Hard Drive Failure Prediction Using Classification and Regression Trees

Conference Paper · June 2014

DOI: 10.1109/DSN.2014.44

7 authors, including:

Jing Li
Nankai University
8 PUBLICATIONS **69** CITATIONS
SEE PROFILE

Gang Wang
Nankai University
130 PUBLICATIONS **680** CITATIONS
SEE PROFILE

Xiaoguang Liu
Nankai Univrsity
114 PUBLICATIONS **578** CITATIONS
SEE PROFILE

2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks

Hard Drive Failure Prediction Using Classification and Regression Trees

Jing Li, Xinpu Ji, Yuhan Jia, Bingpeng Zhu, Gang Wang*
Nankai-Baidu Joint Lab, College of Computer and Control Engineering
Nankai University
Tianjin, China
{*lijing, jixinpu, jiayuhan, zhubingpeng, wgzwp*}*@nbjl.nankai.edu.cn*

Zhongwei Li*, Xiaoguang Liu*
College of Software
Nankai University
Tianjin, China
{*lizhongwei, liuxg*}*@nbjl.nankai.edu.cn*

***Abstract*—Some statistical and machine learning methods have been proposed to build hard drive prediction models based on the SMART attributes, and have achieved good prediction performance. However, these models were not evaluated in the way as they are used in real-world data centers. Moreover, the hard drives deteriorate gradually, but these models can not describe this gradual change precisely.**

This paper proposes new hard drive failure prediction models based on Classification and Regression Trees, which perform better in prediction performance as well as stability and interpretability compared with the state-of the-art model, the Backpropagation artificial neural network model. Experiments demonstrate that the Classification Tree (CT) model predicts over 95% of failures at a false alarm rate (FAR) under 0.1% on a real-world dataset containing 25,792 drives. Aiming at the practical application of prediction models, we test them with different drive families, with fewer number of drives, and with different model updating strategies. The CT model still shows steady and good performance. We propose a health degree model based on Regression Tree (RT) as well, which can give the drive a health assessment rather than a simple classification result. Therefore, the approach can deal with warnings raised by the prediction model in order of their health degrees. We implement a reliability model for RAID-6 systems with proactive fault tolerance and show that our CT model can significantly improve the reliability and/or reduce construction and maintenance cost of large-scale storage systems.

***Keywords*-Hard drive failure prediction; SMART; CART; Health degree**

I. Introduction

Storage systems are growing larger quickly with the rapid development of information technology. Although hard drives are reliable in general, they are believed to be the most commonly replaced hardware components [1], [2]. It is reported that 78% of all hardware replacements were for hard drives in the data centers of Microsoft [1]. Moveover, with the increase of single drive and whole system capacity, block and sector level failures, such as latent sector errors [3] and silent data corruption [4], can not be ignored anymore. For instance, in RAID-5 systems, one drive failure with any other sector error will result in data loss, which may be a disaster to data centers.

A lot of researchers focus on designing erasure codes to improve storage system reliability. This is a typical reactive fault-tolerant technique which is used to reconstruct data when drive failure occurs. By contrast, predicting drive failures before they actually occur can inform us to take actions in advance. At present, Self-Monitoring, Analysis and Reporting Technology (SMART) is implemented inside most of the modern hard drives [5]. However, as reported in [6], this can not reach a desirable prediction performance. To improve failure prediction accuracy, some statistical and machine learning methods have been proposed to build prediction models based on the SMART attributes [6], [7], [8], [9], [10], [11], [12], [13]. Although these methods have reached good prediction performance, there are some problems with them. Firstly, the models are black boxes, such as the artificial neural networks. Therefore, they do not perform well on interpretability as well as stable performance, and it is hard to adjust their prediction performance. Secondly, the prediction models were not evaluated in the way they are used in real-world data centers. Thirdly, they do not pay attention to the changing process of a drive's SMART attributes during its deterioration. They can not rate a drive's health degree nicely, but merely label it to good or failed.

