To Fail Or Not To Fail: Predicting Hard Disk Drive Failure Time Windows

Marwin Züfle, Christian Krupitzer, Florian Erhard, Johannes Grohmann, and Samuel Kounev

Software Engineering Group, University of Würzburg, Würzburg, Germany
marwin.zuefle@uni-wuerzburg.de, florian.martin.erhard@gmail.com,
[christian.krupitzer;johannes.grohmann;samuel.kounev]@uni-wuerzburg.de
Homepage: https://descartes.tools

Abstract. Due to the increasing size of today's data centers as well as the expectation of 24/7 availability, the complexity in the administration of hardware continuously increases. Techniques as the Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) support the monitoring of the hardware. However, those techniques often lack algorithms for intelligent data analytics. Especially, the integration of machine learning to identify potential failures in advance seems to be promising to reduce administration overhead. In this work, we present three machine learning approaches to (i) identify imminent failures, (ii) predict time windows for failures, as well as (iii) predict the exact time-to-failure. In a case study with real data from 369 hard disks, we achieve an F1-score of up to 98.0% and 97.6% for predicting potential failures with two or multiple time windows, respectively, and a hit rate of 84.9% (with a mean absolute error of 4.5 hours) for predicting the time-to-failure.

Keywords: Failure prediction · S.M.A.R.T. · Machine learning · Labeling methods · Classification · Regression · Cloud Computing.

1 Introduction

Large IT companies like Google, Amazon, Microsoft, and IBM have millions of servers worldwide. The administration of those servers is an expensive and time-consuming task. Especially unexpected crashes of servers, e.g., due to hard disk failures, can result in unavailable services and data loss. Hence, hardware is usually equipped with data collection mechanisms to observe the current state.

One such example is the Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) [21]. This technology uses sensors to gather information about the current state of hard disk drives (HDDs) and solid-state drives (SSDs). S.M.A.R.T. also provides an overview of the current state of the drives including vendor-specific thresholds that indicate a current malfunction. However, it misses intelligent analysis and linking of the different parameters. Especially, predictive analytics would help to identify potential faults or crashes in advance, can eliminate delays resulting from unexpected issues and, thus, increase reliability in Cloud Computing. Further, as it enables warning ahead of a failure, it

can support administration, e.g., it offers the potential to guarantee the presence of a backup of the data and the availability of a substitute device.

In this paper, we provide three machine learning based approaches to predict disk failures in advance. Leveraging this method, data center providers can identify their disk drives with imminent failures at an early stage and replace them to avoid unexpected downtime and data loss. Our contributions are threefold:

- A comparison of data preparation steps for binary classification of disk failures in near future.
- A random forest approach for predicting the time window of a disk failure.
- A regression based approach for continuous time-to-failure prediction.

We show the applicability of those approaches in a case study with real data collected from 369 HDDs equipped with S.M.A.R.T. The results are promising: The application of oversampling improves the predictive quality of the binary classifier with an accuracy of 97.642%, the multi-class classification of time windows achieves a multi-class accuracy of 97.628% and downscaled to two classes an accuracy of even 98.885%, while the time-to-failure regression achieves an average hit rate of about 84.9% (with a mean absolute error of only 4.5 hours).

The remainder of the paper is structured as follows: Next, Section 2 introduces the topics HDD monitoring (see Section 2.1) as well as machine learning based on random forest (see Section 2.2). Subsequently, Section 3 summarizes the current state-of-the-art in predicting disk failures and delineates our contribution. Section 4 presents our three approaches for disk drive failure prediction. Afterwards, Section 5 describes the results of the case study for evaluating the applicability of our approaches. Finally, Section 6 discusses the evaluation findings and concludes the paper with an outlook on future work.

2 Background

This section contains the necessary background information on hard disk drive monitoring and random forest, which is used in this paper.

2.1 Hard Disk Drive Monitoring

Failure Indicators. Monitoring internal HDD parameters to improve the reliability was first implemented in HDDs by IBM in 1992 [21]. IBM named this system Predictive Failure Analysis. Compaq, Seagate, Quantum, and Conner together integrated another monitoring system called IntelliSafe just a few years later [17]. In 1995, Seagate aimed to create a version that is compatible also with other hardware manufacturers. Therefore, IBM, Quantum, Western Digital, and Seagate cooperated to develop a novel HDD monitoring system based on IntelliSafe and Predictive Failure Analysis. The result of this collaboration was the Self-Monitoring, Analysis, and Reporting Technology (S.M.A.R.T.) [21]. This technology is used as a monitoring system in most HDDs and nowadays also

SSDs. Here, several internal parameters and operations, e.g., head flying height, spin-up time, and drive calibration retry count, are stored during runtime. Today, most drive manufacturers include the S.M.A.R.T. system for drive monitoring. However, each drive manufacturer can define its own set of attributes to monitor and thresholds for these parameters that should not be exceeded. Yet, there is still a subset of S.M.A.R.T. attributes that most drive manufacturers monitor in common, e.g., spin-up time, read error rate, and start/stop count.

Types of Failures. Failures of HDDs can be separated into two types: (i) predictable and (ii) unpredictable failures. The predictable failures occur due to slow mechanical processes like wear. Since the S.M.A.R.T. technology only uses thresholds of individual parameters, it can only detect the slow degradation effects caused by predictable mechanical failures. According to Seagate [21], mechanical and thus mostly predictable failures make up about 60% of failures. Therefore, the detection rate based solely on S.M.A.R.T. thresholds is not sufficient. In contrast, unpredictable failures usually emerge rather spontaneously. The reasons for such unpredictable failures are typically of an electronic nature or abrupt physical failures. Although using only the manufacturer thresholds does not cover both types of failures, a study by Google [18] shows that some S.M.A.R.T. attributes, e.g., (offline) uncorrectable sector count and reallocation event count, are strongly correlated to HDD failures.

