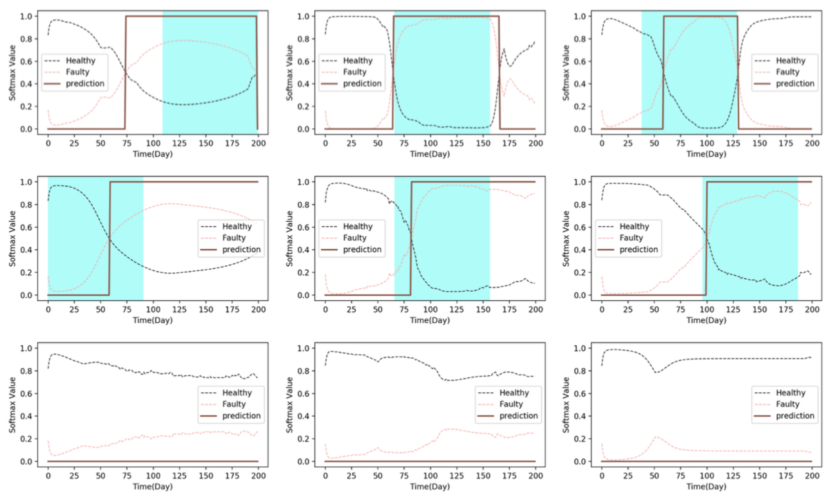


Figure 9: Example results from LSTM.

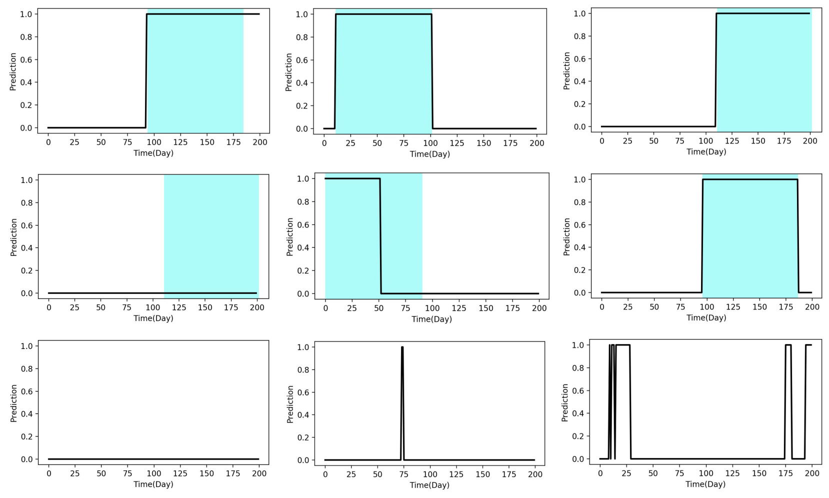


Figure 10: Example results from Random Forest.

class. The corresponding example for Random Forest is shown in Figure 20b, and more examples are presented in Figures 9 and 10.
From these figures, we initially found that LSTM can offer a stable prediction over a certain amount of time without switching. Before a switch happens, there is a distinct tendency that it is increasingly likely to categorize data to another class. Once it reaches the prediction boundary and gives a prediction

on another category, it does not quickly change again. Those features meet the requirement in industry; Companies do not favor a model that provides one prediction at first and switches to another prediction the next day, and they need time to arrange the resources of a workshop and schedule the repair. The proposed LSTM model is able to give stable, consistent prediction overtime. Experiment shows that random forest slightly outperforms LSTM in terms of accuracy and AUC score. However, by analysing the predictions given at different time, we demonstrated that this stability is essential for making repair decisions, and for significant improvement on vehicle uptime.

6.4.2 Stacked Ensemble of Recurrent Neural Networks

Model Stacking

Stacking in its original form is a two-layer architecture. In Wolpert terminology [Wolpert, 1992], first layer data and models are referred as layer_0 data and models, respectively, and the second layer cross-validated data and models are referred as layer-1 data and models, respectively. Here, layer-0 models are called "base models" and layer-1 model is called "meta model". Layer-0 data and layer-1 data are called "original data" and "meta data", respectively. Stacking is categorized as parallel architecture since base models can be trained independently from each other. Figure 11 presents the basic architecture of Stacking.

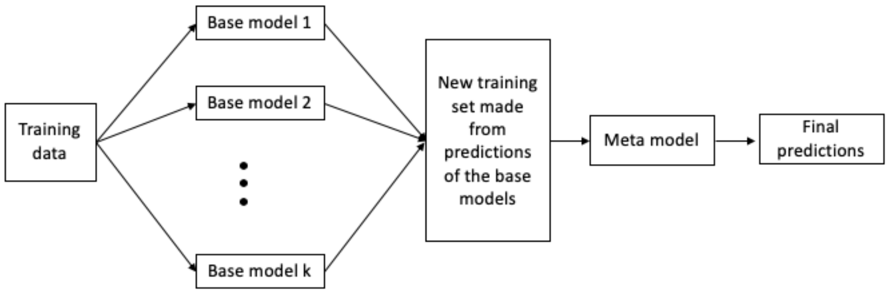


Figure 11: Two-layer stacking architecture.

One important consideration in stacking is to train the models in a way not to leak information. The risk of information leaking originates from the fact that the meta model consumes predictions of the base models as features [Güneş et al., 2017]. The correct way to implement stacking, to prevent information

leaking, is through k-fold cross validation scheme. The process is shown in Algorithm 1. In other words, the main idea is that each base model is trained on k-1 folds and make prediction on the remaining untrained fold.
This way, the models make predictions on parts of the data which label information were not provided for the models beforehand. It means that the labels are only used for the training of the meta model.

Algorithm 1 Stacked ensemble algorithm

```
Input:
    $X_{train}$                          ▷ Training data
    $n_{folds}$                          ▷ Number of folds
    $n_{models}$                         ▷ number of base models
    $base\_models$                       ▷ Set of base learning models
    $meta\_model$                        ▷ Second layer model

Output:
    $base\_models$                       ▷ Trained base learning models
    $meta\_model$                        ▷ Trained second layer model

 1: Split $X_{train}$ into $n_{folds}$ folds
 2: $X_{meta}$ = Create empty array of size $n_{samples} \times n_{models}$
 3: for model in $base\_models$ do
 4:     for $k$ in $n_{folds}$ do
 5:         train model on all folds as input excluding $kth$ fold
 6:         make prediction on $kth$ fold and add to the corresponding rows
    and column in $X_{meta}$
 7:     end for
 8: end for
 9: train $meta\_model$ on $X_{meta}$
10: return $base\_models, meta\_model$
```

Note that for each base model, n_{folds} of them are trained. For the test data, the saved models for each base model make predictions and then averaged together. This process will be done for all base models to generate X_{meta} matrix for the meta model. Finally, the meta model will make the final test prediction.
However, because of time series nature of the data, we cannot simply use the standard k-fold cross validation. To tackle this problem we adopt a time-based

cross validation in which the data folds that a model is trained on, always have lower indexes than the data folds chosen for making predictions. The time-based cross validation is depicted in Figure 12.

