Technical University of Munich

Master's Thesis in Mechanical Engineering

Survival Models for Vehicle Predictive Maintenance

Jonas Felix Haderlein

Technical University of Munich

Master's Thesis in Mechanical Engineering

Survival Models for Vehicle Predictive Maintenance

Survival Modelle zur prädiktiven Wartung von Kraftfahrzeugen

Editor:	Jonas Felix Haderlein
Supervisor:	Prof. Dr.-Ing. habil. Alois Christian Knoll
Advisors:	M.Sc. Michael Truong Le
	Dipl.-Ing. Gereon Hinz
Submission date:	18th October 2017

Sperrvermerk

Die Arbeit beinhaltet vertrauliche Informationen, die der Geheimhaltung unterliegen. Aus diesem Grund darf die Arbeit ohne vorherige schriftliche Zustimmung der BMW AG nicht vervielfältigt oder veröffentlicht werden. Dieser Zustimmungsvorbehalt endet am: 31.12.2022

Jonas Haderlein

Eidesstattliche Erklärung

Ich versichere, dass ich diese Master Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Jonas Haderlein

1 Abstract

Vehicle digitalization paves the way for various cutting-edge approaches to enhance mobility services. Predictive Maintenance for vehicles is an upcoming methodology to monitor deterioration processes and prevent failures and vehicle downtime in turn. The obvious advantages range from customer driving satisfaction without disruptions or malfunctions to several business opportunities like warranty cost savings for vehicle manufacturers. Comprehensive research is now underway to enable future maintenance services in the car industry. In contrast to common predictive analytics, the multitude of vehicles cannot be continuously monitored but only "diagnosed" at for instance dealer workshop visits given aggregated sensor data. In addition, failures occur that have not been considered to be crucial before production and respective components are sometimes not directly surveilled by onboard sensors. Hence, the resulting data poses specific challenges to data-driven predictive maintenance approaches. This paper derives new survival model solutions that improve the state-of-the-art binary classification which is only able to separate into high and low pseudo risks. The proposed methodology provides a scalable machine learning technique to predict real failure probabilities of specific vehicle components learning on historical customer fleet data. Yet, the growing complexity and heterogeneity of vehicle sensor systems and recorded data make it increasingly difficult to determine the root cause of component failures. An implemented feature selection pipeline thus extensively searches for patterns in the provided data and assesses significant influences on the wear process. Survival models are then employed to infer risks of failure over time to derive vehicle-specific maintenance solutions for an arbitrary failure observed in the field.

Abstract

2 Contents

3 Introduction

The ongoing digitalization of the automotive sector changes key functionalities of vehicles and paves the way for several business opportunities to improve customer experience. Cars get autonomous and increasingly connected, enhancing many functionalities and making driving safer and more efficient in many ways. Therefore, digitalization and digital services are key to BMW's corporate strategy [1]. Car to car communication as well as car to manufacturer backend connection are part of BMW's technological progress. Here, Predictive Maintenance (PM) is a cutting-edge approach, where individual vehicle data is monitored and deterioration as well as failures can be diagnosed and even predicted [2]. The goal is to provide mobility services without disruptions or malfunctions. Next to obvious customer satisfaction, a minimization of vehicle downtime opens several business opportunities like warranty cost savings.

The evolution of maintenance in the car industry is depicted in Figure 1. The most common case for non-wearing components is a reactive maintenance, where a device unexpectedly breaks down during operation and is repaired or replaced shortly after. A preventive approach tries to regularly change a device when getting close to the expected lifetime. This lifespan is usually found by experience, engineering considerations or experiments. A condition-based maintenance is already a sensor data-driven approach, indicating equipment deterioration. An example for this are vehicle brakes, where a sensor activates a warning signal when a specific abrasion of brake lining is reached. As final step, PM uses vehicle sensor data to compute component-wise risks of failure. This can be done by e.g. monitoring physical influence factors and predicting the time to failure with analytical methods. PM is preferable to common maintenance procedures since a component repair or exchange prior to break down is by all means advantageous to a break down during operation where a dealer workshop might not be at hand [2].

Comprehensive research is now underway to enable such PM services in the automotive industry. This work discusses new predictive analytics approaches that can improve state-of-the-art techniques to analyze vehicle data and to predict component-wise risks of failure. Therefore, so called survival models are introduced, which are a promising alternative to currently used machine learning methods. In addition, an automated feature selection pipeline is implemented to improve current manual processes and to speed up the process of generating a machine learning solution for a given vehicle component. The introduced algorithms are extensively tested and evaluated on real customer fleet data. It will be shown, that the newly derived solutions show certain advantages in predictive ability, learning the future failure behavior as well as scalability for new failures, i.e. automatically find time-dependent correlations between the failure and car data.

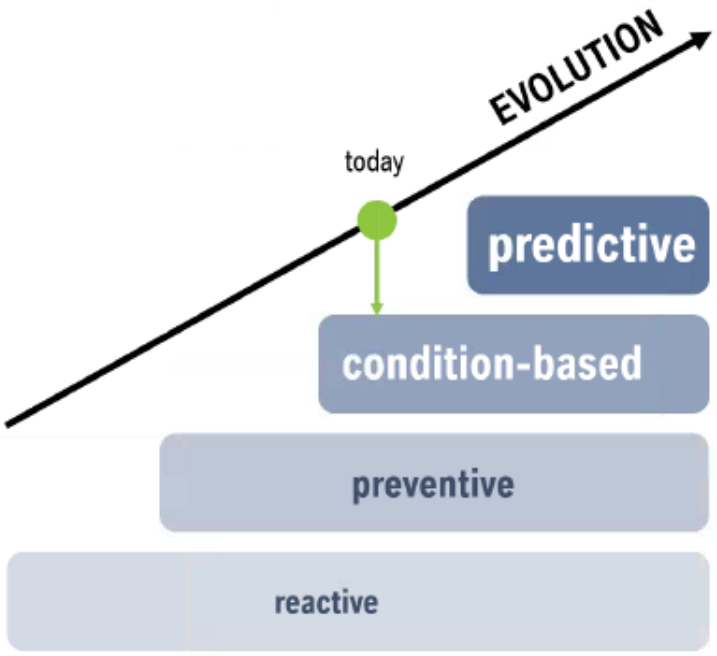

Figure 1: General evolution of maintenance [2].

This work is mainly structured in a background section, a methodology section and a results section. The background section consists of a motivation chapter, a related work chapter depicting existing literature on common PM, a chapter on the available vehicle data as well as a chapter on employed machine learning algorithms. Chapters 8 – 11 contain the work's methodology and results, split into a model training and evaluation part as well as a feature selection part.

4 Motivation

This work is based on real-life customer vehicle fleet PM which needs to be clearly differentiated from common PM. This chapter first describes state-of-the-art techniques of PM followed by a short look at the general process of implementing a PM solution in the case of vehicles. It is then shown, that facing vehicle data rises the need for specific machine learning solutions. These are then shortly described and the methodical necessity for new approaches is depicted. As a conclusion, an outlook on future PM is given.

PM is often employed for manufacturing and production plants [2]. Here, permanent monitoring of machine sensor data is used to detect system failures. As maintenance costs are high, effective PM cuts repair costs and reduces unscheduled downtimes in comparison to simple reactive or preventive maintenance schemes. The monitoring of such stationary machines often relies on a high frequency sensor system, specifically designed for PM methods. Examples for such techniques are e.g. vibration monitoring, thermography or ultrasonics. Data generated by these methods can be analyzed in real time and failures immediately diagnosed or predicted. However, these methods fail when it comes to a comprehensive evaluation of worldwide customer vehicles. Due to their sheer multitude and mobility, vehicles cannot be monitored in high frequency so far. This poses new challenges to the field of PM: Firstly, vehicles are not surveilled in high frequency, but can only be "diagnosed" when coming to a dealer workshop. Here, stored data is obtained from the vehicle and sent to a backend database. As a consequence, vehicle data is only available at discrete times and not as complete time series. As a matter of fact, this vehicle data is created by sensors surveilling key functionalities in the vehicle [3]. Initial purpose of this data is a further processing in the vehicle's electronic control units (ECUs) such as the dynamic stability control. Vehicle sensors are in turn not specifically designed for subsequent data mining. Storage limitations in the vehicle itself impose that ECU data is stored in an aggregated form, for instance as characteristic diagrams, and not as time series. In addition, sensors and process units differ for vehicle series, making available data highly heterogeneous over the complete customer vehicle fleet. As a result, the vast amount of vehicles and their global use make it impossible to implement a homogeneous, high frequency sensor infrastructure on all components. Furthermore, it is not guaranteed that the failure itself afterwards appears in backend data, making it hard to distinguish between a healthy and broken car or component, respectively.

Only recently, PM for single vehicle components which break down in the customer fleet is of high interest. The failures can either be caused by an unexpected shortcoming after start of production or a known wear process such as brake degeneration. In the former case, the failure and its physical background need to be manually analyzed as the components are often ex ante not considered to be deteriorating at all. PM analysis in turn relies on monitored vehicle data describing directly or indirectly how the component deteriorates under certain conditions. For instance, thermodynamic or mechanical strain can cause the failure. Yet, the root cause

might not be a priori known increasing the complexity of the subsequent predictive analytics modelling process.

On the basis of these constraints, analytical solutions other than common PM methods look promising: Modelling the failure is based on specific machine learning techniques that are able to process complex historical data on the given failure. Real customer vehicle data is used to learn and then infer a risk of component break down. Data preprocessing is performed by narrowing down the available data to the affected cars and components. Data sources for training a model can in principle be all sorts of data collected and stored in the vehicle. However, as shown later, data on one specific component failure is usually sparse and a main challenge is to combine different data sources from the field to get the overall picture. Before training the machine learning algorithm, manual feature engineering is often necessary to get characteristic parameters suited for prediction. Models are then evaluated and tested on similar data using resampling or cross-validations. Once a model is built, a business case can determine the further implementation. The predictive solution can be employed in various ways: Either one scores vehicles according to the risk of failure based on the latest available data in the backend or the system repeatedly evaluates the risk of failure onboard, using the vehicle's information directly. The better the model performs the more failures can be avoided by selectively repairing specifically the vehicles at high risk. This is e.g. done by direct risk assessment of a potentially affected vehicle coming to a dealer workshop or a scoring performed in the backend on all vehicles in the fleet based on the latest data available.

This work is mainly concerned with the process of building optimal machine learning models to "learn" the characteristic influences on the failure and to predict it. Such data-driven models are capable of analyzing the vast amount of data created and stored in vehicles. However, there are special issues in using vehicle data which is a main challenge for implementing PM in the car industry. Common PM techniques based on high frequency data are obviously not suited for the problem at hand. Some computational intelligence methods are however capable of scoring components based on the data available. Still, not all data-driven machine learning techniques are well applicable. For legal reasons, it is mandatory for a producer like BMW to justify a PM procedure as customer vehicles are discriminated based on a computed risk score [4]. Therefore, black box models such as artificial neural networks [5] are not used as they do not allow an a posteriori full understanding of the computed risk score. Expert systems are also not suited, as failures commonly appear after start of production and are ex ante not considered critical. The process of experimentally designing an expert system is too complex or the relevant physical parameters are not available by the onboard sensors to correctly apply such an expert system. Requirements for successful models are in turn adaptivity, scalability and a certain simplicity for physical understanding and legal justification. The current methods used at BMW are binary classification models such as general linear models or tree-based classification. These are used to classify a given vehicle or component, described by data from the latest diagnosis, into high or low risk.

Although data is only available for discrete times and not as time series, the timely dependence of deterioration and failure for every component seen in the data needs to be learned by a model. The failure frequency over vehicle age, e.g. a Weibull distribution [6], is crucial for proper predictions of failure rates, too. Yet, these aspects are completely neglected by binary classification models as they do not have an explicit time reference which can thus not be integrated in modelling. Due to the binary design of these models, a risk assessment is only possible for a snapshot in time. The first objective of this work is to overcome these disadvantages with survival modelling techniques [7]. Survival models can inherently display time dependence of component degeneration and thus infer survival probabilities over time. Methods are e.g. Cox Regressions or boosted trees and will be assessed in detail.

Furthermore, maintenance services need to be quickly adjusted to unexpected failures in the customer vehicle fleet. As a result, highly adaptable and scalable methods are necessary to quickly react to unforeseen failures or to enhance individual customer information about the versatile vehicle status. Examples are an intelligent warning of brake degeneration or an early air condition repair prior to failure to avoid customer dissatisfaction. Generating a machine learning system for PM is yet complex and still a mostly manual process. Hence, the scope of this work is to improve and automate the current data-driven toolchains, such as an automated features selection and stepwise model training pipeline.

5 Related work

The concept of PM has been applied to industry applications in multiple ways. Mobley [2] gives a comprehensive review about PM applications and techniques. These are shown to have significant positive influences on overall maintenance costs and machine downtimes. However, these methods exclusively focus on PM for mostly stationary machines that can be continuously monitored or in high frequency such as vibration monitoring. Thus, initial PM were a priori defined systems often supported by expert knowledge and not a posteriori statistical data-driven approaches. Example of such PM solutions are [8],[9] and [10]. Orhan et al. [8] describe vibration monitoring of rolling element bearings. This is done by a real-time expert system based on outlier detections. Garcia et al. [9] present a health monitoring for wind turbine gearboxes by expert systems defining critical values for the data. Jones and White [10] present real-time monitoring of heating and air conditioning systems by temperature sensors. Another example is the degradation model of Kaiser and Gebraeel [11] employing high frequent sensors.

Si et al. [12] give an overview of prognostic data-driven methods for an estimation of a remaining useful lifetime for assets in general, focusing on direct and indirect condition monitoring systems. Possibilities to analyze the former are e.g. Wiener and Gamma processes, for the latter stochastic models like survival model, e.g. hazard models, and hidden Markov models are described. An example is the condition-based inspection policy with environmental covariates by Zhao et al. [13]. Sikorska et al. [14] classify data-driven PM approaches into knowledge based models (fixed rules, fuzzy logic), stochastic and statistical life expectancy models (survival models), artificial neural networks and purely physical models. These are comprehensively compared giving a good overview about the respective modelling advantages. Hybrid approaches are discussed by Liao and Köttig [15]. Choudhary et al. [16] analyze data mining in manufacturing, pointing out that fault prediction in the automotive domain is challenging due to the inaccessibility of customer vehicles and the resulting non-continuous data structure.

Survival models are a widely used machine learning technique for the analysis of failures of any machine component. They are especially suited for modelling time to an event, e.g. time to death of a patient in medical science. An example is the comparison of logistic regression and survival model of Liao et al. [17] to determine the risk of failure of a bearing test rig. They however describe a stationary monitoring not comparable to real-life monitoring of customer vehicle bearings. Survival models were formally introduced by Cox [18]. Further research on this was e.g. performed by Friedman [19], [20], [21] and Ishwaran et al. [22]

In summary, nowadays vehicle on-board data is not tailored to the needs of extensive data mining approaches, especially not for any models that require high frequent or continuous monitoring. In addition, real-world component failures are often ex ante not classified as

deteriorating systems and therefore not directly surveilled by sensors unlike e.g. brakes specifically equipped for condition-based maintenance. Few literature on PM for vehicle generated data is hence published so far. One approach was described by Prytz et al. [23] predicting the need for truck compressor repairs using explicitly logged vehicle data. The component is not directly monitored but only indirect by the available sensor systems in the trucks. They employed supervised learning techniques, i.e. classification models, to assess whether the components have a failure before the next service event. The machine learning methods are trained with historical data, i.e. historical failures of the component. Voronov et al. [24] present a similar study on the modelling of lead batteries with tree-based survival models.

The approach followed in this work is a purely data-driven machine learning methodology able to cope with an arbitrary failure in the customer vehicle fleet. The data background for these statistical models are explained in the next chapter.

6 Data background

All PM solutions by machine learning algorithms are based on historical data which is used for training and evaluating the models. This chapter gives an overview of the relevant customer vehicle data used in this work. The first section describes the source of data, the second describes how the data finds its way to backend servers. The third chapter discusses the data in regards of usability for PM and gives a short outlook on how data sources will possibly look like in the future.

6.1 Data generation in the vehicle

ECUs control the functionality of modern cars [3]. The units process specific information from various sensor systems from different components in the vehicle tailored to regulate a specific vehicle system. For instance, the electronic motor control unit receives information on e.g. engine torque, engine speed, wheel speed and throttle angle to control automatic gear changes. ECU input data, such as sensor data, and ECU output data, e.g. gear change commands, are partially stored in the ECU as well. Due to the heterogeneous ECU development, different ECUs store different signals encoded in different ways. A single ECU's complexity and storage capacity can also vary, often increasing over repeating development cycles.

Initial motivation to analyze ECU data was to enhance vehicle failure diagnosis at dealer workshops [25]. I.e. recorded data is ex ante not designed for extensive data mining requirements and specifically not for PM purposes. The data aggregation and heterogeneous ECU data are in turn a key difficulty in implementing PM solutions. No short term changes on this can be induced due to slow vehicle development cycles focused on electrical safety. Yet, the available data has a huge potential for predictive purposes as the predictive ability of machine learning methods on ECU data described in the following proves.

6.2 Data storage in the backend

Vehicles that come to dealer workshops are usually diagnosed, meaning the contained information in ECUs are read. This information is mainly used to diagnose failures and identify broken components in return. In addition, the information is passed to the producer backend

and stored in databases. Such a data package from a single dealer workshop in the backend is denoted a diagnosis in the following. Diagnosis data is usually highly heterogeneous for different vehicle types containing different ECUs and different versions of them, each storing different information.

Repeated dealer workshop visits then generate multiple diagnoses at time-discrete points in the past. These diagnoses are not necessarily in regular intervals, however new vehicles often visit dealer workshops on a regular basis for inspection and warranty reasons. Due to this sparsity of vehicle diagnoses and therefore limited component information, PM solutions are still constrained to a certain accuracy by the uncertainty about the intermediate vehicle "health" status. A very important requirement is therefore, that component failures need a long term deterioration process to be displayed in the data. Short term failures would need a significantly higher frequency of diagnoses or any other kind of data retrieval to be predictively modelled. Higher frequent or even continuous monitoring is yet complex to achieve as vehicles are moving objects, often with too low network coverage to retrieve data from for instance teleservice. This work deals with state-of-the-art low frequent diagnosis data. The developed machine learning methods are highly adapted to the diagnosis data structure. PM solutions especially have to cope with the irregularity and varying number of diagnoses per vehicle.

In addition to diagnosis data, information needs to be obtained whether a specific vehicle component previously had a failure when diagnosing the vehicle at a dealer workshop. Assuming that a failure causes a subsequent dealer workshop visit, times of failure and diagnosis are closely related. Failure information can hence be retrieved by two possible data sources. The first is the diagnosis data itself: ECUs sometimes store diagnostic trouble codes (DTCs) when certain failures are recognized. A DTC is e.g. stored when brake deterioration has reached a critical limit. Furthermore, data is available on warranty repairs of vehicle components from the dealer workshops. These are also stored in the backend and can be used for labelling. The advantage of the former data source is the exact mileage the DTC occurred, the latter data source only has the information about the mileage of the diagnosis itself. This makes labelling more imprecise but still can be coped with as shown by the results of this work. Neither of the data sources is totally reliable as e.g. a warranty case might be wrongly classified making labelling inaccurate.

6.3 Quality and contained information

The quality of data is mainly influenced by its initial diagnostic purpose, i.e. the root cause analysis of previous failures of a vehicle coming to a dealer workshop. The workshop staff is supported by expert-based off-board diagnostic systems provided by the vehicle producer. These systems use the ECU data to find noticeable deviations formalized by human expert knowledge. The necessary ECU data to be stored as well as rule-based failure diagnosis is usually defined by different development departments. Thus, ECU data is highly heterogeneous, e.g. the same physical quantity is often stored in different units. Another problem often is the storage management of different ECUs. Some ECUs reset measured values after a diagnosis at a dealer workshop. These have to be manually aggregated in the backend to generate valuable information for PM solutions. ECU data storage limitations are also influencing how specific sensor data is aggregated. Engine torque and speed are for instance aggregated from a high frequent time series monitored by onboard sensors to a relatively coarse characteristic diagram eventually stored in the ECU.

Knowledge about the ECU data management and exact physical units of measured values are hence mandatory for a PM process using this information. A high level of coordination with the ECU development departments is in turn needed for a complete understanding of every single ECU value which is integrated into subsequent analysis. This process together with a reasonable data preprocessing is crucial for the success of machine learning methods.

7 Machine learning background

This chapter gives a short overview over the state-of-the-art machine learning techniques for PM. Survival models in the context of vehicle fleet PM are then introduced and methods to evaluate and compare different models are described. This chapter aims for a better understanding of the methodology and is not a comprehensive machine learning introduction. The reader is advised to [7], [26] and [5] for further details.

7.1 Overview

Various techniques are used in the context of PM to analyze and predict a given failure. Yet, the customer vehicle fleet is not monitored in high frequency as it is in the case of PM for stationary machines. Thus, time discrete information with aggregated vehicle sensor data is a main challenge in the context of finding modelling approaches. Main categories of modelling approaches are in order of increasing model complexity [14]:

- Knowledge-based models: Expert systems or fuzzy logic, based on previous experiments and observed situations.
- Life-expectancy models: Statistical and stochastic models where known operating conditions determine the expected risk of failure
- Artificial neural networks: mathematical representation of the component wear derived from observed data.
- Physical models: mathematical representation of the physical behavior of the wear process.

Here, only data-driven models are relevant, otherwise costly a posteriori experiments would be needed and no overall expert knowledge about component wear is available. Hence, knowledge based-models and physical models are not suitable. These techniques are in addition hardly scalable to the multitude of potential components to analyze.

Due to legal reasons, black box models are not used in this work [4]. Artificial neural networks are for instance highly capable of representing complex functions, although a direct input-output relation is complex or hardly be obtained [5]. This is crucial for a justification of selective vehicle maintenance, as legal issues demand a clear understanding of vehicle-wise discrimination according to a computed risk of failure.

The multitude of life-expectancy models provides approaches for various sorts of deterioration models in PM. One class are stochastic approaches, e.g. Weibull distributions, Bayesian

networks like Markov models or Kalman Filters. Kalman Filters might be a good solution to describe the component behavior over time between two dealer workshop visits [14]. This can be matter of future research in this area. Main problem of a Weibull prognosis is to integrate parametric influences to the wear prediction, i.e. the risk of failure.

The complexity of data rises the need for special statistic machine learning approaches enabling a parametric way of modelling. One way can be to use binary classifiers, for example a logistic regression [26]. Here, two classes or labels are defined, indicating healthy and broken vehicles. Training a classifier on this data results in a pseudo probability of failure for instances, i.e. vehicles, as snapshot of the underlying real risk over time dependence. The approach of this work are Proportional hazard models: They are strongly opposed to common classification or regression models and have significant advantages, as explained in the following section.

7.2 Survival models

Survival analysis is used when a time to event, often called survival time, is of interest[1]. Independent covariates can be integrated in the model similar to linear or logistic regression. Survival models are frequently used in medical applications and also industrial time to failure analysis.

7.2.1 Formulation of survival models

The response variable of a model is the time until the event (in this case failure) T, expressed by the survival function, i.e. the probability of event after time t. The survival function $S(t)$ gives the probability that the subject "survives" at least until time t and is given by:

$$S(t) = Prob\{T > t\} \tag{1}$$

The negative log of this function is defined as the cumulative hazard function $\Lambda(t)$ and denotes the accumulated risk up until time t. The hazard function $\lambda(t)$ is the probability that the event will occur at a certain time increment:

$$\Lambda(t) = -\log S(t) \tag{2}$$

$$\lambda(t) = \lim_{u \to 0} \frac{Prob\{t < T \leq t + u \mid T > t\}}{u}$$

The hazard function often resembles a bathtub-shaped curve, with initial wear-in failures, a low random failure rate and increasing wear-out failures [14]. Simple survival models assume, that all samples, i.e. vehicles or components, have the same failure distribution and no features are allowed. One popular survival distribution is the parametric Weibull distribution, given by:

$$\lambda(t) = \alpha\gamma t^{\gamma-1}, \tag{3}$$

$$S(t) = \exp(-\alpha t^{\gamma}),$$

with design parameters α and γ.

[1] Section 7.2 closely follows the explanations in [7].

In contrast to simple models, survival regression models allow for further predictor variables, i.e. features X, so that:

$$\lambda(t \mid X) = \lambda(t) \exp(X \cdot \beta), \qquad (4)$$
$$S(t \mid X) = S(t)^{\exp(X \cdot \beta)}$$

where $X \cdot \beta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$. This is called the proportional hazard model (PHM). As such, PHM assumes a multiplicative effect of model features to the overall hazard and not additively like linear regression. The effect of predictors is the same over all t and therefore no t times predictor interaction is initially given. An advantage is, that PHM combines covariate influence with a baseline hazard function $\lambda(t)$ in contrast to other regression techniques. The predictor coefficients β can, in terms of a supervised learning, be trained using e.g. residual sum of squares or maximum likelihood estimation on historical data. The input variable for training a parametric survival model is survival time or time to event, i.e. failure. As a matter of fact, this survival time can be incomplete, as a component might not break down at all. This is called right-censored data since the survival time is at least the monitored time period. Otherwise, survival time is the time between start of observation and failure. This principle is shown in Figure 2, depicting three exemplary vehicles.