2.2 Random Forest Models

Random forest is a machine learning method which uses the concept of ensemble learning. It was first introduced by Breiman and Cutler [4] and builds up on a former version of Ho [12]. Ensemble learning methods use the concept of the "wisdom of the crowd". That is, these methods combine a large number n of uncorrelated models to derive a final outcome. Typically, ensemble learning methods create many weak learners[1] and fuse their outcomes to derive a model with high predictive power, i.e., a strong learner. The two most common ways of creating and combining such weak learners are boosting and bagging. While boosting aims at iterative improvement of the models by focusing on the instances that were wrongly classified in the previous iteration, bagging generates many individual models and provides a result that is derived by majority decision of the models' predictions [3]. Random forest belongs to the category of bagging ensemble learners. As weak learners, random forest creates decision trees based on the training data. Subsequently, random forest aggregates the individual predictions of the decision trees to produce a final prediction.

By applying the ensemble technique, random forest overcomes the main disadvantage of single decision trees, i.e., their tendency to overfit the training data and therefore generalize rather poorly [12]. However, to prevent overfitting, random forest has to ensure that the decision trees do not correlate with each other.

[1] Weak learners are classification methods that correlate rather weakly with the true classification, while strong learners correlate very well with the true classification.

An initial approach to reach this goal was introduced by Ho [12]. She proposed to use only randomly selected features for model learning of each decision tree. Breiman and Cutler [4] built up on this so-called random decision forest as they introduced the bagging approach. Using bagging, not only the features are selected randomly but also the training samples themselves. That is, for each of the n decision trees, a subset of all training data is sampled with replacement[2] and then the feature space of each sample is also sampled randomly.

Random forest can be parameterized to perform either classification (binary or multi-class) or regression. In terms of classification, the final prediction is the class with the most individual predictions of the decision trees. For regression, the individual predictions are combined using the arithmetic mean.

3 Related Work

The prediction of HDD failures is not only an important task for cloud providers, but also an interesting research area for the scientific community. This field of research arose especially due to the introduction of S.M.A.R.T., which simplified the data collection and thus formed the basis for approaches of statistical and machine learning as well as of artificial intelligence.

The most common approach found in literature is binary classification. The main focus of the papers following this approach is to determine whether an HDD will fail within a certain period of time or not. Botezatu *et al.* [2] propose such an approach based on informative downsampling, regularized greedy forests, and transfer learning. While Shen *et al.* [22] apply an approach based on part-voting random forest, Xiao *et al.* [25] present a mechanism for predicting HDD failures based on an online random forest so that the model can evolve as new data arrives. An algorithm to predict these failures using the multiple-instance learning framework combined with naive Bayesian classifier is introduced by Murray *et al.* [16]. Sun *et al.* [23] propose a CNN-based approach to predict HDD failures as well as memory failures. For this purpose, Sun *et al.* also designed a new loss function to handle the imbalanced training instances effectively. In contrast, Zhao *et al.* [28] treat the data as time series to construct Hidden Markov Models and Hidden Semi-Markov Models.

Hamerly and Elkan [11] model the failure prediction task as an anomaly detection approach. To this end, Hamerly and Elkan introduce two models. The first model is based on a mixture model of naive Bayes submodels with expectation-maximization and the second one applies a naive Bayes classifier to detect the anomalies. Wang *et al.* [24] also propose an approach based on anomaly detection and failure prediction. Therefore, Wang *et al.* apply Mahalanobis distance, Box-Cox transformation, and generalized likelihood ratio tests.

Pitakrat *et al.* [19] compare various machine learning methods with regard to their binary HDD failure classification performance. Aussel *et al.* [1] conduct a study assessing the performance of support vector machines, random forests,

[2] Sampling with replacement means that instances can be selected multiple times in the same sample.

and gradient boosted trees as well as the impact of feature selection. Similar to these works, dos Santos Lima *et al.* [20] evaluate the performance of several recurrent neural models with shallow and deep architectures.

To not only predict whether an HDD will fail or not, some researchers apply the binary classification repeatedly with different failure time windows to provide more information about the time horizon of the predicted failure [14, 15, 29]. Li *et al.* [14, 15] propose a classification approach based on decision trees and an algorithm to estimate the health degree of the HDD using regression trees. Zhu *et al.* [29] present a backpropagation neural network approach and a support vector machine based model for the prediction of imminent failures.

While Chaves *et al.* [7] present an approach to estimating the time-to-failure as a continuous variable using a Bayesian network, Yang *et al.* [27] introduce an approach based on linear regression. An approach to predict the health status of HDDs using a recurrent neural network is proposed by Xu *et al.* [26].

In distinction to the existing work, we compare the performance of different data preparation steps, i.e., oversampling techniques, for the binary classification of upcoming failures. In addition and contrary to most of the approaches presented, we explicitly model the time-to-failure in time windows and predict it by applying multi-class classification using only a single model. Finally, we assess the impact of pre-filtering the training data for regression model learning.

4 Approaches for HDD Failure Level Prediction

This paper aims at three major failure prediction aspects based on S.M.A.R.T. measurements. First, we compare three binary classification alternatives with different preprocessing steps. Second, the time-to-failure is predicted using multi-class classification. Therefore, a new labeling strategy is applied to generate meaningful target values. In a third approach, regression is used to predict the time-to-failure as a continuous value.

4.1 Binary Classification of Failing HDDs

To perform a binary classification task, we create two distinct classes. Therefore, all instances with a time-to-failure of one week or less receive the class label 0. The remaining S.M.A.R.T. data instances get the class label 1. This labeling procedure was also performed by Pitakrat *et al.* [19]. Although Pitakrat *et al.* performed a broad comparison of different machine learning methods, they did not assess the impact of preprocessing techniques. For this purpose, we apply three different ways of training the machine learning model: (I) binary classification without further data preparation (Unmodified), (II) binary classification with Enhanced Structure Preserving Oversampling (ESPO) as oversampling technique, and (III) binary classification using Synthetic Minority Oversampling Technique (SMOTE) as oversampling mechanism. We apply random forest as machine learning method since it is typically comparatively fast for model learning and prediction, robust against overfitting, and can handle multiple classes

efficiently[3]. In addition, random forest achieved very good and robust results in the comparison of machine learning methods in Pitakrat *et al.* [19].