In this paper, we explore building hard drive failure prediction models based on classification and regression trees (also referred to as decision trees), which have high accuracy, ease of interpretability, and stable performance. On a dataset coming from a real-world data center, our Classification Tree (CT) model can predict over 95% of failures at a false alarm rate (FAR) below 0.1%, which outperforms the state-of the-art model, the Backpropagation artificial neural networks (BP ANN) model. We simulate the practical use of our model in real-world data centers - being used with different drive families, being used in small-scale data centers, and being updated periodically. Our model still performs well. Besides the CT binary classifier model, we also present a Regression Tree (RT) model to evaluate the health degree (or fault probability). As a result, deploying the RT model in a storage system, we can deal with warnings in order of their health degrees to reduce processing overhead. We develop a Markov model for RAID-6 systems to evaluate how our prediction models benefit the reliability of large-scale systems. Reliability analysis shows that our CT model can significantly improve reliability and/or reduce cost.

The rest of the paper is organized as follows: In Section II,

978-1-4799-2233-8/14 $31.00 © 2014 IEEE
DOI 10.1109/DSN.2014.44

we survey related work of hard drive failure prediction using SMART attributes. Section III is the introduction of our modeling methodologies for failure prediction. Section IV gives a description of our dataset and the preprocessing of this dataset for building models. We present the experimental results in Section V. In Section VI, we discuss the improvement of reliability if our prediction models are used, followed by conclusions and future work in Section VII.

II. Related Work

SMART is a standard hard disk drive condition monitoring and failure warning technology in industry since 1995 [8]. However, hard drive manufacturers estimate that the threshold-based algorithm implemented in drives can only obtain a failure detection rate (FDR) of $3-10\%$ with a low false alarm rate on the order of 0.1% [6]. The reason is that, to avoid heavy false alarm cost, they set the thresholds conservatively to keep the FAR to a minimum at the expense of failure detection rate.

Hamerly and Elkan [7] employed two Bayesian approaches to predict hard drive failures based on SMART attributes. Firstly, they used a cluster-based model named NBEM. The second approach was a supervised naive Bayes classifier. Both algorithms were tested on a dataset from Quantum Inc. concerning $1,927$ good hard drives and 9 failed drives. They achieved prediction accuracy of $35-40\%$ for NBEM and 55% for naive Bayes classifier with about 1% FAR.

Hughes *et al.* [8] proposed two statistical methods to improve SMART prediction accuracy. Since they found that many of the SMART attributes are non-parametrically distributed, this observation led them to use Wilcoxon rank-sum test. They proposed two different strategies: multivariate rank-sum test and OR-ed single variate test. Both methods were tested on $3,744$ drives containing two different models of which only 36 drives were failed. They achieved failure detection rate of 60% at 0.5% false alarm rate.

Murray *et al.* [9] compared the performance of SVM, unsupervised clustering, and two non-parametric statistical tests (rank-sum and reverse arrangements test). Dataset was collected from 369 hard drives of the same model of which good and failed drives are about half and half. Surprisingly, they found that the rank-sum method achieved the best prediction performance (33.2% detection rate at 0.5% FAR). In their subsequent work [6], a new algorithm based on the multiple-instance learning framework and the naive Bayesian classifier (named mi-NB) was developed. They found that, on the same dataset as [9], the nonparametric rank-sum test outperformed SVM for certain small set of SMART attributes (28.1% failure detection at 0% FAR). When all selected 25 features were used, SVM achieved the best performance of 50.6% detection and 0% FAR.

Zhao *et al.* [10] employed Hidden Markov Models (HMMs) and Hidden Semi-Markov Models (HSMMs) to predict hard drive failures. They treated the observed SMART attributes as time series data. Experimental results (on the same dataset as that used in [9], [6]) showed that these methods outperformed other methods that paid no attention to the relationship of attribute values over time. Using the best single attribute, the HMM and HSMM models achieved detection rates of 46% and 30% with no false alarm, respectively. When combining the best two attributes, the HMM model reached a FDR of 52% at 0% FAR.

Wang *et al.* [12] proposed a strategy for drive anomaly prediction based on Mahalanobis distance (MD). Testing on the same dataset used in [9], they demonstrated that the method with prioritized attributes selected by FMMEA (Failure Modes, Mechanisms and Effects Analysis) performed better than the one with all attributes. In their subsequent study [13], the minimum redundancy maximum relevance (mRMR) was used to remove the redundant attributes from the attribute set selected by FMMEA. Then, they built up a baseline Mahalanobis space using the good drive data of the critical parameters. This model could detect about 67% of the failed drives with zero FAR. 56% of the failed drives could be detected about 20 hours in advance.