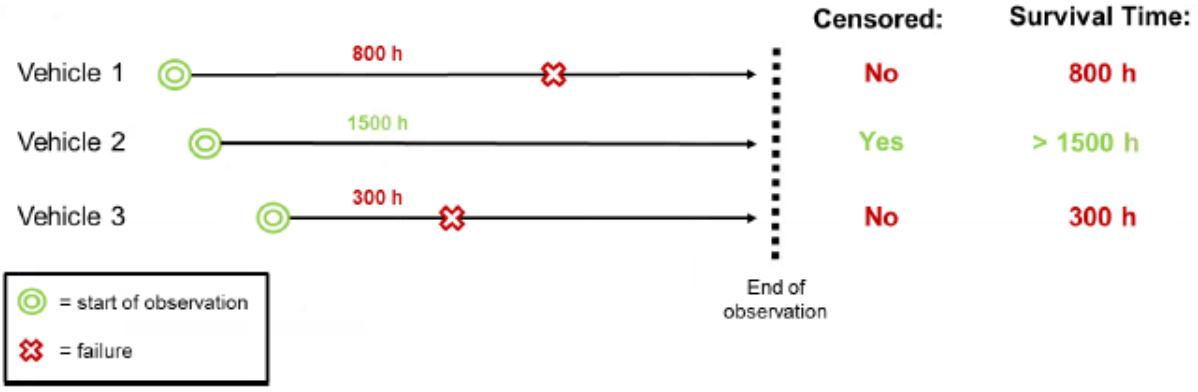

Figure 2: Schematic principle of survival times, in hours of operation, and right censored data.

7.2.2 Cox Proportional Hazard Model

Cox PHM makes no assumptions about the hazard function $\lambda(t)$ but ignores the shape of it completely. Thus, rank ordering of survival times is used as input for training, the exact survival time information is neglected. As a result, Cox regression is less affected by outliers in the data. The model output, i.e. the fitted $X \cdot \beta$, is hence interpreted as risk score, with higher risk for higher $X \cdot \beta$. Due to the simplification in hazard function, training a Cox model does not yield in a direct estimate of $S(t \mid X)$. The output only consists of the $X \cdot \beta$ component, i.e. the regression part. Yet, $S(t \mid X)$ is often not of primary interest and an estimation of feature influence $X \cdot \beta$ is of more importance. Secondary estimates of $S(t \mid X)$ can however be derived as by Kalbfleisch and Prentice [27], [28] or Breslow [29]. With these methods, the baseline hazard function, i.e. the failure frequency distribution, is estimated from the data used in training. This can be advantageous, as the real failure distribution seldom resembles a perfect, a priori known distribution like e.g. Weibull. In addition, results from a Cox fit are shown to be in good agreement with Weibull fits [7]. It is beyond the scope of this work to give a detailed

look into the mathematics of β estimation using maximum likelihood. For a detailed derivation see [7].

Like all machine learning methods, Cox Regression tends to overfit to given data when not carefully trained. Overfitting for instance occurs when using too much predictor variables or data is not sufficiently available for training. For the theoretical background of overfitting and techniques to detect it, see [26]. Regularization methods are commonly used to prevent overfitting in regression models, like the Cox PHM. Regularization adds a shrinkage penalty term to the likelihood function to be minimized. The so called elastic net penalty P, a generalization of shrinkage penalty, is defined by [30]:

$$P := \lambda \left((1-\alpha)\, 0.5\, \|\beta\|_{l_2}^2 + \alpha\, \|\beta\|_{l1}\right) = \lambda \sum_{j=1}^{p} \left[0.5\,(1-\alpha)\,\beta_j^2 + \alpha\,|\beta_j|\right], \tag{5}$$

where β is the coefficient vector of dimension p, l_1 and l_2 are the regularization norms and α is an exogenic tuning parameter. For $\alpha = 1$, the penalty results in a so called *lasso* regularization, for $\alpha = 0$ in a *ridge* regularization. The parameter λ determines how strong the regularization effects the optimization of parameters and is usually determined via cross-validation. Due to the l_2-norm, ridge regression tends to shrink coefficients of correlated features towards each other and less towards zero than the l_1-norm. Hence lasso regression tends to select one and neglect the rest of the correlated features by shrinking the respective coefficients to zero resulting in only few non-zero coefficients [30]. This resembles a kind of feature selection, resulting in non-zero coefficients only for the most influential and uncorrelated features in the model.

7.2.3 Boosting

Boosting is a general technique in machine learning to improve the accuracy of a model, e.g. Cox regression [31], [32]. It is based on the assumption, that fitting several weak, i.e. simple learners, is more accurate than one single, complex learner. Boosting a given algorithm is an iterative and sequential fitting of simple learners on randomly permuted subsets of samples from the complete data set [31]. This randomness usually reduces variance in the final model [19]. Each new leaner is fitted to the residual from the previous one to optimize the overall performance. Exogenic tuning of boosting can be accomplished via the number of iterations and shrinkage: The shrinkage parameter slows the process of adding new leaners down, i.e. slower residual minimization, by multiplying the additive effect of a new leaner with a small number. As a result, slow learners often tend to generalize better than fast learners. The boosting algorithm itself, most commonly *AdaBoost* [33] is depicted in simplified form [34] as follows. $L\big(y,\ F(X)\big)$ denotes a generic loss function for a statistical model F with features X.

1. Initialize the estimate, e.g. $\hat{F}(X) = 0$
2. Compute the negative gradient vector, $u = -\frac{\partial L(y,F(X))}{\partial F(X)}\Big|_{F(X)=\hat{F}(X)}$
3. Compute the possible updates:
   Fit the base learner to the negative gradient vector, $\hat{h}(u, X_j)$
   Penalize the value with learning rate $\hat{f}(X) = \upsilon\, \hat{h}(u, X_j)$
4. Update the estimate, $\hat{F}(X) = \hat{F}(X) + \hat{f}(X)$

Adaboost has been proven to yield a steepest gradient descent in function space [35]. Boosting has advantageous behavior when it comes to overfitting, as the process can be stopped at any iteration step when there are indications that generalization phase is over. This is often evaluated by cross-validation [35]. *R* toolboxes for boosting Cox Regression are *Coxboost* [36], [37] and *mboost* [35], [38]. Both two packages initialize the parameter vector β to zero and stepwise update these parameters given the residual, one parameter update per iteration. However, they differ in the way of doing this, i.e. by likelihood-based and model-based (gradient-based) boosting, respectively. The former only allows for a single parameter update per iteration step in addition. This leads to updates for only a few parameters in the case of multiple highly correlated parameters. Also, penalty terms apply differently. For further detail, see [36], [35] and [34].

7.2.4 Tree-based models

Tree-based survival models are a different approach for time to event modelling. Decision trees are a set of splitting rules and can be used for classification, regression and survival tasks [26], [22]. They are simple and easy to interpret and thus well applicable to PM of vehicles. The baseline algorithm is called recursive binary splitting: It sequentially splits the feature space according to the best possible split determined by a single parameter (see Figure 3). This is derived by the maximum reduction of residual least squares of predictor and split point combination [26]. Trees are highly capable of nonlinear representation of any given function. Trees are usually "grown" fully and then pruned, meaning the last splits are "cutted", i.e. neglected, to prevent overfitting. This cutting, i.e. selecting a subset of splits, is again done by e.g. cross-validation.

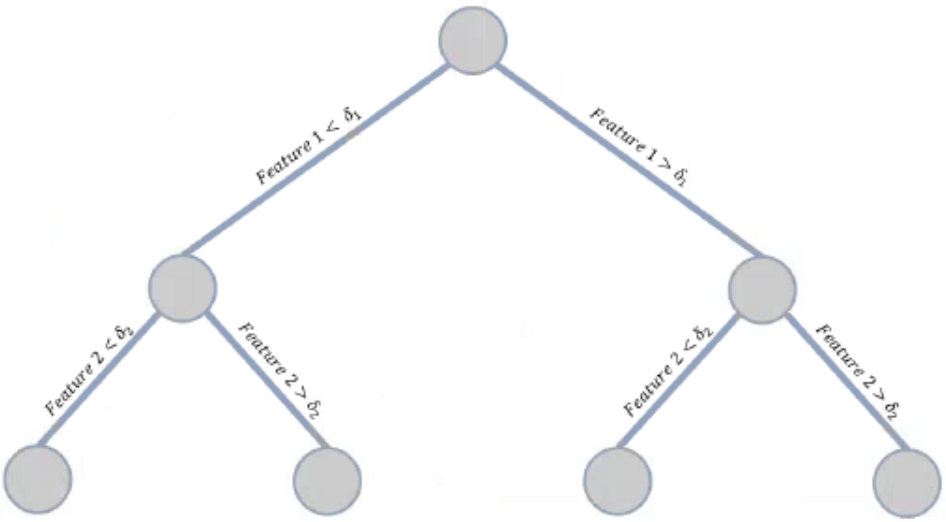

Figure 3: Exemplary decision tree with two splits.

In contrast to pruning, ensemble learning techniques rely on the power of multiple learners combined [39]. One example are boosted survival trees, which are, as described above, a set of sequentially trained simple survival trees, i.e. trees with only a small number of splits (depth). At each split, only a random subset of features is considered. The *gbm* package [40] provides a comprehensive implementation of boosted survival trees which can be optimized by selecting e.g. learning rate, number of trees and tree depth, i.e. the number of splits per tree [32]. The process of fitting a boosted tree also resembles a feature selection according to importance, since only parameters are integrated in the model that can be used for an optimal split and

residual reduction[2]. Exogenic tuning parameters in boosted tree-based survival models are e.g. iteration steps, tree depth and shrinkage [19]. Another ensemble tree-based model are random survival forests [22], [39]. In general, many trees are trained in parallel on bootstrapped training sets, i.e. random samples from the data. The individual predictions of all trees are finally averaged to a single output. In addition, only a random subsample of parameters is considered at each split of a tree. This ensures a decorrelated ensemble of survival trees, reducing the variance of the overall output.

7.3 Model performance evaluation

When applying a previously trained model to new vehicles and components without given label, a classification infers a numeric pseudo risk output from 0 to 1 while a survival model infers a risk score $X \cdot \beta$ or a real survival probability $S(t \mid X)$. This process is called scoring of a vehicle or component. Since both classification and survival model outputs can be used as risk score, both model types can be evaluated by similar techniques. This chapter aims to give an overview about common machine learning evaluation techniques specifically suited for PM. However, before introducing performance metrics, it is important to mention which meta-algorithms are used to slice the existing data into a training data set and a test set for performance evaluation.

7.3.1 Cross-validation

Cross-validation is a general resampling method, where multiple subsets are drawn from the data [7]. This can either be done completely random, random but with mutually disjoint sets adding up to the complete data set or by generating stratified sets. A model is usually fitted on all but one data set, and consequently tested on the remaining set given a performance metric. To reduce variance in the estimate of performance, all combinations of training and test splits are used and the performance metric is averaged. This is often referred to as k-fold cross-validation, with k denoting the number of created subsets. Most models can more or less arbitrarily overfit to given training data and thus have very small errors or residuals on the training data. The test data set, which is not used in training phase, is then employed to evaluate the true performance of the model. Some machine learning models can be stopped in their training, when performance on the test data sample stops reducing, e.g. neural networks or boosting methods [5].

In regards to model variance, random subset sampling is advantageous to a single train test split. This is because the random sampling produces independent samples, reducing the variance in model error estimates. When reducing training data for a later test, less instances are used for training as there are originally, so that bias is relatively high for this technique. This drawback can be coped with using a 5- or 10-fold cross-validation, somewhat empirically appropriate concerning the bias-variance trade-off [41].

Individual evaluation metrics exist for classification, regression and survival tasks. The next section derives the similar basics for classification and survival model evaluation.

[2] This will be discussed in further detail for feature selection in chapter 10.

7.3.2 Area under curve and concordance index

In binary classification, the inferred model output for a new instance is a value between 0 and 1. This can be interpreted as pseudo probability for being in class 0, i.e. component without failure, and class 1, i.e. component with failure. A threshold is set to assign a given instance to either class 0 or 1, e.g. assigning all instances scored over 0.5 to 1 and below to class 0. Such a classification on a test set can make two kinds of errors: It can incorrectly assign an instance to class 0, i.e. healthy, or to class 1, i.e. failure. The former cases are called false negatives (FN), the latter false positives (FP). Accordingly, true negatives (TN) and true positives exist (TP). These four types span a so called confusion matrix, which displays the trade-off of instance classification. The aim is to minimize wrong predictions and hence maximize the domain-specific combination of TP rate and TN rate for given exogenic threshold. Minimization of FP and FN fraction[3] can be displayed as function over threshold as shown in Figure 4.

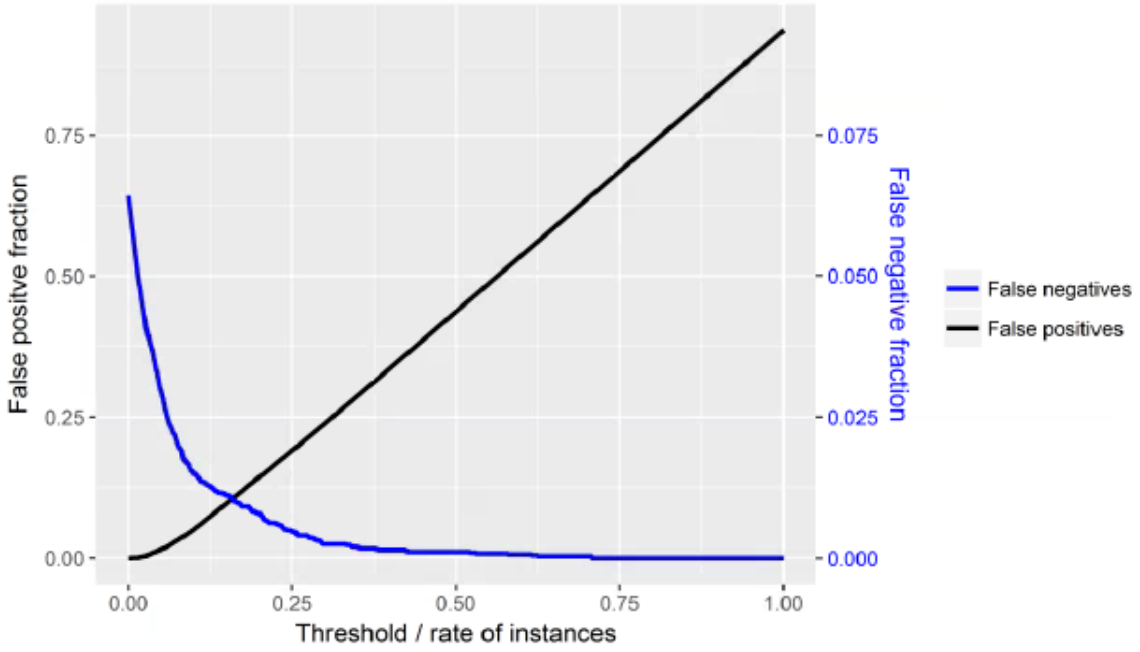

Figure 4: Exemplary trade-off between false negatives and false positives over threshold for a highly imbalanced data set.

The Receiver Operating Characteristic (ROC) is another convenient way to display the information about TP and FP. The curve results from all TP and FP rate combinations over all possible thresholds. Figure 5 displays the ROC curve from the same classification that was shown in Figure 4. The area under curve (*AUC*), denoting the area under the ROC curve, is a measure of performance of the classifier and often used as metric to evaluate and compare the performance of binary classifiers [42]. Its maximum value is 1 for a perfect classification, i.e. FP rate and FN rate equal to zero on the test set.

[3] Here, (TP-, FP-, FN-, TN-) fraction is denoted as percentage of all instances, (TP-, FP-, FN-, TN-) rate as the percentage of overall positives or negatives, respectively.

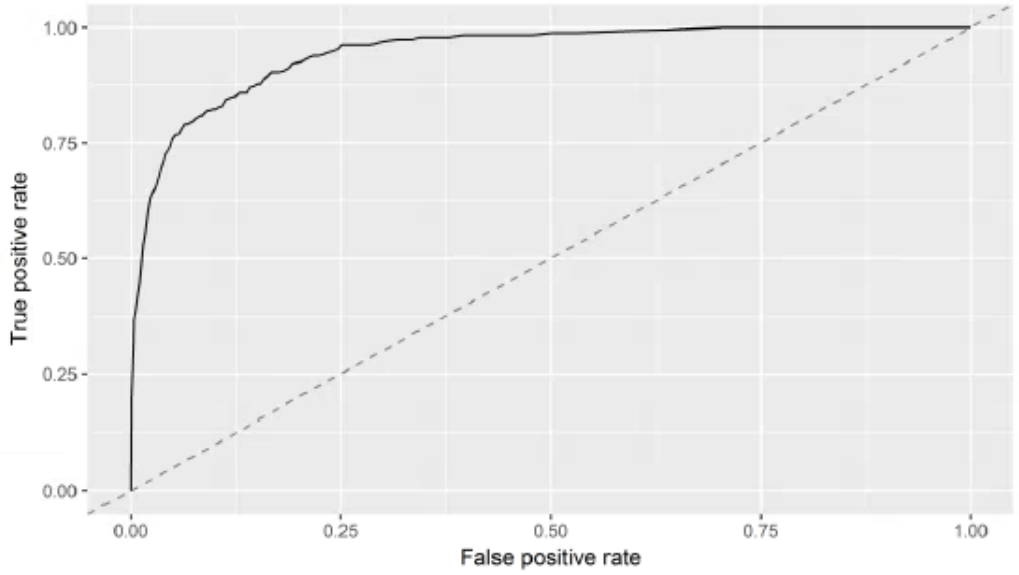

Figure 5: Exemplary Receiver Operating Characteristic. The area under this curve is the AUC.

To quantify performance of survival models, the output, i.e. risk score $X \cdot \beta$ or real survival probability $S(t \mid X)$, can be compared with the actual times to event, i.e. the true survival times. The rate of concordance (*c index*) is the rate of all possible pairs of instances, where higher predicted survival time corresponds to the instance with actual higher time to failure [7]. Censored data cannot be compared and left out for this metric since no exact survival time is given. The amount of censoring does not significantly affect model training and thus the *c index* is relatively unaltered by censoring data. It can be shown, that the *c index* is identical to the *AUC* in classification on uncensored data [7, p. 493].

To evaluate either classification or survival models, *AUC* and *c index* are usually used in combination with cross-validation. In addition, the analogy of both metrics allows for a certain comparison of classification and survival models in PM tasks. This will further be explained in section 8.3.

8 Survival models for Predictive Maintenance

Survival models are well suited to model time to failure in general. Due to the structure of data and performance requirements, specific adaptions are required to enable a predictive ability of survival models on time discrete vehicle data. This chapter describes how survival models were fitted to predict component failures in the customer vehicle fleet.

8.1 Predictive Maintenance process model

Commonly, two vehicle maintenance procedures exist for vehicles affected by a systematic and unforeseen failure. Repairing all by announcing a recall or waiting until the components break down, i.e. a preventive or reactive maintenance. The former is usually done when the failure is critical and one cannot afford to wait until the vehicles break down. Compared to both cases, PM has significant benefits by scoring and ranking vehicles according to risk of failure: A recall can be limited to the vehicles at high risk. If no recall is planned, customers of high risk vehicles can be informed in advance and a repair generates a higher level of customer satisfaction. Other business cases exist, where PM solutions monitor components like breaks in the vehicle and warn the customer of a deterioration in advance. It is the main scope of this work to provide such vehicle-wise failure probabilities using survival models. To do so, a model learns characteristic influences of provided features on known labels, i.e. whether a component had a failure or stayed "healthy", in supervised learning. To generate a PM service for vehicles, historical data on the component failure has to be analyzed in turn. The following process, depicted in Figure 6, is a generalization of the PM process.

As explained in chapter 6, all vehicle data read at dealer workshops is stored in a backend server. Given a component to analyze, i.e. a failure or deterioration that needs to be predicted, relevant data can be read from the server using *SQL* statements. As the failure is commonly affecting only a small subset of all vehicles, for instance a component in only one production series with one specific motor, it is important to restrict the data to this specific subset of vehicles. Thus, criteria to filter data are e.g. vehicle type, manufacturing date, existing ECUs, motor type or country of registration. The result is a data in matrix format, where rows are instances, i.e. vehicle diagnoses, and columns are the available ECU information, i.e. potential features, from this diagnosis. Additionally, labels for all instances have to be provided, determining if the particular component was broken or not at the time of the diagnosis. A main requirement for PM is, that the failure does not occur completely random but is caused by some sort of wear process. Furthermore, characteristic influences on this wear process need to be contained in the data for model training and thus need to be monitored and stored in the vehicle.

Preprocessing is an essential part of machine learning. Features are e.g. centered and scaled to a mean of zero and variance of one. For missing values different strategies exist. One can either leave out instances with missing values or fill the gap with an approximate value. This is called imputation and often done by inserting the mean over all existing values of the respective feature in other instances [7]. Furthermore, the amount of component failures is much less than the total amount of vehicles with this specific component. Thus, the class of instances with failure is much bigger than the class of instances without failure, i.e. censored instances. Such imbalanced data often requires special techniques like down- and upsampling or weighting to enable a good learning process (for details see section 9.2).

Once the data is preprocessed, machine learning models can be trained to infer risk scores or survival probabilities on new data. Usually, the data is first split into a training and test data set. For model training only the bigger training set is used. This set is often split further for a cross-validation to determine the hyperparameters of the learner, for instance finding the optimal exogenic parameters when performing a regularization. The trained models can then be evaluated and compared on the test set using e.g. an *AUC* metric. An additional evaluation methodology for PM is depicted in section 8.3.

Figure 6: Process model for vehicle PM.

A final model, i.e. the model with the best test performance, can be further used to predict failures of new vehicles or vehicles that are still "healthy" and might have a failure soon. An indication is the risk score, in case of a classification or Cox PHM, or the survival probability, in the case of a fully extended survival model. The latest data available for the vehicles at risk are commonly used to infer such risk model outputs. Survival models have significant advantages in extrapolating the survival probabilities to a future date, i.e. estimating the survival probability in the future, as shown later.

8.2 Preprocessing vehicle data for survival models

Due to the time-discrete observations of vehicles, i.e. diagnoses, two main issues have to be handled: How to define an instance and how to label the instances. The exact time of failure is seldom known and failure is usually only observable at the next diagnosis. In analogy to Figure 2, Figure 7 shows a realistic scenario of dealer workshop visits at specific moments in time, i.e. hours of vehicle operation. It is important to mention, that every customer has a different driving frequency and thus individual hours of operation within a given time period. Hence, it is advantageous to define the times to failure of survival models in e.g. hours of operation and not absolute time because a wear process is usually determined by a characteristic operation frequency. Customers that drive less in the same period of time are probable to have a smaller risk of failure than customers with higher operation frequency after an equal amount of time. In principle, any feature that correlates with the characteristic wear process is a possible time variable for survival models. Other possibilities are e.g. total mileage of each vehicle or number of motor starts as time variable instead of hours of operation.

Figure 7 shows three vehicles, each with three diagnoses. As a matter of fact, the number of diagnoses per vehicle highly deviates and somewhat correlates with the hours of operation as customers more or less regularly come to dealer workshops. Vehicles 1 and 3 had a failure, vehicle 2 stays "healthy" until today. An important assumption for modelling is that vehicles with failure will subsequently come to a dealer workshop for repair within a small period of time after failure. This is necessary, because the exact time of failure is in general not known but can only be approximated by the time the broken vehicle comes to a dealer workshop and diagnosis also indicates the previous failure. This induces a certain inaccuracy into modelling as times to failure are not exact but only approximate.

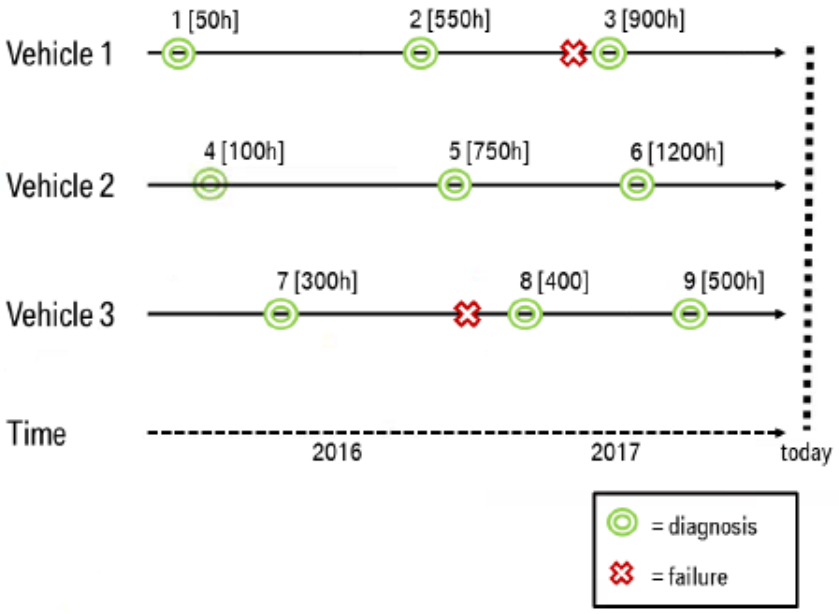

Figure 7: Realistic scenario for vehicles and failures over time [hours of operation].

The first approach here is to define each vehicle as instance. This is called vehicle-based analysis in the following. When using a classification, one takes the latest data on these vehicles as instances and labels them according to whether a failure was observed or not with 1 or 0, respectively. Here, the diagnoses 3, 6 and 8 are used with labels 1, 0, 1. When applying

a survival analysis, there are two labels, an indicator whether data is censored or not, in the following Cox label, and the actual time to failure, in the following Cox time. Similarly, one defines the diagnoses 3, 6 and 8 as instances, each with corresponding Cox time and label. This implies that the time to failure is seen from the viewpoint of the new vehicle, i.e. production or registration date. Diagnosis 8 is in both cases more accurate than 9 since the actual failure happened shortly before diagnosis 8. The corresponding hours of operation are then defined as Cox time. Instance 2 is censored as no failure appeared. Table 1 depicts the corresponding data set with three instances, i.e. vehicles, features as well as classification labels and survival analysis labels.

	Feat. 1	Feat. 2	...	Feat. n	Class. Label	Cox Time [h]	Cox Label
Inst. 1					1	900	1
Inst. 2					0	>1200	0
Inst. 3					1	400	1

Table 1: Vehicle-based data set with three instances and corresponding labels.