Unmodified. In the binary classification approach without further preprocessing steps, we pass the S.M.A.R.T. measurement instances along with their label to random forest to learn a model that represents the training data. After model learning, we provide the learned model with unseen instances to predict whether the instance of the new HDD indicates a future failure or not.

ESPO. Dealing with imbalanced data is a non-trivial task for machine learning methods. In this context, imbalance means that the training data contains significantly more instances of one class (majority class) than of the other class (minority class). This often results in biased models that tend to overestimate towards the majority class. In our case, the number of instances of the class representing HDDs that fail within the next week is much smaller. There are two common ways to deal with such class imbalances: undersampling and oversampling. Undersampling discards instances of the majority class to reach a balance in the number of class instances. Oversampling, in contrast, creates new instances of the minority class based on existing instances. In our approach, we use oversampling instead of undersampling, since removing majority class instances would reduce the size of the training data set and typically decrease the quality of the model[4]. In this second alternative, we apply Enhanced Structure Preserving Oversampling (ESPO) for oversampling. ESPO synthetically generates new instances of the minority class on the basis of multivariate Gaussian distribution [6]. To this end, ESPO estimates the covariance structure of the minority class instances and regularizes the unreliable eigenspectrum [6]. Finally, ESPO extends the set of minority class instances by maintaining the main covariance structure and by generating protection deviations in the trivial eigendimensions [6]. After oversampling the minority class, we apply the same model learning procedure as in alternative (I). However, this alternative offers the machine learning method more training data and with regard to the two classes approximately equally distributed training data.

SMOTE. This alternative is similar to alternative (II), but it uses Synthetic Minority Oversampling Technique (SMOTE) instead of ESPO as oversampling technique. SMOTE creates new instances of the minority class by combining instances that lie close to each other in the feature space [8]. To this end, SMOTE determines the nearest neighbours of each instance of the minority class. Subsequently, it takes a random subset of these neighboring instances and computes

[3] The handling of multiple classes is not explicitly required for comparing data preparation for binary classification. However, it is necessary for multi-class classification in Section 4.2 and to maintain comparability between the approaches.

[4] This does not apply if the data set is sufficiently large to still be large enough after undersampling.

the differences between their features and the feature of the respective instance. The feature difference is weighted with a random number between 0 and 1. Finally, SMOTE adds the resulting weighted difference to the features of the considered instance and provides this as new instance of the minority class [8].

4.2 Classification of multiple Failure Levels

Failure Level Labeling. To predict future failures in multiple levels, a new target variable needs to be defined. For this purpose, we changed the failure variable of the original data and used the new failure label as target variable. In the original data, the failure label is 0 if the HDD is running and only changes to 1 if the HDD stops due to a failure, i.e., the time-to-failure is zero. Since we want to predict multiple failure levels, other class labels are required. Therefore, we define a set of relevant classes, each representing a different failure level: 0, (0,1], (1,2], (2,5], (5,12], (12,24], (24,48], (48,72], (72,96], (96,120], (120,144], (144,168], (168,∞). Thus, each of these labels represents the interval of the time-to-failure in hours. The last class label (168,∞) indicates that no failure is expected within the next week. If, for example, the random forest model predicts label 48, the failure is expected after 24 hours at the earliest (i.e. after the next smaller class label), but no later than 48 hours. For the sake of simplicity and readability, in the following, we will refer to each of these classes only with its upper limit, e.g., we will refer to the label (24,48] as 48.

Model Learning. After calculating the new class labels based on the time-to-failure, we pass the S.M.A.R.T. attributes as features to random forest with the newly created labels as target variable. Thereby, the random forest model learns not only whether the HDD will soon fail or not, but also the time-to-failure in discretized form. This extends the binary classification (cf. Section 4.1) to a multi-class scenario where each class represents a specific time window in which the predicted failure is to be expected.

4.3 Regression for Time-to-Failure Prediction

Applying the binary classification of Section 4.1 only provides predictions on whether an HDD is likely to fail within the next week or not. The multi-class classification approach in Section 4.2 extends the provided information by delivering a time window, i.e., the predicted class, within the failure is expected to occur. However, all these predictions are only discretized.

To obtain continuous predictions about the time-to-failure, regression must be used instead of classification since classification can only predict a discrete output that must be included in a specified set of classes. In addition, the output class must also be present in the training set so that the model can learn the correlation between the features and the output class. Regression, in contrast, can

predict any continuous value based on the input features. This allows regression models to predict values that are not included in the training set.

We implemented two alternatives of time-to-failure regression using random forest: (i) using all available data to learn the random forest regression model (Naive) and (ii) train the random forest regression model exclusively on the training instances where the failure occurs during the next 168 hours (Pre-Filtering). Then, we apply the random forest regression models only on the measurements where a failure will occur within the next week. To retrieve this information, the approaches of Section 4.1 or Section 4.2 can be used, for example. The idea of pre-filtering the training set to HDDs that actually fail within the next week aims to focus the regression model on failing devices. For intact HDDs, a time-to-failure regression does not make sense, because if there is no indicator for an imminent failure yet, it cannot be distinguished whether the HDD will last another year or two years, for example. Therefore, the time-to-failure can only be guessed. By pre-filtering the data set, we explicitly focus our model on the relevant part of the data in order to achieve a more precise prediction.

5 Evaluation

We implemented the approaches in `R` using the libraries `randomForest` [5], `OSTSC` [10], and `unbalanced` [9]. First, this section describes the evaluation design. Then, the achieved predictive qualities of the binary and multi-class classification approaches are provided. Afterwards, the performances of the regression models are presented and, finally, the runtimes required for model training are compared for all approaches.

