Predictive Maintenance of Lead-Acid Batteries with Sparse Vehicle Operational Data

Sergii Voronov[1], Mattias Krysander[2], and Erik Frisk[3]

[1,2] *Department of Electrical Engineering, Linköping university, Linköping, S-581 83, Sweden*
sergii.voronov@liu.se
mattias.krysander@liu.se
erik.frisk@liu.se

ABSTRACT

Predictive maintenance aims to predict failures in components of a system, a heavy-duty vehicle in this work, and do maintenance before any actual fault occurs. Predictive maintenance is increasingly important in the automotive industry due to the development of new services and autonomous vehicles with no driver who can notice first signs of a component problem. The lead-acid battery in a heavy vehicle is mostly used during engine starts, but also for heating and cooling the cockpit, and is an important part of the electrical system that is essential for reliable operation. This paper develops and evaluates two machine-learning based methods for battery prognostics, one based on Long Short-Term Memory (LSTM) neural networks and one on Random Survival Forest (RSF). The objective is to estimate time of battery failure based on sparse and non-equidistant vehicle operational data, obtained from workshop visits or over-the-air readouts. The dataset has three characteristics: 1) no sensor measurements are directly related to battery health, 2) the number of data readouts vary from one vehicle to another, and 3) readouts are collected at different time periods. Missing data is common and is addressed by comparing different imputation techniques. RSF- and LSTM-based models are proposed and evaluated for the case of sparse multiple-readouts. How to measure model performance is discussed and how the amount of vehicle information influences performance.

1. INTRODUCTION

Nowadays, many automotive companies not only sell vehicles with guarantee periods for its components, but offer condition-based maintenance as a service to customers as well. It is possible to avoid the unexpected failures of the components by estimating failure date based on vehicle operation data and scheduling maintenance before the problem becomes critical. Lead-acid batteries are part of the electrical system in any vehicle and in particular in heavy-duty vehicles. Unexpected failures in a component may lead to vehicle unavailability that are costly. Developing predictive models for flexible maintenance of lead-acid batteries in the presence of irregular and quantitatively varying heavy-duty vehicle operational data is the topic of this study.

Heavy-duty vehicles are being operated in diverse conditions that include different climate, quality of roads, different number starts and stops, etc. Moreover, there are many lead-acid battery manufacturers on the market, therefore it is possible that a predictive model has to deal with different degradation profiles that are inherited from a particular battery brand. Detailed physical models of battery degradation are inherently difficult and require sensing battery operational characteristics such as voltage and current which is not always available. An alternative is to use data that describe how a vehicle is operated to build a predictive model. This approach is adopted here which relies on operational data readouts from a vehicle's visits to a workshop, data that is here provided by our industrial partner Scania CV.

In prognostics, a common approach is to estimate the remaining useful life (RUL), which is the remaining time until component failure. In general, RUL estimates rely on information from sensors that can be used to create a health indicator and track its state during the component's lifetime, see, e.g., (Saha et al., 2009; Alghassi et al., 2015). Physics, or model-based, methods for RUL estimation rely on models for component degradation and if the models are accurate, physics-based methods have the potential to provide accurate estimates of RUL. Authors in (Daigle & Goebel, 2011; Hanachi et al., 2015) elaborate how model-based methods can be created for predicting failures in pneumatic valves and gas turbine engines. However, obtaining accurate degradation models is often hard and costly due to lack of sensor information or complex degradation processes. Data-driven methods use machine learning algorithms and statistical methods

to estimate RUL. Here, methods with underlying assumptions on the failure time distributions, e.g., (D. Cox, 1972), and techniques with no assumptions (Ishwaran et al., 2008; Elsheikh et al., 2019; Zhang et al., 2018) are investigated. In hybrid methods predictions from the models, model-based and data-driven, are combined to receive an estimate of failure time. These approaches has the potential to overcome single method weaknesses and make a predictive model stronger (Ahmad et al., 2017).

The paper has the following contributions: 1) prognostic models are developed to estimate reliability and lifetime functions in the presence of sparse vehicle operational data, 2) performance comparison between LSTM network, feed forward neural networks, and RSF models is presented, 3) different imputation strategies for missing data are evaluated with an extension to the case of multiple data readouts per vehicle and 4) an analysis of a connection between performance and the amount of data per vehicle is conducted.

The paper has the following structure. Motivation of the given work together with a statement of the considered problem is given in Section 2. Section 3 gives an overview of the theoretical concepts that are used for the development of the predictive models. The structure of the data has a significant impact on the design of the predictive models, therefore there is a separate Section 4 that describes main characteristic of the data. A problem of model validation together with a suggested strategy is introduced in Section 5. Main contributions of the paper are found in the following three sections. Comparison of different imputation strategies is presented in Section 6 with an approach for performing imputation for the sparse and non-equidistant data. Section 7 and Section 8 present method development for RSF- and LSTM-based predictive models for sparse and non-equidistant data respectively. Comparison of the methods from Section 7 and Section 8 are provided in Section 9 followed by the conclusions in Section 10. It is recommended to read the paper from the beginning to the end, however it is possible to read Section 6, Section 7 and Section 8 independently after reading Section 2 and Section 4.

2. MOTIVATION AND PROBLEM FORMULATION

Components of a truck do not break often as they are designed to be operational for significant amount of time. This means it is expected that a main part of the vehicles do not experience battery failures during the study period. In our case, vehicle operational data is sparse and non-equidistant with different amount of data readouts per vehicle. A vehicle that has not experienced battery failure is called *censored* since the true failure time is not known. The definition of censoring used here is equivalent to right censored in (D. R. Cox & Oakes, 1984). A failed battery here means that a workshop engineer has replaced it. This is considered to be a good indicator of a battery failure. Therefore, there are two groups of vehicles in our study, namely, the ones with *failed* and *censored* batteries.

A basic battery maintenance policy is to use time or mileage, meaning if the vehicle is operated more than a particular time or mileage, the battery is replaced with a new one. This strategy is simple and easy to explain to a customer, unfortunately, time- or mileage-based maintenance do not give good performance.

Consider two types of records: time and mileage which are recorded when a vehicle either had problems with batteries or leaves the study without any problem. Estimation of the probability density functions (PDF:s) for time and mileage of failed and censored vehicles are shown in Figure 1 for the industrial dataset. Red and dashed red curves are PDF estimates of time and mileage records of vehicles with failed batteries and correspondingly, blue and dashed blue curves are PDF estimates of time and mileage records of vehicles with censored batteries. Notice that bottom and left axes correspond to time related information whereas upper and right axes to mileage related information. It is evident from Figure 1 that there is a significant overlap between densities of failed and censored vehicles for both type of records. This indicates that a time or mileage based maintenance policy will either replace batteries too often or too late.

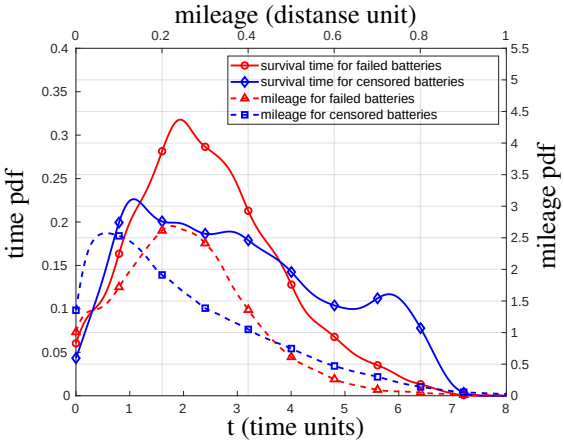


Figure 1. Kernel density estimations of distributions of vehicles with failed and censored batteries. Red and dashed red curves denote probability density function estimations of failure time and covered distance respectively for vehicles with battery problems. Blue and dashed blue curves show probability density function estimations of censoring time and covered distance respectively for vehicles without battery problems.

For every possible maintenance time/mileage, record what fraction of failed batteries are detected correctly and what the false alarm rate is. A Receiver Operating Characteristic (ROC) curve in Figure 2 illustrates the performance for different maintenance thresholds. A black curve represents

the results for mileage-based and a red curve for time-based maintenance and it is clear that using time- or mileage-based maintenance leads to a random guess performance. The analysis above motivates the development of more detailed models using vehicle operational data to predict battery failure times.

2.1. Problem formulation

A common way to perform prognostics is to estimate RUL by tracking and extrapolating a health indicator. In our case, this is not possible since there is no available measurements *directly* related to battery health, see further description of the data in Section 4. Furthermore, a data readout is retrieved at either failure or censoring time, which makes the problem different from a classical classification problem, because in this case it will be impossible to predict a failure in advance.

