Discovering Premature Replacements in Predictive Maintenance Time-to-Event Data

Abdallah Alabdallah[1], Thorsteinn Rögnvaldsson[2], Yuantao Fan[3], Sepideh Pashami[4], and Mattias Ohlsson[5]

[1,2,3,4,5] *Center for Applied Intelligent Systems Research (CAISR), Halmstad, Sweden*
[abdallah.alabdallah, thorsteinn.rognvaldsson, yuantao.fan, sepideh.pashami, mattias.ohlsson]@hh.se

ABSTRACT

Time-To-Event (TTE) modeling using survival analysis in industrial settings faces the challenge of premature replacements of machine components, which leads to bias and errors in survival prediction. Typically, TTE survival data contains information about components and if they had failed or not up to a certain time. For failed components, the time is noted, and a failure is referred to as an *event*. A component that has not failed is denoted as *censored*. In industrial settings, in contrast to medical settings, there can be considerable uncertainty in an event; a component can be replaced before it fails to prevent operation stops or because maintenance staff believe that the component is faulty. This shows up as "no fault found" in warranty studies, where a significant proportion of replaced components may appear fault-free when tested or inspected after replacement.

In this work, we propose an expectation-maximization-like method for discovering such premature replacements in survival data. The method is a two-phase iterative algorithm employing a genetic algorithm in the maximization phase to learn better event assignments on a validation set. The learned labels through iterations are accumulated and averaged to be used to initialize the following expectation phase. The assumption is that the more often the event is selected, the more likely it is to be an actual failure and not a "no fault found".

Experiments on synthesized and simulated data show that the proposed method can correctly detect a significant percentage of premature replacement cases.

1. INTRODUCTION

Maintenance plays a vital role in industrial operations. Aiming at increasing the reliability and the cost-efficiency of industrial systems, predictive maintenance (PdM) techniques, e.g. machine prognostics, have received much attention in recent years. The goal of PdM is to conduct maintenance service in a proactive way so that both the downtime and the maintenance cost of industrial systems are reduced, fully utilizing the lifetime of the equipment and fixing problems before they lead to stops in operation. Machine prognostic methods aim at modeling the reliability of the equipment, using, e.g., Survival Analysis or methods for predicting Remaining Useful Life (RUL). In recent years, this has often been done with machine learning models.

To develop prognostic methods, machine failure data, e.g., sensor readings and the time of the failure event, is required. However, in most industrial applications, equipment is not allowed to run to failure. Under circumstances when the consequences of unplanned downtime are safety-critical or associated with a high cost, preventive maintenance is likely to be implemented. Maintenance personnel will not wait until the full lifetime of a component is exhausted but replace the component before the end-of-life, especially if incipient or intermittent faults have been observed.

Because of the increasing complexity of industrial equipment, it is challenging to acquire thorough coverage in understanding all types of abnormal behavior. With a preventive mindset, abnormal behaviors observed from the operation have an increased chance of being treated as indicating a risk to the operation, and thus, the equipment is scheduled for repair, in many cases, replacement. Such premature replacement of components showing (occasional) abnormal behavior should improve the overall reliability of the system, but it comes with increased maintenance expenses and, more importantly, it introduces significant noise into the TTE data, making it more difficult to improve the prognostic models.

The replaced equipment being determined as healthy in post-replacement tests is referred to as one type of the "no fault found" (NFF) phenomenon (Khan, Phillips, Hockley, & Jennions, 2012), i.e., a false alarm. Common factors causing a component to be replaced prematurely due to a false alarm include a lack of understanding of the component and proper testing methods, lack of training in the workshop, misreporting and -communication, etc.

Abdallah Alabdallah et al. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Survival models are statistical methods used to analyze the time for an event of interest to occur, such as a patient's death or a machine component's failure. Such models endeavor to learn the distribution of events through modeling the Survival function $S(t) = P(T > t)$ referring to the probability of surviving beyond time t, or the hazard function $h(t)$ corresponding to the instantaneous failure rate. Several statistical and machine learning methods have been proposed for survivability prediction, e.g. (Cox, 1972; Ishwaran, Kogalur, Blackstone, & Lauer, 2008; Katzman et al., 2018). In this work, we utilize the Cox Proportional Hazards model CPH (Cox, 1972) as it is the fastest to train, making it a suitable surrogate in our algorithm, which requires numerous iterations of fitting and evaluation. The CPH method is a semiparametric method that models the hazard function $h(t|\mathbf{x}) = h_0(t)e^{\mathbf{w}^\intercal \mathbf{x}}$ conditioned on the covariates $\mathbf{x}$, where $\mathbf{w}$ is a vector of parameters (the weight vector) and $h_0(t)$ is the baseline hazard function.