5.1 Evaluation Design

To assess the performance of our classification and regression models, we use a data set consisting of 369 HDDs[5]. This data set was first used by Murray *et al.* [16]. Although the data set encompasses 64 S.M.A.R.T. features for each instance, we only include 25 in our experiments. Many of the excluded parameters are constant during the entire measurement period and, thus, do not contain any information on the health status of the HDD. The time-to-failure is of course excluded for the classification tasks and used as target variable for the regression task. The remaining 25 features included in the remaining data set are FlyHeight5-12, GList1-3, PList, ReadError1-3, ReadError18-20, Servo1-3, Servo5, Servo7-8, and Servo10. The reduced feature set was also used by Pitakrat *et al.* for their comparison of machine learning methods. Of the 369 HDDs included in the data set, 178 did not fail during the measurement period, while 191 suffered a critical failure. This large number of failed HDDs is due to consumers sending in their failed HDDs containing the S.M.A.R.T. parameters of the past several weeks. Thereby, the number of defective and intact HDDs is

[5] HDD data set: `http://dsp.ucsd.edu/~jfmurray/software.htm`

fairly balanced in this data set. However, the data set contains continuous, i.e., approximately two-hourly, measurements of each HDD. This results in a total amount of 68411 measurement instances. Since we do not only predict for each HDD whether it will fail some day in the future, we predict the time-to-failure for each measurement instance of the S.M.A.R.T. parameters. Yet, this procedure results in an unbalanced data set. Figure 1 shows a histogram of the distribution of the time-to-failure classes. For this purpose, the time-to-failure classes are shown on the horizontal axis and the number for each time-to-failure class on the vertical axis. The figure clearly shows the dominance of the last class, i.e., the class indicating that there will be no failure within the next 168 hours. Furthermore, it can be seen that especially the classes 0 to 5 are very small with only 260 to 653 instances. For binary classification, the class indicating an imminent failure within the next 168 hours comprises 23749 instances, while the class indicating no failure within the next 168 hours consists of 44662 instances.

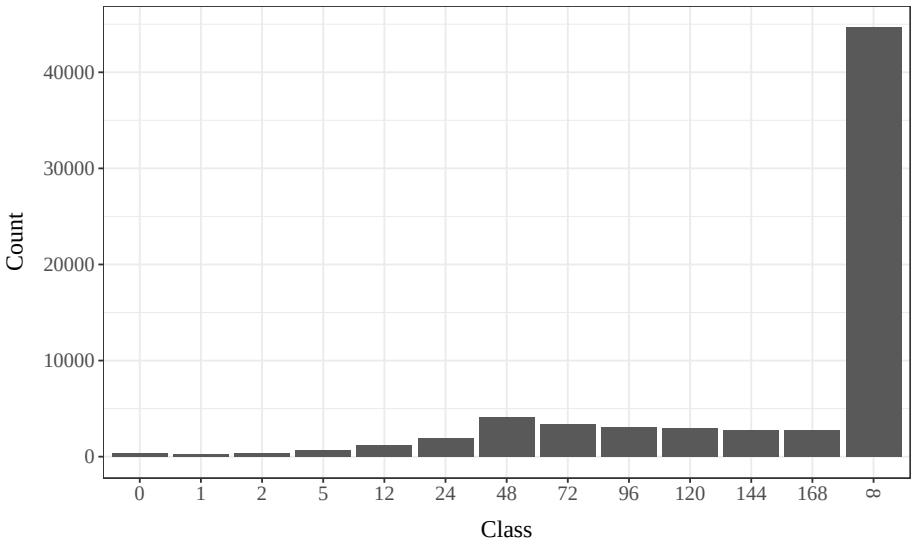

Fig. 1. Histogram of the distribution of failure classes.

In accordance with the literature, we use a split of approximately 80:20 for model training and model testing. That is, about 80% of all data is used to learn the model, while the model is evaluated on the remaining about 20% of the data. Since a single experiment alone is not significant, we perform the experiment 25 times to reduce the random distortion, as most of the techniques and methods applied are based on random numbers.

Although a comparison with existing approaches would be interesting, the reported metrics cannot be compared since the approaches were executed on different data sets (real world and synthetic) –without handling the class imbal-

ance adequately– which affects the evaluation metrics significantly. In addition, the authors presented in Section 3 have not made their code publicly available, hence, it was not possible to reproduce their measurements in our setting.

5.2 Binary Failure Prediction

To ensure reproducibility, Table 1 summarizes the parametrization of the applied methods for the comparison of binary classification random forest models. We set the number of decision trees generated to 100 and the number of features that are randomly selected as candidates for each split to 5 for all three alternatives.

Table 1. Used libraries and methods along with the parametrization for the binary classification random forest models. Alternative (I) is without oversampling, (II) applies ESPO for oversampling, and (III) uses SMOTE to oversample the minority class.

Alternative	Library:Method	Parameters
(I) Unmodified	randomForest:randomForest()	ntree = 100, mtry = 5, replace = TRUE
(II) ESPO	OSTSC:OSTSC()	ratio = 1.0, r = 1.0
	randomForest:randomForest()	ntree = 100, mtry = 5, replace = TRUE
(III) SMOTE	unbalanced:ubBalance()	type = "ubSMOTE", percOver = 300, percUnder = 150, k = 5
	randomForest:randomForest()	ntree = 100, mtry = 5, replace = TRUE

Table 2 presents the average achieved values for the evaluation metrics accuracy, precision, recall, and F1-score. The results show that alternative (II) ESPO yields the highest accuracy, precision, and F1-score. Only in terms of recall, ESPO is outperformed by the unmodified and SMOTE approaches, although all approaches differ only slightly. Considering all four evaluation metrics, ESPO achieves the overall best performance of the binary classification alternatives.

Table 2. Average achieved values of the three binary classification alternatives.