The lack of health indicator together with sparse and non-equidistant data leads to a high degree of uncertainty which makes a probabilistic framework suitable. Therefore, let the random variable T be the failure time, t_0 the current time, e.g., at a workshop visit, and the sequence of data readouts available at time t_0 be denoted by $\mathcal{V}$. Then, the *probability* that the component survives more than $t+t_0$ time units, given that it is working at age t_0 time units with data $\mathcal{V}$ is

$$\mathcal{B}^{\mathcal{V}}(t;t_0) = P(T > t + t_0 \mid T \geq t_0,\ \mathcal{V}). \tag{1}$$

The reliability function D. R. Cox & Oakes (1984), i.e., the probability that a component survives more than t time-units $R^{\mathcal{V}}(t) = P(T \geq t|\mathcal{V})$, can be used to reformulate (1) as

$$\mathcal{B}^{\mathcal{V}}(t;t_0) = \frac{R^{\mathcal{V}}(t+t_0)}{R^{\mathcal{V}}(t_0)}. \tag{2}$$

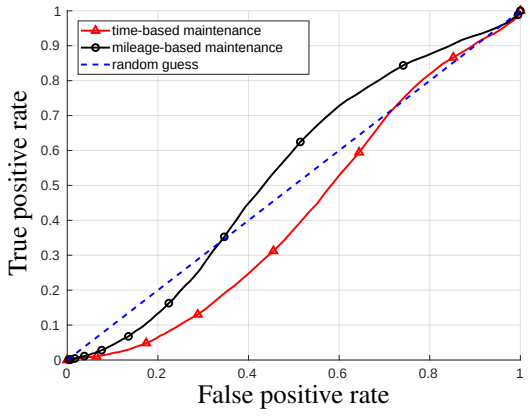


Figure 2. Demonstration of maintenance strategies performance which are based on time and covered distance. Black curve is a ROC curve for mileage-based maintenance and red curve represents ROC curve of time-based maintenance. Dashed blue line corresponds to the random guess strategy.

The function in (1) is here called lifetime function and time t_0 is the time when the prediction is made. The main objective is then to estimate, based on data readout history $\mathcal{V}$, either the reliability function or the lifetime function, both of which can be used in prognostics.

Figure 3 shows an illustration of a pipeline that can be implemented on company infrastructure to predict the lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$ of a vehicle battery that comes for a workshop visit. As shown in the figure, raw sensor measurements from the vehicles are collected in a data lake, then some computational tools are used to extract, transform and load (ETL) data to a predictive model that gives an estimate of the lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$. There is an exchange of information between ETL and model developing and training blocks. ETL block transforms the information that is used to train the model to the form the current architecture of the predictive model requires. At the same time, if the change to the architecture of the predictive model is introduced, this triggers the change in ETL block. In the given paper, the part of ETL stage, i.e., data imputation for the sparse vehicle operational data, is addressed in Section 6, and model development considerations are presented in Section 7 and Section 8.

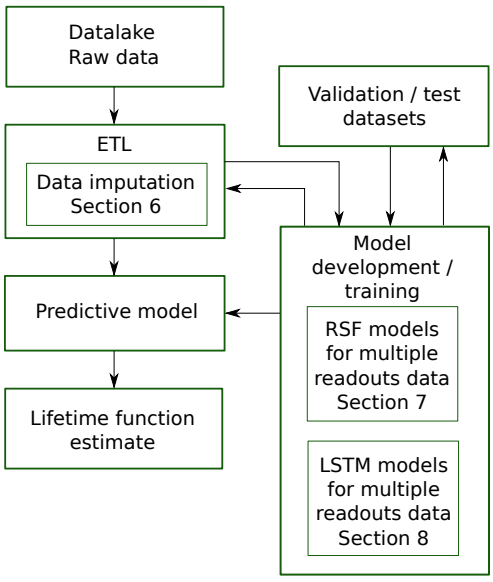


Figure 3. Machine learning pipeline that can be used to predict the lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$ of a battery.

3. THEORETICAL BASIS

The paper uses models based on RSF and LSTM neural networks. Both approaches are machine learning techniques and the basics are briefly summarized in this section. In addition, the section contains an introduction to missing data imputation that is used in the paper by means of an RSF model to tackle missing data problem present in the data under study.

3.1. Random survival forest

Classification and regression trees (Breiman et al., 1984) are binary trees that map a feature or variable space X into a space of outcomes Y. Samples of the feature vector and outcome variable is represented as a pair (x_i, y_i). In a classification problem, the target values y_i are discrete, and in a regression problem the target is continuous valued. Classification and regression trees can be regarded as a non-linear estimator

$$\hat{\theta}(x_i) = \hat{y}_i \tag{3}$$

where $\hat{\theta}(x)$ is built by partitioning the feature space X into disjoint regions $X = \cup_m R_m$ with some estimating model for each region. For a regression problem, an estimate is a real value that fits data in a region R_m best, often the mean. In case of a classification problem, the estimate is often the majority class among all samples in R_m. Classification and regression trees are easy to interpret but, unfortunately trees are weak classifiers with large variance, meaning they do not generalize to unseen data well.

To address this issue a Random Forest (RF) model, an ensemble of trees, was proposed by Breiman (2001). There are two steps in RF that make it perform better than ordinary classification and regression trees, namely, bootstrap aggregation, also known as bagging, and a random variable selection at the split steps. Bootstrap aggregation creates a set of new datasets by sampling uniformly with replacement from the original one, then machine learning models are fitted to every newly created dataset. In the case of regression trees, the output from a bootstrap aggregation model is the mean of outputs of all trees

$$\hat{\theta}_{\text{BAGG}}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{\theta}_i(x) \tag{4}$$

where $\hat{\theta}_i(x)$ is a tree model fitted to the i^{th} bootstrap sample, and B is the number of trees/bootstrap samples. Averaging over all models creates a predictor with a reduced variance compared to the regular classification and regression tree.

An RSF model is a variation of an RF model that is modified for the purpose of right-censored data and survival analysis (Ishwaran et al., 2008). One of the main differences in an RSF model compared to RF is that the cost function used for splitting is typically the log-rank test (Ciampi et al., 1986). At each terminal node in a tree, a node at which splitting no longer is performed, the Nelson-Aalen estimate of the cumulative hazard rate $H(t)$ is computed (D. R. Cox & Oakes, 1984) based on the samples in the node. The estimated cumulative hazard rate $\hat{H}(t)$ of the whole forest is computed by averaging over tree hazard rates. The RSF estimate $\hat{R}(t)$ of the reliability function is then obtained as D. R. Cox & Oakes (1984)

$$\hat{R}(t) = e^{-\hat{H}(t)}. \tag{5}$$

3.2. Long Short-Term Memory networks

Neural networks, similarly to Random Forests, are non-linear models that estimate relationships between input and target variables. Neural network models are interesting since they have proven effective in many different areas, especially when the supporting dataset is large. In a feedforward neural network, the input layer is connected with hidden layers and an output layer by the means of weights. Information is then flowing only in the forward direction. This means that there are no connections between nodes from the current layer with themselves or to previous layers. Since data readouts from a single vehicle forms a sequence, recurrent models are of interest. LSTM networks (Hochreiter & Schmidhuber, 1997) is a class of recurrent neural networks (RNN) (Goodfellow et al., 2016) that allow feedback connections. In recurrent neural networks the output from a layer is fed back through a delay unit as shown in Figure 4. RNNs are designed to work with sequential data such as translating text from one language to another, speech recognition, or analyzing frames in a video stream. The connection to the battery prognostics problem stated here makes RNN networks an attractive model structure to explore.

Training RNNs is in general difficult, in particular due to the problems of vanishing or exploding gradients when sequences of data are long or networks have many layers. LSTM networks mitigate these problems and the core building block is an LSTM cell with a schematic illustration shown in Figure 5. As can be seen, the data from the previous layer x_t and the previous time step h_{t-1} are inputs to the LSTM cell. Flow of information is controlled by three gates, input, forget, and output. Each of the gates has a sigmoid activation function layer, a regular layer as in feedforward networks, which take values in the interval between 0 and 1, and multiplied elementwise with the input data vector. Values that are close to zero indicate which parts of the data that is ignored during the current time step. Conversely, values close to 1 signify active parts of data for making prediction. The most important part of the LSTM cell is the forget gate in combination with internal state s_t. The current internal state is memorized and is multiplied with the forget gate at the next time step. The interesting step happens when the result of filtering an internal state through the forget gate is combined with the result of filtering input data through the input gate. It can be seen that the results are combined by summation which is an important part in tackling vanishing gradients problem. The formula

$$\begin{aligned} f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ s_t &= f_t \otimes s_{t-1} + i_t \otimes \tanh\left(W_v x_t + U_v h_{t-1} + b_v\right) \\ h_t &= o_t \otimes \tanh\left(s_t\right) \end{aligned} \tag{6}$$

represents the LSTM-cell. Here, $\otimes$ denotes element wise

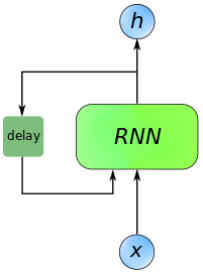


Figure 4. Illustration of an RNN network where x is a sequence of input data and h is a sequence of output from an RNN layer.

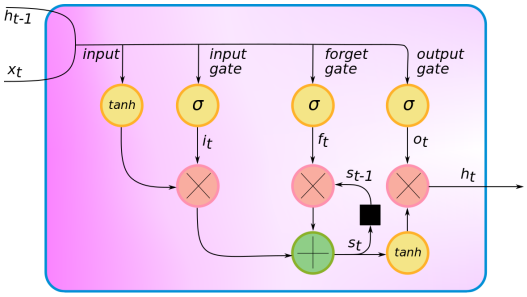


Figure 5. Illustration of an LSTM cell. The sign $\otimes$ stands for element wise multiplication, σ is a sigmoid activation function, $\tanh$ is a tangent hyperbolic activation function.

multiplication of two vectors. Every input to the activation function layer has its own set of weight matrices. Dimensionality of the vectors after passing through the activation function layer is usually different from the dimensions of the vector which combines x_t and h_{t-1} and it is controlled by a network designer.