The strength of survival models is the ability to handle censored data, which refers to the case when some subjects under study did not experience the event (the "death") during the study period. Such censored subjects contain information that they survived past the recorded time, called censoring time. The concept of censoring is useful when dealing with premature replacements and NFF cases. From the survival model's perspective, an event that corresponds to a premature replacement should actually be labeled as a censored case; the equipment was taken out of the study when it was replaced, but it survived beyond that point. Such incorrectly labeled events affect the ranking performance of the survival model, as will be shown experimentally in the results section, which we utilize in this work to uncover the incorrectly labeled events.

In this work, we propose SurvPRD, a new iterative algorithm for discovering premature replacements. The algorithm relies on survival analysis and the concept of censoring as a surrogate to identify premature replacements. It iteratively utilizes genetic algorithms to improve the suggested solution over iterations. To the best of our knowledge, this is the first method that makes the link between premature replacements and censored subjects and utilizes survival analysis to uncover these cases. The code is available on our GitHub repositoy[1]

2. Literature Review

Since survival modeling relies on the time for an event of interest (in our case, replacing worn-out components) to occur, premature replacement in the dataset would affect the modeling in a way similar to mislabeled data samples. There are two general approaches to tackling mislabelled data: i) improving learning algorithms to better tolerate noisy labels; ii) filtering out or correcting potential mislabelled data before the training process starts (Guan & Yuan, 2013). The latter approach is preferable since it is not designed to enhance one specific learning algorithm, and the output, i.e., identified mislabeled data, can be utilized for any algorithm. Most mislabeled data identification methods fall into the following categories: i) Nearest Neighbour based approaches that assume local smoothness in the data (Wilson, 1972; Sánchez, Barandela, Marqués, Alejo, & Badenas, 2003); ii) Majority voting and consensus filtering based on ensembles of multiple trained predictors (Brodley & Friedl, 1999; Zhu, Wu, & Chen, 2003); iii) learning algorithm specific approaches that utilize the property of the predictor, e.g., (Zeng & Martinez, 2003). In addition, genetic algorithms have been applied to iteratively search for the best subset of learning samples that optimize over specific criteria, e.g., maximizing the statistical separability between classes (Ghoggali & Melgani, 2009; Pasolli & Melgani, 2015). Nevertheless, both methods use human experts to validate/invalidate examples. TTE data, the focus of this work, have a special type of labels, where it comes in tuples; the time and the event indicator. To the best of our knowledge, our study is the first to identify premature replacement in repair events based on genetic algorithms and survival modeling, treating the premature replacement as an incorrect event label. Moreover, our proposed method relies on aggregating the optimized solutions' history to gain better confidence in the labels after each iteration.

3. Method

The method is a two-phase iterative algorithm, with the first phase being the Expectation phase and the second being the Maximization phase. The intuition behind the algorithm is to repeatedly find the set of event labels that maximizes the performance of the survival model. Starting from a random initialization of the labels, and aggregating the history of the found solutions over iterations provide statistics of each label. This leads to a more probable solution after each iteration, converging to the final solution after some iterations. The algorithm splits the dataset into two sets. In the Expectation phase, a survival model is trained on the first set with the latest found event labels (the aggregated history of the found labels in the previous iterations). In the Maximization phase, a genetic algorithm is used to search for event labels that maximize the performance on the second set. The two sets are then switched, using the second set and its aggregated history to train the model and the first set to search for the labels. The two phases repeat for a number of iterations until convergence to the final solution.

Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be the input features matrix of n samples of p dimensions, $\mathbf{T} \in \mathbb{R}^{n \times 1}$ the censored time vector, and $\mathbf{E} \in \{0, 1\}^{n \times 1}$ is the event indicator vector. Denoting an element in $\mathbf{E}$ as e_k, $e_k = 1$ indicates that the recorded time t_k is an event, i.e., a component was replaced, and $e_k = 0$ indicates that the component survived beyond t_k, i.e. a censoring time. The problem we are targeting is that a fraction c of the replacements (recorded events) in the workshop repair data

[1] https://github.com/abdoush/SurvPRD

are actual failures, while the rest are premature replacements and should be considered censored cases. From that perspective, our algorithm only switches the recorded event cases, while the originally censored cases are considered censored in all the iterations of the algorithm and not part of the optimization process. To simplify the notations in this section, the originally censored part of the data is omitted. Without loss of generality, we assume that all the times are recorded as observed events. However, in practice, the originally censored part of the data is concatenated to the output of the Maximization phase.

The algorithm, see Algorithm 1, first splits the dataset $(\mathbf{X}, \mathbf{T})$ into two sets: Set A $(\mathbf{X}_A, \mathbf{T}_A)$ and Set B $(\mathbf{X}_B, \mathbf{T}_B)$, between which the algorithm alternates during the two phases. Let H_A and H_B be the histories used to store the found labels over iterations for Set A and Set B, respectively. $\mathbf{E}_A^0$, the initial event labels for Set A are randomly initialized and added to H_A. The algorithm iterates for N_{iter} number of iterations. At iteration i, the two phases are:

Expectation: In the Expectation phase, a CPH model M_i is trained on the set $(\mathbf{X}_A, \mathbf{T}_A, \mathbf{E}_A)$, where $\mathbf{E}_A = \mathbb{1}(\mathbf{E}_A^p \geq (1 - P^c(\mathbf{E}_A^p)))$ is the accumulated and thresholded results for Set A, $\mathbf{E}_A^p = \mathbb{E}_{\mathbf{E}_A^j \in H_A}(\mathbf{E}_A^j)$, and $P^c(.)$ is the c^{th} percentile where c is the fraction of the correctly labeled event. Thresholding with the complement of the c^{th} percentile ensures that the events' fraction converges to c in the final result.

Maximization: In this phase, a genetic algorithm is utilized to search for events' assignments $\mathbf{E}_B^i$ that maximizes the performance of the model M_i on Set B. The history of Set B is then updated with the value $\mathbf{E}_B^i$. The algorithm then switches the two datasets and repeats until the maximum number of iterations is reached. At the last iteration, the algorithm concatenates the two histories H_A and H_B into H and the final output $\mathbf{E} = \mathbb{1}(\mathbf{E}^p \geq (1 - P^c(\mathbf{E}^p)))$ where $\mathbf{E}^p = \mathbb{E}_{\mathbf{E}^j \in H}(\mathbf{E}^j)$.

Algorithm 1 SurvPRD

Input: $X \in \mathbb{R}^{n \times p}$, $T \in \mathbb{R}^{n \times 1}$
Input: c the fraction of correctly labeled events
Split Data : $(\mathbf{X}_A, \mathbf{T}_A), (\mathbf{X}_B, \mathbf{T}_B) \leftarrow (\mathbf{X}, \mathbf{T})$
1) Initialization:
$\mathbf{E}_A^0 \leftarrow$ *Random*
Update the history: $H_A \leftarrow \mathbf{E}_A^0$
for $i \in \{0, \dots N_{iter}\}$ **do**
  1) Expectation:
  Accumulate the history: $\mathbf{E}_A^p \leftarrow \mathbb{E}_{\mathbf{E}_A^j \in H_A}(\mathbf{E}_A^j)$
  Thresholding: $\mathbf{E}_A = \mathbb{1}(\mathbf{E}_A^p \geq (1 - P^c(\mathbf{E}_A^p)))$
  Train $\mathbf{M}_i(\mathbf{X}_A, \mathbf{T}_A, \mathbf{E}_A)$
  2) Maximization:
  $\mathbf{E}_B^i \leftarrow argmax_{\mathbf{E}_B^i} \mathbf{M}_i(\mathbf{X}_B, \mathbf{T}_B, \mathbf{E}_B^i)$
  Update the history: $H_B \leftarrow \mathbf{E}_B^i$
  if $i < N_{iter}$ **then**
    $(\mathbf{X}_A, \mathbf{T}_A, H_A) \leftarrow SwitchSets \rightarrow (\mathbf{X}_B, \mathbf{T}_B, H_B)$
  end if
end for
$H \leftarrow Concat(H_A, H_B)$
Accumulate the history: $\mathbf{E}^p \leftarrow \mathbb{E}_{\mathbf{E}^j \in H}(\mathbf{E}^j)$
Thresholding: $\mathbf{E} = \mathbb{1}(\mathbf{E}^p \geq (1 - P^c(\mathbf{E}^p)))$