The second approach is to define each diagnosis as instance, denoted diagnosis-based analysis in the following. This results in basically nine possible instances depicted in Table 2. In case of a classification, every instance is labelled 0 or 1 whether it was before or after failure. Here, instance 9 is also labelled 1 even though the failure was probably repaired at workshop visit 8 and 9 is "healthy" again. This is to ensure proper learning since without repair diagnosis 9 would still be a failure and the features still indicate high risk. For a survival analysis, no "dead patients" are allowed and therefore diagnoses 3, 8 and 9 have to be left out since the corresponding Cox times are zero or smaller than zero mathematically. However, it can be advantageous to use zero Cox time instances, here instances 3 and 8, for training by artificially giving them a small Cox time greater zero, e.g. one. This indicates a "near to death" instance and positively influences the generalization in some cases as will be shown later. For the other instances, Cox times are defined as time difference in hours of operation between the diagnosis and the first failure, i.e. diagnoses with observed failure. Diagnoses 4-6 are censored as no failure was observed[4]. In case of vehicle 2, one can argue that the resulting Cox times should be defined as the time until today. This holds only if there was no unobserved failure in the meantime between diagnosis 6 and today. Assuming in this case the customer would have come to a dealer workshop for repair, the vehicle is still healthy today. To approximate the hours of operation of vehicle 2 today, one can for instance linearly extrapolate from diagnosis 6 to today. The censored Cox times of diagnoses 4-6 can then be adjusted, i.e. the Cox time of diagnosis 6 is greater than zero. This is done in the following for all censored instances, i.e. diagnoses, to assure more accurate input data. A diagnosis-based analysis implies survival time starting from each diagnosis and not starting with the completely new vehicle.

[4] This does not automatically imply, that there was no failure in reality. It is simply the best guess given the observed data, i.e. vehicle DTC and dealer workshop warranty repair data.

	Feat. 1	Feat. 2	...	Feat. n	Class. Label	Cox Time [h]	Cox Label
Inst. 1					0	850	1
Inst. 2					0	350	1
Inst. 3					1	-	-
Inst. 4					0	>1100	0
Inst. 5					0	>450	0
Inst. 6					0	>0	0
Inst. 7					0	100	1
Inst. 8					1	-	-
Inst. 9					1	-	-

Table 2: Diagnosis-based data set with three instances and corresponding labels.

In this work, diagnosis-based analysis is mainly followed as it provides more information for training the learner. In addition, the Cox time in vehicle-based analysis always refers to the time from vehicle registration date to failure and not to time from diagnosis to failure as in diagnosis-based analysis. Thus, diagnosis-based analysis provides a better understanding of future survival probabilities, as it gives survival probabilities in the future starting at the time of the instance itself and not with vehicle registration which might be years back in the past[5]. Later results show, that a diagnosis-based analysis can also be seen as a valid upsampling of a vehicle-based analysis.

The generated instances with features, Cox label and Cox time can then be used to train a survival model, e.g. Cox regression and boosted survival trees. When applying a learning algorithm, further downsampling of class 0, i.e. Cox label 0, or upsampling of class 1, i.e. Cox label 1, have to be considered since the former class is usually highly overrepresented.

8.3 Evaluation strategy for Predictive Maintenance

The performance of a previously trained model needs to be evaluated. Whether the model has learned to separate vehicles with failure from "healthy" vehicles based on the risk score output on the historical data can be validated by applying e.g. cross-validation and a *c index* performance metric. This however does not yet ensure a predictive ability of the model. To

[5] When assessing future survival probabilities of today's "healthy" vehicle fleet one wants to include the information that all vehicles are still "healthy" today, i.e. the survival probabilities starting with 100% for the time of the "healthy" diagnosis. This is only ensured for a diagnosis-based analysis.

demonstrate this, one can image a pure failure-diagnostic feature in a classification model, i.e. a feature which deviates from normal only when a failure already appeared. Then, the model can perfectly learn to distinguish between the two classes by learning that the failure appeared when the diagnostic feature diverges. This is however a pure failure diagnosis and not a prediction. Given a vehicle with non-diverging diagnostic feature, the model is not able to infer a correct risk score. As explained above, the latest diagnosis with potential failure can be excluded from the instances when using a survival model which inherently prevents the use of pure diagnostic features as survival models do not even "see" the diverging feature after failure in training. However, a performance metric is introduced to measure the real predictive ability of either a classification or survival model specifically for PM purposes. The following describes a performance metric for both classification and Cox PHM models by comparing the model risk score output with real failures. This does not yet consider any survival probability extension of the Cox model but is only based on any risk rank ordering such as the pseudo risk scores of $X \cdot \beta$. The described evaluating procedure was used to validate all used models in this work.

Optimally, the predictive performance of a trained model is evaluated by comparing the inferred model output, i.e. risk score, with real future failures. This is however not always possible as one has to wait a reasonable time to get data on new failures. Instead, the data is split by setting an artificial split date in the past. The model is then trained and tested using a *c index* metric with all data up to the split date. The train-test split is a single random 70 to 30 data split. It is here especially important to generate two disjoint subsets in regards to the vehicle identification number (VIN) for training and testing. As one vehicle, i.e. one VIN, corresponds to multiple diagnoses, a complete random split of diagnoses is not vehicle-wise disjoint. Thus the splits were performed by randomly splitting the VINs in a 70 to 30 ratio and then assigning the corresponding diagnoses to training and test data set, respectively. Only the training set is further used for any model hyperparameter optimizations using cross-validation on this set. Afterwards, the model output can be compared to the real failures after split date. This can either be all failures from split date until today or failures until a cut-off date, i.e. split date plus a predefined time period. The resulting data for a prediction versus actual failure comparison is in the following called validation data set. Figure 8 depicts this procedure schematically. Since a vehicle is not further analyzed after a failure in training, the validation set only consists of healthy vehicles at split date[6]. The validation set is designed to resemble a realistic scoring scenario, as only healthy vehicles are of interest when applying a model to calculate vehicle-wise risks of failure. Only one instance per vehicle, i.e. the latest before split date no matter whether it is initially in training or test set, is in turn contained in the validation data. The diagnoses after split date only influence the labelling: If there is a diagnosis with failure after split date, the respective instance before split date gets a validation label 1 and 0 otherwise. In the given example of Figure 8, the third diagnosis would be part of the validation set with validation label 1 as a failure appeared in the validation time period. A disadvantage of this method is that the information of diagnoses after split date is not used for training except for the labeling of the instances before split date. It also assumes that a vehicle with failure after split date also comes to a dealer workshop. Otherwise, the failure is not monitored and the vehicle wrongly gets a validation label 0.

[6] This is a constraint only for training phase, risk scores can subsequently be inferred for any vehicle.

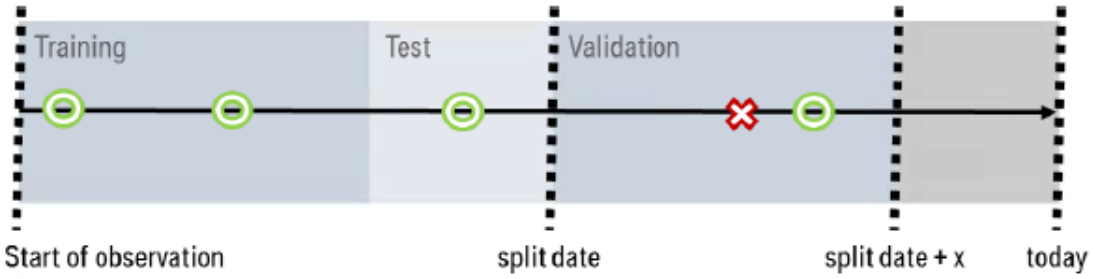

Figure 8: Diagnoses of a vehicle over time with schematic train, test and validation set.

The latest diagnosis before split date is then used to infer a risk score by the model to be evaluated. Subsequently, all instances can be sorted by decreasing risk score. This resembles a real-life procedure of ranking vehicles according to their risk of failure. For both classification and survival models, a higher output $X \cdot \beta$ corresponds to a higher risk. Accordingly, a threshold can be set to divide the total amount of still "healthy" vehicles into vehicles for repair and vehicles without repair. As in real-life, a trade-off between false positives and false negatives arises with this split, which individually needs to be optimized[7]. Figure 9 shows an exemplary density plot of vehicles with failure and without failure after split date over inferred risk score[8]. For models with high predictive ability, both density curves overlap as little as possible. A metric is introduced to measure this overlap, i.e. the predictive ability for a given model and given split date.

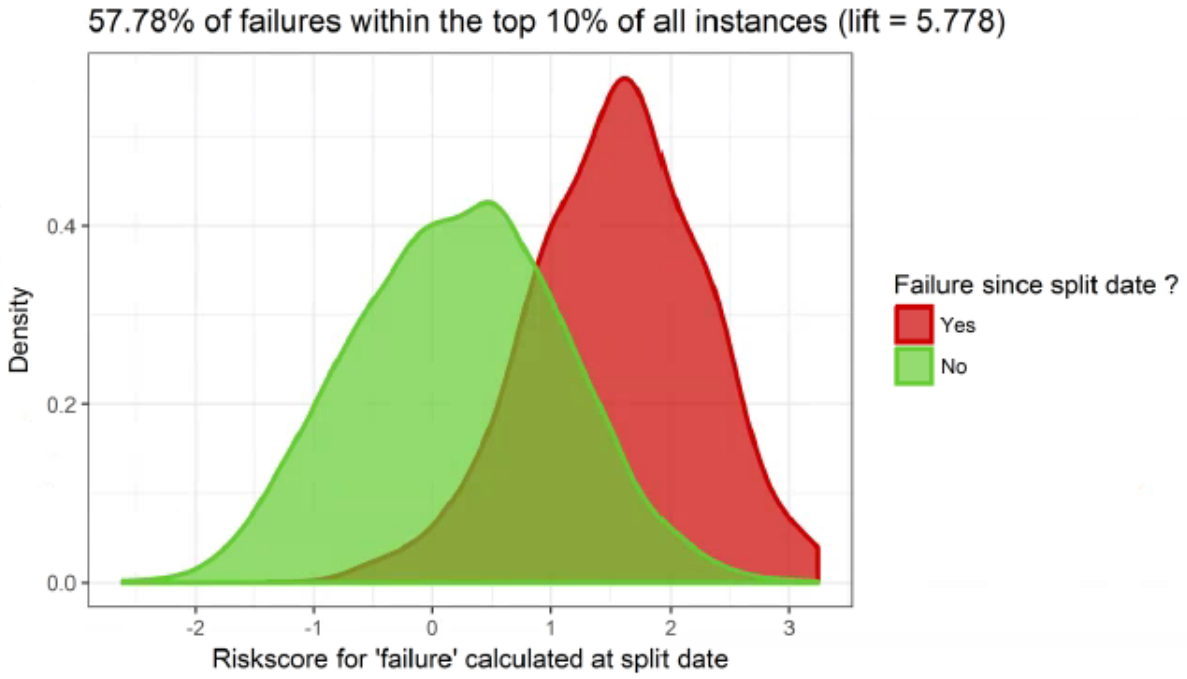

Figure 9: Density plot of vehicles with and without failure after split date over inferred risk score.

Especially important for a real-life business case is the amount of failures within a certain percentage of most risky vehicles of the validation set. If the 10% most risky vehicles are

[7] This optimization is business case-specific and is therefore not subject of this work.
[8] Note that the risk score of a survival model is not limited to zero and one as in the case of a logistic regression.

repaired at split date, i.e. one sets a hypothetical threshold to 0.1, one prevents here nearly 58% of future failures. Therefore, the percentage of instances with validation label one, i.e. true positives, within the respective percentage of all vehicles, i.e. true positives plus false positives, in the validation set for a given threshold is a good indicator for high predictive ability. In this work, the lift is defined as the rate between threshold, i.e. percentage of all vehicles, and respective TP to overall positive rate. For a random classification into 0 and 1 one would approximately get a lift factor of 1. Different lift factors result from different percentages of vehicles, i.e. thresholds, and span a curve from 0 to 100 percent. The area under lift (*AUL*) is, in analogy to the *AUC*, defined as the area under this curve and is to be maximized (compare Figure 10):

$$AUL := \frac{1}{n} \cdot \left[\sum_{i=1}^{n} \frac{TP\left(\frac{i}{n}\right)/P}{i/n} \right], \tag{6}$$

where n is the amount of validation instances and $\theta = i/n$ is the hypothetical threshold from 0 to 100 percent of instances, TP and P are the true positives and overall positives for a validation data set built with the above methodology. An upper bound for the *AUL* is one over the total failure rate of the test data set[9].

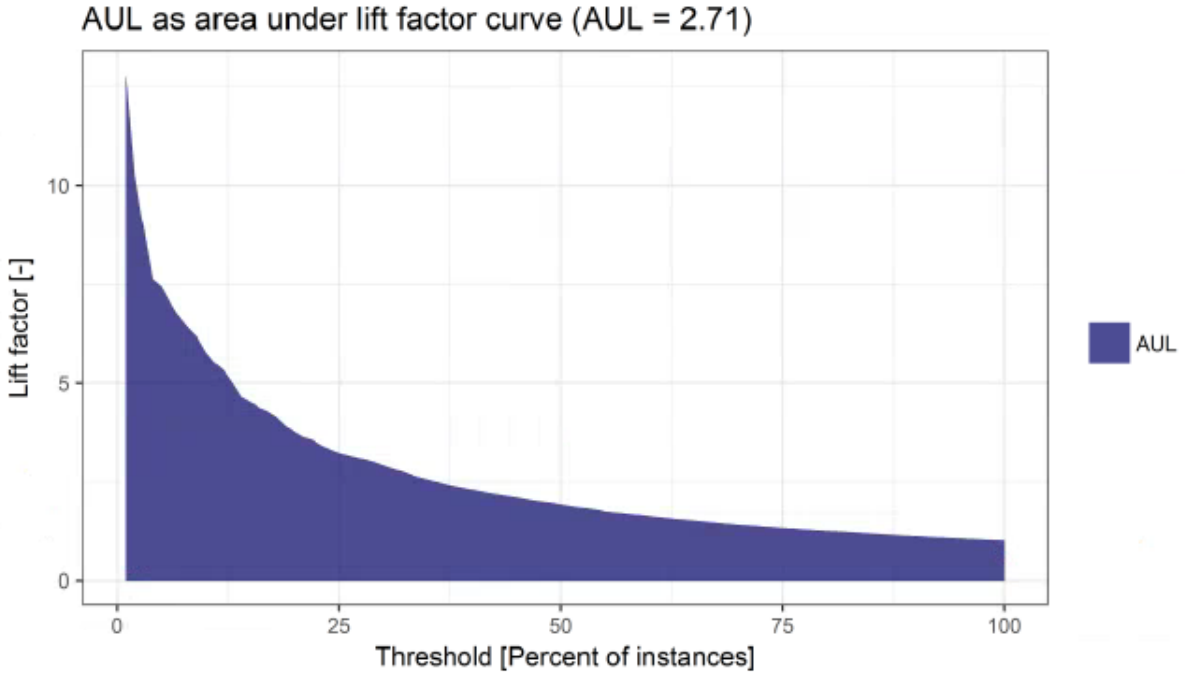

Figure 10: Definition of AUL by the area under the lift factor.

Both *c index* on the test data and *AUL* on the validation data are performance metrics used in the following. The former mainly measures the generalization of the model fit but hardly ensures the predictive ability as it cannot expose pure diagnosis in the model. Predictive ability can be measured with the *AUL* which is therefore a perfect complement to the *c index*. If a

[9] $AUL = \frac{1}{n} \cdot \left[\sum_{i=1}^{n} \frac{TP\left(\frac{i}{n}\right)/P}{i/n} \right] < \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{1}{\varphi} = \frac{1}{\varphi}$ with φ being the failure rate of the test set.

trade-off between *AUL* and *c index* arises one can forfeit in *c index* to increase the *AUL* to get better predictions in the future. For survival model evaluation in section 9.2, both metrics are derived and compared for a given split date. To further evaluate the sensitivity of the *AUL* to the length of validation phase, i.e. the time difference between split date and cut-off, validation data time span is varied from one to six months. This is done to quantify the predictive ability of models for different periods of time in the future. This is also answer to the question, how far into the future a model can well predict failures given today's data.

To further boost the model prediction as well as to ensure comparability between validation set instances, all time dependent features are extrapolated from the actual date of diagnosis to split date. This is done in the same way as the extrapolation of censored survival times in the initial data set as explained in section 8.2. The extrapolation artificially generates instances exactly at split time and not randomly before that date, making extrapolated instances better comparable by their time dependent features. The extrapolation is necessary as the risk score is only used for rank ordering and cannot resemble any time dependence. When using rank ordering, all instances have to be rated at the exact same time, otherwise the ranking is imprecise since the initial instances are evaluated for different times in the past. As a consequence, the extrapolation increases prediction accuracy.

8.4 Survival probability prediction

So far, only the model outputs of classification and Cox models, i.e. $X \cdot \beta$, are compared and evaluated. Neither of these model outputs resembles a real risk of failure but only pseudo risks valid for rank ordering according to higher and lower risk of failure. This risk score ordering is evaluated with the methodology above. The advantage of survival models is however to generate real survival probabilities over time between 1, meaning 100% chance of being "healthy", and 0, meaning a 100% chance of having a failure [7]. Simple survival models, like Weibull distributions, are able to predict the average future risk of failure and therefore a future failure rate. However, these simple models do not discriminate vehicles according to risk of failure but treat every instance the same. Using e.g. Cox regressions, individual survival probabilities can be calculated by approximating the underlying failure frequency over time distribution, e.g. Weibull distributions, in addition to learning regression coefficients [7]. The learned base function of such PHM were depicted in equation (4). $S(t \mid X)$ is an approximation of the future survival probability over time after vehicle diagnosis in case for a diagnosis-based analysis. In this work, this extension of a survival model is assessed for Cox PHMs. After fitting the regression part $X \cdot \beta$ in training, a nonparametric estimation of $S(t)$ is performed. The advantage of a nonparametric estimation is that no underlying failure frequency over time distribution is a priori assumed but the distribution is directly derived from training data. Such secondary estimates of Cox PHMs are e.g. Breslow or Kalbfleisch-Prentice estimators. As both survival function estimators differ little, Breslow is used in the following as it is less computationally complex [7, p. 474]. The scope of this work is to generally validate the survival function extensions of survival models based on a Cox regression. Secondary estimates of survival functions can in turn be computed for boosted Cox regressions or tree-based survival models as well. Figure 11 shows an exemplary survival probability curve for three components and vehicles of a potential customer fleet, i.e. a high, medium and low risk vehicle. Since $S(t)$ only depends on the failure distribution in training data, it is the same function for every instance. Only the predictors X scale the curves differently for each instance. The survival functions start flattening out for higher hours of operation since no failures have been observed in the training data so long after a diagnosis.

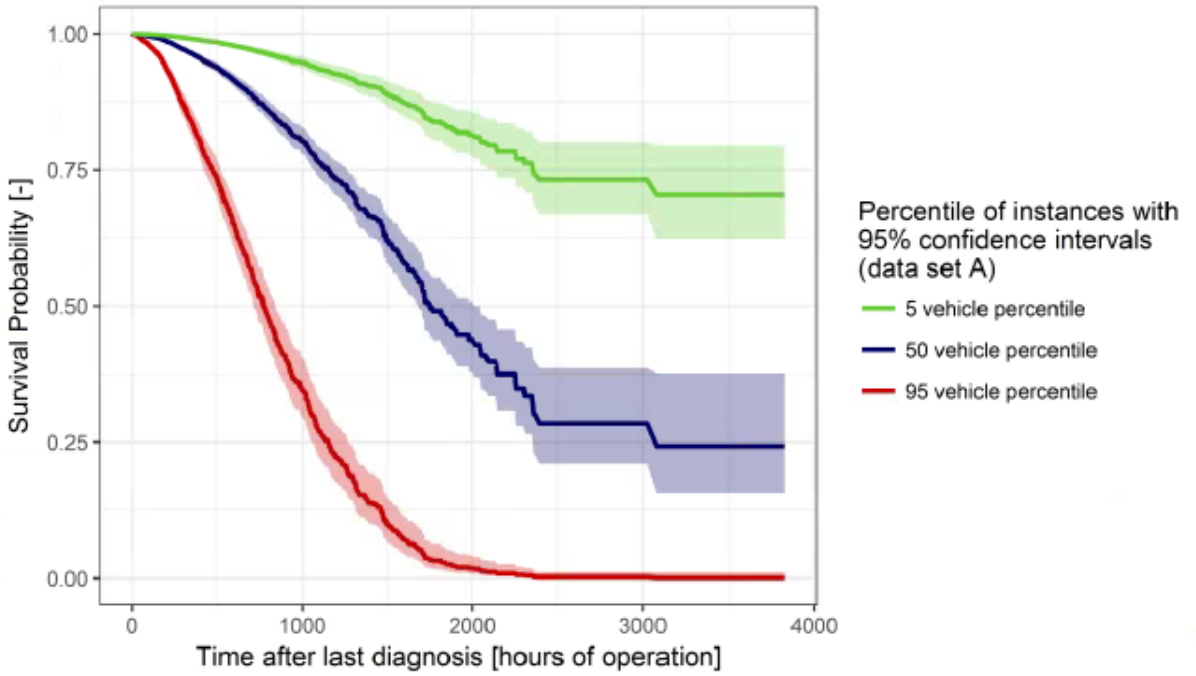

Figure 11: Survival probabilities for the 5, 50 and 95 percentile vehicle over hours of operation as result of a Breslow extension of a Cox regression.

For the non-parametric extension of Cox regression models, *R*'s *survival* package is used. It first fits a Cox model to the data, i.e. estimates the regression part $X \cdot \beta$. Afterwards, a Breslow estimate is performed for all vehicles. This results in a standard estimate of $S(t \mid X)$ without t times feature X interaction. However this work enables a general feature structure, i.e. features of the form $X \cdot t$. Such features, e.g. average temperature times hours of operation, proof very valuable in predicting component survival probabilities as will be shown later. When extrapolating the model time t, this change in t influences the feature itself and has to be included into the model fit. Since the standard Breslow estimate of the toolbox does not account for such interactions, a correction term has to be applied on the model output for all feature-time interactions $X \cdot t$ [7, p. 489]. t_0 denotes the time at split time and Δt the extrapolated time until end of observation, i.e. today:

$$\begin{aligned} S(t \mid X) &= S(t)^{\exp(\beta \cdot X \cdot t)} \\ &= S(t)^{\exp(\beta \cdot X \cdot (t_0 + \Delta t))} \\ &= S(t)^{\exp(\beta \cdot X \cdot t_0 + \beta \cdot X \cdot \Delta t))} \\ &= S(t)^{\exp(\beta \cdot X \cdot t_0) \cdot \exp(\beta \cdot X \cdot \Delta t)} \\ &= \left[S(t)^{\exp(\beta \cdot X \cdot t_0)}\right]^{\exp(\beta \cdot X \cdot \Delta t)} \end{aligned} \tag{7}$$

where the inner $S(t)^{\exp(\beta \cdot X \cdot t_0)}$ is exactly the *R* package output and $\exp(\beta \cdot X \cdot \Delta t)$ is the necessary correction term. Due to the initial normalization of $X \cdot t_0$ with respective mean and variance, $\beta \cdot X \cdot \Delta t$ has to be divided by the standard deviation of the initial $X \cdot t_0$. Applying this correction term every survival probability for every instance is decreased the further one evaluates the survival probability in the future.

Survival probabilities are first validated with the same technique as risk scores by splitting the data into train, test and validation data set with a given split date. Using the same technique as above, the model is trained and its risk scores tested on the test set. Afterwards, the non-parametric estimate is derived which results in survival probabilities over time for every

instance. The features are not timely extrapolated to the exact split date since every survival probability is already a function over time. The validation data set can now be used to evaluate the model fit, i.e. $S(t)$. One gets the exact survival probability for the cut-off time by querying the model output, i.e. the survival probability over time function, at the approximated inherent model time between each diagnosis and cut-off time. This time difference is not necessarily the absolute time but, as explained above, for instance hours of operation. As a matter of fact, the exact hours of operation of every vehicle are not known for a future date. The model time difference is in turn derived with the same technique that was used to extrapolate every instance to spit date, now the model time is just further extrapolated to the end of observation, i.e. cut-off date. The calculated survival probabilities can then be compared with the real failures in the validation data, i.e. failures between split date and cut-off date. The simplest form would be to compare the average survival probability with the actual rate of instances with failures. Is the number of instances high enough, these two rates should equal as much as possible. To enhance this simple evaluation, the vehicles can be split into e.g. instance deciles from high to low risk. The average survival probability of each of these sets should then be similar to the real rate of failure, denoted with validation label one, within these deciles. Thereby, prediction is evaluated independently for high and low survival probability instances.

The main advantage of $S(t \mid X)$ in comparison to only risk scores, i.e. the regression part $X \cdot \beta$, are the real survival probabilities and not just rank ordering pseudo risk scores. This advantage can be exploited by estimating real future risks of failure and accordingly confusion matrices over threshold[10]. Today's overall failure rate of all vehicles can be small, but after e.g. three years this can be significantly higher and estimations of FP and TP rates can be misleading. A Weibull prognosis is a simple approximation of such behavior. A Cox regression inherently displays failure distributions like Weibull and can in addition discriminate vehicles according to an individual risk of failure. Hence, confusion matrices of a future date derived by survival probabilities give a more realistic estimate than confusion matrices derived from today's risk scores. As each vehicle can be assigned with a different survival probability in the future, the amount of total failures is described by a Poisson binomial distribution [43]. Mean and variance for this distribution can be calculated for a given survival probability threshold given the survival probabilities p_i. P denotes the amount of vehicles with failure :

$$E(P) = \sum_{i=1}^{n}(1 - p_i) \tag{8}$$

$$Var(P) = \sigma^2 = \sum_{i=1}^{n} p_i\,(1 - p_i)$$

Exact failure distributions, i.e. the cumulative probability distribution, are however computationally very expensive and exact solutions therefore hard to find [43]. Some approximation methods exist for an exact cumulative Poisson binomial distribution. Instead, a Monte Carlo sampling approach is proposed in this work to find the future failure distributions. It is shown in the following that the convergence over sample size is reasonable good and the failure distributions can thus be approximated.