Alternative	Accuracy	Precision	Recall	F1-Score
(I) Unmodified	97.472%	94.347%	96.993%	95.649%
(II) ESPO	**97.642%**	**94.913%**	96.970%	**95.928%**
(III) SMOTE	95.734%	89.086%	**96.995%**	92.869%

As Table 2 only shows the average values and not the variation within the 25 replications, Figure 2 illustrates the performances using box plots. To this end, the horizontal axis presents the evaluation metrics, while the vertical axis depicts the achieved values. The orange, blue, and purple colored boxes describe the (I) approach without oversampling, the (II) ESPO oversampling approach, and

the (III) SMOTE oversampling approach, respectively. Again, the figure shows that the unmodified and ESPO approaches clearly outperform SMOTE in all metrics besides recall. Regarding recall, all three approaches yield approximately the same value with only marginal variation within the replications. The highest variation can be seen for the metric precision. Here, the interquartile range, i.e., the difference between the third and first quartiles, shows the largest value.

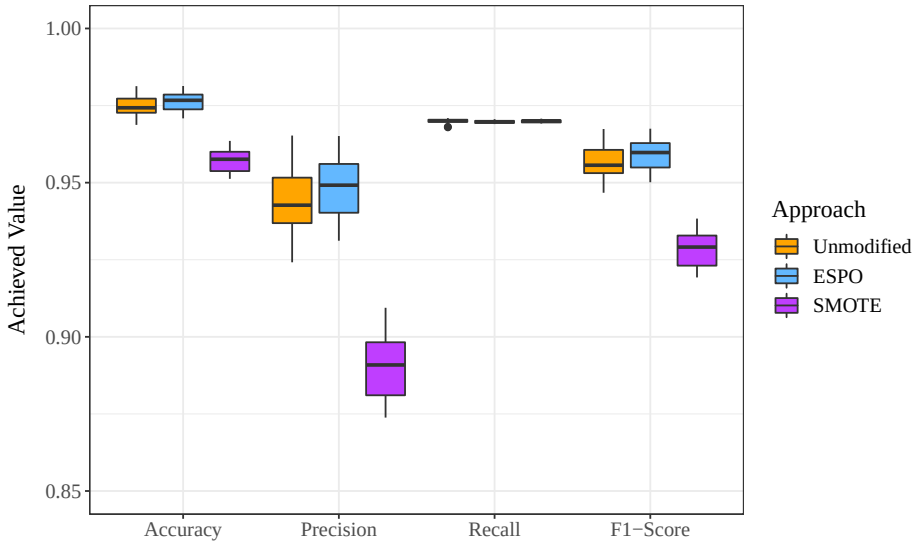

Fig. 2. Performance of the binary classification alternatives for each evaluation metric.

5.3 Failure Level Classification

In contrast to binary classification, we set the number of classification trees to $n = 500$ because predicting multiple classes is more difficult than distinguishing between only two classes. Table 3 shows the average confusion matrix over all 25 runs. The confusion matrix clearly shows that most values are on the diagonal and therefore most instances are correctly predicted. In addition, for the failure classes, most false predictions are predicted in the neighboring classes. This means that the upcoming failure is detected, only the time windows are missed by a few hours up to one day. Thus, these wrongly predicted classes are uncritical in practice since the failure is nevertheless predicted with a relatively accurate time horizon. Moreover, it can be seen that the actual time-to-failure classes 1, 2, and 5 cannot be predicted as accurately as the others. This is due to the fact that there are only very few training instances for these classes and the temporal difference between these classes is very short. However, the rightmost column shows the number of cases in which the HDD failed in the respective

time window, but our multi-class approach did not predict a failure within the next week. This column also contains some instances, which can be explained by the fact that the S.M.A.R.T. parameters cannot cover all aspects that can lead to an HDD failure [21], e.g., sudden electronic or physical impacts.

Table 3. The confusion matrix for the multi-class classification approach. The rows show the actually observed (Ob) time-to-failure classes, while the columns present the predicted (Pr) ones. The value in each cell illustrates the number of instances predicted for that particular set of observed and predicted class labels. The green color indicates the correctly predicted instances. The second cell from the left in the third row, for example, shows a single instance that is predicted as "a failure will occur within the next hour", while the failure actually occurred in the time window of one to two hours after the measurement.

Ob \ Pr	**0**	**1**	**2**	**5**	**12**	**24**	**48**	**72**	**96**	**120**	**144**	**168**	**∞**
0	67	0	0	0	0	0	0	0	0	0	0	0	2
1	0	35	0	0	4	2	9	0	0	0	0	0	0
2	0	1	12	0	0	2	0	0	0	0	0	0	2
5	0	0	0	52	24	17	1	0	0	0	0	0	2
12	0	0	0	0	185	14	2	0	0	0	0	0	8
24	0	0	0	0	17	339	21	0	0	0	0	0	12
48	0	0	0	0	0	6	773	10	0	0	0	0	18
72	0	0	0	0	0	0	12	740	22	0	0	0	24
96	0	0	0	0	0	0	0	6	768	15	0	0	24
120	0	0	0	0	0	0	0	0	14	752	6	0	24
144	0	0	0	0	0	0	0	0	0	20	762	8	29
168	0	0	1	0	0	0	0	0	0	0	16	729	66
∞	0	0	1	1	0	1	0	1	1	0	1	5	14143

To summarize the quality of the random forest multi-class classification model, it achieves an average micro-F1-score of 97.628%. The micro-F1-score is the sum of instances on the diagonal (colored green) divided by the total number of instances in the confusion matrix. It should be noted that for multi-class classification, the micro-F1-score is equal to the multi-class accuracy.