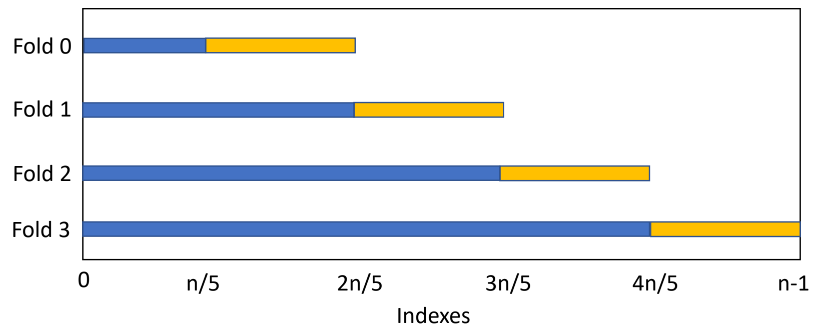


Figure 12: Time-based cross validation

Notice that data with index in range [0,n/5] cannot be used in training the meta model since it can cause overfitting.
In this study, we have considered — feature subsets, past dependency resolution, and modeling structure to create diverse set of models and learn different representation of the data,

Data labeling discussion

One important challenge is to determine how to label the LVD readouts. There are two main approaches for tackling this problem — sliding-box and time to next event (TTE) approach.
In sliding-box model the idea is to define whether a failure is happened within a predefined time interval from the timestep t using Equation 1.
There are two main issues with this approach. The first issue is the need for tuning the time interval (τ) size which can be very difficult to choose. The smallest possible interval-size is two (cannot be one) since we need to predict failure at least one step before a failure happens. Choosing the interval-size of two meaning having the finest resolution; however, the main drawback is that since failures are rare in comparison to non-failures, it makes the problem so imbalanced. On the other hand, the larger the interval-size the lower the resolution. The second issue which can be more serious is the sudden jumps from zero to one (non-failure to failure), a specific amount of time ahead of

failure (depending of interval-size). Mostly, the degradation of components are gradual as in-operation time of a vehicle increases; contradictory, by making the labels binary we attribute this gradual degradation to sudden jumps in the labels. In other words, between two consecutive logged measurements there might not be as much difference as it is in their labels (zero and one). This problem can make machine learning models confused and make it hard to train.
The second approach to labeling the data is Time to Event (TTE) or Remaining Useful Life (RUL) approach. The idea is that instead of assigning binary labels, assign time to the next event to each data as the label. Doing so has two main advantages. First, there is no need for tuning hyper-parameters such as interval-size in sliding-box method. The second advantage is that this labeling is more compatible with the gradual degradation assumption of components and therefore make it easier for models to be trained on. Moreover, if binary prediction of failure is of interest, then after predicting TTE, it can be converted to binary by thresholding. According to the above reasoning, we choose to stick to TTE labeling method.
There is yet another problem arises in both of these labeling methods which is the censoring problem. Let's consider the TTE labeling scheme. The problem is that readout logged after the last failure cannot be assigned labels. That is because in the first place we do not know whether there will be a failure or not, and even if there will be, we do not know when. There are some researches on how to make use of these censored part of the data [Allik et al., 2016b, Allik et al., 2016a]. However, we decided to neglect these censored part for the sake of simplicity. Both TTE labeling and censored data concept are shown in Figure 13.

Covariate shift detection

A very important problem in many real world datasets especially for datasets with time series nature, is the problem of covariate shift. Covariate shift happens when the distribution of features (input variables) changes between train and test datasets or more generally between different partitioning of dataset. Formally, covariate shift arises when,

$$P_{train}(X) \neq P_{test}(X) \;\; and \;\; P_{train}(Y|X) = P_{test}Y|X \tag{2}$$

In other words, change in the joint distribution of X, Y originates from difference in marginal distribution of X_{train}, X_{test}.

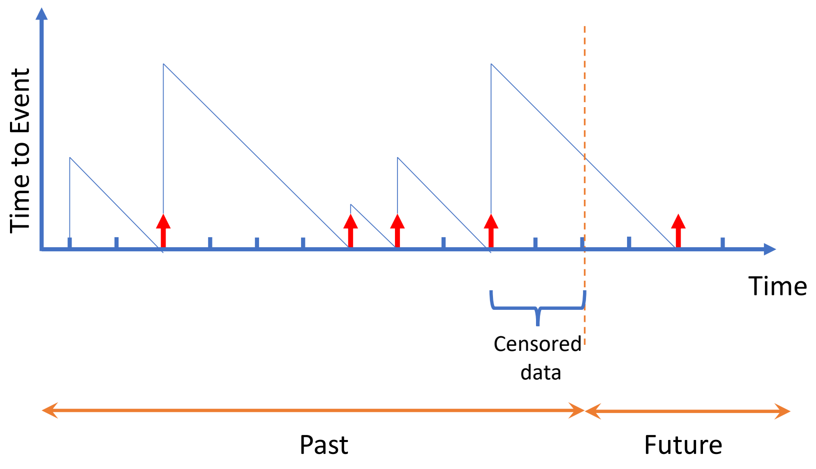


Figure 13: Time to Event (TTE) and censored data

This problem is more serious when the features with covariate shift are considered as important features by the machine learning models. For instance in tree models if a split on a feature with high covariate shift causes a large reduction in impurity, that feature will be considered as an important feature. In the case of neural networks, if weights connected to a high covarite shift feature are large, and if possibly large signal carried through the pathway of that feature towards the end of the network, makes that feature influential. Having influential features with high covariate shift will result in poor generalization.
For this study we adopt a method which has been shown to be practical for many datasets. The idea behind this method is to train a classifier to distinguish between train and test. Algorithm 2 describes the steps of the method.
It is worth mentioning that covariate shift is a subclass of dataset shift. More generally, in additional to marginal distribution shift of features, the conditional distribution of target given features ($P(Y|X)$) can be changed for different splits of dataset. Since Algorithm 2 does not consider actual target of the data into account, it only can detect covariate shift.
In Algorithm 2, a LightGBM [Ke et al., 2017] model is used as the classifier. Now, we need to find a proper threshold (tr) to compare to Lightgbm's AUC for selecting features. Different values of tr result in different subset of features. For each subset of features (obtained for a specific tr), a LSTM

Algorithm 2 Covariate shift detection algorithm

```
Input:
    $X$                          ▷ Entire data
    $tr$                         ▷ Threshold for shift detection
Output:
    $cs\_features$               ▷ Set of features detected as high shift

1: Split X into $X_{train}$ and $X_{test}$
2: Add a label column to $X_{train}$ and $X_{test}$ and assign 1 to $X_{train}$ and 0 to $X_{test}$
3: Concatenate $X_{train}$ and $X_{test}$ and shuffle to create $X_{concat}$
4: Split $X_{concat}$ to create a new $X_{train}$ and $X_{test}$
5: for col in $X.columns$ do
6:     $X^{col}_{train} = X_{train}[col]$
7:     $X^{col}_{test} = X_{test}[col]$
8:     Train a classifier on $X^{col}_{train}$
9:     $test\_auc$ = Calculate auc on $X^{col}_{test}$
10:    if $test\_auc > tr$ then
11:        Add col to $cs\_features$
12:    end if
13: end for
14: return $cs\_features$
```

with the Mean Absolute Error(MAE) cost function is trained. Finally, the obtained MAEs for different values of *tr* are compared to get the best *tr*, and consequently features subset. The result of this experiment is shown in Figure 14, which shows that choosing threshold to be 0.8 leads to the best MAE. Number of features associated with this threshold is 355.