4. Experiments and Results

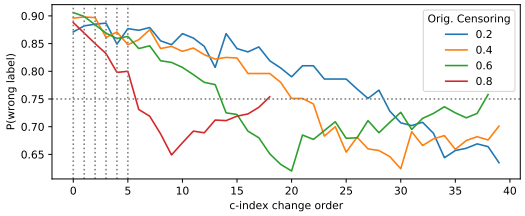
Four experiments were conducted. The purpose of the first two experiments was to study the behavior of the survival model and the C-index metric with respect to erroneous labeling and illustrate that they can be used to guide the genetic algorithm towards a more likely solution. In the third experiment, we ran our algorithm on simulated data with different fractions of correct labeling. The simulated data were created following the simulation study in (Pölsterl, 2020). The datasets consist of varying number of examples with proportional hazards, four covariates with hazard ratios ranging from 1 to 4, and a baseline hazard of 0.1. Survival times are drawn from an exponential distribution linearly related to the log of the hazard ratios. In the last experiment, we ran our algorithm on the C-MAPSS dataset, which we modified to simulate premature replacements. In all experiments, the same hyperparameter values were used for the genetic algorithm: a population size of 100 individuals with a 100% crossover probability, a 50% individual mutation probability, and a 10% bit mutation probability.

4.1. Concordance Index as a Guiding Metric

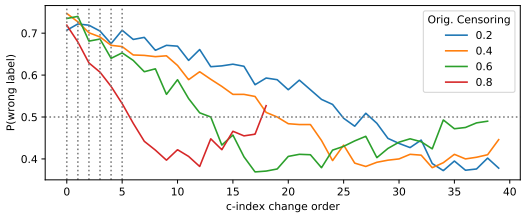
Survival models are usually evaluated using the concordance index (C-index) (Harrell, Califf, Pryor, Lee, & Rosati, 1982), quantifying the rank correlation between survival times and the model's predictions. In this experiment, three datasets of 100 samples with different fractions of correct labels were created corresponding 25%, 50%, and 75% correctly labeled examples. In each of the three datasets, the original censoring percentage ranged from 20% to 80%. In each experiment, a CPH model was trained using the incorrectly labeled data and recorded the C-index score of the model. We flipped one label at a time, computed the difference in C-index performance before and after flipping the label, and ranked the example from highest to lowest change in the C-index (0 being the rank of the biggest change). We repeated each experiment 100 times to get statistics on the change. In every repetition, a different dataset was created with the same characteristics.

Figures 1a, 1b, 1c show plots corresponding to the main three experiments with 25%, 50%, and 75% correctly labeled examples, respectively. In each plot, the curves correspond to different original censoring levels, ranging from 20% to 80%, where the y axis in each curve represents the fraction of times, out of 100 runs, that the flipped label was a wrong label, and the x axis is the rank-order of the change caused by the flip. The horizontal dotted line is the random guess probability of flipping a wrong label.

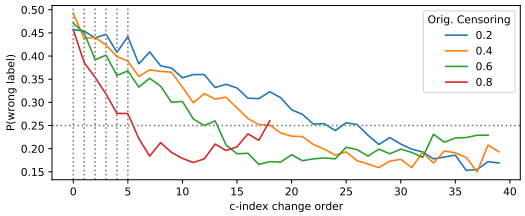
The results show that $P(\texttt{wrong label})$, the probability that the flipped label was wrong, is higher at the highest rank of the change in the C-index, i.e., the flip that caused the biggest change in the C-index. They also show that the probability drops with lower-ranked changes. In other words, correcting an erroneous label (changing an incorrectly labeled event into a censored case) is more likely to lead to the biggest change in the C-index performance of the model. This indicates that the C-index measure can be used to lead the genetic algorithm towards more correct labeling.



(a) 25% Correctly Labeled



(b) 50% Correctly Labeled



(c) 75% Correctly Labeled

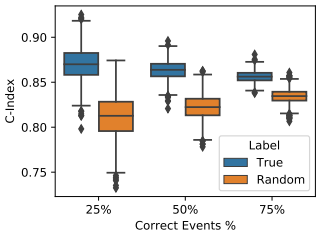
Figure 1. The probability that the largest C-Index change comes from flipping an erroneously labeled event, at different censoring levels and different correctly labeled events%.