As the micro-F1-score of this multi-class approach cannot be directly compared with the F1-scores obtained in Section 5.2, we downscaled the multiple classes to the same two classes presented in Section 5.2. That is, we merge all classes except ∞ into one big class, i.e., the class that indicates an upcoming failure within the next 168 hours. This way, the classes match those used in Section 5.2, which allows us to compare them. However, since ESPO achieved the best values, Figure 3 shows the comparison of the ESPO approach with the downscaled multi-class approach in terms of box plots. Again, the evaluation metrics accuracy, precision, recall, and F1-score are depicted on the horizontal axis, while the achieved values of ESPO and downscaled multi-class approach are shown on the vertical axis. The blue and red boxes represent the ESPO and downscaled multi-class results, respectively. The figure shows that the downscaled

multi-class approach yields even better results regarding accuracy (98.885% vs. 97.642%), precision (99.818% vs. 94.913%), and F1-score (98.019% vs. 95.928%). Yet, the binary classification with ESPO oversampling achieves a little higher recall (96.970% vs. 96.283%). This fact demonstrates that the ESPO approach, on the one hand, detects more failed HDDs. On the other hand, it shows that ESPO also predicts more good HDDs as failed, i.e., ESPO has a higher false positive rate. In numbers, our downscaled multi-class approach achieves a false positive rate of only 0.07% while ESPO shows a value of 2.09%. The false positive rates of the other two binary classification approaches are even higher. The detection of more defective HDDs with a higher false positive rate can be useful if the costs for false alarms are not relevant. However, in cases where false alarms lead to high costs, a more conservative model is advantageous since it detects almost as many failing HDDs and produces less costs for unnecessary HDD replacements.

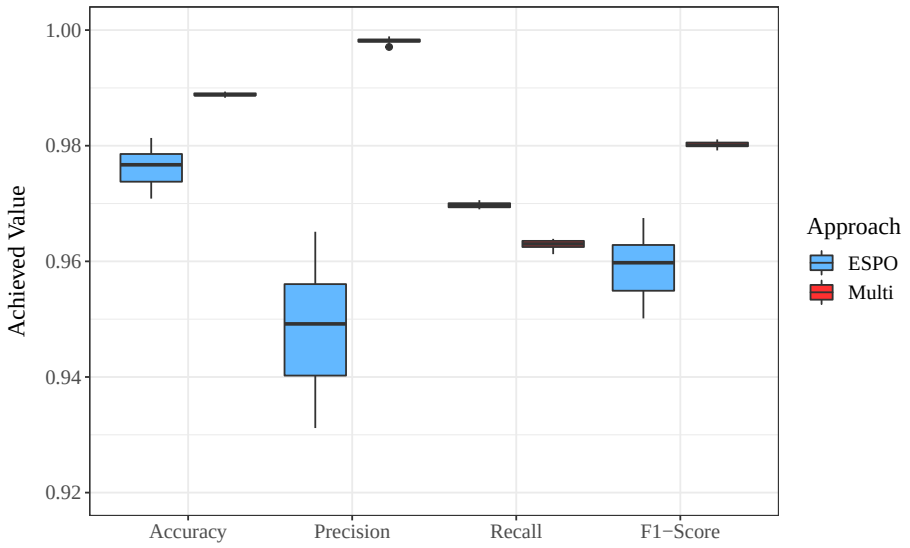

Fig. 3. Comparison of the best binary classification approach (ESPO) and the multi-class classification approach downscaled to two classes.

5.4 Time-to-Failure Regression

Similar to the evaluation of the multi-class classification approach, we set the number of decision trees to $n = 500$. Figure 4 depicts the achieved hit rate, mean absolute error (MAE), and root mean square error (RMSE) for both regression approaches as box plots. The brown and green boxes represent the naive regression approach and our regression approach using pre-filtering, respectively. We define the hit rate as the ratio of correct time-to-failure regressions to the number

of total regressions. Further, a time-to-failure regression is correct if the actual time-to-failure falls within an interval of the predicted time-to-failure ± 10%.

The left subfigure of Figure 4 shows the achieved hit rates. It can be seen that our approach of applying pre-filtering before model learning yields a higher hit rate than the naive version. In numbers, our pre-filtering approach achieves an average hit rate of around 84.9% while the naive version only predicts a correct time-to-failure in about 80.2% of all tries. In contrast to the hit rate, a smaller value is better for MAE and RMSE. Regarding the MAE (middle subfigure of Figure 4), our pre-filtering approach results in an average MAE of about 4.5 hours. The naive regression approach, instead, shows an average MAE of around 10.0 hours. In addition, the interquartile ranges are very small for both approaches. The large difference between the MAEs and the small interquartile ranges imply that our pre-filtering approach predicts the time-to-failure much more accurate. The same conclusion can be drawn by taking a look at the RMSE (right subfigure of Figure 4). Our pre-filtering approach shows an average RMSE of around 12.8 hours while the RMSE of the naive approach reaches around 44.6 hours. Again, the interquartile ranges are very small for both approaches.

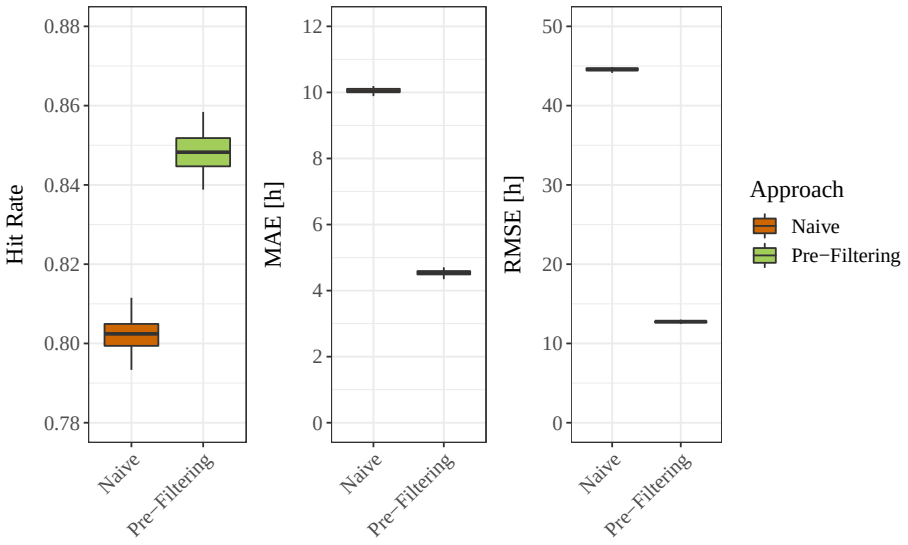

Fig. 4. Achieved values of the two regression alternatives for each evaluation metric.