3.3. Imputing missing data

Many approaches for data imputation have been developed. A thorough overview of the random forest imputation methods can be found in (Tang & Ishwaran, 2017). Even if the paper is focused primarily on random forest-based imputation techniques it gives a good motivation why these method are attractive. Advantages of using the random forest framework for imputation are: 1) it handles heterogeneous missing data, i.e., both numerical and categorical variables, 2) it performs well in the case of a multidimensional feature space with complex interactions between variables, and 3) it scales to big data problems. It is also reported that a MissForest which is a random forest-based technique outperforms in several cases methods such as k-nearest neighbors (Altman, 1992) and multivariate imputation using chained equation (Van Buuren, 2007).

Three imputation techniques are compared in this paper: mean imputation, unsupervised on-the-fly-imputation (OTFI), and a modification of the missforest imputation approach. The two last techniques are described in (Tang & Ishwaran, 2017) and can be found in the RSF package (Ishwaran & Kogalur, 2007) which is also used to implement the RSF models. Mean imputation is based on average values among non-missing records for a particular variable. Unsupervised OTFI is an iterative procedure that treats data imputation as a multivariate regression problem. Data is imputed simultaneously while growing the forest, which is done using a multivariate unsupervised splitting rule. This means that the outcome variables for each split are randomly sampled for that node. The MissForest technique, and its modifications, considers the missing data problem as a prediction problem. Here, random forest models are grown by regressing every variable against all other variables successively and predicting missing data for the current target variable using a random forest model. When comparing, the number of iterations for missForest and OTFI are set to 5 and the value of the parameter α, which defines dimensionality of multivariate regression during one step in missForest algorithm, is set to $\alpha = 0.05$.

4. DATA DESCRIPTION

The industrial vehicle fleet data was not primarily designed for battery prognostic purposes, at first it was used for diagnostics and performance monitoring of the fleet. Further, the data distribution is not constant over time and, for example, sensor installations vary between new and old models leading to non-random missing data in the dataset. The missing data rate is about 40%. The missing values are not uniformly distributed among variables, some variables can have significantly higher missing rate than others. Section 6 discusses several imputation strategies and what is the best way to deal with missing data here.

Available vehicle operational data greatly influences the choice and method development. The data source is provided by Scania CV, a heavy-duty truck manufacturer in Sweden, and represents how a vehicle is operated during the lifetime of the battery. Every vehicle can have different number of data readouts with varying time difference between each readout which makes data sparse and non-equidistant. Readouts are recorded into a database when a vehicle comes to a workshop and workshop visits are determined by vehicle owners which makes data readouts irregular.

A data readout consists of two types of variables: variables with fixed and varying values during vehicle lifetime. Constant value variables typically describe configuration of the vehicle, e.g., what type of cabin is used, where the battery is mounted, if a bed is installed in the cabin, etc. These variables are categorical, i.e., only a finite set of values are possible. Variables with varying values in time are organized into histograms, where each of them shows how a particular part of a vehicle has been used from the beginning of the battery lifetime to a time of readout event. As an example, the ambient pressure is stored as a histogram with 10 bins in the data

and each bin represents the number of times a vehicle has been operated in an atmospheric pressure that falls into the range of the bin. Then, the values of histogram are normalized such that summation of bin values is 100. There are 22 histograms in the data set with varying number of bins. Every bin contributes with a separate variable to the feature space. Examples of other histograms are battery voltage, ambient temperature, and engine load histogram. The ratio between number of categorical and number of numerical valued variables is 1:49, i.e., a large majority are histogram bin values. Each readout consists of 417 variables that is a mixture of the categorical and the numerical variables.

A main characteristic of the data for this work is that there is no health indicator present. Even if there was a health indicator present, it would be difficult to track its value during the lifetime of a battery due to the infrequent and irregular readouts. However, using information about time of failure/censoring and partitioning a set of vehicles into groups with similar operational characteristics by the means of machine learning algorithms, it is possible to build a reliability function for every vehicle that can be used to formulate individual maintenance policies, or condition-based maintenance policies.

4.1. Training, validation and test data

Some main characteristics of the database are as follow. The training set that is used for building models consists of 46,974 trucks whereas validation/test sets contain 2,000 trucks. The trucks in all sets originates from 5 European markets Sweden, Germany, Belgium, Netherlands, and France. For these vehicles from the training set, there are 115,342 data readouts in the database. Three prominent groups of vehicles are found in the database: vehicles with 1, 2, or 3 data readouts where the number of vehicles in each group are 1) 5,192 vehicles in group with 1 data readout, 2) 15,196 vehicles with 2 data readouts, and 3) 26,586 vehicles with 3 data readouts. The censoring rate is about 80%, meaning there are 7,189 vehicles with reported failures. The validation and test sets have 1,000 trucks each with equal split between failed and censored vehicles. As in the training set, the vehicles from the validation/test sets can have different amount of readouts during the lifetime of the battery.

5. MODEL VALIDATION TECHNIQUE

In supervised learning, with labeled output data in a classification and regression problem, it is fairly straightforward to define performance measure and validation techniques. Here, however, the function to estimate is a (conditional-)reliability function, a function we do not know the true value of. Therefore, validation of the estimated models are more involved.

5.1. Principle for validation

For this purpose, the validation and test set of vehicles consists of 2,000 vehicles in total with equal proportion of failed and censored batteries. The requirements on vehicles to be a part of the validation or test set are: 1) the vehicle must have at least two data readouts and 2) a time difference between the last two readouts must be in between 0.25 and 0.5 time units.

All data readouts except the last are fed into the model to receive an estimate of the lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$ where t_0 is the time of the second to last readout. Then, the value $\mathcal{B}^{\mathcal{V}}(t^*;t_0)$ is computed, where t^* is the time difference between the last two readouts. For a model with good predictive power, $\mathcal{B}^{\mathcal{V}}(t^*;t_0)$ should on average be large for censored batteries and small for failed batteries. This comparison is why the requirement on the time difference between the last two readouts is important. It guarantees that compared lifetime values are from the same time interval, otherwise the comparison would not be fair. To illustrate, assume the lifetime function for the broken battery is computed at $t^* = 0.5$ time units, difference between two last readouts, and for the censored battery at $t^* = 3$ time units. If both values are small it is not possible to draw any conclusions on model performance. On the one hand, it could be the case that the model performs poorly, but on the other it could be due to the fact that censored battery is about to fail.

5.2. Validation metric

The validation metric used are ROC curves, introduced in Section 2, and the associated Area under the curve values (AUC) (Friedman et al., 2001). These metrics are computed based on histograms of lifetime functions. Given a model, $\mathcal{B}^{\mathcal{V}}(t^*;t_0)$ is computed for all vehicles in the validation/test set forming two histograms - failed and censored batteries, see Figure 6. Then, an ROC curve is parameterized by all possible values of the maintenance threshold θ. For a given θ, vehicles for which the lifetime function values $\mathcal{B}^{\mathcal{V}}(t^*;t_0)$ are smaller than the threshold are regarded as failed and censored otherwise. The area under the curve is then computed by integration of the ROC curve.

An advantage of using ROC curves are that they are invariant to cost functions and changing class distributions, i.e., changes in imbalance ratio with time. Authors in (Webb & Ting, 2005) criticize usage of ROC curves for the case of changing distributions. However, as mentioned in (Fawcett & Flach, 2005) this can happen only in some particular cases. In this paper, a comparative study of different models with a fixed imbalance ratio is performed, therefore problems mentioned in (Webb & Ting, 2005) are not applicable. ROC curves access separate metrics for both classes of batteries, i.e., failed and censored, which is important in our case as it is of interest not only eliminate battery failures, but also to

minimize the number of unnecessary repairs when a battery is in fact functional.

6. IMPUTATION OF MISSING DATA

Missing data is a common problem in prognostics, machine learning, or data science. In our problem, data is mostly missing due to different sensor installations on vehicles of different model and age. Figure 7 illustrates missing data properties of the data set. With vehicles on the x-axis and variables on the y-axis, every pixel shows the fraction of missing data among available readouts. With the data, there are four possible cases depending on the number of readouts for a vehicle. A missing rate of 1 is represented by the color white, a missing rate of 0 with the color green, a missing rate 0.33 with the color yellow, and missing rates of 0.5 and 0.66 are represented by two shades of orange.

In general, the values of the variables are either missing or present in all readouts. However, some vehicles have intermittent missing values, as shown with the magnifying glass in Figure 7. A key observation is that the majority of missing data does not have a random pattern. One way to deal with the missing data would be to discard vehicles with missing information, which would be appropriate if the number of samples in the whole population with missing data is low. However, here vehicles with missing values constitute the majority of the population. Therefore, instead imputation of missing data is explored.

6.1. Imputation performance for RSF and MLP models

First, the three imputation strategies are evaluated on two models, the RSF and Multilayer perceptron (MLP) single data readout described in (Voronov et al., 2018a) and (Voronov et al., 2018b) respectively. Here, the MLP model is an ensemble multilayer perceptron network with a linear degradation assumption for a reliability function of failed batteries, one of the best performing models from (Voronov et al., 2018b).