The prediction performance of models mentioned above is unsatisfactory. One possible reason is that the datasets used by them are relatively small, which do not contain enough SMART information to build effective prediction models. The dataset provided by Murray *et al.* [9] is used in several literatures. However, it contains 369 drives of which good and failed are about half and half. This is not conformed with the case in real-world data centers. Moreover, it was collected before 2003. The SMART information format is not consistent with the current SMART standard. These factors undermine the practicability of models.

In our previous work [11], we explored the ability of Backpropagation artificial neural networks to predict drive failures based on SMART attributes. A real-world dataset concerning $23,395$ drives was used to evaluate prediction models. We proposed several training and detection strategies to improve prediction accuracy. The BP ANN model could reach a excellent failure detection rate which was up to 95% while with a reasonable low FAR. However, this model does not perform well on performance stability as well as interpretability. Moreover, it is hard to adjust prediction performance.

In this paper, we hope to improve the prediction performance by employing decision trees which have good interpretability and stable performance, and evaluate them on a large real-world dataset.

III. Classification and Regression Tree Models

As mentioned in the last section, the approaches explored in previous researches do not provide an understanding of events which explain the decision given by them. In this paper, we explore the ability of classification and regression

trees [14] to predict hard drive failure based on SMART attributes. Besides high prediction accuracy, they have crucial advantages of yielding stable interpretable results. Users can find out the significant attributes inducing drive failure by analyzing the output regulations of the tree.

A. Classification Tree Model

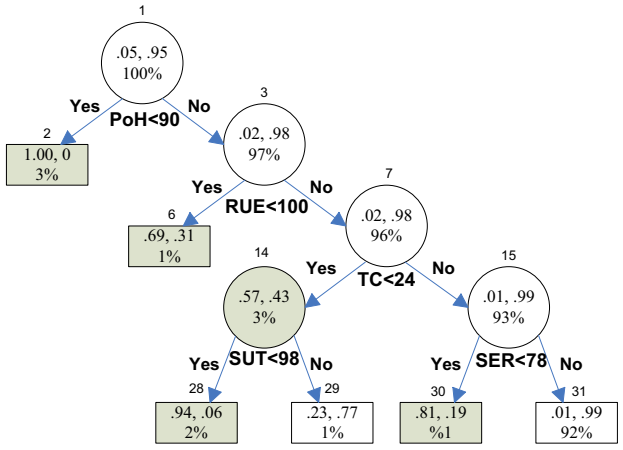

Figure 1. A simplified classification tree for hard drive failure prediction. "POH" denotes the SMART attribute "Power On Hours", "RUE" denotes "Reported Uncorrectable Errors", "TC" denotes "Temperature Celsius", "SUT" denotes "Spin Up Time" and "SER" denotes "Seek Error Rate".

Figure 1 is a simplified classification tree for hard drive failure prediction. It starts with a single root (node 1) which contains all of the data denoted by the percentage 100% at the bottom of the node. The fractions of 0.05 and 0.95 represent the probability distributions of the failed and good samples at that node, respectively. The root node is labeled as good (denoted by a white node) with the majority vote of the data contained at it. This node is split based on the value of a SMART attribute "Power On Hours". If the value is less than 90, those samples are put in the first terminal node or called leaf node, denoted by node 2, which contains only one class or do not have enough data to be split. Node 2 contains 3% of all samples of which failed samples make up 100%, and it is labeled as failed denoted by a shaded node. The remain 97% of all samples, whose value of "Power On Hours" is equal or greater than 90, are placed in node 3 labeled as good for good samples making up 98%. Node 3 continues to be split into two child nodes based on the value of "Reported Uncorrectable Errors". The process continues until there are no more nodes which can be split. Each leaf node is labeled with the majority class of the samples at it.

We use the Information Gain as the splitting function in our models. The split procedure searches through all values of the input SMART attributes to find out the best partition variable which maximizes the gain in information. For a binary target, assuming node D is split into child nodes D_1 and D_2 based on feature (SMART attribute) v_i, We can calculate the Information Gain for this split as

$$gain(D, v_i) = info(D) - info(D, v_i) \tag{1}$$

where $info(D)$ is the information entropy at node D, and $info(D, v_i)$ is the sum of information entropy of child nodes after this split. The information entropy is calculated as

$$info(D) = -plog_2(p) - qlog_2(q) \tag{2}$$

where p, q are constrained by $p+q=1$, denoting probability distributions of the two classes samples at node D, respectively. The sum of information entropy of child nodes is calculated as