Taking all three measures into account, it can be concluded that our pre-filtering approach predicts the time-to-failure much more accurately than the naive version. With an average hit rate of nearly 85% and a mean absolute error of only 4.5 hours, the time-to-failure regression is very precise.

5.5 Training Time Comparison

After assessing the quality of the predictions of all presented approaches, we also analyzed the runtimes for model training. The time needed for the prediction is negligible compared to the training time. We executed the experiments in our private cloud using Apache CloudStack and Kernel-based Virtual Machine (KVM). The virtual machine is deployed on a host with 32 cores each providing 2.6 GHz and 64 GB memory, having hyperthreading activated. The virtual machine runs Ubuntu 16.04 (64-bit) with 4 cores each with 2.6 GHz and 8 GB memory. We implemented the appraoch in R version `3.4.4`.

Figure 5 illustrates the training times of all classification and regression approaches using box plots. To this end, the horizontal axis depicts the approaches, while the vertical axis presents the required training time. The boxes are colored according to the respective approach, whereby the coloring corresponds to the previous figures. With an average training time of about 27 seconds, the unmodified binary classification approach yields the shortest training time followed by our multi-class classification approach with about 174 seconds. Both of the binary classification with oversampling approaches require a much larger training time of on average around 420 seconds and 335 seconds for ESPO and SMOTE, respectively. Thus, our multi-class classification does not only achieve an overall better prediction performance but also a much shorter training time than the best binary classification approach, i.e., ESPO. Regarding the regression approaches, the naive procedure takes on average about 2175 seconds, while the approach that pre-filters the data only requires around 346 seconds. That is, our pre-filtering approach is more than 6 times faster than the naive version and

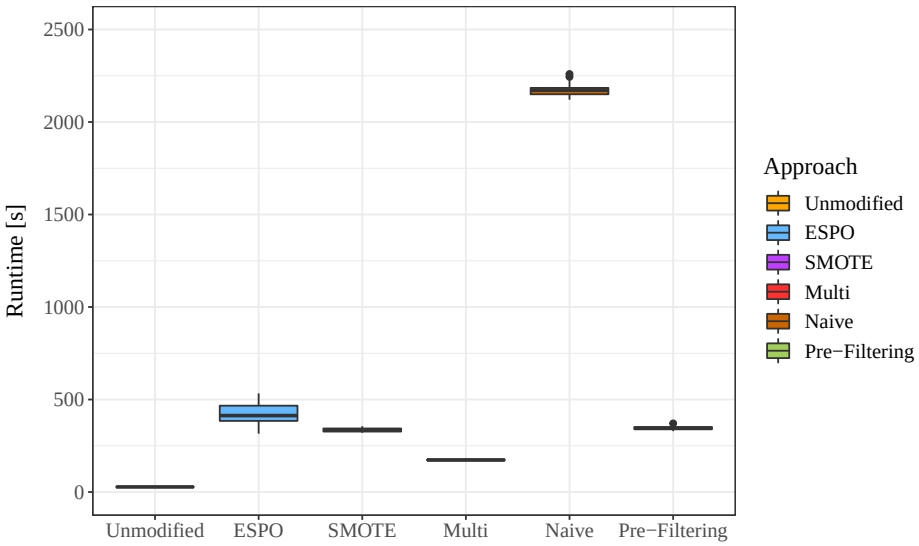

Fig. 5. The required training times for all of the presented approaches.

even faster than the binary classification approach with ESPO oversampling, while significantly improving the hit rate, MAE, and RMSE.

6 Conclusion and Discussion

In this paper, we aim at the early prediction of HDD failures. For this purpose, we present three contributions: (i) a comparison of data preparation steps for binary classification of imminent HDD failures, (ii) the introduction of a random forest based approach for time-to-failure classification using multiple time-to-failure classes, and (iii) a pre-filtering and regression based approach to predict the time-to-failure as a continuous variable.

(i) For the first contribution, we compare different data preparation steps for binary classification of imminent failures. The results show that applying ESPO oversampling to balance the number of instances per class improves the prediction quality with an F1-score of 95.928% but also requires more runtime to build the model and perform predictions. In contrast, the application of SMOTE oversampling performed worse than the unmodified version.

(ii) Since the time horizon of upcoming failures is essential for cloud administrators to prevent downtimes and data losses, our second contribution focuses on predicting the time window of upcoming failures using multiple classes by grouping close times-to-failures into the same classes. Based on these classes and the respective S.M.A.R.T. features, we learned a random forest classification model. This approach yields a micro-F1-score of 97.628% for multi-class classification. If downscaled to the two classes used in the first contribution, it achieves an even higher F1-score of 98.019%. Thus, it clearly outperforms the binary classification approaches while maintaining a very low false positive rate of only 0.07%.

(iii) Finally, we present a pre-filtering prior to learning a random forest regression model for the prediction of the time-to-failure as a continuous value. We show that this pre-filtering improves the hit rate from 80.2% to 84.9% and significantly reduces the runtime compared to learning the regression model using all available data.

As future work, we plan to compare the predictive quality and runtime of different machine learning methods for multi-class classification of drive failure time windows. In addition, we want to assess the performance of our approaches on SSDs as well. Furthermore, we plan to integrate time series forecasting methods from our previous work [30] to forecast future states of the HDDs for integrating an additional facet into the analysis. Additionally, it would be possible to use the forecast in combination with a self-adaptive system [13] to dynamically trigger additional backups or other maintenance tasks for critical HDDs.