The imputation evaluation is performed by training models with different imputation strategies, and then computing ROC curves, shown in and Figure 8, and the corresponding AUC values in Table 1. First, notice that the MLP model outperforms the RSF model for all imputation strategies. Further, the mean imputation strategy gives the best performance for the MLP model and unsupervised on the fly imputation is the best for the RSF model. A conclusion from the results is that there is no big difference in using mean imputation or more advanced strategies for the MLP and RSF models. This result is consistent with results in (Tang & Ishwaran, 2017) where the more advanced methods only gave a small performance increase for the case of non-random missing data with medium and high correlation between features. This is exactly the situation with the vehicle fleet data here. Also, taking into account that the advanced methods for imputation take significant computational resources, and little performance gain is expected, the simple mean imputation will be used.

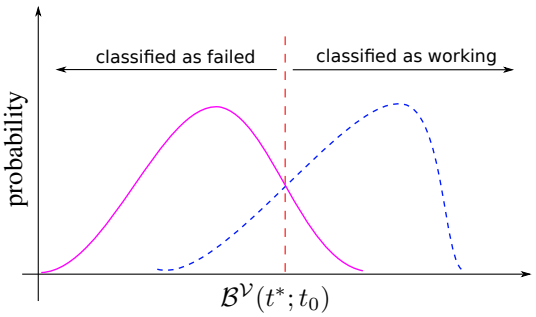


Figure 6. Illustration of two histograms of $\mathcal{B}^{\mathcal{V}}(t^*;t_0)$ values for failed batteries, pink curve, and censored batteries, blue dashed curve. Threshold θ, red dashed line, separates working and failed batteries. Time t^* is time of either failure or censoring.

6.2. Imputation strategy for multiple data readouts

Thus, for the single data readout case it was concluded that a mean imputation strategy is appropriate. Next question is then how to extend this strategy to the multiple readout case. A possible mean imputation extension is to use all readouts that contain information for a particular variable. This approach is implemented for the majority of the models that are considered in the paper. However, a drawback is that the mean can be biased towards a particular class of vehicles with many readouts. For example, consider again Figure 1. Many batteries are failed or censored between 1 and 4 time units which means there are many readouts from this period and the mean will then be biased towards this set of vehicles. One way to address this situation is to partition readouts based on age and compute the mean value for imputation within its respective partition. Thus, a mean value in this case will represent how a vehicle is used on average within a certain age interval.

The dataset contains vehicles with survival time up to 8 time units which, initially, are divided into 8 equidistant time intervals. If a vehicle has several readouts in an interval then only one with the latest timestamp is used in the imputation. In Figure 9(a), the distribution of readouts among the 8 time intervals is shown. It is evident that the number of readouts in the last three intervals are much fewer than in the first three. In addition, the missing rate for the first three intervals is about 33%, while for the last three 67%, 74%, and 83% respectively. If the missing rate is as high as in the last two intervals combined with a relatively small number of readouts, this leads to a few vehicles with non-missing data for some of the variables which makes the estimation of the true mean value less reliable. Therefore, to balance the intervals,

the last three intervals were merged into one interval. The resulting distribution is shown in Figure 9(b) where the missing rate for the sixth interval is 68%. Evaluation of the strategies that impute missing values with a mean over all readouts and a mean over readouts from a particular interval is given in Section 9.

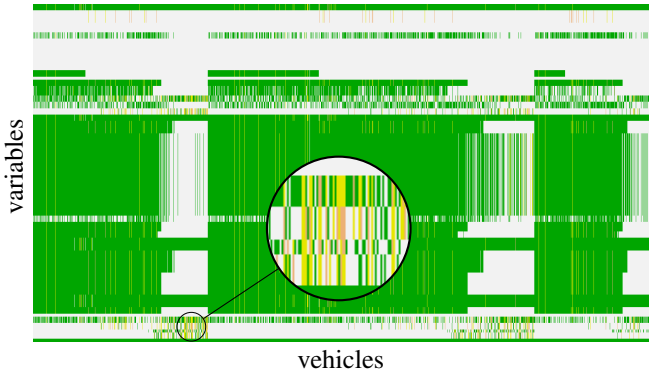


Figure 7. Illustration of missing data among multiple data readouts. For every vehicle, x axis, there is a set of variables that form a readout, y axis. White color shows that a value of a variable is missing for a given vehicle in all available readouts. If a variable is available in all readouts then color is green. Yellow and two shades of orange show that values of variable are present in some readouts and missing in others which can be observed in the magnified part of the image.

7. EXTENDING RSF METHOD FOR THE CASE OF MULTIPLE DATA READOUTS

This section discusses how to extend the single data readout RSF model from (Voronov et al., 2018a) to the the situation with multiple, sparse, and irregular data readouts. Gomes et al. (2017) presents modifications of Random Forest models to the case of data streams. However, these data streams differ from the fleet data case since there are many vehicles with very few data readouts versus the long sequences as, for instance, in (Rosset & Inger, 2000) for the adaptive random forests in (Gomes et al., 2017). Another problem when applying modifications of random forest models to data streams is that these models are designed for the classification problem and not for survival analysis as in our case.

To the best of our knowledge there is no straightforward RSF model for multiple readout problems. A simple approach is to reduce the problem to the one readout case by considering every readout as from an independent vehicle and, then, apply the single readout RSF model directly. However, this violates a basic assumption on independence of the readouts and a bias towards the vehicles with the most readouts can be expected. Another problem is then that the imbalance between censored and failed vehicles will increase significantly, which is not desirable as the failed to censored vehicle is already low at 20%.

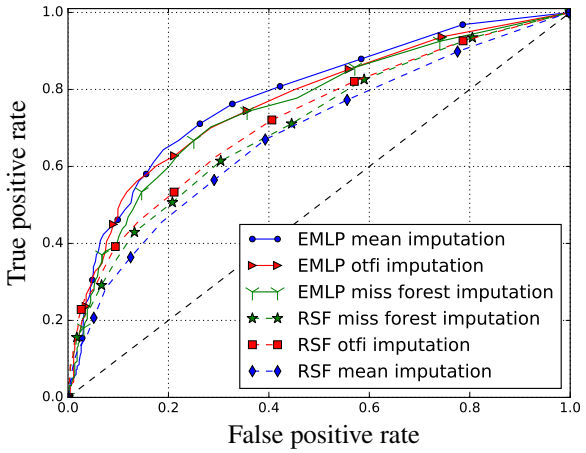
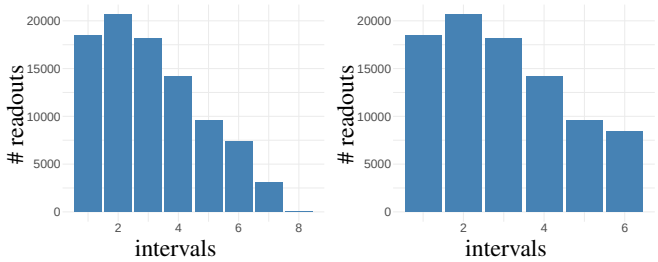


Figure 8. ROC curves for different imputation strategies for MLP and RSF models. Only one (last) data readouts is used per vehicle. Black dashed line corresponds to random guess performance. EMLP stands for an ensemble multilayer perceptron model.

Table 1. AUC values for ROC curves in Figure 8.

Imputation approach	RSF	MLP
Mean imputation	0.681	0.772
OTFI imputation	0.715	0.764
MissForest imputation	0.700	0.761



(a) Split in 8 intervals. (b) Split in 6 intervals.

Figure 9. The figure illustrates the distribution of vehicle data readouts based on time interval splitting. Figure (a) shows the distribution when splitting into 8 intervals and Figure (b) when splitting into 6 intervals.

7.1. Feature augmentation vs whole population of readouts

It is suggested in the paper to deal with the short sequences of multiple readouts by augmenting a feature space with data from multiple readouts. Figure 10 illustrates this procedure where there are three types of vehicles with 3, 2 and 1 data readouts. For example, consider *vehicle1* where the augmented

features corresponds to concatenating the readouts, starting with the last, and shown to the right in Figure 10. The procedure for *vehicle2* and *vehicle1* is similar, except they have less data readouts and corresponding features are denoted as missing. The static categorical data that do not change with time should not be repeated in the augmented readout. Denote the static variables for vehicle j with s_j and all remaining variables as h_{ji} where i stands for i:th data readout. The augmented readout as ar_j is then

$$\text{ar}_j = (s_j, h_{j3}, h_{j2} \text{ or missing}, h_{j1} \text{ or missing}). \quad (7)$$

7.2. Time related information and missing readouts

In (7) it is clear that the augmented readout ar_j does not contain any time information such as time and mileage. Voronov et al. (2018a) showed that time and mileage is best not used as features directly as we are interested in estimating degradation based on vehicle operation and not on age or mileage. However, time related information on readout separation should be introduced in the augmented readout to let the RSF model adapt to the difference in time and mileage between initial readouts. Let the timestamp and mileage of the i:th readout of the j:th vehicle be denoted by t_{ji} and m_{ji} respectively, then two new variables for the difference in time and mileage between k:th and l:th readouts are defined as

$$\Delta t^j_{k,l} = t_{jk} - t_{jl} \text{ and } \Delta m^j_{k,l} = m_{jk} - m_{jl}. \quad (8)$$

Two cases are now: 1) use the differences in time and mileage between two neighboring readouts, $\Delta t^j_{i+1,i}$ and $\Delta m^j_{i+1,i}$, 2) use the differences in time an mileage between the last readout and the current one, $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$. Here, the last readout corresponds to the third readout for *vehicle1*, the second readout for *vehicle2*, and the first readout for *vehicle3*. Notice that $\Delta t^j_{k,l}$ and $\Delta m^j_{k,l}$ are not defined if one of the readouts in the definition (8) is missing.