Experimental Results

In this section the result of applying the stacked ensemble will be reported from both classification and regression perspective. Since the labeling of the data provided with TTE is in regression format, first, the result of regression is reported. Then by applying threshold on predictions, they will be converted and reported as the classification results. For comparison, the result of stacked ensemble will be compared to simple ensemble aggregation such as average

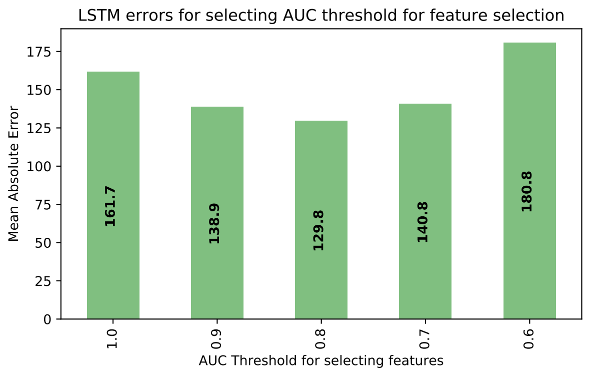


Figure 14: LSTM error results for choosing threshold in feature selection in Algorithm 2.

and median of all base models predictions. Moreover, ensemble result will be compared to the best individual base model result.
Starting from regression result, Table 8 shows the MAE of different models including ensembles and best individual model. The figures indicate that both mean and median aggregation of the result of base models on test data beat the best individual model. However, the result of Stacked ensemble outperforms the mean and median ensembles.

Table 8: Model comparison

Model	MAE
Best individual model	139.32
Mean of all base models	125.63
Median of all base models	124.26
Stacked ensemble	105.79

A sign of diversity in stacking is that the base models' predictions have low correlation with respect to each other. Table 9 depicts some statistics about correlation of the base models.
The average correlation of 0.56 is a good indicator of diversity between base models. Note that in the context of stacking 0.75 correlation (the max correlation) is not still a high correlation given that the models are trying to

Table 9: Correlation statistics of base models

Mean correlation of all models	0.56
Max correlation	0.75
Min correlation	0.42

predict the same target.
Figure 15 and 16 show the prediction of stacked ensemble predictions and best model prediction against target values, respectively. In Figure 17 the two previous figures are plotted together for better comparison of stacked ensemble predictions and best model predictions.

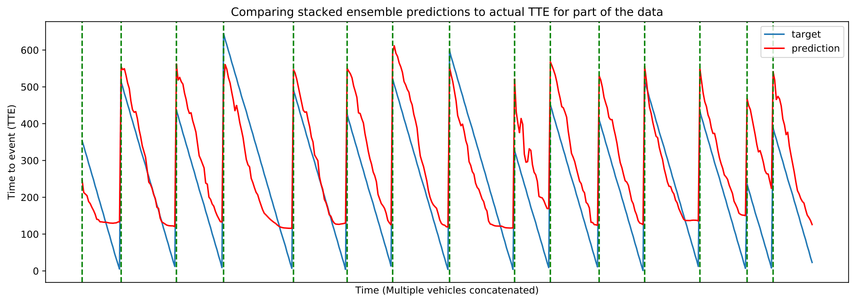


Figure 15: Comparing stacked ensemble results to the target values

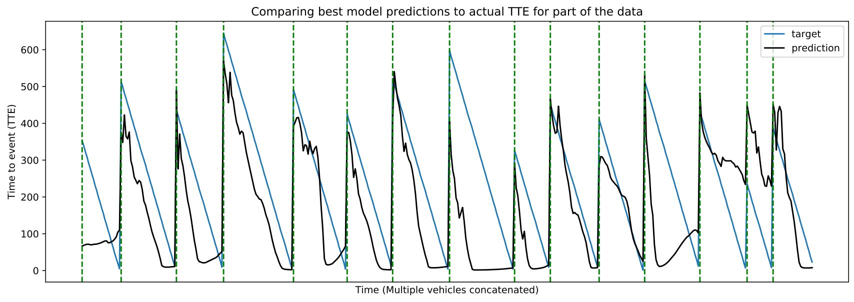


Figure 16: Comparing best model results to the target values

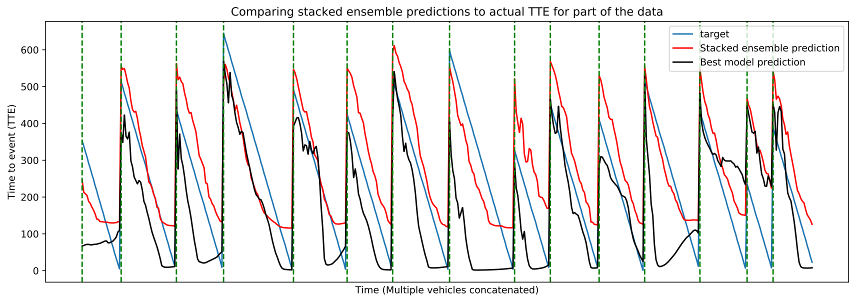


Figure 17: Stacked ensemble predictions, best model predictions, and target value comparison.

As can be seen in Figure 16 and 17, predictions of the best model drop too early in comparison to target TTE. Also, it can be seen that there are more unnecessary fluctuations in its predictions. On the other hand, the stacked ensemble predictions drop much closer to the target TTE, while the predictions of the ensemble method are much more stable.
In order to convert the TTE predictions to binary fault classification, the TTE predictions are compared to a threshold derived from the training part of the data and then applied to test partition of the data.
Figure 18 illustrates the result of converting TTE to binary fault prediction using AUC curves for the best base model and stacked-ensemble model. As can be seen, there is a considerable boost achieved using stacked ensemble method. It is also noticeable from Figure 17 that since the base model predictions drops faster relative to target TTE, applying the threshold causes more false positive in comparison to stacked ensemble predictions. In other words, since the predictions of stacked ensemble are more aligned with the target TTE, for the same value of FPR, TPR is higher (elbow of stacked-ensemble).

6.4.3 Recurrent Neural Networks for Fault Detection using 2D histogram data

Similar to the previous implementation, in this experiment we took the advantage of Recurrent Neural Network (RNN) aimed to predict the components' failure by exploiting 2D histogram data. Indeed, the objective of this experiment is to perform a binary classification of healthy and unhealthy trucks –focusing on steering-pump component–, using truck-data readouts