Acknowledgements

This work was co-funded by the German Research Foundation (DFG) under grant No. (KO 3445/11-1) and the IHK (Industrie- und Handelskammer) Würzburg-Schweinfurt.

References

1. Aussel, N., Jaulin, S., Gandon, G., Petetin, Y., Fazli, E., Chabridon, S.: Predictive models of hard drive failures based on operational data. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 619–625. IEEE (2017)
2. Botezatu, M.M., Giurgiu, I., Bogojeska, J., Wiesmann, D.: Predicting disk replacement towards reliable data centers. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 39–48. ACM (2016)
3. Breiman, L.: Bagging predictors. Machine learning **24**(2), 123–140 (1996)
4. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
5. Breiman, L., Cutler, A., Liaw, A., Wiener, M.: Breiman and cutler's random forests for classification and regression (2018), `https://cran.r-project.org/web/packages/randomForest/randomForest.pdf`
6. Cao, H., Li, X.L., Woon, D.Y.K., Ng, S.K.: Integrated oversampling for imbalanced time series classification. IEEE Transactions on Knowledge and Data Engineering **25**(12), 2809–2822 (2013)
7. Chaves, I.C., de Paula, M.R.P., Leite, L.G., Gomes, J.P.P., Machado, J.C.: Hard disk drive failure prediction method based on a bayesian network. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2018)
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
9. Dal Pozzolo, A., Caelen, O., Bontempi, G.: unbalanced (2015), `https://cran.r-project.org/web/packages/unbalanced/unbalanced.pdf`
10. Dixon, M., Klabjan, D., Wei, L.: Ostsc (2017), `https://cran.r-project.org/web/packages/OSTSC/OSTSC.pdf`
11. Hamerly, G., Elkan, C., et al.: Bayesian approaches to failure prediction for disk drives. In: ICML. vol. 1, pp. 202–209 (2001)
12. Ho, T.K.: Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition. vol. 1, pp. 278–282. IEEE (1995)
13. Krupitzer, C., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A Survey on Engineering Approaches for Self-Adaptive Systems. Pervasive and Mobile Computing Journal **17**(Part B) (2015)
14. Li, J., Ji, X., Jia, Y., Zhu, B., Wang, G., Li, Z., Liu, X.: Hard drive failure prediction using classification and regression trees. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. pp. 383–394. IEEE (2014)
15. Li, J., Stones, R.J., Wang, G., Liu, X., Li, Z., Xu, M.: Hard drive failure prediction using decision trees. Reliability Engineering & System Safety **164**, 55–65 (2017)
16. Murray, J.F., Hughes, G.F., Kreutz-Delgado, K.: Machine learning methods for predicting failures in hard drives: A multiple-instance application. Journal of Machine Learning Research **6**(May), 783–816 (2005)
17. Ottem, E., Plummer, J.: Playing it smart: The emergence of reliability prediction technology. Tech. rep., Technical report, Seagate Technology Paper (1995)
18. Pinheiro, E., Weber, W.D., Barroso, L.A.: Failure trends in a large disk drive population. In: 5th USENIX Conference on File and Storage Technologies (FAST 2007). pp. 17–29 (2007)

19. Pitakrat, T., Van Hoorn, A., Grunske, L.: A comparison of machine learning algorithms for proactive hard disk drive failure detection. In: Proceedings of the 4th international ACM Sigsoft symposium on Architecting critical systems. pp. 1–10. ACM (2013)
20. dos Santos Lima, F.D., Pereira, F.L.F., Chaves, I.C., Gomes, J.P.P., de Castro Machado, J.: Evaluation of recurrent neural networks for hard disk drives failure prediction. In: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS). pp. 85–90. IEEE (2018)
21. Seagate Product Marketing: Get s.m.a.r.t. for reliability. Tech. rep., Technical report, Seagate Technology Paper (1999)
22. Shen, J., Wan, J., Lim, S.J., Yu, L.: Random-forest-based failure prediction for hard disk drives. International Journal of Distributed Sensor Networks **14**(11), 1550147718806480 (2018)
23. Sun, X., Chakrabarty, K., Huang, R., Chen, Y., Zhao, B., Cao, H., Han, Y., Liang, X., Jiang, L.: System-level hardware failure prediction using deep learning. In: Proceedings of the 56th Annual Design Automation Conference 2019. p. 20. ACM (2019)
24. Wang, Y., Ma, E.W., Chow, T.W., Tsui, K.L.: A two-step parametric method for failure prediction in hard disk drives. IEEE Transactions on industrial informatics **10**(1), 419–430 (2013)
25. Xiao, J., Xiong, Z., Wu, S., Yi, Y., Jin, H., Hu, K.: Disk failure prediction in data centers via online learning. In: Proceedings of the 47th International Conference on Parallel Processing. p. 35. ACM (2018)
26. Xu, C., Wang, G., Liu, X., Guo, D., Liu, T.Y.: Health status assessment and failure prediction for hard drives with recurrent neural networks. IEEE Transactions on Computers **65**(11), 3502–3508 (2016)
27. Yang, W., Hu, D., Liu, Y., Wang, S., Jiang, T.: Hard drive failure prediction using big data. In: 2015 IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW). pp. 13–18. IEEE (2015)
28. Zhao, Y., Liu, X., Gan, S., Zheng, W.: Predicting disk failures with hmm-and hsmm-based approaches. In: Industrial Conference on Data Mining. pp. 390–404. Springer (2010)
29. Zhu, B., Wang, G., Liu, X., Hu, D., Lin, S., Ma, J.: Proactive drive failure prediction for large scale storage systems. In: 2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST). pp. 1–5. IEEE (2013)
30. Züfle, M., Bauer, A., Lesch, V., Krupitzer, C., Herbst, N., Kounev, S., Curtef, V.: Autonomic Forecasting Method Selection: Examination and Ways Ahead. In: Proceedings of the 16th IEEE International Conference on Autonomic Computing (ICAC). IEEE (2019)