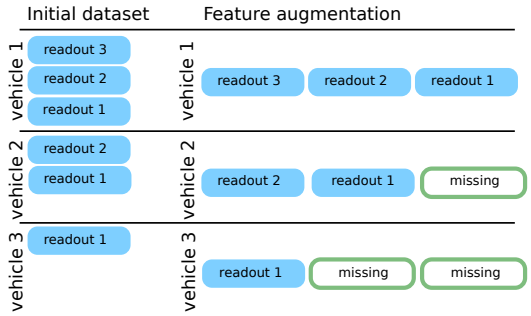


Figure 10. Demonstration of feature space augmentation for the case of multiple readouts. Left part of the figure shows how many readouts are initially available for three vehicles. Right part of the figure shows how the given readouts are transformed to form a new feature space.

How to treat variables in missing readouts depends on the way $\Delta t^j_{k,l}$ and $\Delta m^j_{k,l}$ are computed. For the approach with neighboring readouts, $\Delta t^j_{i+1,i}$ and $\Delta m^j_{i+1,i}$, the neighbor readout is copied into the missing one and values of differences between time and mileage are set to zero. For the second approach when the differences in time and mileage are computed with respect to the last readout, $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$, the last readout is copied into the missing one and both time and mileage differences are set to zero. Based on the discussion above two models for the the augmented readout are listed below. For the model which uses $\Delta t^j_{i+1,i}$ and $\Delta m^j_{i+1,i}$ the augmented readout is defined

$$\text{ar}_j = (s_j, h_{j3}, h_{j2}, \Delta t^j_{3,2}, \Delta m^j_{3,2}, h_{j1}, \Delta t^j_{2,1}, \Delta m^j_{2,1}). \quad (9)$$

For the model that uses $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$, the augmented readout is

$$\text{ar}_j = (s_j, h_{j3}, h_{j2}, \Delta t^j_{3,2}, \Delta m^j_{3,2}, h_{j1}, \Delta t^j_{3,1}, \Delta m^j_{3,1}). \quad (10)$$

Another approach to define the augmented readout for a vehicle is to consider the rate of change, i.e., the derivative, between two readouts instead of using the second h_{j2} and first h_{j1} readouts as they are. Analogously to (8), the difference between k:th and l:th readouts for the j:th vehicle is defined as

$$\Delta h^j_{k,l} = h_{jk} - h_{jl} \quad (11)$$

The derivative is a ratio $\frac{\Delta h^j_{k,l}}{\Delta t^j_{k,l}}$ that is obtained by diving the difference of values between two readouts $\Delta h^j_{k,l}$ by the time difference $\Delta t^j_{k,l}$. The values for derivatives that involve a missing readout are set to zero. The augmented readout for a vehicle j for a model that uses $\Delta t^j_{i+1,i}$ and $\Delta m^j_{i+1,i}$ is

$$\text{ar}_j = \left(s_j, h_{j3}, \frac{\Delta h^j_{3,2}}{\Delta t^j_{3,2}}, \Delta m^j_{3,2}, \frac{\Delta h^j_{2,1}}{\Delta t^j_{2,1}}, \Delta m^j_{2,1} \right). \quad (12)$$

The augmented readout for a vehicle j for a model which uses $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$ is

$$\text{ar}_j = \left(s_j, h_{j3}, \frac{\Delta h^j_{3,2}}{\Delta t^j_{3,2}}, \Delta m^j_{3,2}, \frac{\Delta h^j_{3,1}}{\Delta t^j_{3,1}}, \Delta m^j_{3,1} \right). \quad (13)$$

Here, several approaches to augment the feature space for multiple readouts have been summarized as

1. RSF model with an augmented readout as in (9)
2. RSF model with an augmented readout as in (10)
3. RSF model with an augmented readout as in (12)
4. RSF model with an augmented readout as in (13)

Performance of the models will be analyzed in Section 9.

8. LSTM MODELS FOR MULTIPLE DATA READOUTS

LSTM networks, a specific kind of RNNs, have been successfully applied to different machine learning problems with varying length sequential data in the fields such as natural language processing, image captioning, and voice recognition. The success is mainly due to the ability of the network to capture long term dependencies between features. A problem with general recurrent neural networks is that with deep architectures and long data sequences there is a risk of encountering vanishing gradients leading to convergence problems, which is one of the reasons recurrent neural networks are less common in practice. LSTM networks in turn addresses the problem of vanishing gradients with the help of gates and internal or hidden states.

The authors in (Elsheikh et al., 2019; Zhang et al., 2018) use models that are based on LSTM networks to predict the RUL of turbofan engine and lithium-ion batteries. However, the type of data in the aforementioned works is different from the one considered in the paper. Articles (Elsheikh et al., 2019; Zhang et al., 2018) work with run-to-failure data, and the type of data in this paper is sparse survival data. Therefore, there is a need for the new methods that take into account characteristics of the survival data. Recently, LSTM networks have been applied to medical data for the purpose of giving a diagnosis, see for example (Lipton et al., 2015) where the dataset has similar characteristics as the dataset here, e.g., containing a varying number of samples from each patient. This section will extend the results from (Voronov et al., 2018b), where a fully connected neural network model is applied to vehicle data with only one readout per vehicle, to an LSTM network for sequential data.

8.1. LSTM network models for battery prognostics

As stated in the problem formulation, the output of the predictive models can either be the lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$ or the reliability function $R^{\mathcal{V}}(t)$. Therefore, there are two main LSTM network architectures, as illustrated in Figure 11. The target, or output data, is represented by a fully connected neural network layer with sigmoid activation function with possible values in the range $[0,1]$ that corresponds to survival probabilities. Each output node β_i or R with a time span of 8 time units and a step of 0.25 time units gives 32 time point estimates of either the lifetime or the reliability functions resulting in dimensionality of node β_i or R to be 32.

The blocks that are denoted by *LSTM* are LSTM cells with 64 internal/hidden states. The size of the layer is chosen as a balance between computational time and performance and, as will be shown in Section 9, the selected number of hidden states is sufficient for our use-case.

Further, it was decided to use two layers of LSTM cells in the networks. An architecture with three LSTM layers was investigated but did not lead to any performance improvement. In addition, to counteract overfitting a dropout technique (Srivastava et al., 2014) is used when training the LSTM networks. Dropout is introduced for the linear transformation of the inputs and for the linear transformation of internal states with the values of 0.5 and 0.4 respectively. The blocks *TimeDistributed* emphasizes that weights for the fully connected output layer share the same values when applied independently for every time step in the LSTM network. The notion is borrowed from the deep learning library Keras (Chollet, 2015) that is used to implement the models. Question marks in shape triples (batch size, number of readouts, size of readout) in Figure 11 indicate the varying batch size and varying number of readouts between vehicles.

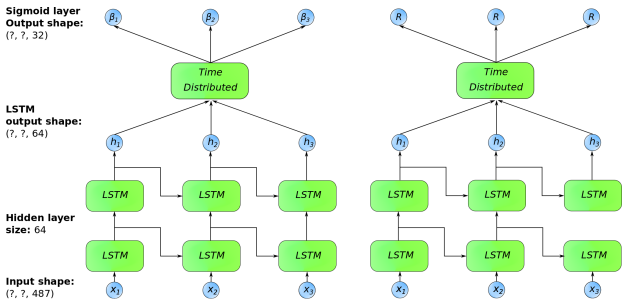


Figure 11. Basic architectures of LSTM networks. The left network uses lifetime functions β_1, β_2 and β_3 as outputs, and the right network uses reliability function R as an output.

8.2. Inputs and target values for the LSTM networks

The input data is denoted by a vector containing all data from one data readout x_1, x_2, and x_3. Each x_i is a data readout augmented with two variables $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$, i.e., the time and mileage difference between the readouts as described in Section 7. The representation of input x_i for jth vehicle is

$$x_i = \left(s_j, h_{ji}, \Delta t^j_{\text{last},i}, \Delta m^j_{\text{last},i}\right) \tag{14}$$

where s_j is a collection of all categorical variables and h_{ji} is a collection of all histogram variables, as introduced in Section 7. The dimensionality of the input vector x_i is 487. The values of variables $\Delta t^j_{\text{last,last}}$ and $\Delta m^j_{\text{last,last}}$ are always zero, therefore a small random value from a uniform distribution is added to avoid introducing constant features which might cause problems in the learning phase.

The procedure for data preprocessing for a neural network is described in detail in (Voronov et al., 2018b) and is applied to every time step in the network. Categorical variables are transformed to vector representation using 1-of-(C-1) encoding where C is number of possible values. Numerical valued variables are standardized, i.e., normalized to zero mean and unit standard deviation. Effects of using raw data input, without any preprocessing, are also considered in the comparison

in Section 9.

The procedure for creating target values for networks with the reliability function as output is presented in detail in (Voronov et al., 2018b). Briefly, the reliability function R for a censored vehicle is 1 in all nodes prior to the censoring time and the remaining part of the reliability function is estimated with a Kaplan-Meier estimator (D. R. Cox & Oakes, 1984) where vehicles with failed and censored batteries after $t_{\text{censoring}}$ are used in the estimate. See the green dashed curve in Figure 12 for an example. For a vehicle with a failed battery, the reliability function R decays monotonically from 1 to the time of failure t_{failure} as shown by the red curve in Figure 12.