$$info(D, v_i) = \frac{|D_1|}{|D|} info(D_1) + \frac{|D_2|}{|D|} info(D_2) \tag{3}$$

where $|D|$ denotes the total number of samples contained at node D. We can learn from formula (2) that the entropy reaches its maximum value 1 when the probability distribution is uniform to 0.5, and reaches its minimum value 0 when $p=1$ or $p=0$. The concept of information entropy implies how bias the dataset is in relation to the value of the target. If the dataset contains samples belonging to only one class, then there is zero entropy which means no further split needed. Hence, a larger bias in the distribution generates a smaller entropy. At each step of tree building, the CT algorithm calculates the information gain for each possible split, and then chooses the split which provides the greatest information gain. Therefore we can increase the homogeneity of each class of the samples in relation to the target value every step.

The classification tree is fully grown by recursive partitioning approach, until the node does not satisfy the split conditions or contains only one class. Minimum Split (Minsplit) and Minimum Bucket Size (Minbucket) [15] are used to control node split. Minsplit limits the minimum number of samples that must exist at a node before it is considered for splitting. Minbucket limits the minimum number of samples at any leaf node. A complete tree will be built to maximum depth depending on the values of Minsplit and Minbucket, which can overfit the training data and then not perform very well on new data. We can avoid the overfitting by pruning. Subbranches with low overall information gain will be pruned back from the grown tree, thereby the classification tree is simplified. So there are two steps in the CT algorithm for training prediction models. The first step is to build a full classification tree on the training set. The second is to prune back some subbranches with low gain in order to avoid overfitting. We can use the Complexity Parameter (CP) to control the size of the tree and to select an optimal size by controlling the process of pruning. The CP governs the minimum gain that must be obtained at each split of the classification tree in order to make a split

worthwhile. The detailed CT algorithm for training drive failure prediction models is shown in Algorithm 1.

We build classification and regression tree models using SMART attributes and their change rates as input vectors together with the target values representing good or failed drives. To separate good and failed drives effectively, we also propose some training strategies. We change the probability distributions of the good and failed samples by adjusting their weights, which impacts the prediction performance dramatically. To reduce false alarms, we can weight the two kinds of errors (false alarms and miss detections), which will affect the choice of variable on which to split the dataset at each node. We use loss weight to denote the unwelcome degree of the error.

Algorithm 1 CT algorithm for training prediction models

Input: Training data (composed of SMART attributes, attribute change rates and target values), split conditions (Minsplit, Minbucket), and pruning parameter (CP)
Output: CT model for drive failure prediction

```
Begin
create root node T which contains all the data
label T with the majority vote of the data at it
push T onto a stack S
while S is not empty do
    pop the top element from S and store it to D
    if D does not satisfy the split conditions then
        set D as a leaf node
    else
        for each possible split based on v_i at D do
            calculate gain(D, v_i) using Formula (1), (2) and (3)
        end for
        select the split maximizing gain(D, v_i)
        split D into child nodes D_1 and D_2 based on v_i
        label nodes D_1, D_2 and push them onto S
    end if
end while
for each node P in the tree do
    if the gain induced by P's split is less than CP then
        prune back the entire sub-tree rooted at P
    end if
end for
End
```

B. Regression Tree Model

The prediction models explored by previous works, including the classification tree model presented in the last subsection, are all binary classifiers. When we train a model, we label each sample as good or failed. When we test a sample, the model also simply outputs a binary classification result. This result cannot describe the drive's health condition finely. As a result, although those models can achieve satisfactory prediction performance, they can only deal with accurate predictions and false alarms indiscriminately. This will induce heavy processing cost. In fact, drives do not deteriorate suddenly, but gradually. Our idea is a mechanism that can deal with drives closer to failure more priority than those more healthy. Therefore, a prediction model that can evaluate drives' healthy finely is essential.

We build a Regression Tree (RT) model to evaluate drives' health degree. In the RT model, each test has a quantitative target value describing the drive's health degree rather than a class label indicating good or failed. The training algorithm is mostly similar with the CT training algorithm. In order to find the best split, the algorithm checks all possible splitting attributions, as well as all possible values of the attribution to be used to split the node. However, the measure of the best split is the minimum of squares instead of the greatest gain in information. For each possible split, the algorithm calculates the within-node sum of squares about the mean of each child node on the target variable. We choose the best split that yields the smallest overall sum of squares within the child nodes. The sum of squares within a node is calculated as

$$sq = \sum_{i=1}^{n} (y_i - \overline{Y})^2 \tag{4}$$

where n is the number of samples at this node, y_i is the target variable of the $i-th$ sample, and $\overline{Y}$ is the mean of the n samples on the target variable. The detailed algorithm for training RT prediction model is shown in Algorithm 2.