As a matter of fact no labels exist whether a component experienced a failure or not when deriving survival probabilities for a given date in the far future for today's population of "healthy" vehicles. This is crucial as commonly labels and risk scores are used to derive confusion matrices. To generate the missing information whether the vehicles are actually "healthy" at that time or the component had a failure, a 0/1 label is sampled for each instance with given

[10]Note that these confusion matrices can also be computed with current risk scores, there is however no way to estimate the failure rate in the future with an underlying correct failure frequency over time distribution.

survival probability. Subsequently, one possible set of confusion matrices can be derived by comparing the sampled labels with predicted survival probabilities[11]. This combination of labels for all instances is however only one guess for real future failures and has to be repeated multiple times to generate an overall average guess of future failure rates and confusion matrices. The amount of sampled failures for each single instance then naturally converges against its predicted survival probability at that time. Besides the approximation of realistic confusion matrices, this technique predicts the amount of future failures and its probability distribution at any time in the future by sampling many failure rates. This is especially valuable when the failure rate is expected to increase over time and today's risk scores are only based on today's failure rate[12]. It has to be noted, that this confusion matrix estimation relies on a correct model fit, i.e. a valid $S(t)$. The validity of $S(t)$ is shown in section 9.3 before sampling failure rates. If the correct failure frequency over time distribution cannot be estimated from the historical data, a parametric fit like a Weibull fit can perform better.

[11] Again, by sorting according to risk and evaluating TP, TN, FN and FP over threshold.

[12] The future failure rate is generated by a parametric or non-parametric estimation of $S(t \mid X)$ by incorporating the failure frequency over time distribution into a survival model fit.

9 Results of survival model fitting

This chapter depicts the results of fitting survival models on two datasets representing two different kinds of real vehicle component failures. Both components where ex ante not identified as deteriorating system and are therefore not directly monitored with onboard sensors. The data hence mainly consists of motor and electronic stability control unit information. At first, the failures will be described and the failure distribution over time presented. In the second section, diagnosis-based classification and survival models are fitted and the risk scores evaluated and compared. The third section depicts the results of a Breslow estimate for survival probabilities of a Cox regression.

9.1 Presentation of data

The two datasets display two empirically monitored component failures of different vehicle series [44]. The first, in the following failure A or dataset A, is a mechanical failure which often appeared in high temperature regions. The root cause of the failure was known by the time these analyses were performed and so this dataset was used mainly for evaluation reasons. Dataset B or failure B is an electric failure of a relais component. The root cause of this failure is by the time of this thesis not understood and the data analysis is also used to determine possible influences on the deterioration.

Both failures do not affect the whole fleet but only a subset of vehicles which contain the respective component. Therefore, used filters to preselect the data were vehicle type and relevant ECU firmware version. An important prerequisite for a successful model is the amount of available data after filtering, i.e. the amount of observed failures in the data. The first step of the analysis was therefore to extract the diagnosis data and warranty data for this specific subset of vehicles as well as the warranty failure code for labelling. The failures were in turn defined by a combination of warranty repair of the component and corresponding DTC in the diagnosis data. As explained above, the failure time is attributed to the subsequent diagnosis as no exact mileage is known where the components broke.

Both datasets were first descriptively analyzed. This is necessary to ensure that failures are not randomly appearing in the data but are caused by a long term deterioration process. These requirements can e.g. be checked by a failure frequency distribution over time. As explained above, model time is here not absolute time but can be any monotonic process variable describing the deterioration process. Is has been proven advantageous here to define time as vehicle mileage or hours of operation. The following analyses are all performed with hours of operation as model time variable to treat different driving velocities as equally abrading, i.e.

consider idle time as abrading. To include high velocities as negative influence in the model, average velocity is a possible feature in a survival model later. The failure distributions are displayed as relative failure frequency over hours of operation for all diagnoses and diagnoses with attributed failure in Figures 12-13 for the two datasets [44]:

- **Dataset A** contained in total 7,138 vehicles, 21,578 diagnoses and 1290 diagnoses corresponding to a failure. These diagnoses were recorded between April 2012 and January 2017. The median diagnosis date is the 18/09/2015.
- **Dataset B** contained in total 34,931 vehicles, 65,799 diagnoses and 3,184 diagnoses corresponding to a failure. These diagnoses were recorded between April 2014 and June 2017. The median diagnosis date is the 15/09/2016.

It is noted, that at this point multiple failures per vehicle are allowed. In further analyses, only the first failure is of interest and all subsequent diagnoses are left out[13].

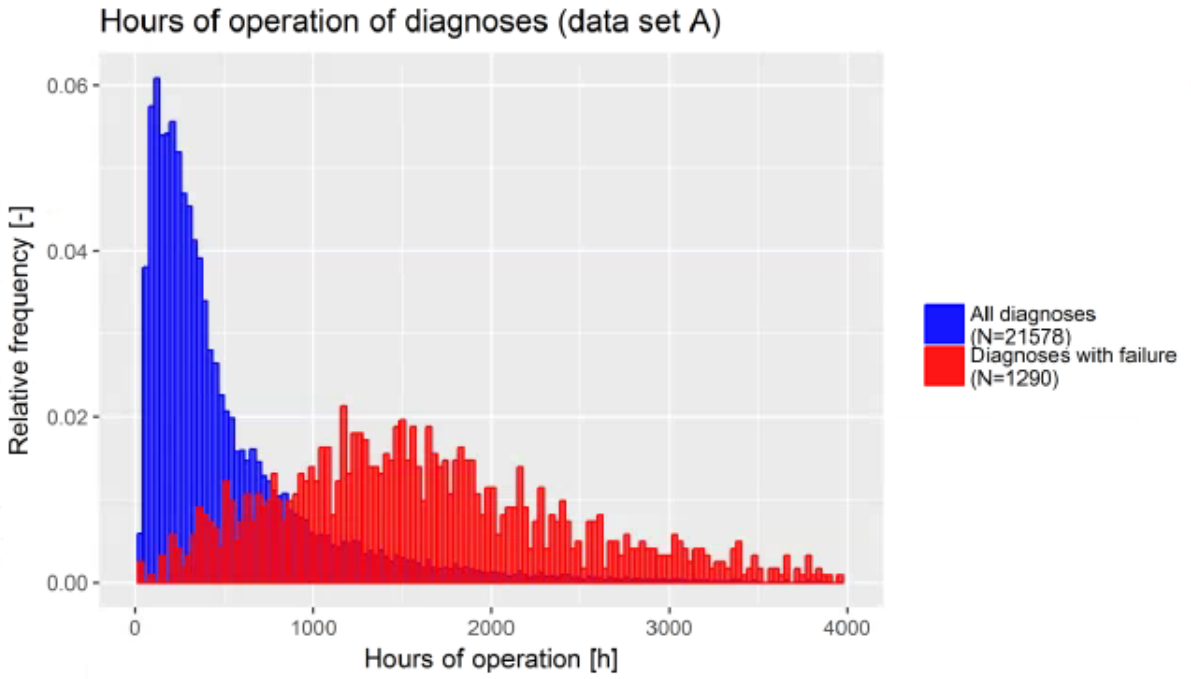

Figure 12: Comparison of relative failure frequency and relative diagnosis frequency over hours of operation for failure A [44].

[13] This is a simplification only for training, of course a model would be able to infer risks after a (repaired) failure as well.

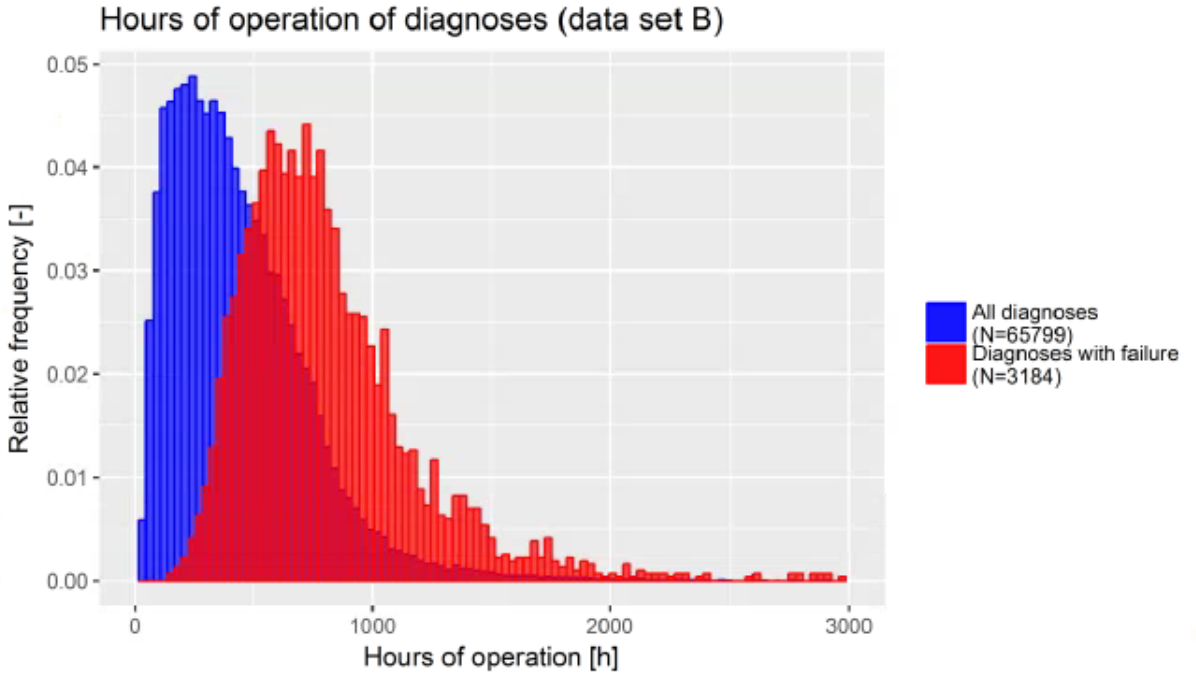

Figure 13: Comparison of relative failure frequency and relative diagnosis frequency over hours of operation for failure B [44].

The depicted frequency distributions confirm a possible deterioration process over vehicle usage time and make the assumption of random failures extremely unlikely. Both failures are thus modelled and evaluated with the above methodology. Further data filters are a minimum mileage of 1000 km and minimum absolute time difference between vehicle registration and diagnosis of 365 days to ensure reliable data. Specifically aggregated temperature data is only trustworthy after sensors have recorded all seasons.

All features of validation data instances, i.e. all vehicles without previous failure at split date, are extrapolated to split data. Instances with survival time equal to zero, i.e. instances with failure at that time, are not excluded here. The 01/09/2016 was set as split date for dataset A. The 01/12/2016 was set as split date for dataset B. This results in the following characteristics for datasets A and B shown in Table 3 and 4 for a diagnosis-based analysis.

Dataset A		Amount	Median survival time	Mean survival time	Maximum survival time
Train/test (before split date)	*Uncensored instances*	2,698	230.3 h	359.7 h	4309.0 h
	Censored instances	18,593	70.5 h	341.1 h	3083.0 h
Validation (after split date)	*Instances with failure*[14]	180	-	-	-
	Instances without failure	4,557	-	-	-

Table 3: Summary of instances of failure A [44].

Dataset B		Amount	Median survival time	Mean survival time	Maximum survival time
Train/test (before split date)	Uncensored instances	2,248	0 h	100.6 h	1273.0 h
	Censored instances	39,152	157.7 h	219.8 h	4611.0 h
Validation (after split date)	Instances with failure	585	-	-	-
	Instances without failure	14,425	-	-	-

Table 4: Summary of instances of failure B [44].

9.2 Model training and risk score evaluation

This section describes the results of different survival models inferring a risk of failure and comparing them to a binary classification. The data preprocessing, i.e. the composition of a diagnosis-based input data set, model training and evaluation follow the methodology described in chapter 8. All following model training and evaluation steps are performed with a set of predefined features that were found to generate good models. These features were defined by expert knowledge on the physical deterioration process and earlier predictive analytics results. This section thus evaluates the general performance of survival models and paves the way for future model selection considerations. This very chapter is not concerned with finding other or better features. Chapter 10 describes a detailed feature selection methodology to find equally well performing features automatically. Both datasets were reduced to only the few necessary features to model the failure and the labels. All other diagnosis information is irrelevant for the modelling.

[14] The validation data set contains all instances that are considered "healthy" at split date. Instances with failure correspond to a failure after split date, i.e. a component which broke down after split date. For evaluation reasons, no survival time is given here but only the information whether a failure appeared or not. For details see section 8.3.

The features are:

- **Dataset A:**
  Relative time at idle speed times absolute hours of operation
  Relative time at idle speed times relative operation time at high temperature
  Relative time at idle speed times absolute mileage
- **Dataset B:**
  Relative time at idle speed times absolute motor starts
  Relative time at idle speed absolute hours of operation
  Relative time at high temperature times absolute hours of operation

These features are interactions of characteristic influences on the wear process. They are also time dependent and have to be handled with extrapolations and correction terms (see section 8.4) since usually feature-time interactions are not included in survival models. This time interaction is however very advantageous for PM solutions as shown by the feature selection in chapter 10. All features are normalized using the mean and standard deviation over all instances. In addition, instances with missing values are neglected for training.

Different model types and model parameter combinations are evaluated and compared in the following. As a matter of fact, the degree of freedom resulting from only a few exogenic model parameters is high and the optimal parameter combination for any machine learning model is hardly to be found here. This work does not follow a brute force approach to find an optimum but evaluates general properties of the models and their respective design parameters.

The described methodology is used to evaluate the performance of Cox regressions, boosted Cox regressions, boosted survival trees and random survival forests for a fixed validation set. Toolboxes for training were *glmnet* [30], *Coxboost* [36], *gbm* [40] and *randomForestSRC* [22], respectively. All test-train splits are vehicle-wise disjoint, censored survival times are extrapolated to split date and validation instances are extrapolated to split date. Subsequently, the cut-off of the validation set is varied to evaluate the predictive ability in comparison to a binary classification.

9.2.1 Model training and performance

For the first evaluation, the validation set is not cut so that it includes all information up to end of observation, i.e. the last recorded diagnosis of the datasets. As the two classes, uncensored and censored instances, are highly imbalanced, techniques to equalize these are evaluated, too. Here, complete downsampling, complete upsampling and weighting are considered. The first randomly chooses as many censored instances as there are uncensored instances for training. This reduces the amount of instances in the training data set significantly. Upsampling generates artificial instances for training sampled and interpolated from the original uncensored instances. By this procedure the amount of instances for training increases by almost 100%. Weighting gives every instance an importance weight for training according to the proportion of censored to uncensored instances. This weight then affects the internal model optimization to value the smaller class accordingly higher. These techniques are only applied to the training set. All evaluation calculations are performed on the unaltered test and validation data sets[15]. The following Tables 5-12 show the results of model training, i.e. the *c index* on

[15] Unaltered test and validation sets are necessary, as the model performance has to be tested on the real proportion of failures to "healthy" instances.

the test data and *AUL* on validation data, for unaltered, downsampled, upsampled and weighted training data sets.

Furthermore, it is tested whether it is advantageous to include instances with Cox time equal to zero, i.e. diagnoses with observed failures, in training phase. Commonly, such instances cannot be processed by survival models. However, shifting the Cox time by adding one hour of operation to the Cox time of all instances makes these instances a valid input for training[16]. The procedure is denoted Cox time shifting in the following. When including the instances with failure in a model one has to make sure that no diagnosis feature is in the model, otherwise the model would just learn to diagnose a failure in the aftermath. Yet, the provided features are expected to indicate failures beforehand and the *AUL* ensures a proper model prediction.

Tables 5 and 6 show the results of fitted Cox regressions by the *glmnet* toolbox. The Cox regression is regularized by an elastic net penalty as generalization of ridge and lasso regularization. The implementation internally optimizes the magnitude λ of regularization. Different combinations of the regularization parameter α with Cox time shift and data set weighting are evaluated. For failure A, the best *c index* is reached by a downsampled training set with a Cox time shift. The *AUL* is highest for the upsampled training set with Cox shift. However, the *c index* is overall very similar and down- and upsampling do not bring significant increases of *AUL* compared to the unaltered training data. For failure B, the unaltered training set results in the best performance, with a trade-off between *c index* and *AUC* with and without Cox time shift.

***glmnet* (dataset A)**	**Survival time shift**	**Parameter (λ *opt.*)**	***C index***	***AUL***
Unaltered dataset	0 Cox time shift	α= 0,	0.885	2.547
		α = 1	0.886	2.546
	+1 Cox time shift	α = 0	0.901	2.717
		α = 1	0.900	2.717
Downsampled dataset	0 Cox time shift	α = 0	0.890	2.570
		α = 1	0.888	2.561
	+1 Cox time shift	α = 0	**0.903**	2.725
		α = 1	**0.903**	2.725
Upsampled dataset	0 Cox time shift	α = 0	0.889	2.571
		α = 1	0.889	2.561
	+1 Cox time shift	α = 0	0.902	2.722
		α = 1	0.902	**2.727**
Weighted dataset	0 Cox time shift	α = 0	0.888	2.566
		α = 1	0.888	2.556
	+1 Cox time shift	α = 0	0.902	2.716
		α = 1	0.902	2.725

Table 5: Results of fitting Cox regressions to failure A [44].

[16] No loss of accuracy due to the rank ordering of instances.

glmnet (dataset B)	Survival time shift	Parameter (λ *opt.*)	*C index*	*AUL*
Unaltered dataset	0 Cox time shift	$\alpha = 0$	0.868	2.100
		$\alpha = 1$	0.857	**2.107**
	+1 Cox time shift	$\alpha = 0$	**0.882**	2.018
		$\alpha = 1$	0.881	1.955
Downsampled dataset	0 Cox time shift	$\alpha = 0$	0.705	1.769
		$\alpha = 1$	0.709	1.818
	+1 Cox time shift	$\alpha = 0$	0.879	1.980
		$\alpha = 1$	0.880	1.946
Upsampled dataset	0 Cox time shift	$\alpha = 0$	0.786	1.789
		$\alpha = 1$	0.787	1.788
	+1 Cox time shift	$\alpha = 0$	0.880	1.973
		$\alpha = 1$	0.880	1.946
Weighted dataset	0 Cox time shift	$\alpha = 0$	0.802	1.843
		$\alpha = 1$	0.790	1.802
	+1 Cox time shift	$\alpha = 0$	0.880	1.974
		$\alpha = 1$	0.880	1.946

Table 6: Results of fitting Cox regressions to failure B [44].

As for basic Cox regression down- or upsampling or weighting do not look promising, boosted Cox regression is here only performed for the unaltered dataset. The *CoxBoost* implementation initializes all feature coefficients with zero and stepwise optimizes these on the training data. The step number, i.e. the number of boosting steps, is internally optimized by cross-validation. The results are depicted in Tables 7 and 8. In case of failure A, a Cox time shift looks more promising while in case B both metrics favor different training data.

CoxBoost (A)	Survival time shift	Parameter	*C index*	*AUL*
Unaltered dataset	0 Cox time shift	*Step no. optimized*	0.885	2.552
	+1 Cox time shift	*Step no. optimized*	**0.901**	**2.732**

Table 7: Results of fitting boosted Cox regressions to failure A [44].

CoxBoost (B)	Survival time shift	Parameter	*C index*	*AUL*
Unaltered dataset	0 Cox time shift	*Step no. optimized*	0.860	**2.105**
	+1 Cox time shift	*Step no. optimized*	**0.883**	1.961

Table 8: Results of fitting boosted Cox regressions to failure B [44].

Tables 9-10 show the results of fitting boosted survival trees with the *gbm* package. Exogenic parameters to define are mainly tree depth, number of trees and shrinkage. The first is the number of allowed splits per tree, the second the total amount of trees, i.e. the number of boosting steps. The third is a multiplication term which shrinks the change a new tree brings into the model, i.e. a parameter that slows the gradient descent process. Therefore, the shrinkage is similar to a learning rate, the smaller the value the slower the learning process is in training phase. An additional *k*-fold cross-validation for every boosting step can be performed to ensure the model is not overfitting. While the *c index* on the training part of data from the *k*-fold split monotonically decreases, the performance on the test part of the *k*-fold data split indicates overfitting. For both datasets, models on the unaltered data perform best with a Cox time shift. It is also noticeable, that a tree depth of two usually performs best. The higher number of trees does not result in better predictions.

gbm (A)	Survival time shift	Parameter	*C index*	*AUL*
Unaltered dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.877	2.556
		Depth = 3, trees = 10000, shr. = 0.001	0.878	2.560
		Depth = 2, trees = 1000, shr. = 0.01	0.879	2.576
		Depth = 4, trees = 1000, shr. = 0.01	0.875	2.526
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.893	2.693
		Depth = 3, trees = 10000, shr. = 0.001	**0.894**	**2.700**
		Depth = 2, trees = 1000, shr. = 0.01	**0.894**	2.696
		Depth = 4, trees = 1000, shr. = 0.01	0.893	2.694
Downsampled dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.876	2.468
		Depth = 3, trees = 10000, shr. = 0.001	0.875	2.484
		Depth = 2, trees = 1000, shr. = 0.01	0.876	2.499
		Depth = 4, trees = 1000, shr. = 0.01	0.875	2.449
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.893	2.688
		Depth = 3, trees = 10000, shr. = 0.001	0.893	2.687
		Depth = 2, trees = 1000, shr. = 0.01	0.892	2.685
		Depth = 4, trees = 1000, shr. = 0.01	0.893	2.676
Upsampled dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.874	2.491
		Depth = 3, trees = 10000, shr. = 0.001	0.874	2.503
		Depth = 2, trees = 1000, shr. = 0.01	0.875	2.515
		Depth = 4, trees = 1000, shr. = 0.01	0.873	2.495
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.892	2.670
		Depth = 3, trees = 10000, shr. = 0.001	0.892	2.672
		Depth = 2, trees = 1000, shr. = 0.01	0.891	2.669
		Depth = 4, trees = 1000, shr. = 0.01	0.892	2.651

Weighted dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.874	2.496
		Depth = 3, trees = 10000, shr. = 0.001	0.875	2.503
		Depth = 2, trees = 1000, shr. = 0.01	0.876	2.526
		Depth = 4, trees = 1000, shr. = 0.01	0.872	2.479
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.892	2.661
		Depth = 3, trees = 10000, shr. = 0.001	0.892	2.660
		Depth = 2, trees = 1000, shr. = 0.01	0.891	2.678
		Depth = 4, trees = 1000, shr. = 0.01	0.893	2.653

Table 9: Results of fitting boosted survival trees to failure A [44].

***gbm* (B)**	**Survival time shift**	**Parameter**	***C index***	***AUL***
Unaltered dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.724	1.519
		Depth = 3, trees = 10000, shr. = 0.001	0.731	1.544
		Depth = 2, trees = 1000, shr. = 0.01	0.737	1.632
		Depth = 4, trees = 1000, shr. = 0.01	0.715	1.442
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	**0.887**	1.993
		Depth = 3, trees = 10000, shr. = 0.001	**0.887**	1.984
		Depth = 2, trees = 1000, shr. = 0.01	**0.887**	**2.002**
		Depth = 4, trees = 1000, shr. = 0.01	**0.887**	1.983
Downsampled dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.681	1.392
		Depth = 3, trees = 10000, shr. = 0.001	0.681	1.412
		Depth = 2, trees = 1000, shr. = 0.01	0.685	1.453
		Depth = 4, trees = 1000, shr. = 0.01	0.676	1.335
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.883	1.926
		Depth = 3, trees = 10000, shr. = 0.001	0.883	1.929
		Depth = 2, trees = 1000, shr. = 0.01	0.884	1.961
		Depth = 4, trees = 1000, shr. = 0.01	0.881	1.895
Upsampled dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.695	1.306
		Depth = 3, trees = 10000, shr. = 0.001	0.697	1.308
		Depth = 2, trees = 1000, shr. = 0.01	0.714	1.389
		Depth = 4, trees = 1000, shr. = 0.01	0.684	1.276
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	**0.887**	1.945
		Depth = 3, trees = 10000, shr. = 0.001	0.886	1.949

		Depth = 2, trees = 1000, shr. = 0.01	0.886	1.951
		Depth = 4, trees = 1000, shr. = 0.01	0.886	1.931
Weighted dataset	0 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.703	1.317
		Depth = 3, trees = 10000, shr. = 0.001	0.702	1.329
		Depth = 2, trees = 1000, shr. = 0.01	0.715	1.415
		Depth = 4, trees = 1000, shr. = 0.01	0.695	1.288
	+1 Cox time shift	*Depth = 3, trees = 1000, shr. = 0.01*	0.886	1.941
		Depth = 3, trees = 10000, shr. = 0.001	0.886	1.945
		Depth = 2, trees = 1000, shr. = 0.01	0.886	1.970
		Depth = 4, trees = 1000, shr. = 0.01	0.886	1.921

Table 10: Results of fitting boosted Cox regressions to failure A [44].

As in the case of a basic Cox regression, down- and upsampling or weighting do not perform any better than an unaltered training set for boosted survival trees. The second tree-based models, i.e. random survival forests, are thus only trained on an unaltered data. The design parameter is here the parallel number of trees to train on the data. The performance for random survival trees is shown in Tables 11 and 12 only for the unaltered training datasets. Both *c index* and *AUL* are significantly lower than for any other model type regardless of the number of used trees. Therefore, random forests are not considered in the following assessments.

*RandomForest*SRC (A)	Survival time shift	Parameter	*C index*	*AUL*
Unaltered dataset	0 Cox time shift	n_{trees} = 100	0.866	2.366
		n_{trees}= 1000	0.868	2.379
		n_{trees}= 10000	0.868	2.376
	+1 Cox time shift	n_{trees}= 100	0.879	2.478
		n_{trees}= 1000	0.881	2.515
		n_{trees}= 10000	**0.882**	**2.518**

Table 11: Results of fitting random survival forests to failure A [44].

RandomForestSRC (B)	Survival time shift	Parameter	*C index*	*AUL*
Unaltered dataset	0 Cox time shift	n_{trees}= 100	0.666	1.465
		n_{trees}= 1000	0.689	1.494
		n_{trees}= 10000	0.690	1.393
	+1 Cox time shift	n_{trees}= 100	0.849	1.839
		n_{trees}= 1000	0.853	1.851
		n_{trees}= 10000	**0.855**	**1.855**

Table 12: Results of fitting random survival forests to failure B [44].