4.2. Survival Model and Noisy Labels

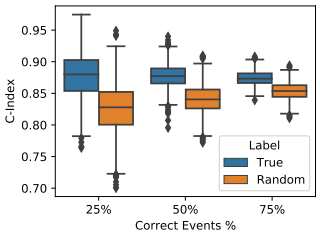
In this experiment, we analyze the performance of the CPH survival model under noisy labels. The idea behind this experiment is to show that the model performance on noisy labels, i.e., trained and tested on noisy labels, is more likely to be worse than on correct labels. This then indicates that the survival model performance can be used as the genetic algorithm's fitness to guide it towards better labeling.

We performed three scenarios, corresponding to 25%, 50%, and 75% correctly labeled instances. Each scenario was repeated $1,000$ times. At each run, a dataset of $1,000$ examples and 50% originally censored cases was created, and a CPH model was trained and tested on both the correct labels and random assignments of the labels. Figures 2a and 2b show the train and test performance, respectively, comparing the models' performances in the three scenarios. The results in Figure 2 show that the model's performance on the true labels is significantly better than on the random labels. Statistically, from the $1,000$ runs, we can compute the probability that training and testing on the correct labels has a better performance than training and testing on random labels $P(\texttt{True > Random})$, which corresponds to (97%, 98%, and 98%) on the training sets and (84%, 90%, and 86%) on the testing sets in the three scenarios of 25%, 50%, and 75% correctly labeled instances, respectively.

These results suggest the plausibility of using the survival model's performance as a fitness function of the genetic algorithm and maximizing the performance of the models leads to more correct assignments of the events' labels.



(a) Train



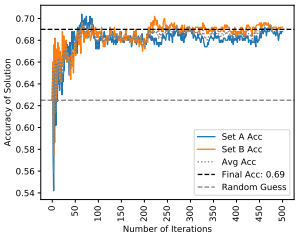
(b) Test

Figure 2. Difference between the CPH model performance with correct labels vs. random labels, at different levels of correctly labeled events%.

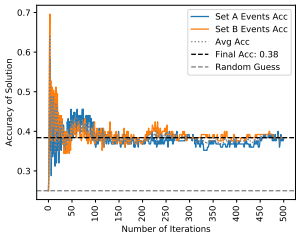
4.3. SurvPRD with Different Fractions of Correct Labels

This experiment shows the results of running the algorithm for $1,000$ iterations (500 iterations on each set) on three simulated datasets of variable fractions of correct labels. In each case, the dataset consists of $2,000$ examples with an original censoring level of 50%. Out of the remaining 50% observed event examples, the three datasets contain different percentages of correctly labeled examples. Figures 3, 4, and 5, show the results of the three experiments with 25%, 50%, and 75% correctly labeled examples, respectively.

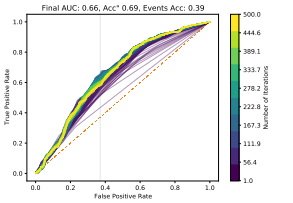
The accuracy on all samples is computed as Accuracy $= \sum_{i\in 0,1} P(\hat{e}=i|e=i)P(e=i)$. While the accuracy on the event samples only is computed as Events Accuracy $= P(\hat{e}=1|e=1)$. The random guess accuracy, illustrated in Figures 3, 4, and 5 as a horizontal gray dotted line, is the accuracy of a random classifier. It is computed for each case based on the probability of correct labeling, i.e., the fraction of correct labels in each dataset. This results in a random guess total accuracy and events' accuracy of (62.5%, 25%), (50%, 50%), (62.5%, 75%) in the three cases of 25%, 50%, 75% correct labels, respectively.