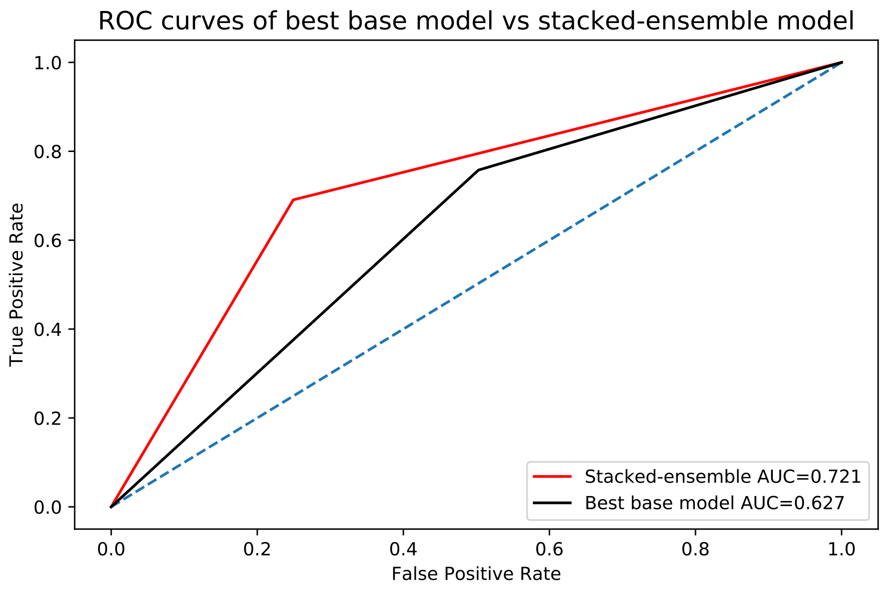


Figure 18: AUC curves for comparing stacked-ensemble and best base model

represented as sequences of samples of 2D histogram-data originating from engine speed and torque.
The data includes the samples, which are captured and logged from heavy duty trucks over a time-period of two years. One sample is defined as a histogram, expressing the relation between 'engine speed' and 'engine torque' signals, with 20 x 20 bins. This in turn defines one sample as 400 features and their labels. An example of a 2D histogram is shown in Figure 19.
In order to label the data, we have used Equation 1 having in mind 90 days time windows. We took into account this 90 days to be the interval in which the symptoms of a forthcoming failure are most likely to be visible, and when the vehicle usage has the highest impact on a failure. Hence, samples are labeled to 1 as unhealthy if and only if a repair for the interested component has been reported, otherwise they are labeled to 0 as healthy vehicles. Consequently, we defined 90 days as the prediction horizon in this binary classification problem using RNN to predict the component failures/repairs before they really happen.
Once the labeling took the place, we have pre-processed the data so that all the 2D histogram samples grouped into sequences of five time-ordered. As

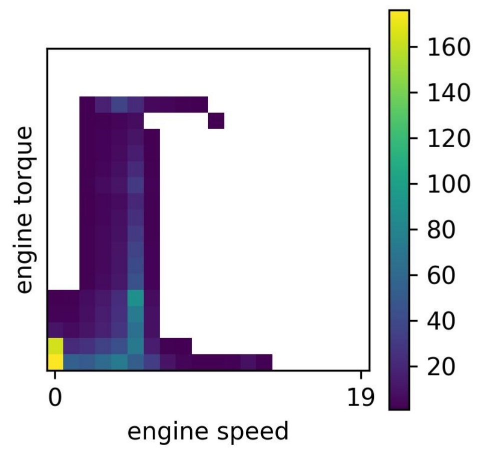


Figure 19: Engine speed and torque histogram data

Data Sets	Characteristics		Note
	# of Samples and Sequences	Distribution of Data	
V1=original	Samples= 521273	very unbalanced	The biggest data-set is used
V1=extracted	Sequences= 4426	—	Derived from V1 as sequences of 5 samples

Table 10: The characteristics of the original data and extracted data.

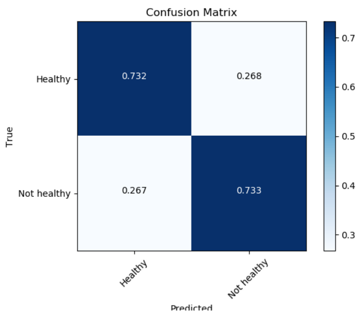
it is represented in Table 10, 521273 histogram samples have been grouped into sequences, then 4420 sequences are randomly selected to be used for this experiment.
To conduct the experiment, we partitioned the data into three sets of train, validation and test including 70%, 20% and 10% of the data, respectively. Then, we have implemented the RNN using tensorflow[1]and ConvLSTM2D Dense objects/layers to train the model. The input layer is set to sequence of inputs where each sequence consists of 5 samples with 20x20 dimensions. We have used 2 ConvLSTM2d, 1 Flatten and five Dense layers in this network.

[1]We took the advantage of tensorflow.keras package using ConvLSTM2D and Dense available in Python: `https://www.tensorflow.org/api_docs/python/tf/keras/layers/ConvLSTM2D`

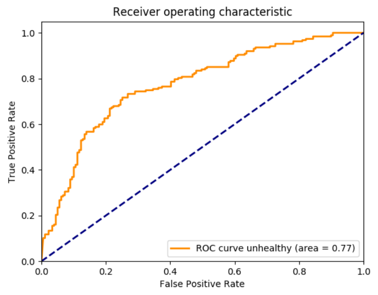
More details of the configuration in our network are illustrated in Table 11.

RNN Layers and settings	**Notes and details**
Input-layer	input-Shape=5 samples (1 sequence) with 20x20 dimensions)
ConvLSTM2D	filters=5, kernel_size=(5, 5), padding=same, return_sequences=True, stateful=False, activation=relu
ConvLSTM2D	filters=10, kernel_size=(5, 5), dilation_rate=[2, 2], padding=same, return_sequences=False, stateful=False, activation=relu
Flatten layer	Flattening of previous layer
Dense	neurons=400, activation=relu
Dense	neurons=400, activation=relu
Dense	neurons=200, activation=relu
Dense	neurons=200, activation=relu
Dense	neurons=20, activation=relu
Output layer	neurons=2, activation=softmax
Optimizer	Adagrad(lr=0.01)
Loss-function	sparse_categorical_crossentropy

Table 11: Table of best RNN-model configuration



(a) Normalized confusion matrix.



(b) ROC-curve.

Figure 20: RNN experimental results.

Figure 20 shows the confusion matrix and ROC curve obtained by the RNN algorithm. The normalized confusion matrix indicates that our RNN provides an admissible result, such that the model could correctly predict both healthy

and unhealthy vehicles almost with similar performances $\sim 73\%$. It can be seen that False Positive and False Negative as prediction errors of the classifier are relatively low in this complex problem by $\sim 27\%$. In addition, Receiver Operating Characteristic (ROC) curve, which is depicted in Figure 20b, shows that the classifier with our setting performs well in predicting the two classes with $AUC = 0.77$.
All the above figures obtained from RNN implementation highlight that 2D histogram data offers a valuable knowledge/pattern, so that classifier could provide prediction with admissible results.

6.5 Results for WP4: Causal discovery using clusters from observational data

Many methods have been proposed over the years for distinguishing causes from effects using observational data due to its complexity we progress by creating powerful heuristics, capable of inferring necessary information present in real data.
The idea of explicitly considering cluster as latent variables previously proposed by Sgouritsa et. al [Sgouritsa and Scholkopf, 2013]. They used a kernel method to infer a common cause of a sets of variables by clustering the variables. However, there is no hint for selecting these variables efficiently which makes their method computationally expensive. Clusters are a very typical occurrence that should be taken into account, and exploited, in the process of identifying causes and effects. In this experiment we show that inhomogeneities in the data, appearing due to a confounding factors, can hide existing causal relationship between two variables. We try to resolve this problem based on structure of discovered causal network that can reduce ambiguity in the causal structure. We also propose a new method that, as shown by experiments performed on synthetic and real data, is able to improve the quality of learned causal structures.
Figure 21 represents the existence of correlation among variables in many different ways. It is often implicitly assumed that the relationship between two continuous-valued variables will take some functional form, such that the value of one variable can be expressed as a (possibly approximate or noisy) function of the other. However, existence of clusters in the data, while a very common occurrence, leads to a very different manifestation of the correlation between variables. We claim that clusters should be addressed explicitly.