Target values for networks with the lifetime function as an output, the node targets β_i, are computed by taking the modeled reliability function R described above and applying formula (2).

8.3. Loss function

When choosing loss function for the training of the network, it is important to notice that the output, for both reliability and lifetime output functions, has the general property that it is monotonically decreasing. First, the loss function used for single readout MLP models in (Voronov et al., 2018b) is described and then an extension to the LSTM models is proposed.

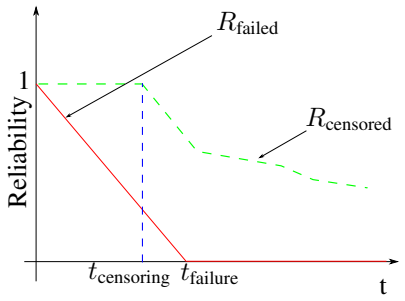


Figure 12. Models of reliability function R for a failed, red curve, and a censored, green dashed curve, vehicle.

Let q_i be the modeled reliability function in node i. Then q_i represents a probability to survive at least t_i time units where t_i is the corresponding time for node i and let $\mathcal{Q} = \{q_i, 1-q_i\}$ define a discrete probability distribution with two outcomes. Similarly, let the discrete probability distribution $\mathcal{P} = \{p_i, 1-p_i\}$ be defined based on the probability of survival p_i that is estimated by a multilayer perceptron network. The Kullback-Leiber divergence e_i is used to quantify difference between $\mathcal{Q}$ and $\mathcal{P}$ as

$$e_i = KL(\mathcal{Q}||\mathcal{P}) = q_i \ln \frac{q_i}{p_i} + (1-q_i) \ln \frac{(1-q_i)}{(1-p_i)}. \quad (15)$$

The relative entropy loss function for the MLP models is then defined as

$$E_{\text{KL}}^{gl} = \sum_{i=1}^{32} e_i^{gl} + \lambda \sum_{i=1}^{31} \max(0, p_i^{gl} - p_{i-1}^{gl}) \quad (16)$$

where the term after λ parameter penalizes non-monotonicity in the solution.

The extension of the relative entropy loss in (16) to an LSTM network with target outputs for every time step/data readout is then

$$E_{\text{LSTM}} = \frac{1}{K \cdot M \cdot N} \sum_{g=1}^{K} \sum_{l=1}^{M} E_{\text{KL}}^{gl} \quad (17)$$

where N denotes number of nodes in reliability function, namely 32, M is number of readouts per vehicle, and K is a size of a sample which is propagated through the network before the weight update.

8.4. Batch formation for sparse vehicle operational data

In training, all examples in a mini-batch, which are selected randomly before feeding them into the network, must have the same dimensionality, i.e., number of readouts. This implies for our problem that if a batch of size, for example, 128 is selected, then all vehicles in the batch must have equal amount of readouts and since vehicles have a varying number of readouts this need to be addressed.

Three approaches to deal with varying number of readouts is analyzed in this work. The first approach is to use a batch of size 1, and then by construction all vehicles in the mini-batch have the same number of readouts and no additional data manipulation is needed. A drawback with such an approach is that the estimate of the gradients in the loss function will be noisy and may lead to reduced performance of the model. Here, batch size 1 is used as a baseline for other strategies.

A second approach for batch formation of size n is to sample randomly a class of vehicles with 1, 2, or 3 readouts proportionally to their frequency in the data and, then, randomly sample n vehicles from the chosen class. Finally, the third approach for batch formation is to zero-pad missing readouts. This means that all 487 input and 32 output nodes for the given readout are set to zero for missing readouts.

8.5. Data balancing by ensembles of LSTM and weighted loss

Learning from imbalanced data can create a bias towards the majority class depending on severity of the imbalance. Some machine learning methods, for example tree-based methods including RSF, handle imbalances better than some other methods, for example neural networks.

In (Voronov et al., 2018b) an ensemble of MLP models is used to address the data imbalance problem. The method

showed promising results and a similar strategy is used here for LSTM networks. The ratio between failed and censored batteries is about 1 to 5 in the dataset. Therefore, vehicles with censored batteries are randomly partitioned into 5 groups. Then, for each group, the data is merged with the group of failed vehicles to form a training dataset for one LSTM network. The procedure is repeated for every network in the ensemble. When the networks are trained, the prediction from the last time step is taken from all networks and averaged to receive a final estimate of the lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$ or the reliability function $R^{\mathcal{V}}(t)$ as shown in Figure 13. A main advantage of using ensemble models for data balancing is that it can be directly applied to the heterogeneous dataset and variance of the predictive model is reduced. Note that the applicability of the method depends on the severity of imbalance. It is useful with the ratio 1:5 as in our problem, but would be less applicable if the ratio is, for example 1:100.

An alternative way to handle imbalance is to consider one LSTM network, and weight censored samples in the loss function according to the imbalance. This directly influences the computation of the gradients and leads to less biased model towards a majority class. The imbalance weighted loss function is

$$E_{\text{LSTM}} = \frac{1}{K \cdot M \cdot N} \sum_{i=1}^{K} \mu_i \left(\sum_{j=1}^{M} E_{\text{KL}}^{ij} \right) \tag{18}$$

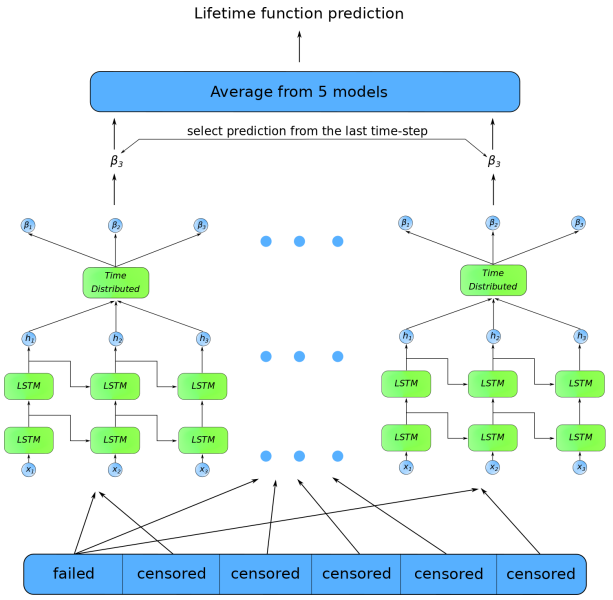


Figure 13. Demonstration of the approach to balance failed and censored batteries with a help of an ensemble of LSTM networks. The ensemble consists of 5 LSTM networks with the architectures described in Figure 11. Every network has its own training dataset. Predictions from LSTM networks are averaged to receive the estimation of the lifetime function.

where the weight μ_i has a value of $\frac{1}{6}$ for the censored batteries or $\frac{5}{6}$ for failed batteries. The values of μ_i are defined by the ratio between the number of censored and failed batteries. Both approaches to handle imbalanced data are evaluated in Section 9.

8.6. Feeding $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$ at later stage in network

A main objective in this work is to find dependencies between vehicle operation and battery degradation behavior and therefore time and mileage is not fed into the feature vector, only time differences $\Delta t^j_{\text{last},i}$ and mileage $\Delta m^j_{\text{last},i}$ are provided to the LSTM models as part of the augmented input vector x_i. It is possible to detach vehicle operation characteristics from time and mileage even more by considering the network architecture shown in Figure 14. Here, the input vector y_i for j:th vehicle is

$$y_i = (s_j, h_{ji}) \tag{19}$$

and a vector $t_i = \left(\Delta t^j_{\text{last},i}, \Delta m^j_{\text{last},i}\right)$ contains time and mileage difference. First, vehicle operational data is processed by LSTM layers to find dependencies between variables and then the representation of the dependencies h_i is concatenated with the vector t_i and propagated to the last dense layer. This approach is also evaluated in Section 9.

9. PERFORMANCE ANALYSIS OF MODELS

In the previous sections, a number of different model architectures has been introduced and this section evaluates performance for both RSF- and LSTM-based models. In addition, baseline performance is determined using models for one data readout based on standard Cox regression and an ensemble multilayer perceptron model with 5 networks from (Voronov et al., 2018b). Performance will be quantified as outlined in

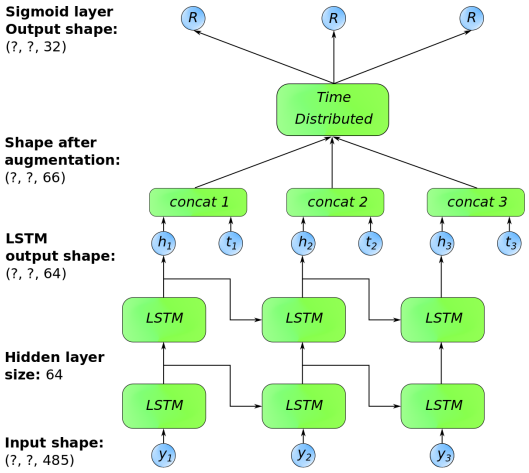


Figure 14. Architecture of a network where $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$ are considered as one vector (denoted as t_1, t_2, and t_3) and fed into the network after the recurrent layers.

Section 5 based on ROC curves and AUC scores. The ROC curve and AUC score are estimated using validation dataset, i.e. 1,000 trucks with equal split between failed and censored batteries.