It seems that complex models, i.e. boosted Cox regressions and random survival forests do not at all have advantages given these features and this data compared to simpler model types. The runtime of such model trainings is also considerably higher than for a Cox regression or boosted survival trees. In addition, down- and upsampling or weighting do not significantly increase the model performances.

The evaluated parameter combinations give a good but not optimal set of design choices. Further analyses are performed using parameter choices as best guess from these validations.

9.2.2 Model predictions in the future

Both Cox regression and boosted survival trees are tested on their predictive ability for different validation data sets in this section. In addition, a logistic regression is trained on the exact same training data and evaluated on the validation data. The package *h2o* [45] is used for the classification. A regularization of the model is performed by optimizing α and λ automatically by a cross-validation of training data.

How far an instance risk scoring is valid in the future is evaluated by setting different validation data cut-off dates for both failures, i.e. cut-off between one and six months after split date. Only one parameter combination for each model is used here: Cox regression with $\alpha = 1$ and $\alpha = 0$ for failures A and B, respectively. Boosted survival trees with a depth of 3 and 2 for failures A and B, respectively, with a total of 1000 trees and a shrinkage term of 0.01. For both model types, a Cox time shift is performed for training data. The resulting *AUL* of the three model types is compared in Figures 14 and 15.

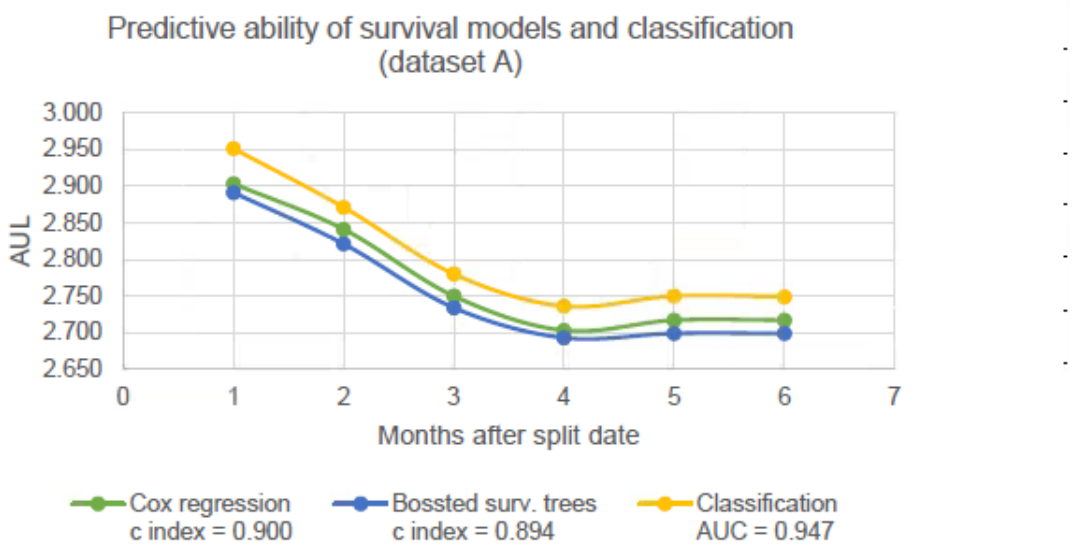

Figure 14: AUC over cut-off in months after split date for Cox regression, boosted trees and logistic regression for failure A [44].

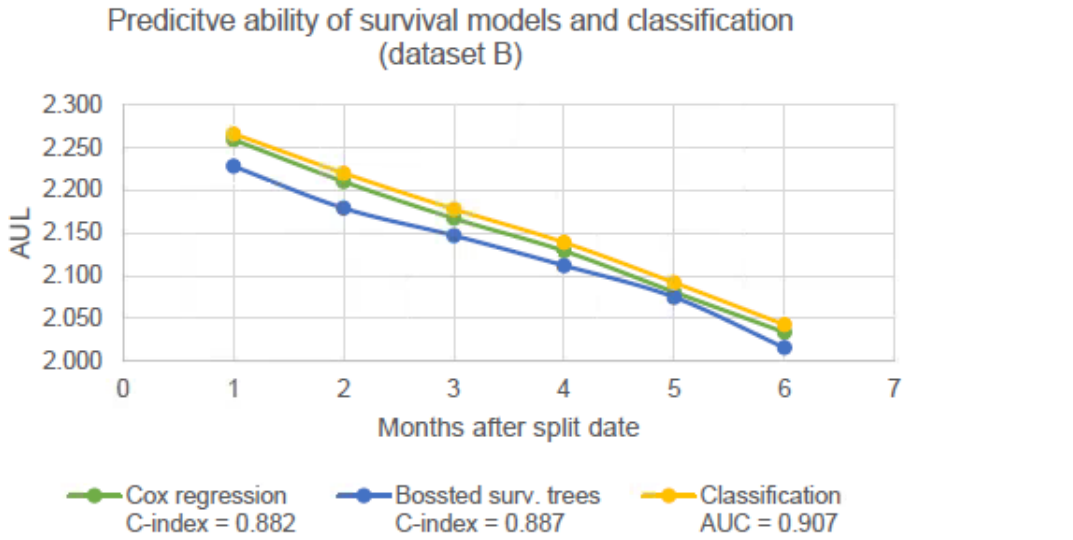

Figure 15: AUC over cut-off in months after split date for Cox regression, boosted trees and logistic regression for failure B [44].

All models tend to predict failures better in the near future than for long periods of time. This is proven with the decreasing *AUL* over increasing cut-off date. As risk scores are derived at split date, the resulting rank order of instances gets more and more inaccurate for future dates. For both model types, i.e. survival models and classifications, this can be coped with by different techniques. In the case of a classification, all time-dependent features can simply be extrapolated to the respective future date resulting in an estimation of future risk scores. This however completely neglects the underlying failure frequency over time distribution. In the case of survival models, this failure frequency over time can be integrated into the model. These extensions, resulting in real survival probabilities over time, inherently display valid risk scores for any future date as depicted in the next section.

In general, the risk scoring performance of survival models is very similar to the performance of the classification as *c index* and *AUC* differ little for the three model types. However, these evaluations only consider the regression part of survival models, not yet the real survival probabilities which can be derived by parametric or non-parametric extensions of a trained model. Most importantly, survival models have equally learned to distinguish between high and low risk vehicles. This is a necessary condition for the further analyses of real survival probabilities extending a basic Cox regression. Nevertheless, risk scoring with survival models is performing as precise as with binary classification.

9.3 Survival probability derivation

The models trained so far only generate pseudo risk scores able to rank vehicles according to this risk. This score is however no real failure probability. To generate these, survival models are extended to infer failure probabilities over time. This chapter describes the results of a non-parametric estimate for survival probabilities of all "healthy" vehicles at split date for both datasets. Similar to above, a Cox regression model is first fitted using the training data and tested on the test data. The derived probabilities are then validated before being used to generate predictions of failure rates in the future.

9.3.1 Breslow estimate and survival probability validation

The Breslow estimate [29] for survival probabilities relies on the failures and respective survival times in training: $S(t)$ is derived from training data which can be further scaled by $\exp(\beta \cdot X)$ to get a vehicle-wise $S(t\,|X)$, i.e. a real survival probability. The *R* package *survival* [46] was used for both training and further non-parametric extension. For a proper derivation of $S(t)$ it is important to not include diagnoses with failure into training, i.e. train the model without Cox time shift. This would otherwise result in a $S(t)$ which gives survival probabilities significantly less than one directly after a healthy diagnosis as the model has artificially seen many instances fail directly after their diagnosis. This artificially induced behavior is not realistic and not valid for deriving $S(t)$ but, as shown above, can positively influence the learning of feature coefficients β.

The Breslow estimate generates survival probability curves for every instance. These are monotonically decreasing up to highest survival time observed in training. The curves are then constant up to the highest observed censored survival time in training. The model is not able to deduce any further prediction. This behavior makes it necessary to train the model ideally on a complete failure frequency distribution. As this is not possible with a limited training set, a non-parametric estimation always leaves uncertainties in the model fit. A parametric fit, e.g. a Weibull distribution, models a complete failure distribution but does not necessarily resemble the real failure frequency distribution. Figures 16-17 each show three survival probability curves for three vehicles over time for dataset A and B. Each curve is shown with its 95 percent confidence interval. The vehicles resemble the 5, 50 and 95 risk percentile of failure probability within the total amount of instances in the validation set corresponding to a low, medium and high risk vehicle.

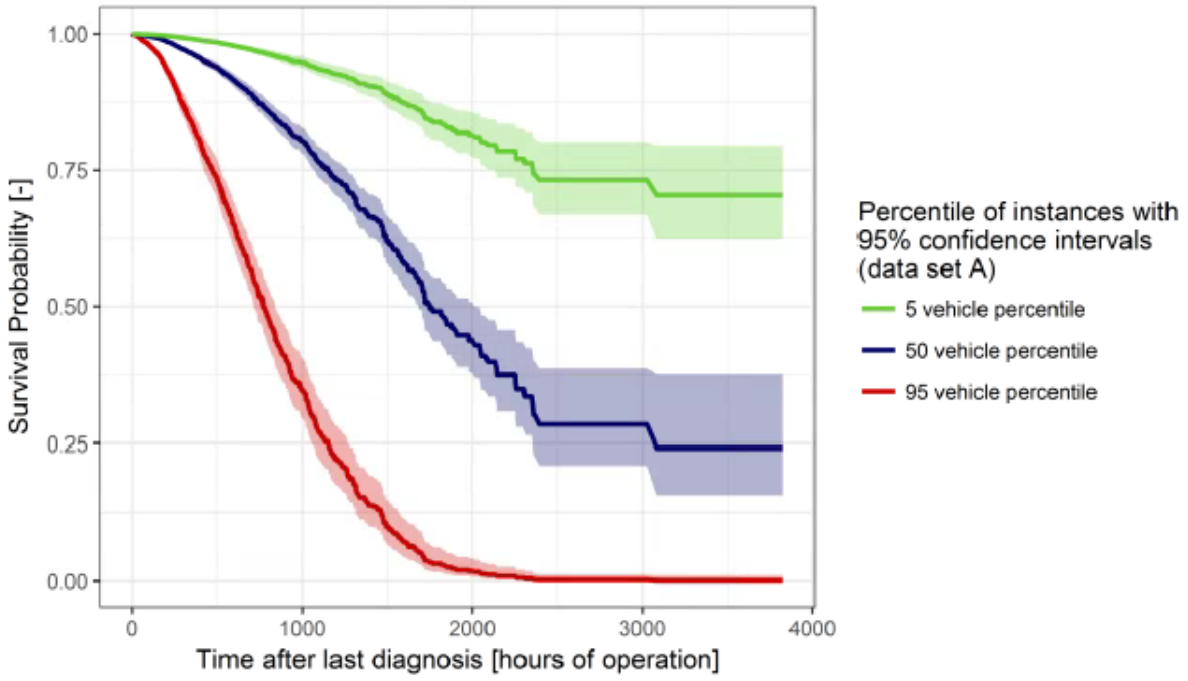

Figure 16: Breslow estimate of survival probabilities for a high, medium and low risk vehicle for failure A [44].

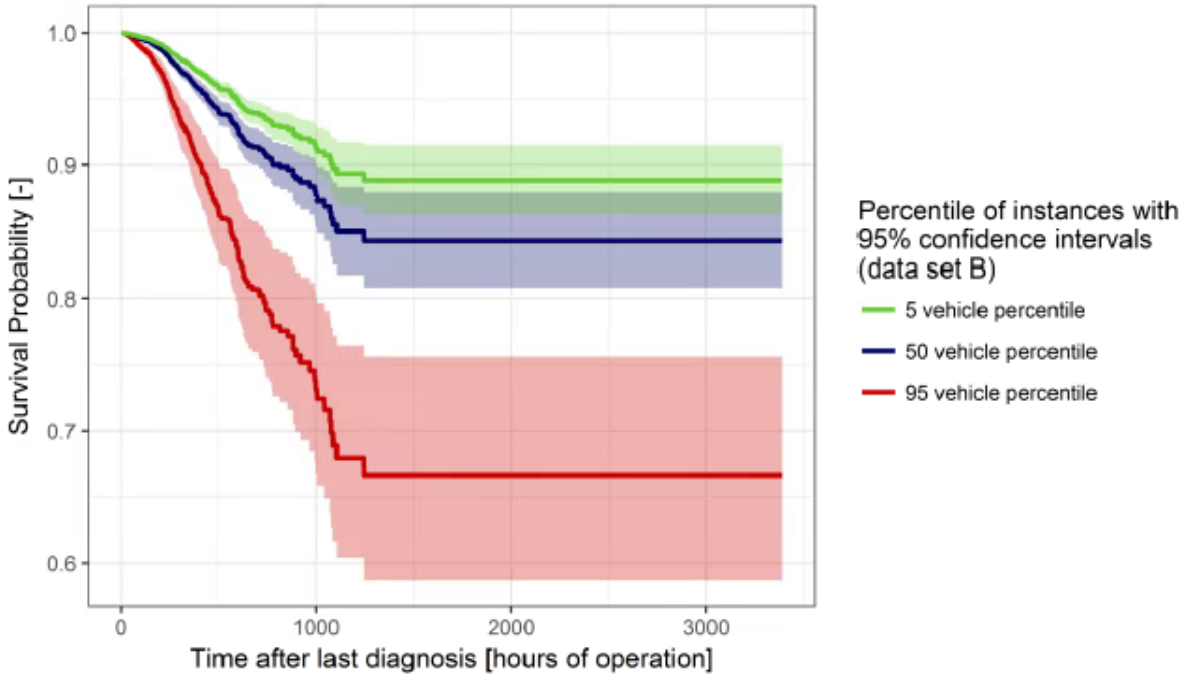

Figure 17: Breslow estimate of survival probabilities for a high, medium and low risk vehicle for failure B [44].

In order to evaluate the resulting $S(t\,|X)$, the validation data set can again be used. Here, a cut-off date of six months after split date is applied to generate the validation set. For each vehicle, the respective survival probability at cut-off date can be queried by extrapolating each vehicle's hours of operation from the last diagnosis to cut-off date. Then, the instances are ranked according to the survival probability at cut-off date. The resulting survival probability distribution over vehicles is shown in Figures 18-19 for datasets A and B. It can be seen, that only a small percentage of vehicles has small survival probabilities while the majority does

probably not fail within this six month period. The overall estimation at split date of failure rates six months after split date for all "healthy" vehicles is in both cases significantly less than one percent off the actual failure rate within this period of time.

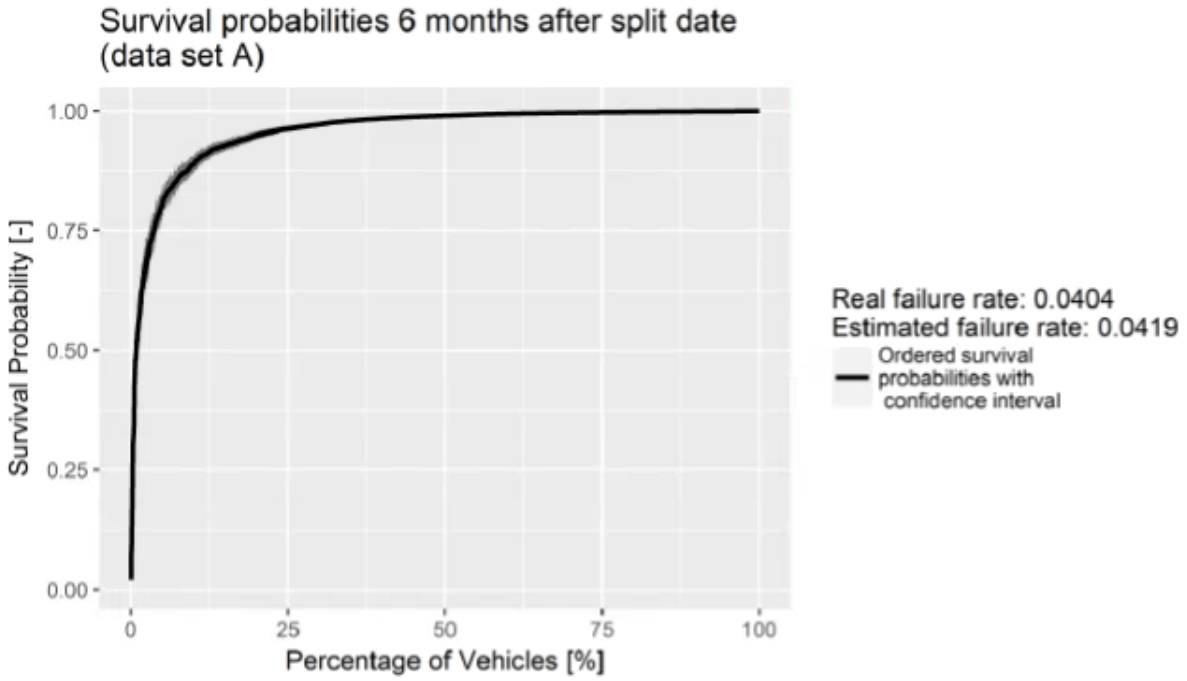

Figure 18: Resulting survival probabilities six months after split date for all vehicles ordered by survival probability for failure A [44].

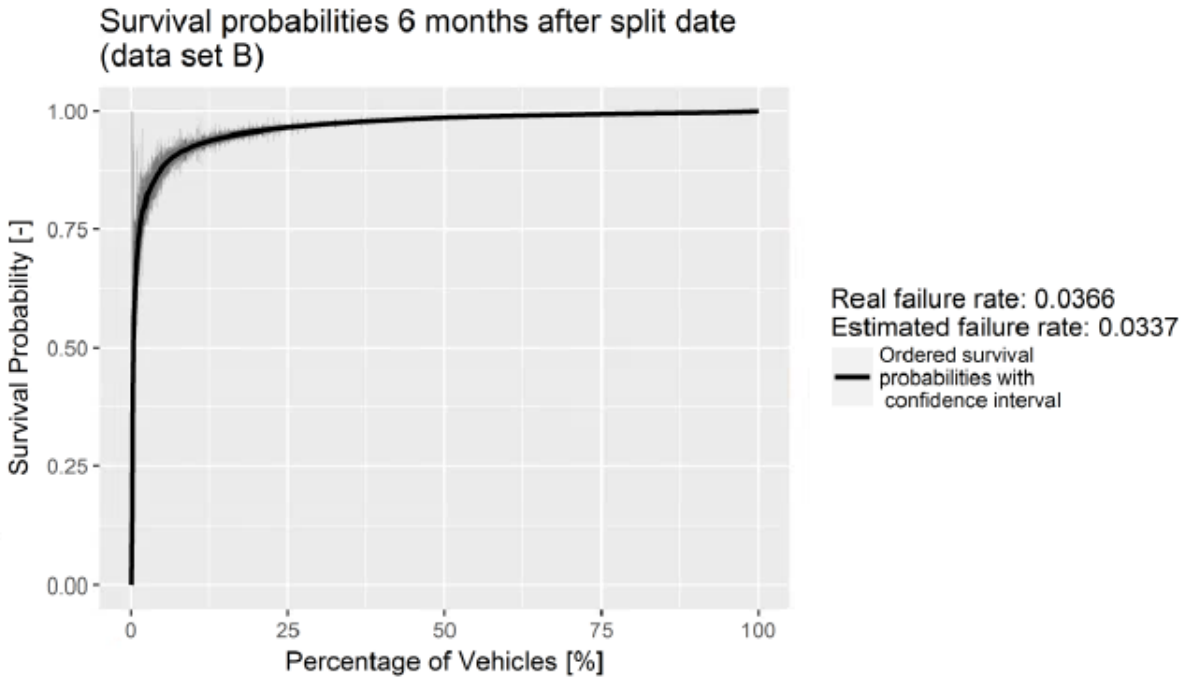

Figure 19: Resulting survival probabilities six months after split date for all vehicles ordered by survival probability for failure B [44].

The ranked instances are split into deciles according to their respective survival probability, resulting in ten sets with equal amount of vehicles from very high to very low risk. For each set, the mean expected failure rate, i.e. one minus the average survival probability, can be compared to the real failure rate of the respective instances labeled with 1 in the validation data set. For the expected failure rate, confidence intervals are derived by using the upper and lower 95% survival probability confidence bound for calculating the mean. The 95% confidence bound for the real failures is determined by a Clopper-Pearson interval [47]. The comparison in Figures 20 and 21 shows a high level of accordance between estimated failure rate and real failure rate. Such an evaluation is mandatory for any subsequent usage of survival probabilities.

The confidence intervals of the estimated failure rate overlap with the Clopper-Pearson interval for the real failure rate in all cases except for some deciles with very small estimated failure rates. Here, estimated failure rates are close to zero failures per decile. The real failure rate within these deciles is very small but in most cases greater than zero. This discrepancy between prediction and reality can however easily be caused by a single failure induced by any unknown reason or even completely random. Due to the respective decile's amount of failures the discrepancy is by no means significant. As the validation thus shows good agreement between predicted and real failures, the Breslow estimate from the training data seems valid.

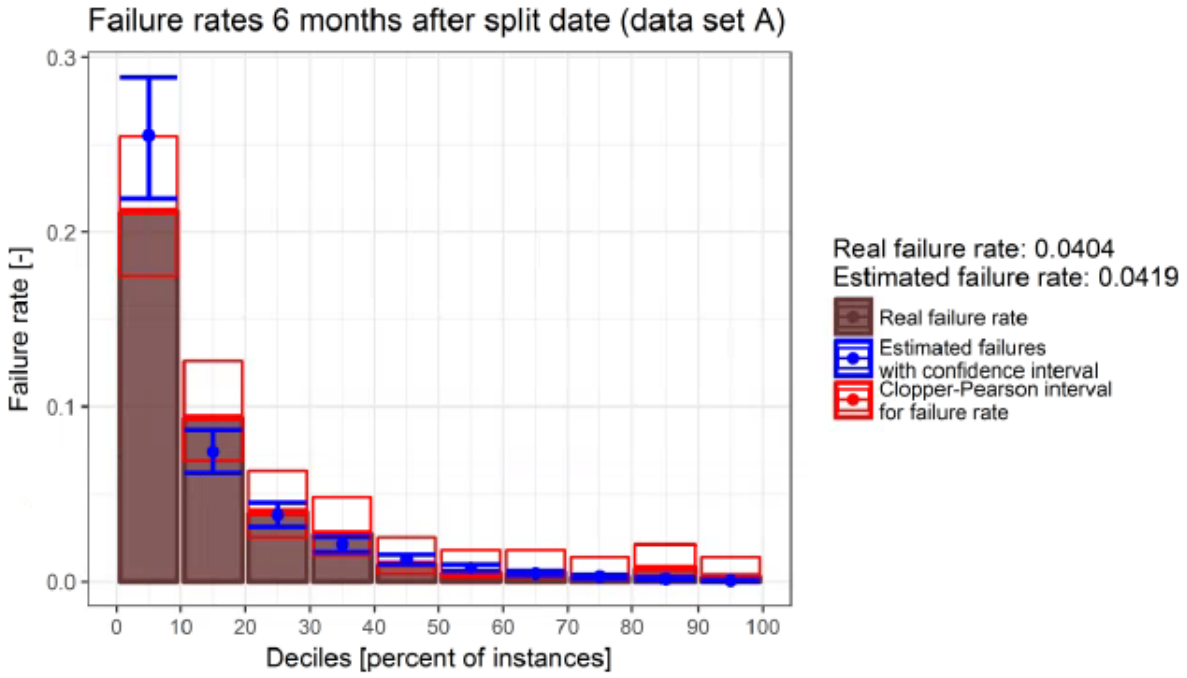

Figure 20: Estimated and real failure rates with confidence intervals for instance survival probability deciles for failure A [44].

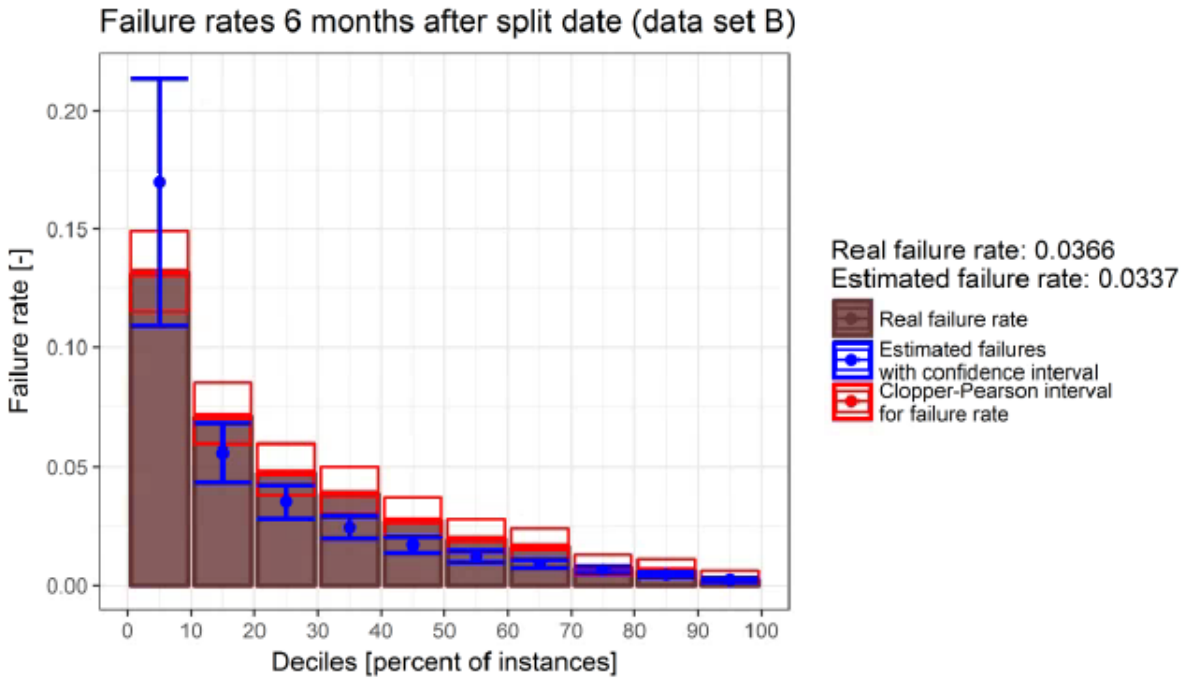

Figure 21: Estimated and real failure rates with confidence intervals for instance survival probability deciles for failure B [44].