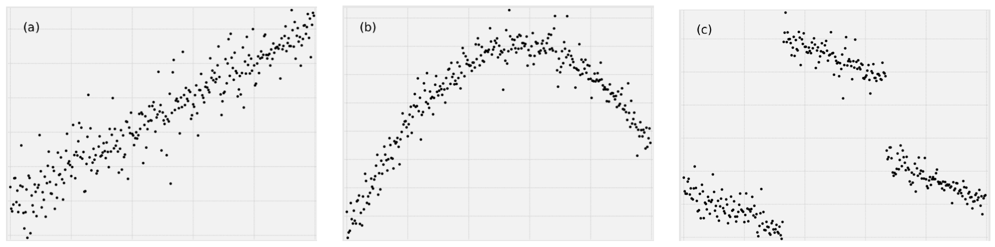


Figure 21: Different kinds of data: (a) linear correlation (b) non-linear functional correlation (c) correlation with clusters.

Method

The intuition behind our idea lies in the notion that “lumpiness in the data must have a reason”. This is similar to the principle used in the Information Geometric Causal Inference (IGCI) class of methods [Janzing and Schlkopf, 2012], where entropy is used to estimate whether an attribute is likely a cause or an effect. In our case, it corresponds to noticing varying density between cluster variables has lower entropy than variables with same variance therefore clusters among larger sets of attributes are considered for calculations.
The proposed method starts by using an existing algorithm for finding an initial causal structure. For this, we have here used the PC algorithm [Spirtes et al., 2000][Kalisch and Bhlmann, 2012]. The PC algorithm builds a Markov equivalence class which contains the underlying causal graph and represents it by a Completed Partially Directed Acyclic Graph (CPDAG). The calculations are based on conditional independence tests. In many cases, the graph produced contains bi-directed edges. Such bi-directed edges are undesirable, as they imply the existence of a confounding variable.
The next step of the method is to look for clustered data in the subset of variables constituting such cliques of ambiguous causal direction. The clustering method used here is a Gaussian Mixture Model (GMM) [Zivkovic, 2004] trained by Expectation Maximization (EM). This is combined with a test for “homogeneity”, in this case, defined as being similar to a (multivariate) Gaussian distribution, since it is the distribution (in Euclidean space) which has the highest entropy for a given variance. However, most tests for Gaussian distributions is based on checking moments of the distribution. If the local density is much higher than expected, this indicates a clear clustered structure. The homogeneity measure is used both to check whether there are any clusters for a clique of variables and to determine the number of clusters required.

Our contribution is to extend existing causal inference algorithms with a cluster-based conflict resolution mechanism. In the final step, the PC algorithm is run again, this time with the added new variable, and values corresponding to cluster indices. If successful, the added variables will resolve the cliques of ambiguous causal direction.
Furthermore, for determining the causal direction in the last phase of the PC algorithm, we can use the assumption that the clustered structure in the data is correspond to a cause variable. Therefore, the causal direction for this variable is from the clustered index to the variables showing traces of this clustering.
The above steps are then repeated until no more unexplained clusters in the data are left. In this way, the existence of clusters in data will help resolve the causal directions. We now turn to the evaluation of the proposed method.

Empirical evaluation

The proposed method is tested on the real-world data set collected by a fleet of six city buses. In particular, the aim is to identify the causal relation between the set of signals influencing fuel consumption.

Fleet of city buses

One application domain on which this approach has been evaluated is to identify the causal relations between signals measured on-board heavy duty vehicles. A causal network can be utilized to understand how various parameters are affecting the vehicle's performance.
The attributes in the fleet of city buses, that we used, are AcceleratorPedalPos, AmbientAirTemperature, BrakePedalPos, EngineCoolantTemperature, EngineSpeed, FuelRate, RelativeSpeedFrontLeft, RelativeSpeedFrontRight, SelectedGear, SteeringWheelAngle, VehicleSpeed.
In this dataset, there is one discrete variable i.e selected gear which appears to cause clear clusters in the data. Figure 22 shows the relation between the engine speed and the vehicle speed. In this example, selected gear causes the grouping into clusters of different slopes. Hence, these form of clusters must be taken into account when trying to distinguish causes from effects in the data. There is another clustering where a vertical streak is observer at higher density for all gears around of engine speed of 600 rpm corresponding to an engine that is idling. Variation in engine speed was observed during start off

the vehicle due to idling. There is no variable in the data corresponding to idling, hence this is one candidate for a latent variable. It is very likely that the causal effects in the engine differ between idling and normal operation.

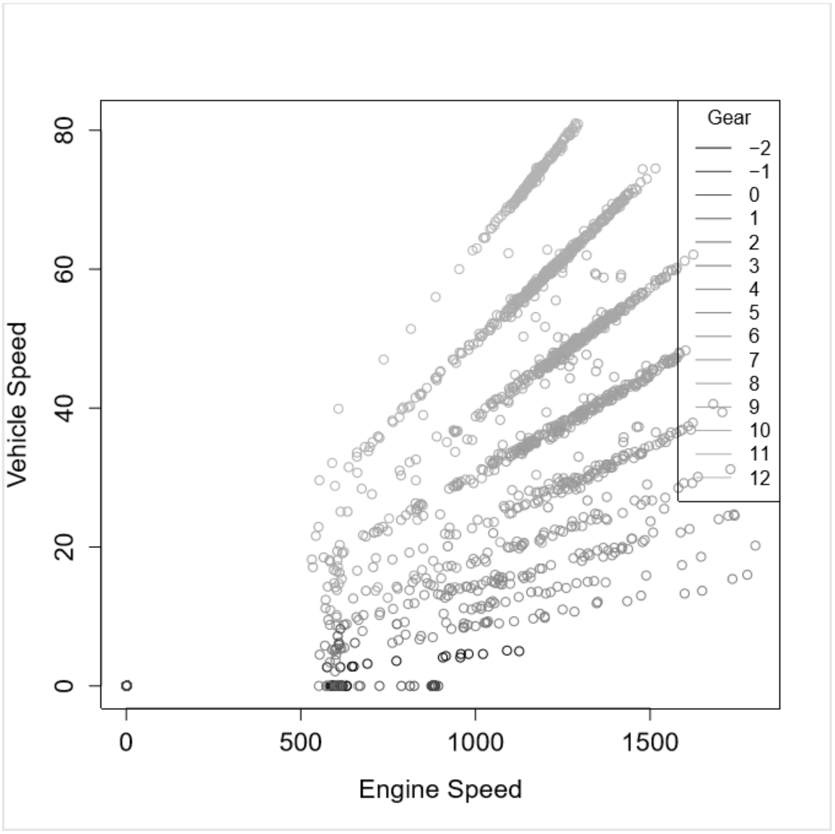


Figure 22: The clusters in the engine vs. vehicle speed plot can be explained by the selected gear.