9.1. Performance of RSF-based models

Table 2 summarizes AUC scores for the standard Cox regression, the RSF model with only one data readout, and four RSF models with augmented feature space introduced in Section 7. The Cox regression model is estimated using the *Python* package for survival analysis *lifelines* (Davidson-Pilon & et al., 2019). All RSF models are estimated using the *R* package (Ishwaran & Kogalur, 2007) and trained with number of trees ntree = 2000, average number of unique samples in the terminal node nodesize = 200, candidates variables selected for split mtry = $\sqrt{p}$, and logrank as the splitting rule.

Here and throughout the paper, the difference in the performance is measured in terms of relative change. It can be seen in Table 2 that all RSF models are doing significantly better than the Cox regression model, in particular the RSF model with only one data readout is about 14% better than a Cox regression model. Note that the performance of Cox regression is close to random guess. This is consistent with the observation in Voronov et al. (2018a) that the vehicle fleet data does not seem to fulfill the proportional hazards assumption. Comparing the RSF models that have an augmented feature space with the RSF model that have only one last readout, leads us to conclude that the feature augmentation does not improve model performance. One possible reason for this result is the accumulative nature of the histogram variables, i.e. information is accumulated from the beginning of the lifetime of the battery until the time of data retrieval.

In the case of RSF models with derivatives and differences in time and mileage that are computed with respect to the last readout, the performance of the models is similar to the one with only one data readout with AUC values of 0.69 and 0.675 respectively. The performance of the models is lower in the case of derivatives and differences in time and mileage, which are computed between neighboring readouts, with AUC values of 0.663 and 0.63 respectively. At this point one can think that having extra readouts does not lead to improved prediction performance of the models. However, as will be shown below, a reason can be irregular, non-equidistant vehicle operational data.

9.2. Performance of LSTM-based models

Several kinds of LSTM models were introduced in Section 8 by choice of preprocessing, mini-batch formation during training, choice of target function, and approach to balancing data. Table 3 summarizes performance for several different architectures and options. The models are grouped by type of input data, i.e., raw or preprocessed, and by type of target functions,

Table 2. AUC scores for RSF models with multiple data readouts.

Model	AUC
Cox regression	0.596
RSF only last readout	0.681
RSF with an augmented readout as in (9)	0.63
RSF with an augmented readout as in (12)	0.663
RSF with an augmented readout as in (13)	0.69
RSF with an augmented readout as in (10)	0.675

i.e., lifetime or reliability function. Every model in the table represent one of the possible approaches mentioned in Section 8 and can address, for example, an approach to form a batch, an approach for data balancing, etc. All models were implemented in python using the libraries *Keras* and *Tensorflow*.

Table 3. Area under the ROC curve values for different LSTM networks. Possible values for the model characteristics: type : {S - single LSTM model, E - ensemble LSTM model, L - LSTM model with input of $\Delta t^j_{\text{last},i}$ and $\Delta m^j_{\text{last},i}$ after recurrent layers, O - LSTM model with target function only at the last time step, M - multilayer perceptron model}; batch size : numerical value; forming batch : {RB - standard batch sampling, CS - sampling class according to frequency of data, ZP - zero padding}; epochs : numerical value; data balancing : {EN - ensemble, WL - weighted loss, NB - no balancing}; imputation : {M - mean imputation, MOTI - imputation with mean computed within groups, i.e., time intervals}

#	Model						AUC	
	type	batch size	forming batch	epochs	data balancing	imputation	raw inputs	std inputs
Lifetime functions as targets								
1	S	1	RB	1000	NB	M	0.570	0.70
2	S	128	ZP	1000	WL	M		0.777
3	S	128	ZP	1000	NB	M	0.718	0.793
4	S	128	CS	1000	NB	M	0.738	0.794
5	E	128	ZP	1000	EN	M	0.730	0.806
6	S	128	ZP	1000	NB	MOTI		0.816
7	E	128	ZP	1000	EN	MOTI		0.829
Reliability functions as targets								
8	S	128	ZP	1000	NB	M	0.670	0.693
9	M	128	RB	300	EN	M		0.769
10	O	128	ZP	1000	EN	M	0.713	0.775
11	E	128	ZP	1000	EN	M	0.717	0.777
12	L	128	ZP	1000	EN	M	0.720	0.779

One conclusion that can be made so far is that preprocessing data, in our case standardizing histogram variables, leads to significantly improved performance. For example, consider model 5 in the table, which is an ensemble of LSTM networks with batch size 128 where readouts for vehicles that

have less than 3 readouts are zero-padded and the model is trained for 1,000 epochs. Both AUC values for raw and standardized inputs are larger than corresponding values of RSF models. It is evident, not surprising, that the preprocessing step is important for the model performance.

Now, consider how different approaches to form a mini-batch in training influence the performance. For this purpose consider models 1, 3 and 4. The architecture of the LSTM network is the same for all those models, the only difference is batch size and the approach used to form mini-batches. It is clear that batch size 1 gives the worst performing model. The two other approaches give comparable results in terms of AUC values. There is a small difference when one considers the models with raw inputs, on the other hand the results are similar when inputs are standardized. Therefore, all remaining models in the table use zero padding to form mini-batches, except model 9 which is baseline ensemble multilayer perceptron model with 5 networks from (Voronov et al., 2018b).

Next architectural choice to explore is the choice of target function, lifetime or reliability function, and in particular performance for imbalanced data. For this, consider the pairs of models $(3, 5)$ and $(6, 7)$ where model 3 and 6 use a single LSTM network and model 5 and 7 use an ensemble of LSTM networks. The missing data in pair $(3, 5)$ is imputed using a mean over all data readouts while in pair $(6, 7)$ the mean is computed within groups partitioned in time intervals. The ensembles of LSTM networks perform better than a single model by approximately 1.6%. Now, consider the difference in performance when reliability functions are used as targets. For this, consider the pair of models $(8, 11)$. The difference in performance between an ensemble and a single model is now 12,1%, which almost 10 times the difference in the case of lifetime function as target. A similar difference in performance between an ensemble and a single model is detected for multilayer perceptron model with a single readout, see (Voronov et al., 2018b). A conclusion from this is that using lifetime function as a target in the LSTM networks seems to handle imbalanced data better than with the reliability target functions since with only a single LSTM network the performance is close to the performance of the ensemble. Overall, the ensemble of LSTM networks with the lifetime functions as targets, model 5, perform better than model 11 with reliability functions as targets.

An observation, by comparing models 2 and 5, are that balancing the data with weights in the loss functions as in (18) gives worse performance than balancing with an ensemble. Further, injecting time and mileage difference after the recurrent layers do not change prediction performance which can be seen by comparing models 11 and 12.

A majority of the LSTM models which are used here have targets present for every time step/input readout in LSTM network, see Figure 11. For the case of reliability functions this procedure is similar to the technique called target replication which is used by authors in (Lipton et al., 2015). It is reported that using this technique in classification problems improves the model performance. In this paper one model that uses an LSTM network with target reliability function for only the last time step in the network is considered as model 10 in Table 3. The results show, if models 10 and 11 are compared, that replicating target reliability function for every time step/input readout in the network does not lead to the improved performance here.

All models in Table 3 except model 9, the model from (Voronov et al., 2018b), are trained for 1,000 epochs and maximum performance on the validation set is achieved at an earlier epoch for every model. Therefore, early stopping, as presented in Figure 15, is employed and the suggested number of epochs for early stopping is about 230 that is indicated by dashed blue line. The four most promising models from Table 3 are selected and retrained with early stopping and the results are reported in Table 4.

9.3. Best performing model

The best performing model among all of the considered is an ensemble of LSTM networks where missing data is imputed with a mean of available readouts within groups which are based on time as in Section 6. Batch size of 128 is used in training, zero-padding of non-existing readouts, and training for 110 epochs. The model achieves an AUC score of 0.851 on the test dataset. Below, the full set of steps for training the model is summarized in the procedure.

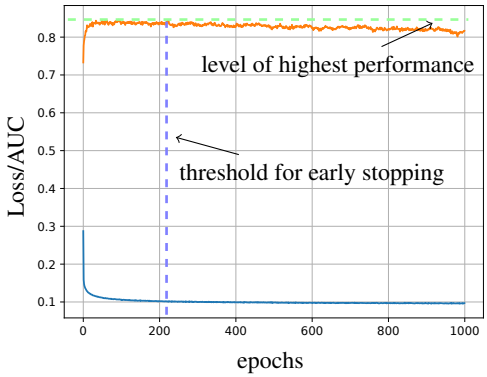


Figure 15. Demonstration of the loss and AUC curves for model 6 in Table 3 during the process of training the model. Blue curve is a loss curve and orange curve is AUC curve. Blue dashed line denotes the suggested number of epochs when to stop training the model. Green dashed line shows the highest AUC value which is reached during the training.

Figure 16 shows histograms of the lifetime function values

Table 4. Area under the curve values for LSTM models re-trained with early stopping.

Model						AUC
type	batch size	forming batch	epochs	data balancing	imputation	
Reliability functions as targets						
L	128	ZP	100	EN	M	0.787
Lifetime functions as targets						
E	128	ZP	80	EN	M	0.818
S	128	ZP	230	NB	MOTI	0.832
E	128	ZP	110	EN	MOTI	0.843

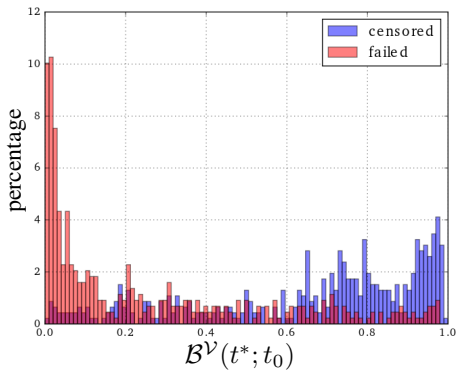


Figure 16. Histograms of lifetime functions at survival time t^* for failed batteries, red bars, and censored batteries, blue bars, for the best performing model.