A further evaluation of the resulting survival probabilities after six months can be undertaken by using the vehicle-wise survival probabilities for vehicle rank ordering and calculating an *AUL* given the validation labels at the end of the six month validation data set. This *AUL* can be compared to the *AUL* of the initial Cox regression model trained with data up to split date as shown in Tables 13-14. The *c indices* of both datasets are relatively high. In case of failure A, a rank ordering according to survival probabilities $S(t\,|X)$ is preferable to a ranking according to regression-based risk scores $X \cdot \beta$. For dataset B, survival probability-based rank ordering is slightly less performing than risk score-based ordering.

Cox regression (Dataset A)	***AUL*** *(c index = 0.889)*
Risk score-based rank ordering	2.444
Survival probability-based rank ordering	2.555

Table 13: AUL determined with survival probability-based and risk score-based ranking for a validation data set with a cut-off of 6 months after split date for failure A [44].

Cox regression (Dataset B)	***AUL*** *(c index = 0.833)*
Risk score-based rank ordering	2.100
Survival probability-based rank ordering	2.018

Table 14: AUL determined with survival probability-based and risk score-based ranking for a validation data set with a cut-off of 6 months after split date for failure B [44].

9.3.2 Future failure rate prediction

So far, classification and survival models have only been evaluated on data today available. As a matter of fact, today's data only comprises today's failure rate. The failure rate of a component is however potentially increasing over time, making the fitted model inaccurate in predicting the future. I.e. performance metrics as TP, TN, FN, FP or the *AUC* are not correct in the future when not considering an increasing failure rate over time. This drawback causes the future failure rate as well as the future model performance impossible to assess from classifications.

A Breslow estimate of survival probabilities paves the way for calculating such future failure rates as shown above: Given the survival times in training, a baseline survival function is calculated displaying the real failure frequency. The mean failure rate can be calculated as the sum of one minus survival probability. Therefore, a mean future failure rate can be determined for any future date by exploiting the vehicle-wise survival probabilities over time. Setting a survival probability threshold splits the vehicles into failure group and non-failure group whether their survival probability is higher or lower than the threshold. Estimations for confusion matrices, i.e. TP, TN, FN and FP fraction, over threshold can hence be derived from the resulting survival probabilities in the future by calculating the failure rates above and below threshold[17]. Confusion matrices are shown in Figures 22-23 for a survival probability estimate three years after split date.

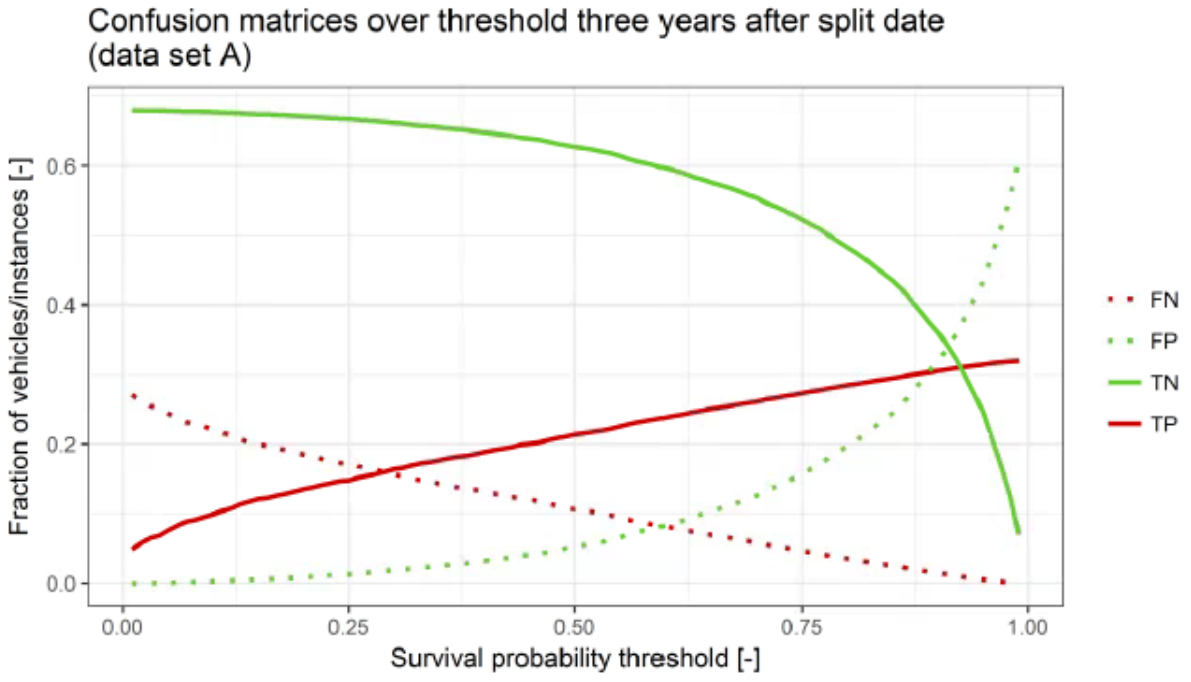

Figure 22: Confusion matrices resulting from average failure rates from survival probabilities three years after split date for failure A [44].

[17] This estimation is a lower bound for model performance as the model tends to perform better when applied multiple times with actualized vehicle data (see section 9.2.2).

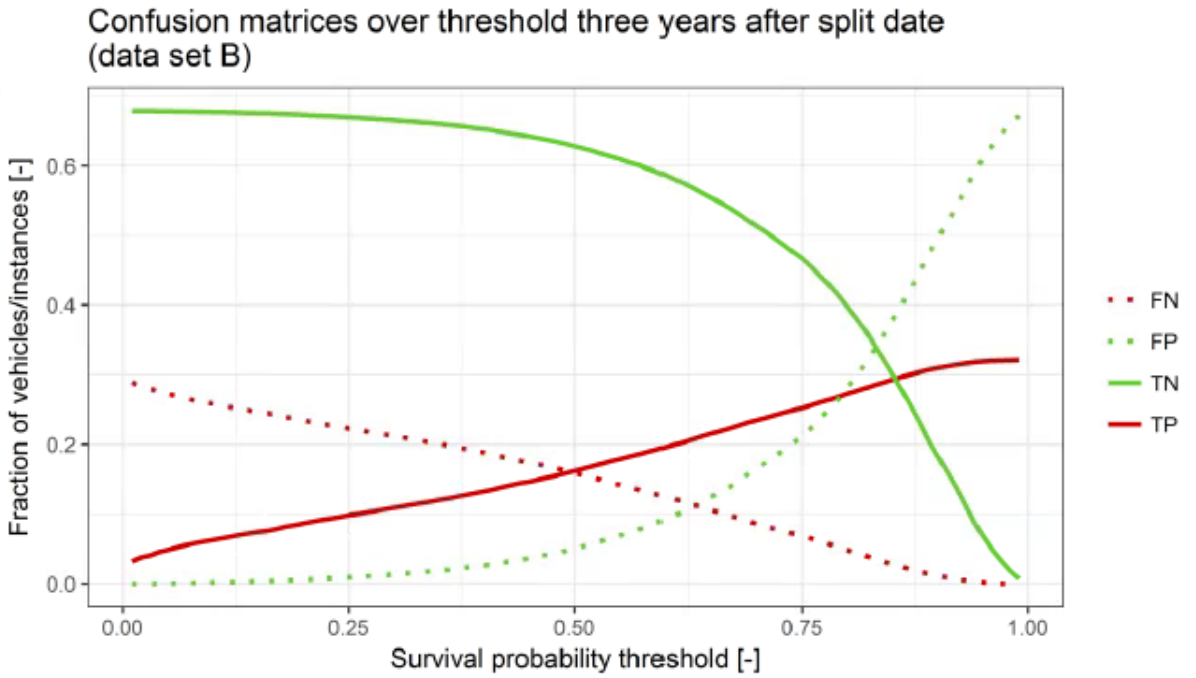

Figure 23: Confusion matrices resulting from average failure rates from survival probabilities three years after split date for failure B [44].

For Poisson binomial distributions, such as the failure rate of vehicles each with a different survival probability, mean and variance can easily be calculated. However, the exact cumulated failure distribution is analytically not definable [43]. The above sampling methodology is in turn used to infer labels for each instance according to their survival probability and therefore derive TP, TN, FN and FP. This then leads to an approximation of future failure frequency distributions. To validate the sampling and resulting failure rates, the TP, TN, FN and FP can be compared to the exact TP, TN, FN and FP calculated by the average over survival probabilities (see Figures 22-23). The sampling size, i.e. the amount of individual samples to generate confusion matrices over threshold, is varied from 1 to 20.000. The results are shown in Figures 24 and 25. The measure of deviation is the threshold-wise difference between exact and sampled TP, TN, FN and FP in percentage of exact TP, TN, FN and FP, respectively. As sampled TP, TN, FN and FP clearly converge to the exact TP, TN, FN and FP, the methodology generates not only valid estimates of TP, TN, FN and FP but also valid failure rates and probability distributions from the sampling. Every sample generates a new guess for a failure rate over threshold which can be accumulated to an overall failure rate probability distribution when sampling size is high. Figures 26 and 27 show the failure distributions in three years for a survival probability threshold of 1, i.e. the FN probability distribution when no vehicles are predictively repaired in the meantime. For both failures, the expected rate of broken components within the amount of today's "healthy" vehicles is around 32%. This is way higher than the so far observed failure rate and in turn makes business cases on today's data impractical and misleading. Cumulative failure distributions as approximation of the respective Poisson binomial distribution can in turn easily be derived.

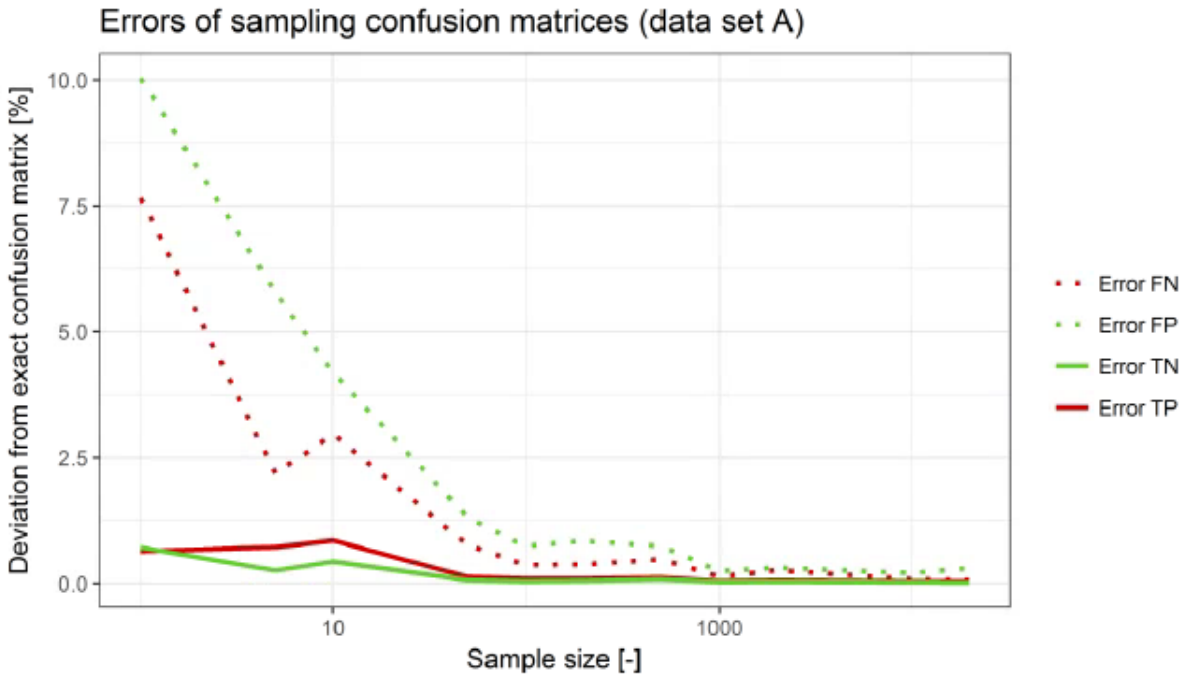

Figure 24: Convergence of confusion matrices over sample size resulting from survival probabilities three years after split date for failure A [44].

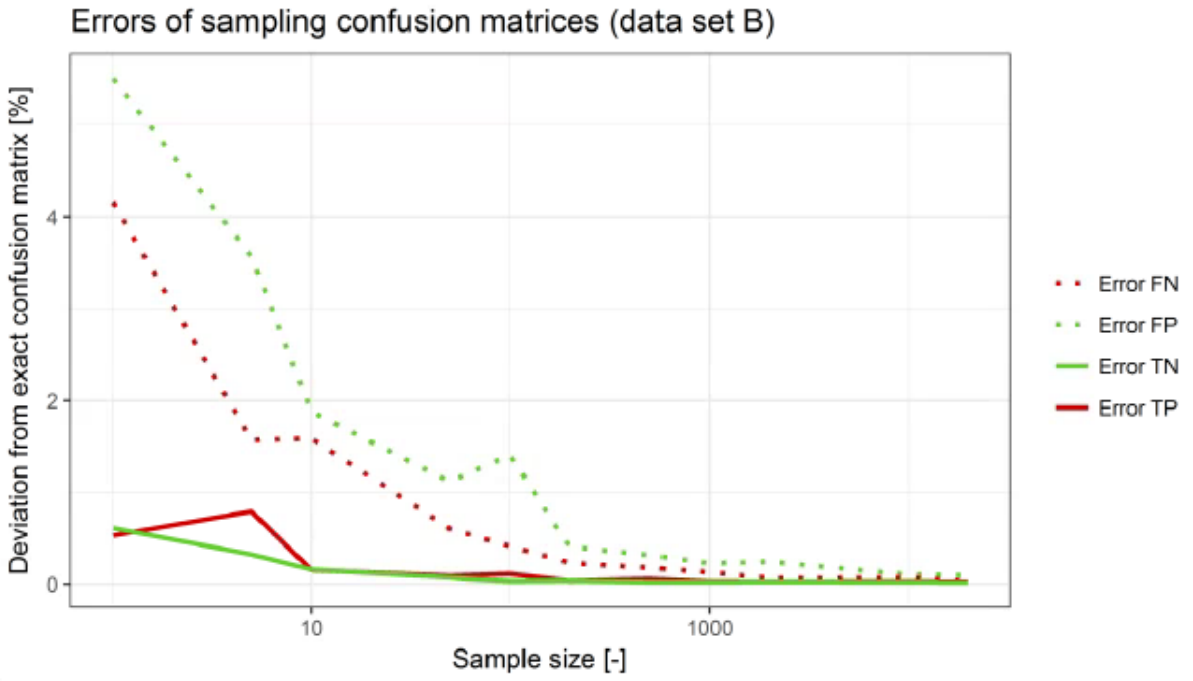

Figure 25: Convergence of confusion matrices over sample size resulting from survival probabilities three years after split date for failure B [44]

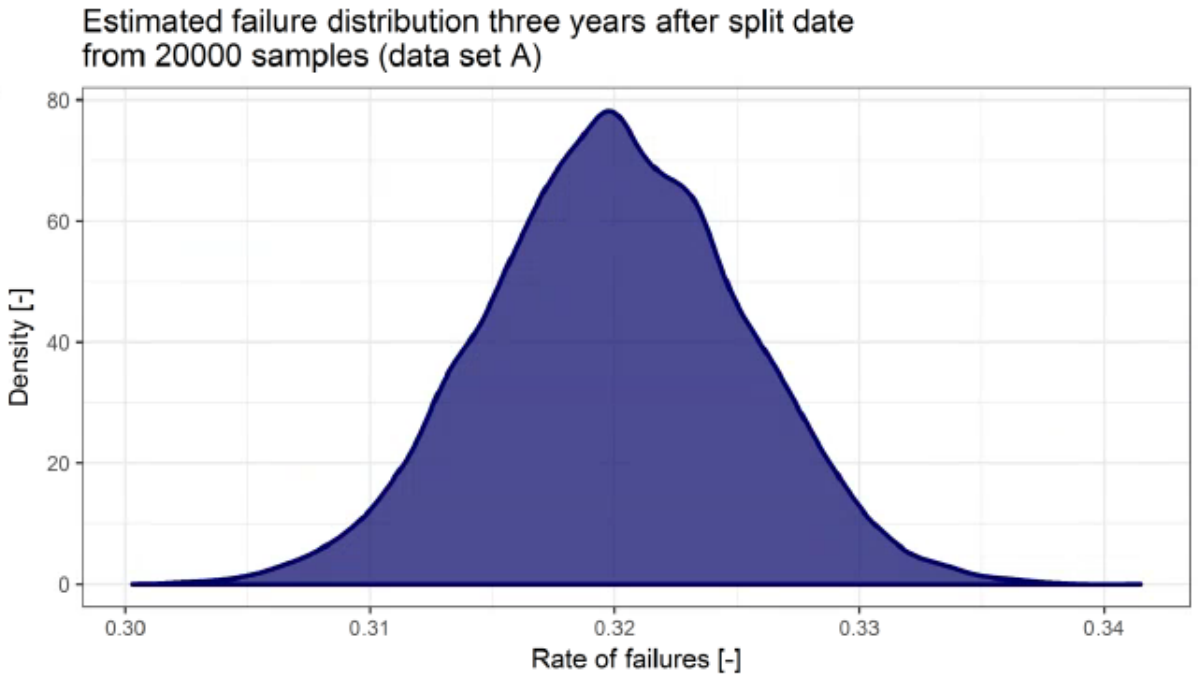

Figure 26: Failure rate probability distribution resulting from sampling 20000 confusion matrices three years after split date for failure A [44].

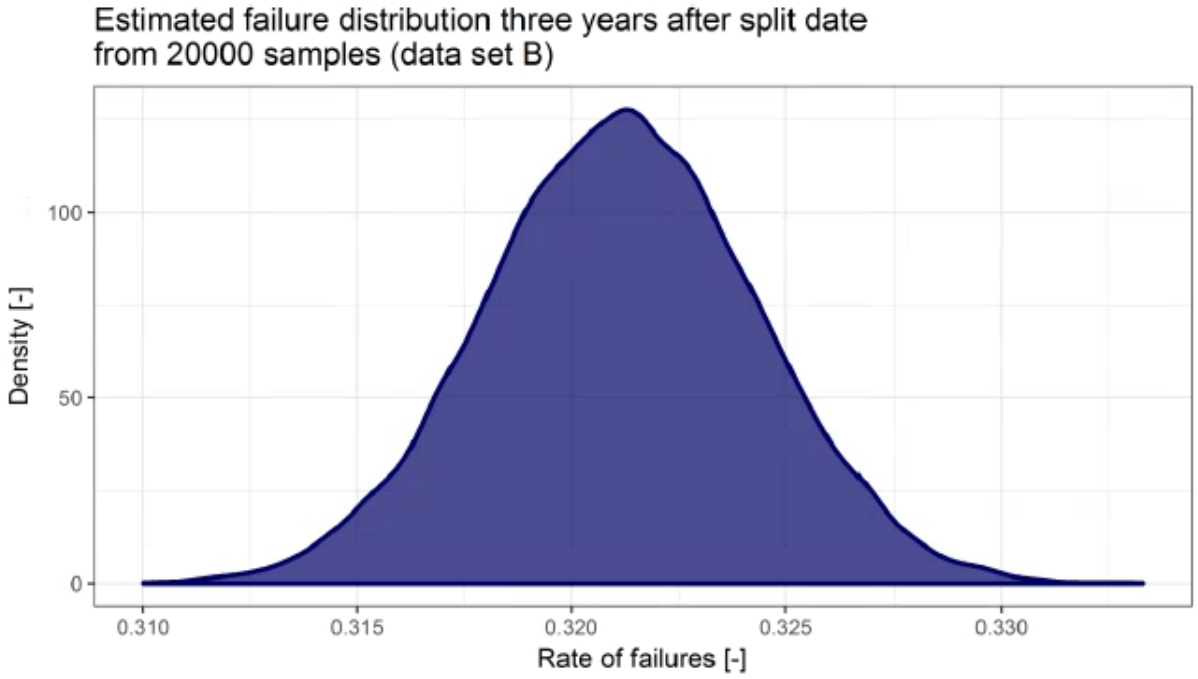

Figure 27: Failure rate probability distribution resulting from sampling 20000 confusion matrices three years after split date for failure B [44].

10 Feature selection with survival models

10.1 Background

The previous analyses assumed a predefined set of features that are characteristic for the failure at hand. The features were defined by expert knowledge on the physical deterioration process as well as many modelling loops to find well performing features for the machine learning models. However, the growing complexity of vehicle sensor systems and recorded data make it increasingly hard to determine the root cause of failures. This becomes crucial for diagnosing vehicle failures at dealer workshops and subsequently predicting them [25]. The challenge of high amounts of possible features contained in a diagnosis is worsened by their heterogeneity over the vehicle and ECU diversity. The result is a high dimensional feature space for a given failure in the field, which needs to be narrowed down for a successful failure prediction. It is therefore indispensable to find highly scalable and adaptable machine learning approaches to replace a manual feature selection process.

Certain conditions arise for a feature selection pipeline. The main goal is to find features that are interpretable and physically correlating to the failure. Main challenges are the multitude of possible features with different distributions, noise and missing values. Spurious correlations between failure and features are not uncommon and are a crucial problem to tackle when automatically selecting features. A final model is desired to utilize only a few features to ensure interpretability and to avoid overfitting. Hence, the multitude of features needs to be heavily reduced. In addition, possible feature interactions should be analyzed, increasing the dimensionality of the feature space further.

Interpretability of the final model is key for a successful PM solution as understanding and justification are mandatory for a model to be utilized. Hence, techniques that alter the feature space by e.g. coordinate transformations (see principal component analyses [48]) are not considered here [49]. An optimal subset of features needs to be found though. The following describes a feature selection methodology able to rank thousands of features according to a measure of importance for modelling accuracy. A forward stepwise model fitting shows that no more than ten features are needed to predict the failures with sufficient accuracy.

10.2 Feature selection pipeline

Finding feature subsets is typically achieved by using model-based filter, wrapper or embedded selection techniques [49]. Filter methods are simple importance measures that evaluate the feature's feasibility without training a model, i.e. prior to seeing any modelling results and therefore independent from a learner. They are fast and scalable and thus a good first step in reducing high dimensional feature spaces. Wrappers train a given model with different features. The better the model performes, the more important the respective feature in the model is. Usually, forward or backward stepwise methods select or deselect features from the overall multitude of possible features. This process is often computationally expensive as many feature combinations have to be tested. Embedded methods rely on inherent model abilities in selecting and ranking features according to importance or influence on the accuracy. A multitude of features can be presented to such models while it selects the most performing ones by its own internal optimization.

This chapter describes a feature selection pipeline designed for high adaptability and scalability for an arbitrary failure at hand. It is based on survival modelling logic, i.e. instance labels are Cox time and Cox label. The pipeline takes a data matrix consisting of instances (diagnosis-based instances with survival time) and all possible features, no matter how much missing values or distribution the feature has. The pipeline (see Figure 28) consists of five layers: A preprocessing layer for preparation, a filter and wrapper layer for feature selection, a layer to find possible interactions between single features and a third feature selection layer with an embedded selection approach. The result is a ranking of most important features according to performance influence. The feature selection pipeline is mainly developed for metric features, but most of the steps are also applicable for categorical features with proper preprocessing.

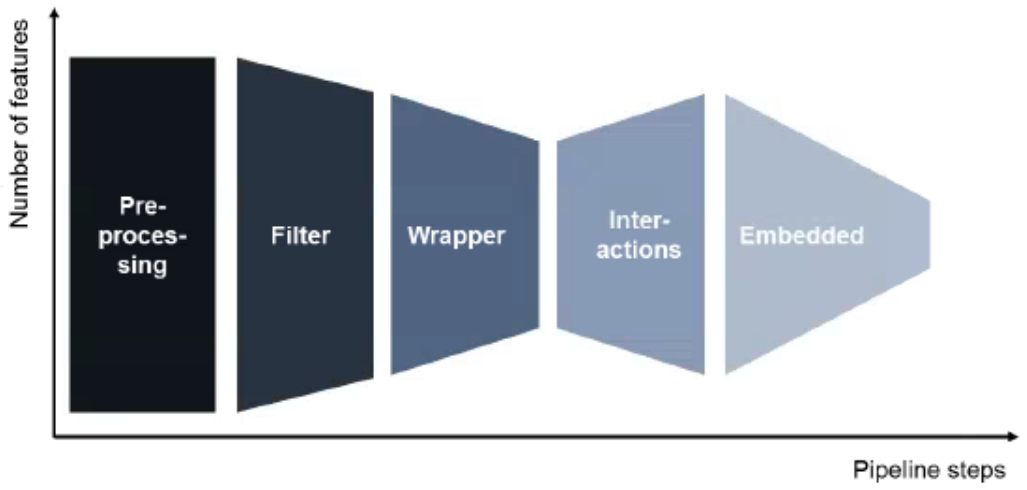

Figure 28: Overview of the feature selection pipeline.

10.2.1 Preprocessing layer

The preprocessing layer basically performs three steps: First, default feature interactions with vehicle age are introduced as explained in the following. Often a specific feature determines a negative influence on the component, e.g. high average ambient temperature. This feature

alone however does not much correlate with survival times as "exposure time" of the component to the damaging influence, i.e. any vehicle age indicator, is essential. Therefore, average temperature times hours of operation of the respective vehicle is a good indication for a deteriorated component and thus a well performing feature later in model training. As parameter times corresponding hours of operation features have already empirically proven to be well performing in the analyses before, all initial metric features, which are per se not already vehicle age-correlated, are by default multiplied with the hours of operation of every instance. This method creates one new feature per initial feature. Thus, the metric feature space is doubled by this step. Other interaction terms, e.g. feature over vehicle age or vehicle age over feature, can be plausible, too. These possibilities were however not further evaluated due to feature times vehicle age interaction results from modelling failure A and B.