We have added two latent variables to the data to account for clustered structures, “LatentVar1” and “LatentVar2”. The first added cluster variable (“LatentVar1”) represents “idle run” (engine speed below or around 620 rpm), and the second (“LatentVar2”) indicates whether the bus is an old or new generation (for example, the “Steering Wheel Angle” signal is not measured in the old generation).
We then run the PC algorithm, first without the explained two latent variables, then with them. The causal relation of the 11 measured attributes as a result of the PC algorithm is shown in Figure 23. Note that the number of bi-directed

edges (ambiguous causal direction) differs in the resulting graphs: there are 5 fewer bi-directed edges when the latent variables are added. This shows that adding these variables helps in identifying causal directions between the other variables. Moreover, some of the remaining bi-direction edges in the graph with latent variable (Figure 23b) make sense.

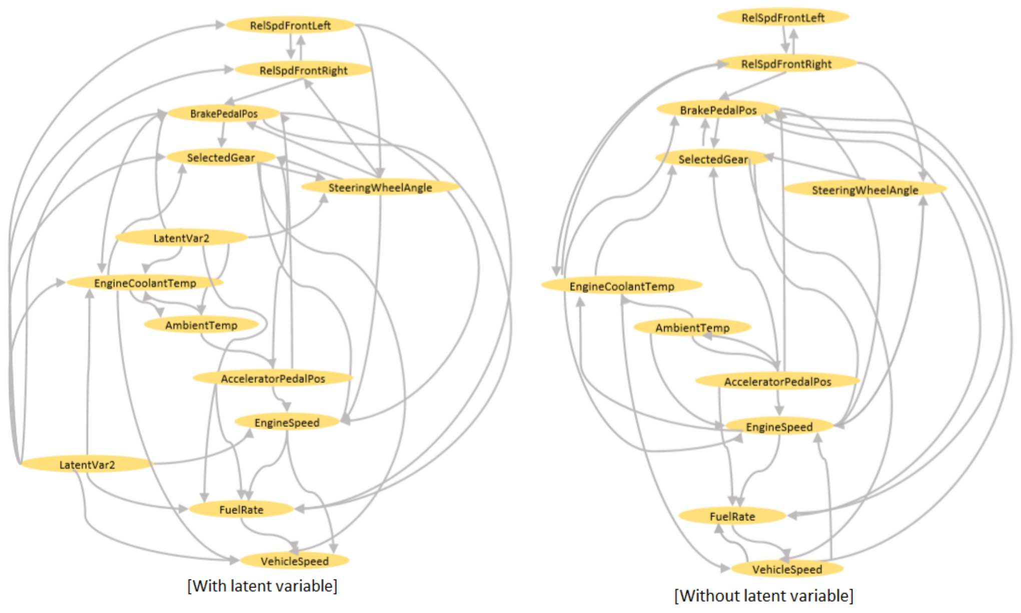


Figure 23: Results of the proposed algorithm for the bus fleet data.

We propose and discuss the idea that clusters in the observational data should be used as a clue for causal discovery. The underlying idea is that inhomogeneities in the data are unlikely to appear spontaneously; instead, they probably have a cause. If none of the variables existing in the data can explain the cluster structure, it is an indication that a latent variable may be behind it. However, it should be noted that this is only a heuristic. There are many counterexamples, where inhomogeneities in the data can be created from continuous processes.
Still, this is the same for most methods based on, for example, entropy differences; that it is an indication and not the absolute truth. Deducing causal direction from data is hard enough we must use all available information provided, and actively look for more ideas. To the best of our knowledge, information related to cluster structure has not been addressed in any of the

previously works.
Clusters are abundant in real world data, which makes this an important addition to the causal discovery tool box.

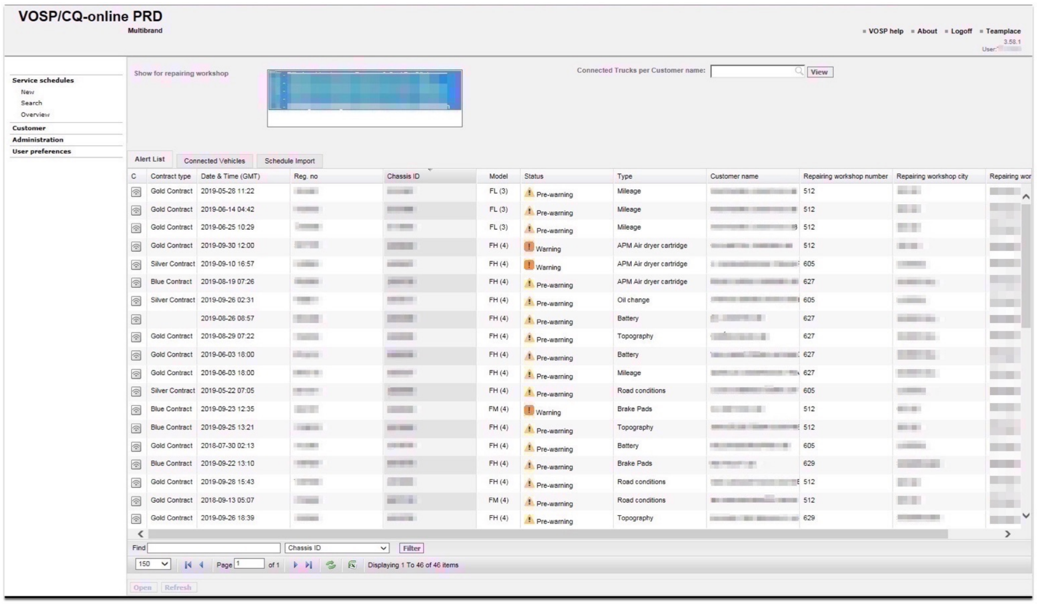
7 Dissemination and publications

7.1 Dissemination

7.1.1 Deployment on the VOSP server

The results of the HEALTH project has been deployed in Volvo Trucks production environment called VOSP for prediction of vehicle component status and breakdown prognosis of trucks. VOSP is a software developed by Volvo to optimize the service planning of the trucks, which is used by authorized workshops
The deployment has been offered as an advanced service for the trucks under the gold contracts between Volvo dealers and customers. The work has been limited to gold contract trucks because of three main reasons. First, we wanted to make sure that potentially we have access to the entire service and maintenance history of the truck. They are also chosen since customers covered by this type of contracts do not need to go to any other dealer than authorized workshops. The second reason is associated with the branding and commercial purposes of the company to offer advanced maintenance services to gold contracts so that other customers would be encouraged to go under its coverage. The third reason is that there is a data management agreement in place for gold contract trucks, which allows using of the data for Uptime service.
The deployment aimed to run a pilot on a set of selected markets for a specific set of components. There are four markets included in the pilot i.e., Sweden, Switzerland, Finland, and Denmark. Usually, the predictive maintenance alerts are intended for sophisticated components that a physical measurement by a simple sensor could not detect the wear out or sometimes failure of the part. The list of the selected target components includes the following five vehicle parts: air compressor, turbocharger, front and rear air below, and alternator.
Figure 24 displays the user interface of the VOSP server. As it can be seen in the screenshot, the predictive maintenance system produces warning messages

on the server for the Volvo dealers. The warning message informs the dealer of an increased risk of failure for a specific component of a specific truck. Finally, based on the part, the system recommends some actions to the dealer to take like planning an earlier visit for the customer to check the component or to consider repairing or replacing the component for the next visit.