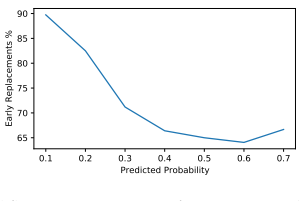
(a) Accuracy over iterations



(b) Events acc over itrations

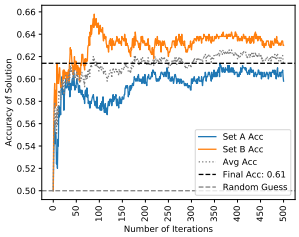


(c) AUC over iterations

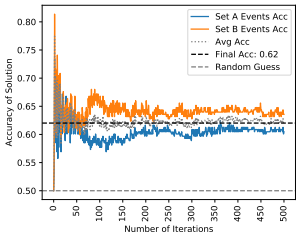


(d) Premature replacements % per confidence

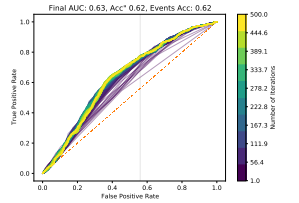
Figure 3. Results of running the algorithm 1,000 iterations on a simulated dataset with 25% correctly labeled events.



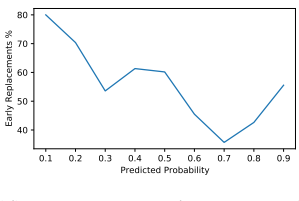
(a) Accuracy over iterations



(b) Events acc over itrations



(c) AUC over iterations



(d) Premature replacements % per confidence

Figure 4. Results of running the algorithm 1,000 iterations on a simulated dataset with 50% correctly labeled events.

In the three experiments with 25%, 50%, and 75% correct labeling, the algorithm successively improved the accuracy and the AUC over iterations and converged towards accuracies of 69.0%, 61.4%, and 70.4%, as shown in Figures 3a, 4a, 5a, corresponding to AUCs of 66%, 63%, and 65%, respectively. Figures 3b, 4b, and 5b, show the Event accuracies over iterations which converged to 38.4%, 62.0%, and 80.4% in the three experiments respectivly. Event accuracy fluctuations during early iterations is due to the accumulation process of the candidate solutions during iterations, which causes the percentage of events to fluctuate a lot until enough knowledge is gathered, at which stage the event percentage converges to the desired value, which is the percentage of the correctly labeled events. Details are shown in Table 1.

Table 1. Results of the experiments on the three simulated datasets with 25%, 50%, 75% correct labels.

Correctly Labeled Events%	25%	50%	75%
Total Accuracy %	69.0	61.4	70.4
Total Rand. Acc %	62.5	50.0	62.5
Events Acc %	38.4	62.0	80.4
Events Rand. Acc %	25.0	50.0	75.0
Premature Replacements Acc %	79.2	60.8	40.4
Premature Replacements Rand. Acc %	75.0	50.0	25.0

Figures 3d, 4d, 5d, show the percentage of actual premature replacements against the predicted probability of an instance. The higher the predicted probability, the higher the confidence that an instance is an actual event (a genuine component failure). The lower the predicted probability, the higher the confidence that an instance is a premature replacement. The figures show that the percentage of actual premature replacements is highest in examples where the model is confident that they are premature replacements (the lowest predicted probabilities).

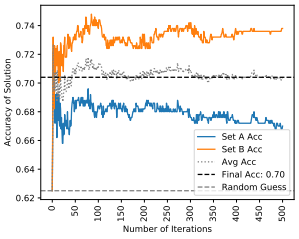
4.4. Turbofan Engine Dataset Use Case

The NASA C-MAPSS engine degradation data has four sets of simulated data, for different combinations of operational conditions and fault modes (Saxena, Goebel, Simon, & Eklund, 2008). We used the set with two failure modes. We simulated the censoring by limiting the observation period to 300 cycles resulting in an original censoring of 20.41%. We subsampled the dataset, selecting 98 engines with 20 readings per engine, resulting in 1,960 samples. Finally, we simulated a scenario where 50% out of the 79.59% recorded events were incorrectly labeled.

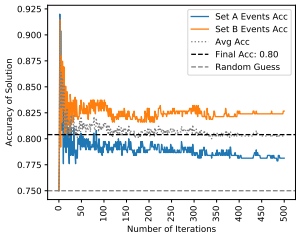
As in previous experiments, we ran the algorithm for 1,000 iterations (500 iterations on each set), converging to an AUC of 72% and an accuracy of 66.0%. With an event accuracy of 66.4% and premature replacements accuracy of 65.6%. Figure 6 shows the progress over iterations. Figure 6d shows that almost all the instances with a probability of less than 30% of being events are actually premature replacements.