In the second step, all features are centered and scaled with their respective mean and standard deviation. Missing values are imputed with the mean over all existing values. There is no threshold for a minimum number of existing observations per features as later on, a variance filter copes with such little information gain features.

In addition, Cox labels and Cox times are automatically calculated for given diagnoses with or without failure. The data, which is initially given by all diagnoses with respective failure label, is converted to a data set valid for all further diagnosis-based analyses as described in section 8.2. Subsequently, the whole data set is divided into train, test and validation set with a Cox time shift of 1. The results of modelling failure A and B showed that a downsampling of censored instances in the training data does not negatively affect the learning of survival models. Hence, for large data sets with many features and instances, a downsampling can be considered to speed up the pipeline, especially in the computationally expensive embedded layer.

10.2.2 Filter layer

The filter layer is used to sort out features completely uncorrelated to the failure and features without or very little information content. An initial filter is a "near zero variance" filter, excluding all features that have very few unique values relative to the number of samples and where the ratio of the frequency of the most common value to the frequency of the second most common value is large [50]. The cut-off for the former condition is ten percent of distinct values out of the number of total samples. The cut-off for the latter condition is a ratio of 95 to 5 of most common value to second most common value. Clearly, features that contain no information, i.e. features with only a single value, and features with a high number of missing values that were imputed and therefore have a lot of equal values afterwards, are excluded.

A second filter is applied based on correlation metrics. From the total data set, a second data set is deduced only containing the non-censored instances, i.e. instances corresponding to vehicles that experienced a failure. At least one of two conditions needs to be met by the features: Either the features have a correlation greater than 0.1 on the complete data set with the Cox label 1, or they have a correlation greater than 0.1 with the Cox time on the reduced data set. This ensures all further used features either to correlate with vehicles with failures in general or to correlate directly with the survival time, i.e. correlate with the physical deterioration process. Features that do not meet either condition are excluded as they show no significant influence or dependence on relevant deterioration characteristics.

10.2.3 Wrapper layer

The subsequent wrapper layer is the second feature reduction layer. It uses a simple wrapper approach to further downsize the feature space. A single survival tree is fitted with every single feature individually. The tree is pruned by default and tested by resampling, i.e. cross-validation. The performance of the single tree on every feature is, in this case measured by the *c index*, recorded and ranked. All features that have a corresponding performance of 0.5, i.e. are not performing any better than random, are excluded. This procedure was performed using *mlr*'s *univariate.model.score* wrapper [51].

10.2.4 Interaction layer

In contrast to the previous pipeline steps, the feature space is expanded in this layer. The aim is to find feature interactions advantageous for prediction. Previously, feature times model time interactions have already been calculated in the preprocessing layer. Now, feature times other feature interactions are determined.

The method to find interactions was designed to find worthwhile interactions between any two features given at this point, i.e. possible interactions with the previous calculated features times vehicle age interactions. The highest feature interaction order evaluated in this approach is thus feature one times feature two times vehicle age squared $X_1 \cdot X_2 \cdot t^2$. The following describes the methodology to find such interactions only for reasonable feature combinations. A brute force approach, i.e. building every possible feature tuple would increase the feature space dimensionality by $0.5 \cdot n \cdot (n-1)$, with n being the number of features at this point in the pipeline. Especially for higher n this is not practical at all. As a consequence, this approach only derives feature cluster interactions, where every cluster is an agglomeration of features containing similar information, i.e. features with high positive correlation[18]. These clusters are found by a hierarchical clustering of features in the transposed data set, i.e. initial features being viewed as instances and initial instances or diagnoses as features. Hence, the features are clustered according to similarity over the instance or diagnosis space. This methodology thus creates clusters of features with similar characteristics over all instances. The data set is here reduced to only non-censored instances to better find clusters that similarly correlate with the survival time. The clustering gives a hierarchical set of stepwise merged clusters which further need to be analyzed to find a valid cut defining the final amount of clusters, described below.

Figure 29 depicts a possible scenario of two diagnoses corresponding to two vehicles and customers. The first vehicle has experienced in average high dynamical driving behavior and high ambient temperature. The second vehicle was in average exposed to high ambient temperatures and low dynamical driving behavior. One can image a component deterioration and finally failure caused by the interaction of high ambient temperature and high dynamics. Then, a feature like high temperature indicator times high dynamics indicator is performing best in a subsequent feature importance ranking. A cluster algorithm would in return find four clusters in the overall amount of features corresponding to the high and low temperature and dynamical driving features. The objective is to automatically find the interaction of a high temperature feature times a high dynamics feature. Given the four found feature clusters, i.e. one representative feature for every cluster, all $4 \cdot 3 / 2 = 6$ possible interactions can be calculated of which one is the desired high temperature times high dynamics interaction. The

[18] A principle component analyses is due to its coordinate transformation not used.

representative is here found by searching for the feature located nearest to the cluster center[19]. When training a model with these 6 new features, clearly the one characterizing the deterioration process ranks highest in importance and can be selected for a final model. Importantly, the dimensionality of features only increased by 6 and not by $16 \cdot 15 / 2 = 120$ in the case of a brute force search with 16 features[20].

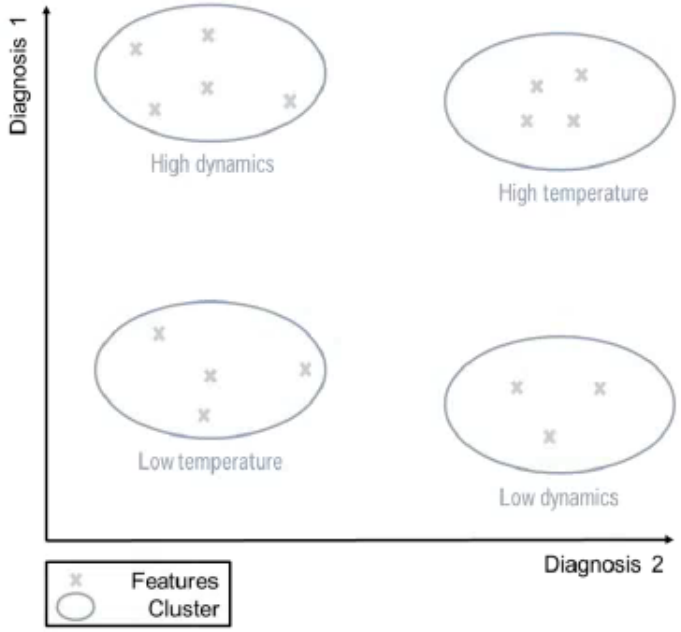

Figure 29: Schematic scenario of 16 features and 4 feature clusters in a two dimensional instance space.

Finding the right amount of cluster is yet crucial in a hierarchical clustering approach. Criteria to find a proper cut in the hierarchical dendrogram created by the clustering, i.e. a point where no further cluster merging is reasonable, are for instance silhouette coefficients. Another metric is introduced by a comparison of the real data set with random data [52]. Here, the merging costs over cluster number of the actual data set are compared to the merging costs of an equivalent dataset with random numbers. This artificial data set is initialized with equal dimensionality, i.e. with the same amount of artificial features and instances. For each instance in this data set, the feature values are randomly sampled from a normal distribution defined by the mean and standard deviation of the respective feature of the actual data set. This creates an equivalent sized and distributed data set, which however does not contain any feasible clusters. The random data set is in turn clustered with the same hierarchical cluster algorithm and merging costs are monitored. One can repeat the procedure to generate average merging costs for random data. When merging costs over cluster number are finally compared, an optimal number of clusters in the actual data set can be found by the following consideration: Merging costs of the random data set are initially higher than the merging costs of the actual data set since real clusters are found and merged in the actual data set. Since these instances or clusters are close together in the feature space, the distance and thus the merging costs of

[19] Here, Euclidian distance measures are applied.

[20] A factorial analysis [52] is another feasible way to find "clusters" of features corresponding to similar characteristics over all instances. However, a factorial analysis would map negatively correlating features together into a factor. As a matter of fact, low and high temperature- as well as low and high dynamics-features naturally correlate negatively as they are inversely proportional. Hence, factorial analysis would result in only two factors, or feature "cluster", corresponding to temperature and dynamics features. Then there is no way in finding the right interactions except for randomly multiplying features from factor one with factor two.

those two clusters is smaller than for merging random numbers. At the point where the costs of merging clusters in the random set get less than in the actual data set, merging is stopped. Here, valid clusters are found and a further merging is "expensive" as the clusters lie relatively far apart in feature space. For random numbers, no such clusters form and thus the merging costs stay relatively small. By this method, a reasonable amount of clusters is found when the algorithm stops merging clusters.

The resulting clusters are used to generate the interactions by first finding one representative feature of each cluster which is closest to the cluster center in instance space. Then, all interactions of these features are calculated by building all possible representative feature tuples. If the amount of clusters is too high to build all possible interactions or one wants to reduce the amount of resulting interactions, certain thresholds can to be defined whether to calculate the interaction of two representative features or not. Possible criteria are e.g. correlation thresholds, where the interaction is only calculated when both features, naturally with low correlation between each other since in different clusters, have a very low correlation smaller than for instance 0.1. Another possible restriction for interaction building could be to only take those clusters with lowest variance. The validity of such methods is not further evaluated in this work as the dimensionality of the data at hand is acceptable.

10.2.5 Embedded layer

The embedded feature selection layer is the fifth and last layer. It assess all features including all interactions resulting from the previous interaction layer. As the amount of features is potentially high, this layer focusses on a feature importance ranking. Models that enable such an embedded feature ranking are e.g. regularized Cox regressions, boosted Cox regressions or boosted survival trees. Random survival forests apparently are a poor choice as their learning performance was significantly worse than for all other models.

A regularized Cox regression tries to shrink the feature coefficient towards zero and leaves only the most important ones unequal to zero [7]. The regularization parameter α determines the way the coefficients are optimized with the penalty term. The regularization in turn resembles a "backward" feature selection, as all features are presented to the model and it inherently reduces the feature space to only a few most important features. The feature importance can then be expressed by the feature coefficients: The higher the coefficient, the more important the feature is[21]. A boosted Cox regression would initialize all feature coefficients with zero and stepwise update these coefficients to values greater than zero [36]. The optimization of a performance measure, e.g. determined by cross-validation, stops after a certain amount of steps where the performance metric cannot be improved anymore. Hence, boosted Cox regressions are similar to "forward" feature selection processes, selecting more and more features for training. The feature importance can then again be read from the feature coefficients after model training. Boosted survival trees also stepwise select single features for building small inaccurate trees [53]. At every split, features are selected from a random subset of all features in order to perform the optimal split on training data. Sequentially many trees are fitted to the residual of the previous trees. The training stops when the performance evaluated by e.g. cross-validation stops to increase. The final model can then be used to get a feature importance measure by checking how often and at what tree depth the features were employed, as shown by Friedman [40]. Features are again "forward" stepwise incorporated into the model.

[21] Assuming equal feature distributions. Thus the previous center and scale preprocessing is mandatory.

This work ranks features according to their importance by implementing a boosted survival tree. It is trained on the training data set with all possible features. Eventually only a few will be included into the model and a feature importance can be derived. Hence the features are not only reduced but also ranked by importance. Optimally, this training is repeated for multiple train-test data splits to remove any noise in selecting optimal features per tree branch resulting from the data split. In addition to the embedded model feature importance ranking, a feature permutation technique is applied to generate a second feature importance ranking: Feature values in the final tree model are swapped with values from a randomly chosen instance. The decrease in model performance averaged over many random value swaps is recorded for every feature. The greater the decrease in model performance averaged over all random swaps, the more dependent the model is on this feature and thus higher the feature importance [54]. The metric was determined by the *permutation importance* function of *mlr* [51]. Both importance metrics, tree based feature importance and permutation importance of said model are eventually normalized to a range of 0 to 1 and feature-wise averaged. This then yields the final feature importance ranking. As validation of the generated ranking, a survival model can be stepwise trained with these features according to importance. The performance on a test and validation data set of the model over feature number is a good indication of feature performance, i.e. generalization and predictive ability. It will be shown that no more than a few of the highest ranked features are necessary to yield similar accuracies than models with the "hand-crafted" features from chapter 9. Again, multiple train-test data splits for stepwise trainings can generate more reliable results.

11 Feature selection results

The results of the feature selection are again based on dataset A and B, this time however not reduced to the three features used before but to all possible metric features [44]. These include for instance, total vehicle mileage and hours of operation. Dataset A also contains histograms for temperature and engine speed encoded in several single features representing each class of the histograms. Dataset B is a more extensive feature space, containing histograms for e.g. ambient temperature, motor intake air temperature, motor start, engine speed, vehicle accelerations, steering wheel angle and gear oil temperature. In addition to these features, features are included that are a priori not useful for predictive analytics, for instance the metrical encoded dealer workshop number or an encoded unique number of the respective diagnosis. Usually, training with such features only leads to the representation of random noise in the data, yet they are not excluded to test the feature selection pipeline to filter them automatically. The following results focus on purely metric features.

The preprocessing layer first performs the automatic feature times vehicle age interaction for which again the hours of operation feature is used. All features are then centered and scaled to a mean of zero and standard deviation of one and imputed with mean values. Both datasets are in turn split into a train, test (70:30) and validation set given the same split dates as above. The pipeline is only employed on the training set, the other two are primarily deprived to test the ranked features in the end. The training dataset is not altered for the two failures at hand, i.e. not down- or upsampled. For higher dimensional data, a downsampling can be performed beforehand the wrapper layer as survival models showed equivalent performance on downsampled datasets A and B.

The filter layer applies the "near zero variance" filter to drop all features with not enough variance. In addition, spearman correlation[22] was used to measure the correlation with the Cox label and the Cox time to further exclude features as explained above. The wrapper layer subsequently trains a single survival tree on each feature for a univariate model score. All features that result in a *c index* of 0.5 or less are filtered.

Within the interaction layer, hierarchical clustering is performed using a Euclidian distance metric in combination with a Ward clustering, i.e. dissimilarities are squared before clusters are merged and updated [55]. The number of final clusters are found by a comparison to an equally spaced and distributed random dataset. The representative feature per cluster was found by a minimum distance to the cluster center, derived by a mean over all features contained in the respective cluster. All interactions of all representative features are calculated here. Any thresholds for such interaction terms, for instance cluster "compactness" or representative feature correlation, are not examined.

[22] In order to account for potentially non-linear relations.

The embedded layer employs a boosted survival tree with a depth of 2, 1000 sequential training steps and a shrinkage parameter of 0.01 using the *gbm* package[23]. The inherent feature importance was fused with the respective model permutation importance.

The results of the pipeline for both failures A and B are depicted in the following. Figures 30-31 show the resulting amount of features over the pipeline steps. Especially layer one, i.e. the "near zero variance" and correlation filter, exclude a majority of features not useful for further prediction. The interaction layer in turn increases the feature space, yet by far not to initial complexity. The disadvantage of dimensionality increase generates in return potentially useful interactions of per se predictive features. The boosted tree then ranks these features while some of them are not even included into the model, shown at step five in the Figures. Dataset A and B start with 74 and 835 features, which are reduced to 36 and 153 after the wrapper layer, respectively. After the interaction layer, the total amount of features is increased to 51 and 363, of which 50 and 249 are contained in the final boosted tree model, respectively.

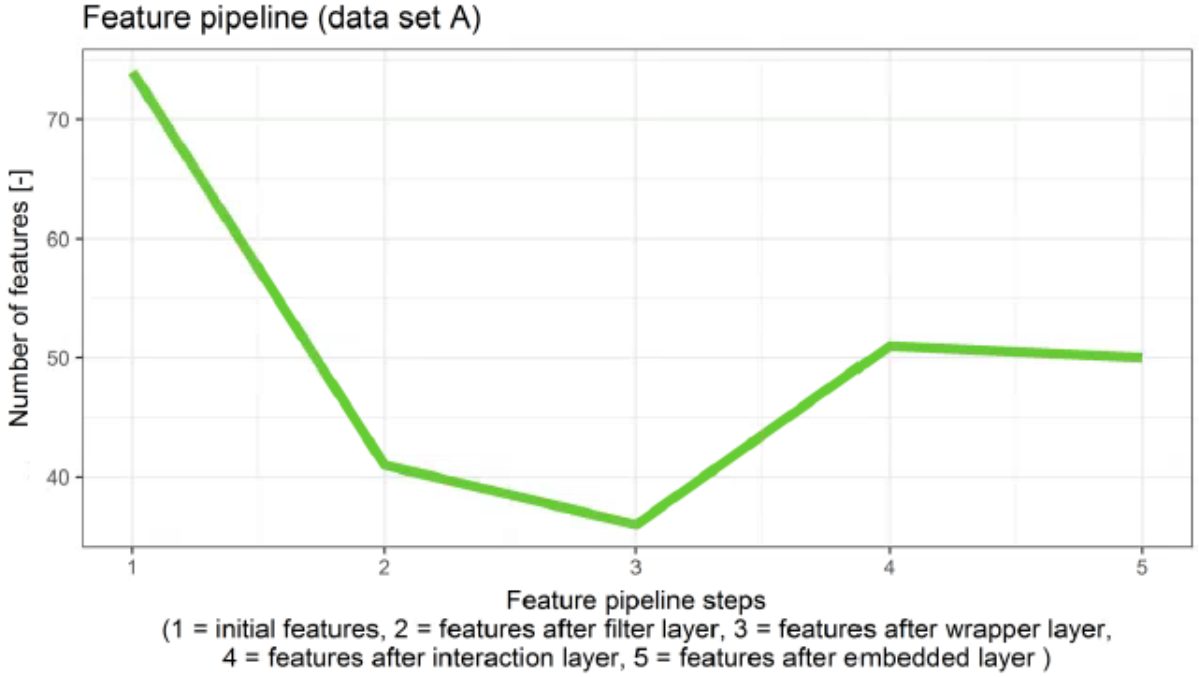

Figure 30: Amount of features over pipeline steps for failure A [44].

[23] These parameter were found to perform well on such data in section 9.2.

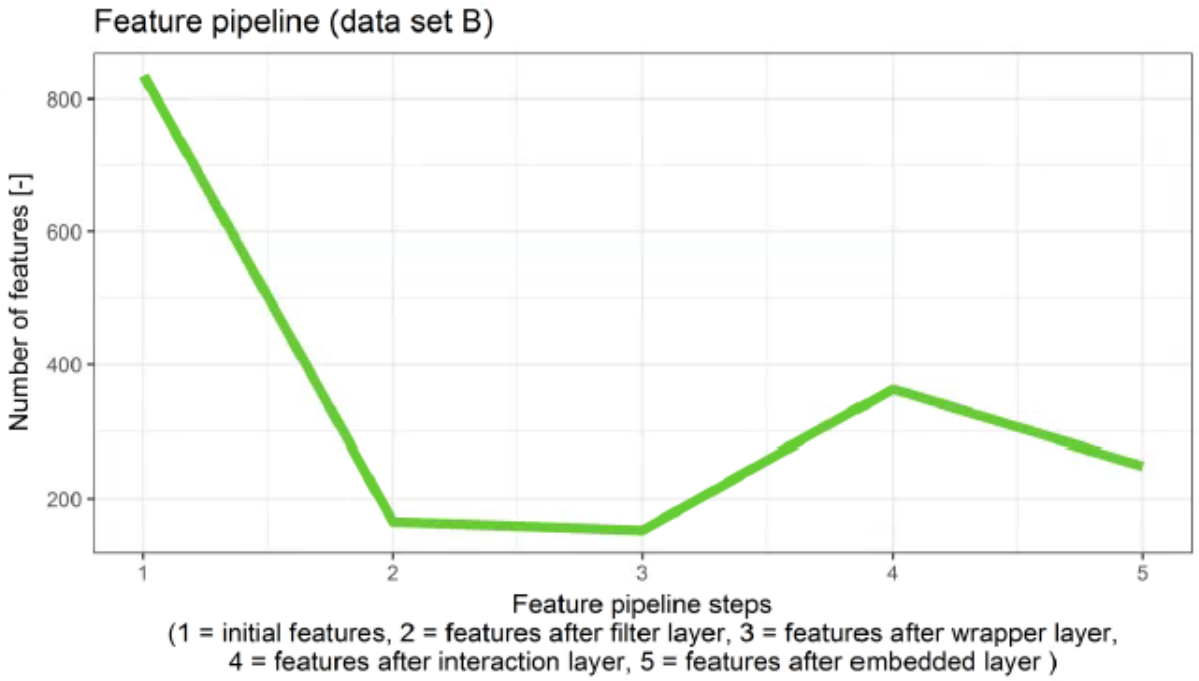

Figure 31: Amount of features over pipeline steps for failure B [44].

The interaction layer strives to find clusters of highly positively correlated features. This can be validated by calculating the average feature to feature Pearson correlation of features within each cluster and comparing it to the average correlation these features have with all other features not included in the respective cluster. The results (see Tables 15-16) show significant higher correlations for the former than the latter relations. Therefore, the hierarchical clustering and cluster number derivation produce clusters of very similar features in regards to the provided instances, i.e. vehicle diagnoses. Yet, some clusters (e.g. cluster 4 and 6 of dataset A or cluster 11 of dataset B) are not necessarily reasonable as their feature correlation is small. Filters in generating clusters and interactions are not evaluated here, however these clusters could be left out for interaction calculations as their "compactness" or "isolation" from other clusters seems small.

Cluster dataset A	**1**	**2**	**3**	**4**	**5**	**6**
Cluster size	7	10	6	5	1	7
Correlation within clusters	0.751	0.846	0.204	0.219	-	0.867
Correlation to features of other clusters	0.401	0.338	0.152	0.226	0.292	0.244

Table 15: Correlations of features within a cluster and with all features outside the respective cluster for failure A [44].

Cluster dataset B	1	2	3	4	5	6	7	8
Cluster size	10	44	8	4	11	5	3	5
Correlation within clusters	0.666	0.893	0.633	0.511	0.532	0.712	0.668	0.752
Correlation to features of other clusters	0.430	0.419	0.390	0.222	0.176	0.423	0.227	0.346
	9	10	11	12	13	14	15	16
Cluster size	2	7	14	3	3	2	2	2
Correlation within clusters	0.732	0.505	0.212	0.721	0.838	0.778	0.763	0.620
Correlation to features of other clusters	0.268	0.214	0.190	0.192	0.206	0.221	0.228	0.105
	17	18	19	20	21			
Cluster size	13	9	1	4	1			
Correlation within clusters	0.815	0.672	-	0.811	-			
Correlation to features of other clusters	0.495	0.445	0.386	0.348	0.160			

Table 16: Correlations of features within a cluster and with all features outside the respective cluster for failure B [44].

The embedded layer results in a ranking of the remaining 50 and 249 features. The ranking is fused and normalized to a scale of 0 to 1[24]. As shown in Figures 32 and 33, only a few features have significant influences in the model to predict the failures A and B. In both cases, more than 80 percent of remaining features have a relative importance of less than 0.1. For failure A, the highest ranked features are:

- Relative time at high ambient temperature times hours of operation
- Relative time at idle speed times hours of operation
- Relative time at idle speed
- Average motor speed
- Relative time at high motor speed times relative time at high ambient temperature times hours of operation

These are mostly equivalent to the features used in chapter 9, which were found completely manual supported by engineering considerations. For failure B, the highest ranked features are:

- Relative time at high motor air intake temperature times hours of operation
- Relative time at high ambient temperature times hours of operation
- Relative time at a specific motor characteristic diagram section times hours of operation
- Relative time at high gearbox oil temperature times hours of operation
- Relative time at medium motor air intake temperature times hours of operation

These features point at a mainly ambient temperature- and motor operation-based deterioration which increases over vehicle age. The modelled effects are physically similar to

[24] This does not imply that feature importances add up to 100%.

the used features above although the features are different. It is noted that these features still can express spurious correlations. Such effects are particularly hard to spot and sometimes the ECU information provides many cases of spurious dependencies.