C	Contract type	Date & Time (GMT)	Reg. no	Chassis ID	Model	Status	Type	Customer name	Repairing workshop number	Repairing workshop city	Repairing wor
	Gold Contract	2019-05-28 11:22	[redacted]	[redacted]	FL (3)	Pre-warning	Mileage	[redacted]	512	[redacted]	[redacted]
	Gold Contract	2019-06-14 04:42	[redacted]	[redacted]	FL (3)	Pre-warning	Mileage	[redacted]	512	[redacted]	[redacted]
	Gold Contract	2019-06-25 10:29	[redacted]	[redacted]	FL (3)	Pre-warning	Mileage	[redacted]	512	[redacted]	[redacted]
	Gold Contract	2019-09-30 12:00	[redacted]	[redacted]	FH (4)	Warning	APM Air dryer cartridge	[redacted]	512	[redacted]	[redacted]
	Silver Contract	2019-09-10 16:57	[redacted]	[redacted]	FH (4)	Warning	APM Air dryer cartridge	[redacted]	605	[redacted]	[redacted]
	Blue Contract	2019-08-19 07:26	[redacted]	[redacted]	FH (4)	Pre-warning	APM Air dryer cartridge	[redacted]	627	[redacted]	[redacted]
	Silver Contract	2019-09-26 02:31	[redacted]	[redacted]	FH (4)	Pre-warning	Oil change	[redacted]	605	[redacted]	[redacted]
		2019-08-26 08:57	[redacted]	[redacted]	FH (4)	Pre-warning	Battery	[redacted]	627	[redacted]	[redacted]
	Gold Contract	2019-08-29 07:22	[redacted]	[redacted]	FH (4)	Pre-warning	Topography	[redacted]	627	[redacted]	[redacted]
	Gold Contract	2019-06-03 18:00	[redacted]	[redacted]	FH (4)	Pre-warning	Battery	[redacted]	627	[redacted]	[redacted]
	Gold Contract	2019-06-03 18:00	[redacted]	[redacted]	FH (4)	Pre-warning	Mileage	[redacted]	627	[redacted]	[redacted]
	Silver Contract	2019-05-22 07:05	[redacted]	[redacted]	FH (4)	Pre-warning	Road conditions	[redacted]	605	[redacted]	[redacted]
	Blue Contract	2019-09-23 12:35	[redacted]	[redacted]	FM (4)	Warning	Brake Pads	[redacted]	512	[redacted]	[redacted]
	Blue Contract	2019-09-25 13:21	[redacted]	[redacted]	FH (4)	Pre-warning	Topography	[redacted]	512	[redacted]	[redacted]
	Gold Contract	2018-07-30 02:13	[redacted]	[redacted]	FH (4)	Pre-warning	Battery	[redacted]	605	[redacted]	[redacted]
	Blue Contract	2019-09-22 13:10	[redacted]	[redacted]	FH (4)	Pre-warning	Brake Pads	[redacted]	629	[redacted]	[redacted]
	Gold Contract	2019-09-28 15:43	[redacted]	[redacted]	FH (4)	Pre-warning	Road conditions	[redacted]	512	[redacted]	[redacted]
	Gold Contract	2018-09-13 05:07	[redacted]	[redacted]	FM (4)	Pre-warning	Road conditions	[redacted]	512	[redacted]	[redacted]
	Gold Contract	2019-09-26 18:39	[redacted]	[redacted]	FH (4)	Pre-warning	Topography	[redacted]	629	[redacted]	[redacted]

Find: Chassis ID

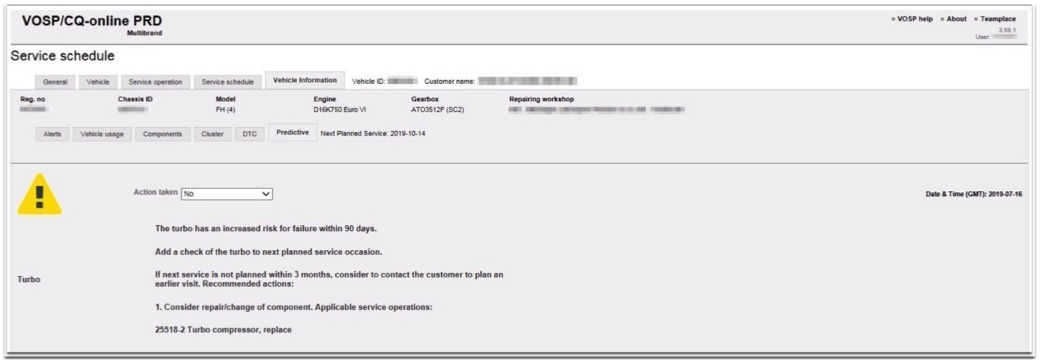
150 Page 1 of 1 Displaying 1 To 46 of 46 items

Figure 24: User interface of the VOSP server

The alerts produced by the system are intended to support the service planning process of the truck. This means that by predicting possible future failures, not only we avoid extra visits to the workshop, but also, we avoid extra costs related to unexpected breakdowns like towing costs or uptime promises. On the other hand, the workshop people can preplan the repair process based on the prognosis warning messages and order the required hardware in advance and make sure that they have the right technician available for delivering the service. Figure 25 illustrates the details of a prognostic alert produced by the system for a specific truck.

7.2 Publication

The HEALTH project has led to the following publications and master thesis that corroborate the objectives dened in Section5:



Service schedule

User: [redacted]

Vehicle ID: [redacted] Customer name: [redacted]

Reg. no	Chassis ID	Model	Engine	Gearbox	Repairing workshop
[redacted]	[redacted]	FH (4)	D16K750 Euro VI	ATO3512F (SC2)	[redacted]

Next Planned Service: 2019-10-14

Action taken: No

Date & Time (GMT): 2019-07-16

Turbo

The turbo has an increased risk for failure within 90 days.

Add a check of the turbo to next planned service occasion.

If next service is not planned within 3 months, consider to contact the customer to plan an earlier visit. Recommended actions:

1. Consider repair/change of component. Applicable service operations:

25518-2 Turbo compressor, replace

Figure 25: Details of a predictive alert on the VOSP server for a truck

- Published:
  - K. Chen, S. Pashami, Y. Fan and S. Nowaczyk, ”Predicting Air Compressor Failures Using Long Short Term Memory Networks”, EPIA Conference on Artificial Intelligence, 2019.
  - P. Pirasteh, S. Nowaczyk, S. Pashami, M. Löwenadler, K. Thunberg, H. Ydreskog, and P. Berck ”Interactive feature extraction for diagnostic trouble codes in predictive maintenance”, Proceedings of the Workshop on Interactive Data Mining (WIDM 19)..
  - S. Pashami, et al. ”Causal discovery using clusters from observational data.” FAIM'18 Workshop on CausalML, 2018.
  - MR. Bouguelia, A. Karlsson, S. Pashami, S Nowaczyk, and A. Holst ”Mode tracking using multiple data streams”, Information Fusion 43, 33-46.
- Accepted:
  - R. Khoshkangini, S. Pashami and S. Nowaczyk, ”Bayesian Network for Failure Prediction in Different Seasons”, ESREL2020-PSAM15.
  - M. Rahat, S. Pashami and S. Nowaczyk ”A: Predicting Turbocharger failures using knowledge representation learning over Logged vehicle datarocess for Proactive mainte”, ESREL2020-PSAM15.
  - M. Rahat, S. Pashami, and S. Nowaczyk, ”Modeling turbocharger failures using Markov process for Proactive maintenance”, ESREL2020-PSAM15.