$\mathcal{B}^{\mathcal{V}}(t;t_0)$ for failed and censored batteries on the test dataset. The lifetime function $\mathcal{B}^{\mathcal{V}}(t;t_0)$ is evaluated at survival time t^*, i.e., the time when the battery either fails or leaves the study without any problem. The figure shows that, as expected, on average failed batteries have lower values of the lifetime function than censored. The corresponding ROC curve, computed based on values of the histograms, is shown in Figure 17 which corresponds to a maintenance policy based on the value of the lifetime function. Taking into account uncertainties in the data, lack of a battery health indicator, the fact that data is not initially intended for prognostic purpose, and the complex nature of battery degradation, the performance of the best model is regarded as an important result with significant improvement compared to time- or mileage-based maintenance as well as standard methods such as Cox regression.

10. CONCLUSIONS

This paper addresses the problem of predicting failures of lead-acid batteries of heavy-duty vehicles based on sparse operational data. Sparse means that few, possibly as few as one, data readouts from a vehicle is available to the prognostics algorithms. The data are mainly accumulated sensor readings collected during the battery lifetime. Data is typically recorded during irregular workshop visits. The data structure motivates the choice of the probabilistic framework to predict battery failure. The proposed method is not tailored to batteries in particular, but can be applied to other components as well as long as the data is similar to the case studied here.

Procedure: building predictive model
Data: *Sparse vehicle operational data*
Steps to build the best performing predictive model:
Imputation:
1 Create groups which are based on some criterion, for example time in our case.
2 Assign every data readout of each vehicle to one of the groups.
3 If a group contains more than one readout from a vehicle, keep only the latest readout.
4 In every group find the mean of non-missing variables and use it to fill the missing values.
Input variables:
5 Encode discrete variables with 1-of-(C-1) encoding.
6 Standardize histogram variables.
Targets:
7 Build the target lifetime function for every data readout of every vehicle.
Data balancing:
8 Randomly split censored batteries into 5 equal in size data sets.
9 Create 5 new training data sets by combining the set of failed batteries with one of the 5 sets of censored batteries.
Ensemble of LSTMs:
10 Build 5 LSTM networks with 2 hidden layers.
11 Use 64 hidden states in the LSTM cells.
12 Set the dropout rate for linear transformations of inputs to 0.5 and the dropout rate for linear transformations of internal states to 0.4
13 Train the networks for 110 epochs with batch size 128.

Two methods for predicting probability of component failure are proposed and compared, RSF and LSTM-based neural networks that incorporate the structure of sparse vehicle operational data into their architectures. The available data contains several uncertainties where one is a high rate of non-random missing values. One result is that a mean imputation approach to impute missing data gives comparable performance to more involved imputation algorithms and this is consistent with that the non-random missing data behavior. Models based on neural networks give significantly better performance than RSF-based models, both in the case of only one data readout and multiple data readouts per vehicle. In turn, both models are better than a standard non-parametric algorithm such as Cox regression used as a baseline comparison. The LSTM-based neural network model requires careful handling of the imbalanced data. In the data set used in the study, vehicles with functioning batteries are 5 times more

common than vehicles with failed batteries and this lead to an ensemble of balanced predictive models.

Key research questions of this work are how to handle the sparse data and the effects of the accumulative nature of sensor readings. On the one hand it is shown that having more frequent and regular readouts will lead to improved predictive performance of the models, which is natural since more information is provided for the predictive model. This result helps in decision making process regarding a new data collection approach. At the same time, due to the accumulative nature of sensor readings from a vehicle, there is no need to perform data logging at a high rate, e.g., every minute or hour. Performing a data readout from a vehicle once in, e.g., a couple of months is enough for a quality predictive model. This result significantly reduces amount of transmitted information from a vehicle to a database which in turn reduces the cost of equipment on the vehicle.

ACKNOWLEDGMENT

We acknowledge Scania CV and FFI (Vehicle Strategic Research and Innovation) for partial funding of this work.

NOMENCLATURE

AUC	area under the curve
CS	class sampling according to frequency of data
EMLP	ensemble multilayer perceptron
ETL	extract, transfor, load
KL	Kullback-Leiber divergence
LSTM	long short-term memory
MissForest	random forest-based imputation technique
MLP	multilayer perceptron
MOTI	imputation with mean computed within groups
mtry	number of candidates variables selected for split in a random survival forest model
nodesize	number of unique samples in a terminal node of a random survival forest model
NB	no balancing
ntree	number of trees in a random survival forest model
OTFI	on-the-fly-imputation
PDF	probability density function
RB	standard batch sampling
RF	random forest
RNN	recurrent neural networks
ROC	receiver operating characteristic
RSF	random survival forest
RUL	remaining useful life
WL	weighted loss
ZP	zero padding

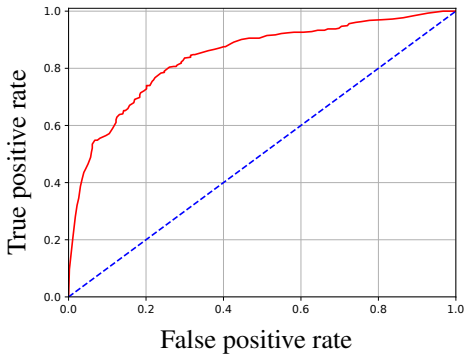


Figure 17. Receiver operating characteristic curve, red curve, for the best performing model. Blue dashed line is random guess performance.

REFERENCES

Ahmad, W., Khan, S. A., & Kim, J.-M. (2017). A hybrid prognostics technique for rolling element bearings using adaptive predictive models. *IEEE Transactions on Industrial Electronics*, *65*(2), 1577–1584.

Alghassi, A., Perinpanayagam, S., & Samie, M. (2015). Stochastic rul calculation enhanced with tdnn-based igbt failure modeling. *IEEE Transactions on Reliability*, *65*(2), 558–573.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Taylor and Francis.

Chollet, F. (2015). *Keras*. https://github.com/fchollet/keras. GitHub.

Ciampi, A., Thiffault, J., Nakache, J.-P., & Asselain, B. (1986). Stratification by stepwise regression, correspondence analysis and recursive partition: A comparison of three methods of analysis for survival data with covariates. *Computation Statistics and Data Analysis*, *4*, 185-205.

Cox, D. (1972). Regression model and life-table. *Journal of the Royal Statistical Society*, *34*(2), 187-220.

Cox, D. R., & Oakes, D. (1984). *Analysis of survival data* (Vol. 21). CRC Press.

Daigle, M. J., & Goebel, K. (2011). A model-based prognostics approach applied to pneumatic valves. *International journal of prognostics and health management*, *2*(2), 84–99.

Davidson-Pilon, C., & et al. (2019, October). *Lifelines: v0.22.8*. https://lifelines.readthedocs.io/. Zenodo.

Elsheikh, A., Yacout, S., & Ouali, M.-S. (2019). Bidirectional handshaking lstm for remaining useful life prediction. *Neurocomputing*, *323*, 148–156.

Fawcett, T., & Flach, P. A. (2005). A response to webb and ting's on the application of roc analysis to predict classification performance under varying class distributions. *Machine Learning*, *58*(1), 33–38.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1) (No. 10). Springer series in statistics New York.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., ... Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, *106*(9-10), 1469–1495.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Hanachi, H., Liu, J., Banerjee, A., Chen, Y., & Koul, A. (2015). A physics-based modeling approach for performance monitoring in gas turbine engines. *IEEE Transactions on Reliability*, *64*(1).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Ishwaran, H., & Kogalur, U. (2007). Random survival forests for r. *Rnews*, *7/2*, 25-31.

Ishwaran, H., Kogalur, U., Blackstone, E., & Lauer, M. (2008). Random survival forests. *The Annals of Applied Statistics*, 841–860.

Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzel, R. (2015). Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*.

Rosset, S., & Inger, A. (2000). Kdd-cup 99: knowledge discovery in a charitable organization's donor database. *SIGKDD Explorations*, *1*(2), 85–90.

Saha, B., Goebel, K., & Christophersen, J. (2009). Comparison of prognostic algorithms for estimating remaining useful life of batteries. *Transactions of the Institute of Measurement and Control*, *31*(3-4), 293–308.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Tang, F., & Ishwaran, H. (2017). Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, *10*(6), 363–377.

Van Buuren, S. (2007). Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical methods in medical research*, *16*(3), 219–242.

Voronov, S., Frisk, E., & Krysander, M. (2018a). Data-driven battery lifetime prediction and confidence estimation for heavy-duty trucks. *IEEE Transactions on Reliability*, *67*(2), 623–639.

Voronov, S., Frisk, E., & Krysander, M. (2018b). Lead-acid battery maintenance using multilayer perceptron models. In *2018 ieee international conference on prognostics and health management (icphm)* (pp. 1–8).

Webb, G. I., & Ting, K. M. (2005). On the application of roc analysis to predict classification performance under varying class distributions. *Machine learning*, *58*(1), 25–32.

Zhang, Y., Xiong, R., He, H., & Pecht, M. G. (2018). Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, *67*(7), 5695–5705.