The important difference between the CT and RT algorithms is how to set target values. In the CT algorithm, the target value of every good sample is set to 1 and that of every failed sample is set to -1. In the RT algorithm, the target values of good samples remain the same which represent their absolute health. For each failed sample, we set its target value to a real value representing its health degree. A simple function to determine the health degree of the failed sample i hours before failure is as:

$$h(i) = -1 + \frac{i}{w} \tag{5}$$

where w denotes the size of the *global deterioration window*. That is, we think all samples w hours before failure represent a borderline condition between good and failed, and drives deteriorated gradually after that. As such, all of the failed sample i hours before failure have the same health degree. However, this function does not perform very well. We develop another function based on *personalized deterioration window* to generate the input for Algorithm 2:

$$h_d(i) = -1 + \frac{i}{w_d} \tag{6}$$

where w_d denotes the size of drive d's deterioration window. We set w_d to the time in advance of d which can be obtained by building a prediction model, such as a CT

Algorithm 2 RT algorithm for training prediction models

Input: Training data (composed of SMART attributes, attribute change rates and target values), split conditions (Minsplit, Minbucket), and pruning parameter (CP)
Output: RT model for drive failure prediction

Begin
create root node T which contains all the data
label T with the mean of health degree
push T onto a stack S
while S is not empty **do**
    pop the top element from S and store it to D
    if D does not satisfy the split conditions **then**
        set D as a leaf node
    else
        for each possible split at D **do**
            calculate the sum of squares sq_j for each child node D_j using Formula (4), $j = 1,2$
            calculate the sum $sq = sq_1 + sq_2$
        end for
        select the split minimizing sq
        split D into child nodes D_1, D_2
        label D_1, D_2 and push them onto S
    end if
end while
for each node P in the tree **do**
    if the sq induced by P's split is less than CP **then**
        prune back the entire sub-tree rooted at P
    end if
end for
End

model, using the training set, and then applying it to d. Since personalized deterioration window distinguishes different individual drives' deterioration process more precisely, this method achieves better prediction performance than the method based on global deterioration window.

IV. Dataset Description And Preprocessing

A. Data Collection

Our dataset is collected from a real-world data center, and contains two drive families represented by "W" and "Q". There is a total of 25,792 drives in the dataset, labeled good or failed. Table I lists the details of our dataset. During a period of eight weeks, each good drive was sampled every hour. Some samples were missed because of sampling or storing errors. For failed drives, samples in a period of 20 days before actual failure were recorded. Some failed drives might lose some samples if they had not survived 20 days of operation since we began to collect data.

For every drive, we can read out 23 meaningful attributes from a SMART record. However, some attributes are useless for failure prediction because their values are the same for good and failed drives and are changeless during operation.

Table I
Dataset details.

Family	Class	Disks	Period	Samples
"W"	Good	22,790	56 days	30,631,028
"W"	Failed	434	20 days	158,190
"Q"	Good	2,441	56 days	3,155,735
"Q"	Failed	127	20 days	40,017

Table II
Preliminary Selected SMART attributes (basic features).

ID #	Attribute Name
1	Raw Read Error Rate
2	Spin Up Time
3	Reallocated Sectors Count
4	Seek Error Rate
5	Power On Hours
6	Reported Uncorrectable Errors
7	High Fly Writes
8	Temperature Celsius
9	Hardware ECC Recovered
10	Current Pending Sector Count
11	Reallocated Sectors Count (raw value)
12	Current Pending Sector Count (raw value)

So we filter out them and use only ten attributes to build our prediction models. Each SMART attribute has a six-byte raw value and a one-byte normalized value ranging from 1-253 which is transformed from the raw value [5]. The formats of the raw values are vendor-specific and not specified by any standard. Since some normalized values lose accuracy and their corresponding raw values are more sensitive to the health condition of drives, we select two raw values besides the ten normalized values to build our models. Table II lists the preliminarily selected features (called basic features).

B. Feature Selection Using Statistical Methods

We found that in our dataset the SMART attributes are non-parametrically distributed, which agrees with the observations in previous works [6], [8]. So we use three non-parametric statistical methods - reverse arrangement test, rank-sum test and z-scores [6] - to select features.