- Submitted:
  - R. Khoshkangini, S. Pashami and S. Nowaczyk," Feature Extraction for Prediction of Vehicles with High Risk of Failure", IDA2020.
  - P. Berck, S. Narayanan, S. Pashami, "Resolving Class Imbalance using Generative Adversarial Networks"
  - P. Mashhadi, S. Nowaczyk and S. Pashami, "Stacked Ensemble of Recurrent Neural Networks for Predicting Turbocharger Remaining Useful Life", Applied Science Journal.
  - P. Berck, S. Thondgere, V. Nataraj and S. Pashami, "Resolving Class Imbalance using Generative Adversarial Networks", ESREL2020-PSAM15.
- Master Theses:
  - K. Chen,"Recurrent Neural Networks for Fault Detection"
  - N. Sushmitha and N. Vismitha, "Resolving Class Imbalance Using Generative Adversarial Networks"

8 Conclusion and future research

Early prediction of component failures is one of the possible ways to increase the maintenance quality service which assures up-time, safety, and availability of parts.
The HEALTH project provided several machine leaning solutions to early detection of vehicles' component failure accommodating multiple available data sources. In this regard, we have illustrated how we can take advantage of Recurrent Neural Network and Markov process for considering operation history of the vehicles. Different sources of data including logged vehicle data, vehicle service records and diagnostic trouble codes have been used in this study. Furthermore, various Generative Adversarial Neural Networks were implemented to alleviate the imbalanced problem of the faulty class, which is one of the big challenges in this research area. Based on our analysis creating a balanced training set improve the failure prediction of different components. The journey for the deployment of machine learning based predictive maintenance has also taken place during this project. IBM solutions have been used for the deployment and training of the methods in the production environment. The results are presented through the Volvo truck production environment called VOSP producing a warning message to the dealer to express a high risk of component failures for each individual vehicle. This potentially helps the workshops people to do actions beforehand.
We are planning to continue the collaboration between Halmstad University and Volvo Group on data-driven failure forecasting. We applied together for a new project founded by Knowledge Foundation (KKS). This project concentrates on knowledge representation for predictive maintenance.

9 Participating parties and contact person

For more information, please contact:
Magnus Löwenadler, magnus.lowenadler@volvo.com
Sławomir Nowaczyk, Slawomir.Nowaczyk@hh.se

References

[Allik et al., 2016a] Allik, B., Miller, C., Piovoso, M. J., and Zurakowski, R. (2016a). The tobit kalman filter: An estimator for censored measurements. *IEEE Transactions on Control Systems Technology*, 24(1):365–371.

[Allik et al., 2016b] Allik, B., Piovoso, M. J., and Zurakowski, R. (2016b). Recursive estimation with quantized and censored measurements. In *2016 American Control Conference (ACC)*, pages 5130–5135.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia. PMLR.

[Berthelot et al., 2017] Berthelot, D., Schumm, T., and Metz, L. (2017). Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*.

[Chinnam, 1999] Chinnam, R. B. (1999). On-line reliability estimation of individual components, using degradation signals. *IEEE Transactions on Reliability*, 48(4):403–412.

[Dash and Liu, 2000] Dash, M. and Liu, H. (2000). Feature selection for clustering. In *Pacific-Asia Conference on knowledge discovery and data mining*, pages 110–121. Springer.

[Eddy, 1998] Eddy, S. R. (1998). Profile hidden markov models. *Bioinformatics (Oxford, England)*, 14(9):755–763.

[Fergus et al., 2013] Fergus, P., Cheung, P., Hussain, A., Al-Jumeily, D., Dobbins, C., and Iram, S. (2013). Prediction of preterm deliveries from ehg signals using machine learning. *PLOS ONE*, 8(10):1–16.

[Ganganwar, 2012] Ganganwar, V. (2012). An overview of classification algorithms for imbalanced datasets. *International Journal of Emerging Technology and Advanced Engineering*, 2(4):42–47.

[Ganin et al., 2016] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[Graves et al., 2013] Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. pages 6645–6649.

[Güneş et al., 2017] Güneş, F., Wolfinger, R., and Tan, P.-Y. (2017). Stacked ensemble models for improved prediction accuracy. In *Proc. Static Anal. Symp.*, pages 1–19.

[Heusel et al., 2017] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium.

[Janzing and Schlkopf, 2012] Janzing, Dominik, M. J. Z. K. L. J. Z. J. D. P. S. B. and Schlkopf, B. (2012). Information-geometric approach to inferring causal directions.artificial intelligence.

[Japkowicz, 2000] Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proc. of the Int'l Conf. on Artificial Intelligence*.

[Kalisch and Bhlmann, 2012] Kalisch, Markus, M. M. C. D. M. M. and Bhlmann, P. (2012). Causal inference using graphical models.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.

[Khoshkangini et al., 2019] Khoshkangini, R., Pashami, S., and Nowaczyk, S. (2019). Warranty claim rate prediction using logged vehicle data. In *EPIA Conference on Artificial Intelligence*. Springer.

[Liu et al., 2008] Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2008). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550.

[Lu and Meeker, 1993] Lu, C. J. and Meeker, W. O. (1993). Using degradation measures to estimate a time-to-failure distribution. *Technometrics*, 35(2):161–174.

[Maillart and Pollock, 2002] Maillart, L. M. and Pollock, S. M. (2002). Cost-optimal condition-monitoring for predictive maintenance of 2-phase systems. *IEEE Transactions on Reliability*, 51(3):322–330.

[Mirza and Osindero, 2014] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[Nowaczyk et al., 2013] Nowaczyk, S., Prytz, R., Rögnvaldsson, T., and Byttner, S. (2013). Towards a machine learning algorithm for predicting truck compressor failures using logged vehicle data. In *12th Scandinavian Conference on Artificial Intelligence, Aalborg, Denmark, November 20–22, 2013*, pages 205–214. IOS Press.

[Prytz et al., 2015] Prytz, R., Nowaczyk, S., Rögnvaldsson, T., and Byttner, S. (2015). Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering applications of artificial intelligence*, 41:139–150.

[Sgouritsa and Scholkopf, 2013] Sgouritsa, Eleni, J. D. P. J. and Scholkopf, B. (2013). Identifying finite mixtures of non-parametric product distributions and causal inference ofconfounders. *arXiv preprint*.

[Shao and Nezu, 2000] Shao, Y. and Nezu, K. (2000). Prognosis of remaining bearing life using neural networks. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 214(3):217–230.

[Spirtes et al., 2000] Spirtes, P., Glymour, C. N., Scheines, R., Heckerman, D., Meek, C., Cooper, G., and Richardson, T. (2000). *Causation, prediction, and search.* MIT press.

[Stratonovich, 1965] Stratonovich, R. L. (1965). Conditional markov processes. In *Non-linear transformations of stochastic processes*, pages 427–453. Elsevier.

[Wagstaff et al., 2001] Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *Icml*, volume 1, pages 577–584.

[Wolpert, 1992] Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

[Zivkovic, 2004] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31. IEEE.