5. Conclusion

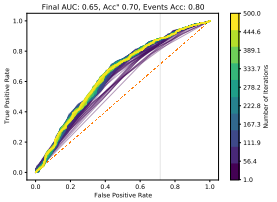
We presented an algorithm for discovering premature component replacement; an algorithm that relies on survival analysis models and their ability to handle censored cases. We experimentally illustrated the effect of incorrect labeling on the survival model's ranking performance. Moreover, we utilized such an effect on ranking to drive the genetic algorithm to discover a considerable percentage of incorrectly-labeled in-



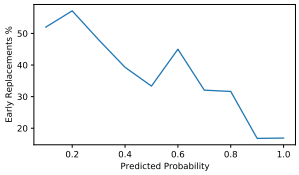
(a) Accuracy over iterations



(b) Events acc over itrations



(c) AUC over iterations



(d) Premature replacements % per confidence

Figure 5. Results of running the algorithm 1,000 iterations on a simulated dataset with 75% correctly labeled events.

stances corresponding to prematurely replaced components.

REFERENCES

Brodley, C. E., & Friedl, M. A. (1999). Identifying mislabeled training data. *Journal of artificial intelligence research*, *11*, 131–167.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, *34*(2), 187–220.

Ghoggali, N., & Melgani, F. (2009). Automatic ground-truth validation with genetic algorithms for multispectral image classification. *IEEE transactions on geoscience and remote sensing*, *47*(7), 2172–2181.

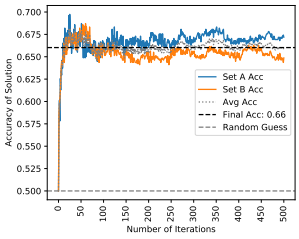
Guan, D., & Yuan, W. (2013). A survey of mislabeled training data detection techniques for pattern classification. *IETE Technical Review*, *30*(6), 524–530.

Harrell, J., Frank E., Califf, R. M., Pryor, D. B., Lee, K. L., & Rosati, R. A. (1982, 05). Evaluating the Yield of Medical Tests. *JAMA*, *247*(18), 2543-2546. doi: 10.1001/jama.1982.03320430047030

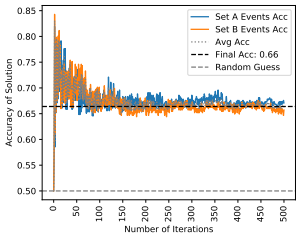
Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008, 09). Random survival forests. *Ann. Appl. Stat.*, *2*(3), 841–860. doi: 10.1214/08-AOAS169

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, *18*(1), 24. doi: 10.1186/s12874-018-0482-1

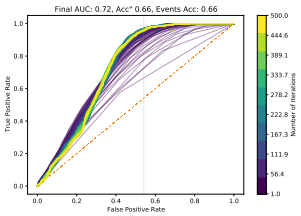
Khan, S., Phillips, P., Hockley, C., & Jennions, I. K. (2012). Towards standardisation of no fault found taxonomy.



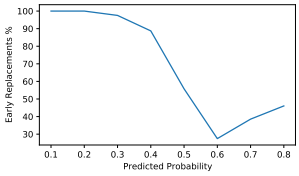
(a) Accuracy over iterations



(b) Events acc over itrations



(c) AUC over iterations



(d) Premature replacements % per confidence

Figure 6. Results of running the algorithm 1,000 iterations on the C-MAPSS dataset with 50% correctly labeled events.

Pasolli, E., & Melgani, F. (2015). Genetic algorithm-based method for mitigating label noise issue in ecg signal classification. *Biomedical Signal Processing and Control*, *19*, 130–136.

Pölsterl, S. (2020). scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *Journal of Machine Learning Research*, *21*(212), 1-6.

Sánchez, J. S., Barandela, R., Marqués, A. I., Alejo, R., & Badenas, J. (2003). Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, *24*(7), 1015–1022.

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management* (p. 1-9). doi: 10.1109/PHM.2008.4711414

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*(3), 408–421.

Zeng, X., & Martinez, T. (2003). A noise filtering method using neural networks. In *Ieee international workshop on soft computing techniques in instrumentation, measurement and related applications, 2003. scima 2003.* (pp. 26–31).

Zhu, X., Wu, X., & Chen, Q. (2003). Eliminating class noise in large datasets. In *Proceedings of the 20th international conference on machine learning (icml-03)* (pp. 920–927).