We first apply three non-parametric statistical methods to the basic features. By testing these features and some of their combinations, ten of them are selected for model building. The tenth feature "Current Pending Sector Count" and the twelfth feature the raw value of "Current Pending Sector Count" are excluded. We also test the change rates of SMART attributes. However, unlike [11], for every attribute, we test change rates with different intervals and finally select three of them as features - the 6-hour change rates of "Raw Read Error Rate", "Hardware ECC Recovered" and "Reallocated Sectors Count (raw value)".

As a result, each sample used in model training and failure detection has 13 features including 9 normalized

values, 1 raw value and 3 change rates. Section V shows that the features selected by statistical methods yields better prediction performance compared to the features selected by expertise [11].

V. Experimental Results

The experimental results in our previous work [11] show the advantage of the Backpropagation artificial neural network model in prediction performance over the other previous models. Meanwhile, the AdaBoost method [11] does not provide significant performance improvement and is much more computationally expensive. So, when we evaluate our classification and regression tree models, the plain BP ANN model is served as the control group. It has been reported by many studies [1], [2], [3], [16] that hard drive models, manufacturers and other environment factors can influence the statistical behavior of failures. To eliminate the impact of these factors, the SMART dataset is separated by drive model when building and evaluating our models.

A. Evaluating Classification Tree Model

1) Experimental Setup: When we evaluate our classification tree model, we use only "W" drive family and good samples collected within a single week. To evaluate the model more practically, we divide the dataset into training and test sets according to time rather than randomly. For each good drive, we take the earlier 70% of the samples within the week as training data, and the later 30% as test data. Since failed drives are much less than good drives and the chronological order of them was not recorded, we use all failed drives and divide them randomly into training and test sets in a 7 to 3 ratio. We randomly choose 3 samples per good drive in the training set as good samples to train models. By this way, we can eliminate the bias of a single drive's sample in a particular hour and provide enough information to describe the health condition of the drive. Like [11], we take out the failed sample within a *time window*, that is, the last n hours before the failure actually occurs, to train models, based on the assumption that the last n samples could demonstrate the failed signature. We test $n = 12, 24, 48, 96, 168, 240$. In the following experiments except for Section V-B3, we divide the dataset into training and test sets in the same way.

The Receiver Operating Characteristic (ROC) curve is used for presenting the prediction performance of our models. For drive failure prediction problem, the ROC curve indicates the trade-off between the failure detection rate (FDR) and the false alarm rate (FAR). FDR means the fraction of failed drives that are correctly classified as failed. FAR is defined as the fraction of good drives that are incorrectly classified as failed. We can trade off between them by tuning algorithm parameters. Another important metric of drive failure prediction is the time in advance (TIA) which describes how long in advance we can detect the impending failures.

2) Feature Selection: As mentioned in the last section, we select 13 critical features including 10 SMART attributes and 3 change rates. To verify the effectiveness of statistically selected features, we apply BP artificial neural network [11] and classification tree models to three different feature sets. They are the 12 basic features detailed in Table II, the 13 critical features selected by statistical methods, and the 19 features selected by expertise [11]. In this experiment, we set the failed time window to 12 hours [11]. That is, the samples collected within last 12 hours before failure are used as failed samples. For the BP ANN model based on 19-features, the input, hidden and output layers contain 19, 30, and 1 nodes respectively. For the 13-features set, the three layers have 13, 13 and 1 nodes respectively. For the 12-features set, they have 12, 20 and 1 nodes respectively. The maximum number of iterations is set to 400 and the learning rate is set to 0.1. Some important CT parameters are set as follows: $MinimumSplit = 20$, $MinimumBucketSize = 7$, $ComplexityParameter = 0.001$. Unless otherwise stated, we keep these settings in the following experiments. When we detect a drive, we check its sample in chronological order, and predict the drive is going to breakdown if any sample is classified as failed. Otherwise, the drive is classified as a good drive.

Table III shows the results. The feature set selected by the statistical methods outperforms other two feature sets in prediction performance with both prediction models. Therefore, we use it in the following experiments.