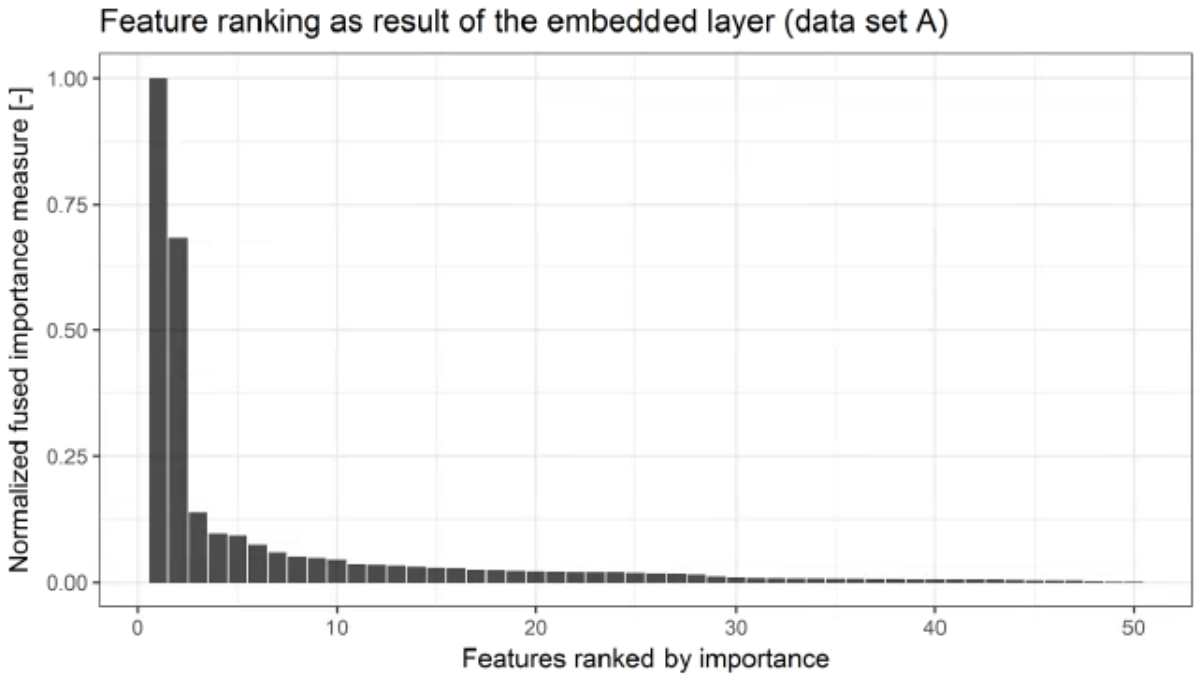

Figure 32: Feature ranking as normalized fused ranking of a boosted survival tree importance and respective permutation importance for failure A [44]

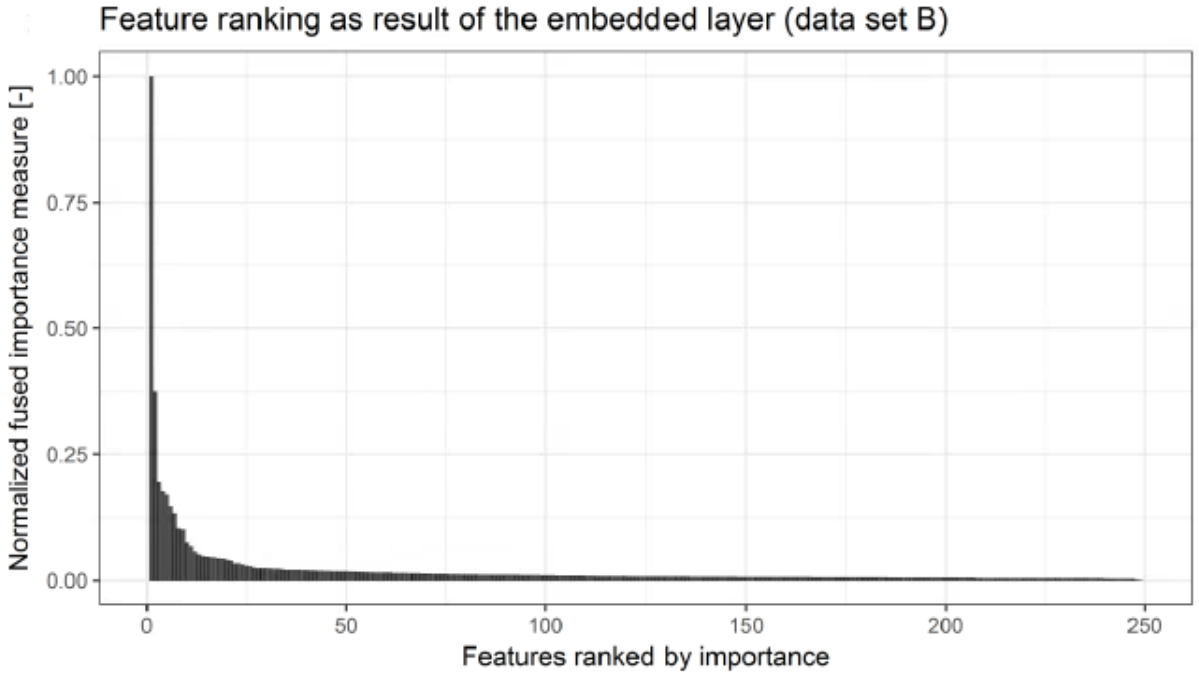

Figure 33: Feature ranking as normalized fused ranking of a boosted survival tree importance and respective permutation importance for failure B [44]

To prove the validity of these feature rankings, a stepwise survival model training is performed. Thus, a regularized Cox regression is trained stepwise with the highest ranked 1 to 40 features (*glmnet* toolbox). The model performances, i.e. *c index* on the test and *AUL* on the validation set, are depicted in Figures 34-35 averaged over ten random 70:30 train-test set splits. For comparison, *AUC* and *AUL* of a regularized logistic regression are computed as well (*h2o* toolbox). Both models are ridge regularized ($\alpha = 0$) so that features are not completely excluded from the model as in the case of a lasso regularization. This ensures that stepwise additional features are influencing the model performance and do not have negligible coefficients.

The results show high model performances in regards to both *c index* and *AUL*. For two model parameters, the performances are close to the performances yielded with "hand-crafted" features (compare section 9.2). For additional included features, the performance stays relatively constant. This shows on the one hand, that the automatically ranked features are able to predict the failure as well as manually selected features. On the other hand, as performance does not decrease over number of features, no apparent overfitting takes place. The non-decreasing behavior is probably also caused by correlated features in the ranking: The effect of correlated features is spread upon them while performance is neither significantly increasing nor decreased[25]. For a final feature choice, one can for instance only include highly uncorrelated features into the model. As a conclusion, all features are quite feasible for prediction but also correlated and are not used to "learn" noise in the data.

A second characteristic of the depicted curves is the similar behavior of *c index* and *AUL* over number of features. This can be seen as further validation of the *AUL* methodology as both performance measures similarly indicate good model fit in a parallel way.

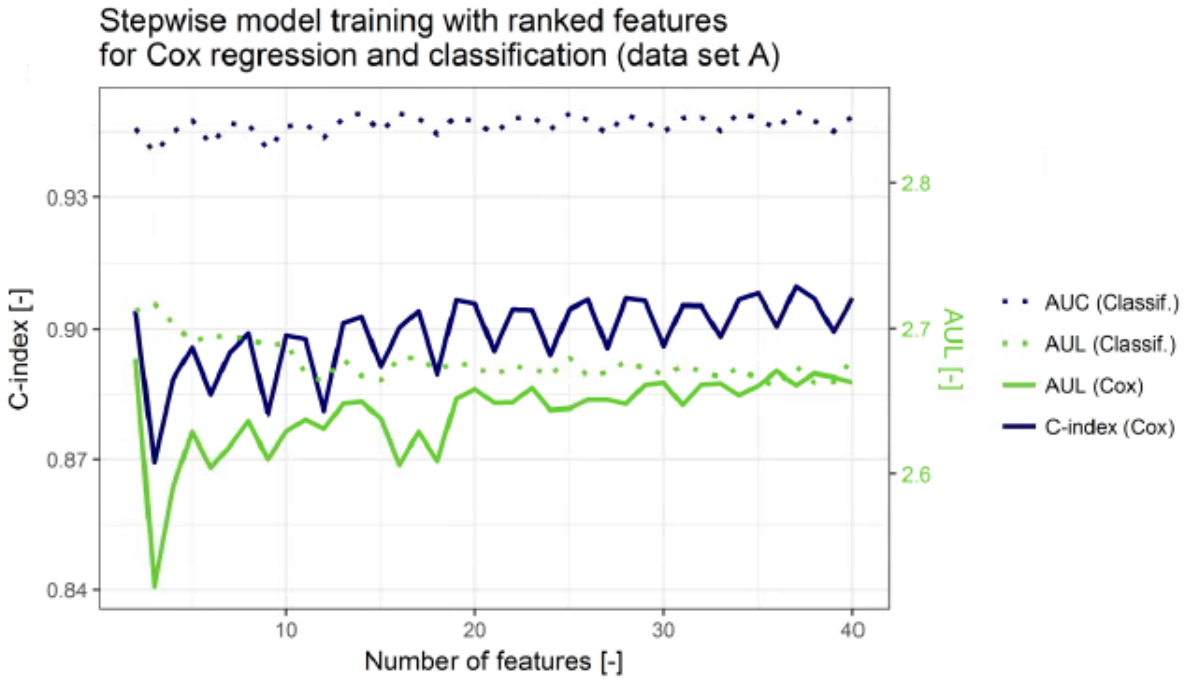

Figure 34: Stepwise training of a regularized Cox regression and a logistic regression on the ranked features averaged over ten random train-test splits for failure A [44].

[25] The regularization of both models does not suppress the effect of correlated features.

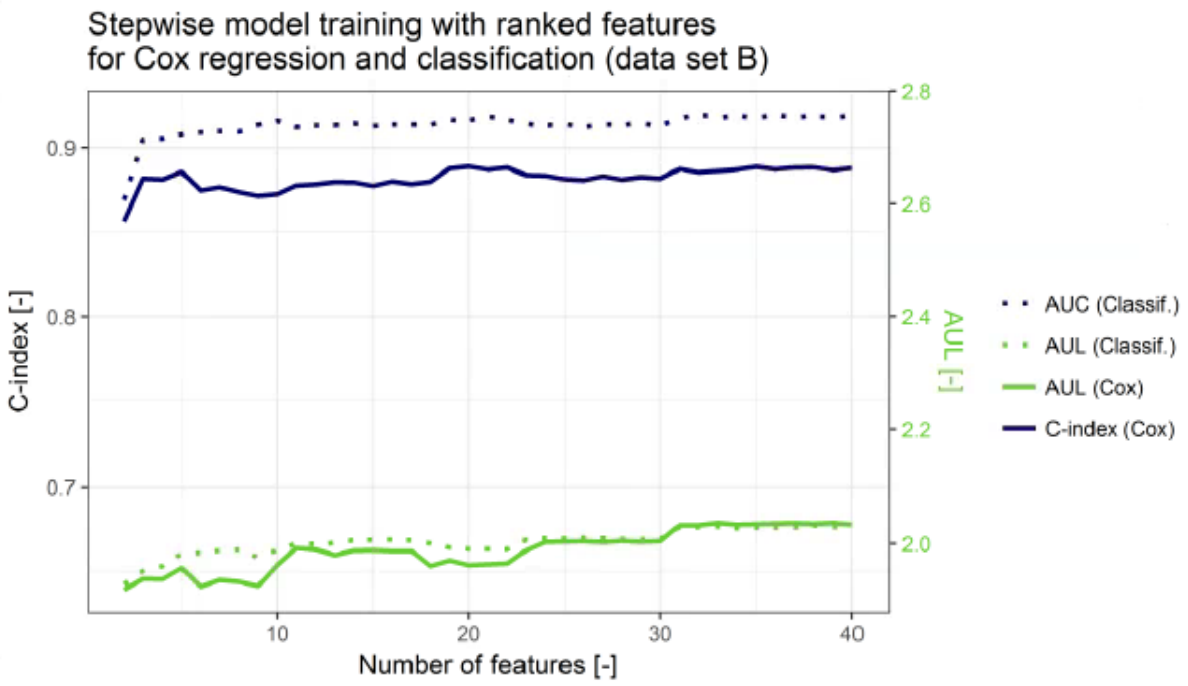

Figure 35: Stepwise training of a regularized Cox regression and a logistic regression on the ranked features averaged over ten random train-test splits for failure B [44].

12 Summary and discussion

Survival models were introduced in the context of vehicle predictive maintenance in this work. The goal was to create a methodological framework to train and test survival models on state-of-the-art car data from dealer workshops. In addition, survival models were compared to basic binary classification models. While both approaches are able to discriminate vehicle components according to risk, survival models have the ability of inferring real failure probabilities over time.

Real customer fleet data poses challenges to predictive analytics solutions. First and foremost, this work aims to predict failures of components that were ex ante not considered to break down in the field. Hence, the components are often not directly monitored by sensor systems tailored for a broad data mining approach. Then, sensors and ECUs in vehicles generate highly heterogeneous data being stored in accumulated form due to the storage limitations directly in the ECUs. Finally, the resulting data of ECUs cannot be retrieved in high frequency from the vehicle, but only "diagnosed" at dealer workshops. The presented methodology copes with these difficulties, just provided that the failure is caused by some sort of long term deterioration and not by random or a short term influence. Otherwise, no characteristic failure patterns can be found and modeled with the given data. It will be subject of further research to enable predictive solutions given higher frequent vehicle monitoring. However, requirements for sensors and ECUs better suited for predictive analytics cannot be implemented in short term due to long development cycles of vehicles.

An automatic and scalable feature selection pipeline is introduced to find significant influences on the wear process of failures in the customer fleet. It is designed to require no a priori physical understanding of the failure at all. Feature selection can support a manual root cause analysis in turn. The pipeline consists of five layers in order to reduce the multitude of features to a few ranked by importance for further predicting the failure. After general preprocessing steps to tailor the data to the diagnosis-based instance scheme proposed here, several simple filter and wrapper methods reduce the feature amount significantly. Still, the dimensionality is again increased by feature interactions determined by a hierarchical feature clustering approach. The results show a high positive correlation of the features within found clusters, pointing out very similar information in regards to predicting the failure at hand. However the best performing features were simple feature times vehicle age interactions. The additional search for reasonable feature interactions was hence dispensable for both analyzed failures. Yet, the approach might proof valuable for other components and failures caused by specific deteriorating influences not displayed in regular features. By doing so, the search space for the subsequent tree-based feature ranking is increased while at the same time the additional amount of features does not hinder the feature selection by the tree as the selection ensures relatively few new interaction terms.

A machine learning model can in turn be trained on historical data, given a defined set of features, either manually selected or the top ranked ones from feature selection. The scope of this work was to compare several survival models and binary classification. *C index* and *AUC* on a common train-test data split were used as a first evaluation. In order to really assure the predictive ability of a final model, a second metric, the *AUL*, was introduced in addition to measure the performance of predicting real failures in the future. Cox regression, boosted Cox regression, boosted survival trees, random survival forests and logistic regression were compared. They all yield similar results, although the simpler models, i.e. Cox regression and boosted survival trees, were preferred for following analyses. Most importantly, results show that survival models are equally capable of vehicle discrimination according to risk score as classification methods. Based on the discrete data, the predictive ability of survival models is assumed not to unleash its full potential. In this context, a "time-continuous" survival model has no advantageous compared to a "time-discrete" binary classification. Also, approaches like down- or upsampling of the highly unbalanced amount of censored and uncensored instances could not significantly improve results. It was shown, that models tend to loose predictive ability the further one wants to predict failures in the future given today's data. While some hyperparameters could be optimized by cross-validation on the training data, others were manually evaluated. No brute force search was undertaken for such model parameters so that the general behavior for different parameter combinations were determined but not overall optimized.

For both datasets, the automatically found features were stepwise used to predict the failure with previously found exogenic parameters. Very good results in terms of predictive ability were in general observed. A few features are already sufficient and more would unnecessarily increase model complexity but can also lead to overfitting. An AIC criteria [56] can be introduced in the future to evaluate how many features are reasonable given a feature ranking. Importantly, the model performance with features found by the automatic feature selection is equally high as with manual features selected by a priori physical expert knowledge. This proves the validity of the proposed feature selection methodology.

State-of-the-art classification methods order vehicles according to their pseudo risk of failure in order to design a predictive recall for a specific failure in the field. An artificial threshold needs to be set to decide which vehicles are predictively repaired and which not. While binary classification cannot be further extended to infer real failure probabilities, survival models can incorporate vehicle-wise failure probabilities very well. A Breslow estimate of a Cox regression was employed to predict such survival probabilities in this work. These probabilities can in turn be used to predict average failure rates for any given date in the future. A sampling approach is proposed to approximate the Poisson binomial distribution of failures which was shown to converge within less than 50,000 samples to sufficient accuracy. The approach is highly beneficial for PM projects and paves the way for more realistic failure assessments.

13 Conclusion and outlook

This work describes a general methodology for survival models to enable vehicle PM. The scalable and adaptable approach is tailored to predict arbitrary failures as long as patterns describing the deterioration process are given by historical data. Data retrieval from the whole vehicle fleet cannot be realized in high volume and frequency over e.g. telematics. Thus, state-of the art information source is data from vehicle dealer workshop visits. This time discrete and highly aggregated data originally stored in ECUs is a main difficulty to tackle. Common PM solutions based on continuous monitoring, like e.g. vibration monitoring, cannot be used in turn. As failures still underlie a timely deterioration process and do not appear random, survival models were used to learn and predict failure influences and resulting risks of failure by defining each vehicle diagnosis as instance, i.e. as "patient". To enable such a supervised learning on historical fleet data, both vehicles with and without failure needed to be coped with in training and testing the machine learning models. Survival models were trained on the survival times of provided diagnoses identifying characteristic influences, i.e. features, on two independent failures of not directly sensorially monitored components. The result is on the one hand a feature-based vehicle discrimination according to risk as well as on the other hand a prediction of survival times, i.e. survival probabilities in the future.

The new approach especially proves effective, as often ex ante unforeseen failures lead to worldwide recalls for a specific vehicle production series containing the respective component. Such recalls can in turn be limited to only the vehicles at high risk by a machine learning solution. Recently, binary classification methods were used to assign each vehicle a pseudo risk by training on failures and "healthy" components. As a matter of fact, classifications infer pseudo risks of belonging to "failure class" as a result of their binary design and yet no real failure probabilities. Survival models in turn overcome a crucial disadvantage: The new methodology yields vehicle rankings according to risk of failure as well as inference of real failure probabilities over time in the future. This leads to significant advantages, for instance failure rate estimations for future dates and the inherent depiction of true failure frequency over vehicle age distributions.

New automotive developments are committed to more connectivity between cars, customers and producer backend. Vehicle PM can then be significantly boosted by higher frequent data on the "health state" of vehicles. Future research will assess the possibilities of such developments. An interesting use case will be the calculation of risks of failure directly in the car. Future research will examine algorithms that can directly access and parse ECU data to determine a failure probability at any desired time without the detour of discrete and aggregated data. Therefore, PM algorithms for any component have to be transferred "over the air" into the vehicle to evaluate the component's risk of failure. Until then, the presented approach leverages the existing vehicle technology and data for optimal PM solutions tailored to maximum customer benefit.

14 List of figures

15 List of tables

16 References

[1] H. Krüger, "BMW Pressekonferenz," 2016. [Online]. Available: https://www.bmwgroup.com/content/dam/bmw-group-websites/bmwgroup_com/ir/downloads/de/2015/Reden_Kueger_Eichiner_AIK_2016.pdf. [Accessed: 20-Sep-2017].

[2] R. K. Mobley, *An Introduction to Predictive Maintenance (Second Edition)*. Butterworth-Heinemann, 2002.

[3] K. Reif, *Automobilelektronik*. Springer, 2007.

[4] "Gesetz über die Haftung für fehlerhafte Produkte (Bundesministerium der Justiz und für Verbraucherschutz)." [Online]. Available: http://www.gesetze-im-internet.de/prodhaftg/index.html#BJNR021980989BJNE000201311. [Accessed: 29-Sep-2017].

[5] C. M. Bishop, *Pattern Recognition And Machine Learning*. 2006.

[6] W. Weibull, "A statistical distribution function of wide applicability," *Journal of applied mechanics*, vol. 18. pp. 293–297, 1951.

[7] F. E. Harrell, *Regression modeling strategies. With applications to linear models, logistic regression, and survival analysis*. Springer, 2001.

[8] S. Orhan, N. Aktürk, and V. Celik, "Vibration monitoring for defect diagnosis of rolling element bearings as a predictive maintenance tool: Comprehensive case studies," *NDT and E International*, vol. 39, no. 4, pp. 293–298, 2006.

[9] M. C. Garcia, M. A. Sanz-Bobi, and J. del Pico, "SIMAP: Intelligent System for Predictive Maintenance. Application to the health condition monitoring of a windturbine gearbox," *Computers in Industry*, vol. 57, no. 6, pp. 552–568, 2006.

[10] J. K. Jones and J. White, "Method and apparatus for predictive maintenance of HVACR systems." US Patent 5596507, 1997.

[11] K. A. Kaiser and N. Z. Gebraeel, "Sensor-Based Degradation Models," vol. 39, no. 4, pp. 840–849, 2009.

[12] X. S. Si, W. Wang, C. H. Hu, and D. H. Zhou, "Remaining useful life estimation - A review on the statistical data driven approaches," *European Journal of Operational Research*, vol. 213, no. 1, pp. 1–14, 2011.

[13] X. Zhao, M. Fouladirad, C. Bérenguer, and L. Bordes, "Condition-based inspection/replacement policies for non-monotone deteriorating systems with environmental covariates," *Reliability Engineering and System Safety*, vol. 95, no. 8, pp. 921–934, 2010.

[14] J. Z. Sikorska, M. Hodkiewicz, and L. Ma, "Prognostic modelling options for remaining useful life estimation by industry," *Mechanical Systems and Signal Processing*, vol. 25, no. 5, pp. 1803–1836, 2011.

[15] L. Liao, "Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems , and an Application to Battery Life Prediction," vol. 63, no. 1, pp. 191–207, 2014.

[16] A. K. Choudhary, J. A. Harding, and M. K. Tiwari, "Data mining in manufacturing: A review based on the kind of knowledge," *Journal of Intelligent Manufacturing*, vol. 20, no. 5, pp. 501–521, 2009.

[17] Haitao Liao, Wenbiao Zhao, and Huairui Guo, "Predicting remaining useful life of an individual unit using proportional hazards model and logistic regression model," *RAMS '06. Annual Reliability and Maintainability Symposium, 2006.*, vol. 0, no. C, pp. 127–132, 2006.
[18] D. R. Cox, "Regression models and life tables," *Journal of the Royal Statistical Society. Series B:*, vol. 34, no. 2, pp. 187–220, 1972.
[19] J. Elith, J. R. Leathwick, and T. Hastie, "A working guide to boosted regression trees," *Journal of Animal Ecology*, vol. 77, no. 4, pp. 802–813, 2008.
[20] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics and Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.
[21] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for Cox's proportional hazards model via coordinate descent," *Journal of Statistical Software*, vol. 39, no. 5, pp. 1–13, 2011.
[22] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests," *Annals of Applied Statistics*, vol. 2, no. 3, pp. 841–860, 2008.
[23] R. Prytz, S. Nowaczyk, T. Rögnvaldsson, and S. Byttner, "Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data," *Engineering Applications of Artificial Intelligence*, vol. 41, no. May, pp. 139–150, 2015.
[24] S. Voronov, D. Jung, and E. Frisk, "Heavy-duty truck battery failure prognostics using random survival forests," *IFAC-PapersOnLine*, vol. 49, no. 11, pp. 562–569, 2016.
[25] B. Schlegel and B. Sick, "Design and optimization of an autonomous feature selection pipeline for high dimensional, heterogeneous feature spaces," *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, no. Section IV, 2017.
[26] G. James, D. Witten, T. Hastie, and R. Tibishirani, *An Introduction to Statistical Learning*. Springer, 2013.
[27] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data*, vol. 5. John Wiley and Sons New York, 1980.
[28] J. D. Kalbfleisch and R. L. Prentice, "Marginal likelihoods based on Cox's regression and life model," *Biometrika*, vol. 60, no. 2, pp. 267–278, 1973.
[29] N. Breslow, "Covariance Analysis of Censored Survival Data," *Biometrics*, vol. 30, no. 1, p. 89, 1974.
[30] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization Paths for Generalized Linear Models via Coordinate Descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–24, 2010.
[31] R. E. Schapire, "The Boosting Approach to Machine Learning: An Overview," in *Nonlinear Estimation and Classification*, D. D. Denison, M. H. Hansen, C. C. Holmes, B. Mallick, and B. Yu, Eds. New York, NY: Springer New York, 2003, pp. 149–171.
[32] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
[33] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297–336, 1999.
[34] R. De Bin, "Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages CoxBoost and mboost," *Computational Statistics*, vol. 31, no. 2, pp. 513–531, Jun. 2016.
[35] P. Buhlmann and T. Hothorn, "Boosting Algorithms: Regularization, Prediction and Model Fitting," *Statistical Science*, vol. 22, no. 4, pp. 477–505, 2007.
[36] G. Tutz and H. Binder, "Boosting ridge regression," *Computational Statistics and Data Analysis*, vol. 51, no. 12, pp. 6044–6059, 2007.
[37] H. Binder, A. Benner, L. Bullinger, and M. Schumacher, "Tailoring sparse multivariable regression techniques for prognostic single-nucleotide polymorphism signatures," *Statistics in Medicine*, vol. 32, no. 10, pp. 1778–1791, 2013.
[38] P. Bühlmann and B. Yu, "Boosting With the L2 Loss: Regression and Classification," *Journal of the American Statistical Association*, vol. 98, no. 462, pp. 324–339, 2003.
[39] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[40] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals*

of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.
[41] J. Hastie, Trevor, Tibshirani, Robert, Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2009.
[42] A. J. Hanley and J. B. McNeil, "The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve," *Radiology*, vol. 143, pp. 29–36, 1982.
[43] Y. Hong, "On computing the distribution function for the Poisson binomial distribution," *Computational Statistics and Data Analysis*, vol. 59, no. 1, pp. 41–51, 2013.
[44] "BMW Group Munich, BA-74; BMW-internal data base." 2017.
[45] S. Aiello, T. Kraljevic, and P. Maj, "Package 'h2o,'" *Cran*, 2016.
[46] T. M. Therneau, "A Package for Survival Analysis in S.," *Survival*. 2015.
[47] C. J. Clopper and E. S. Pearson, "The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial," *Biometrika*, vol. 26, no. 4, p. 404, 1934.
[48] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. 2001.
[49] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19. pp. 2507–2517, 2007.
[50] M. Kuhn, J. Wing, S. Weston, A. Williams, C. Keefer, and A. Engelhardt, "Caret: Classification and Regression Training," *R package version 6.0-64*. 2012.
[51] B. Bischl *et al.*, "mlr: Machine Learning in R," *Journal of Machine Learning Research*, vol. 17, pp. 1–5, 2016.
[52] J. Haderlein, "Cluster analysis of customer driving patterns for car efficiency optimization," in *4th International conference on Energy Efficient Vehicles (ICEEV 2015)*, 2015.
[53] G. Ridgeway, "Generalized Boosted Models: A guide to the gbm package," *Compute*, vol. 1, no. 4, pp. 1–12, 2007.
[54] D. Lenz, F. Diehl, M. Truong Le, and A. Knoll, "Deep neural networks for Markovian interactive scene prediction in highway scenarios," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*, 2017, pp. 685–692.
[55] F. Murtagh, "Multidimensional clustering algorithms," *COMPSTAT Lectures 4. Lectures in Computational Statistics*. pp. 8–130, 1985.
[56] H. Akaike, "Information theory and an extensión of the maximum likelihood principle," *International symposium on information theory*, no. 1973, pp. 267–281, 1973.