Table III
Effectiveness of Three Different Feature sets

Model	Dataset	FAR (%)	FDR (%)	TIA (hours)
BP ANN	12 features	0.44	89.47	347.7
	19 features	0.25	90.23	345.5
	13 features	0.20	90.98	342.5
CT	12 features	0.57	95.49	352.4
	19 features	0.63	94.74	351.4
	13 features	0.56	95.49	351.4

3) Evaluating CT Model: To separate good and failed drives effectively, we modify the probability distributions of the good and failed samples in training set to affect node splitting when building tree. We boost the failed sample set by giving it a higher weight, which adjusts the failed sample set to occupy 20% of the total and the good sample set to occupy 80%. In addition, due to good drives being the absolute majority in reality, a high FAR implies too many false alarms and results in heavy processing cost. To lower FAR, the loss weight specified for FAR is 10 times higher than that for FDR, which will affect the choice of variable on which to split the dataset at each node.

We test the impact of time window on the prediction performance of the CT model. Six different time windows

are respectively used to train CT models while keeping good training samples fixed. The results are illustrated in Table IV. As expected, adjusting the time window provides a coarse way to trade off between FDR and FAR. When the time windows is set to 168 hours (i.e. 7 days), the CT model obtains the best performance with a FDR of 95.49% at the FAR of 0.09%. Certainly, users can use other time windows to obtain higher FDR at the expense of FAR. In the following experiments in this paper, we use the same time window and probability distribution settings to train CT models and use a 12 hours time window to train BP ANN models.

Table IV
IMPACT OF TIME WINDOW ON CT MODEL.

Time Window	FAR (%)	FDR (%)	TIA (hours)
12 hours	0.31	93.98	354.4
24 hours	0.33	93.98	355.3
48 hours	0.39	95.49	350.6
96 hours	0.21	96.24	351.7
168 hours	0.09	95.49	354.6
240 hours	0.11	93.23	361.4

Since an abnormal sample can not give the confident information of the fault drive due to the measurement noise, it is not appropriate to predict that a drive is going to fail if only one sample of it is classified as failed by the model. We apply the voting-based detection algorithm [11] to our CT model. When detecting a drive, we check the last N consecutive samples (voters) before a time point, and predict the drive is going to fail if more than $N/2$ samples are classified as failed, and the next time point is tested otherwise. If all time points pass, the drive is classified as a good drive. Figure 2 shows the prediction results of CT and BP ANN models using the voting-based failure detection algorithm. We can see that, as N increases, the FAR of the CT model drops quickly while its FDR decreases slowly. When 27 voters are counted, the CT model predicts over 93% failures at a FAR of 0.009%. Even at the end (left) of the curve, the FAR of the CT model continues dropping. The BP ANN model is inferior to the CT model both in FDR and FAR. Moreover, when N exceeds 5, its FDR drops sharply while its FAR decreases slowly. We can conclude that, compared to the BP ANN model, our CT model achieves both higher failure detection rate and lower false alarm rate, and also can reach a very low false alarm rate while maintaining a high detection rate.

Besides prediction accuracy, providing sufficient time (measured by TIA) to users for backing up or migrating data, is also important. For correct predictions, Figure 3 and Figure 4 show the distribution of hours in advance, respectively plotted for BP ANN at the point 84.21% detection and 0.07% false alarms and for CT at 93.23% detection and 0.009% false alarms. In both models, almost all of correct detections are predicted 24 hours before failure, which is the

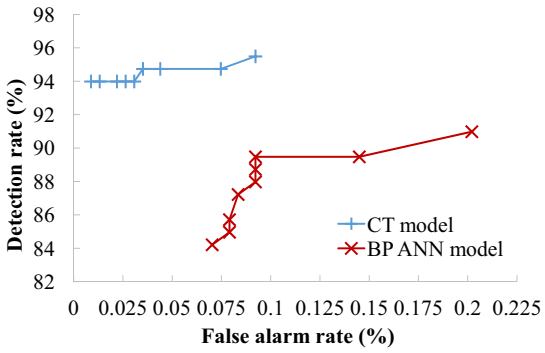

Figure 2. Impact of voting-based detection method on prediction performance. The points on each curve are obtained by the number of voters $N = 1, 3, 5, 7, 9, 11, 15, 17$, and 27 from right to left.

goal hard drive manufacturers want SMART technology to achieve. And both model achieve an average TIA over two weeks, which is sufficient for backing up data before the failure actually occurs. For other points in Figure 2, their TIA distributions are similar to those showed in Figure 3 and 4. We can conclude that the CT model achieves very good TIA as well as prediction accuracy.

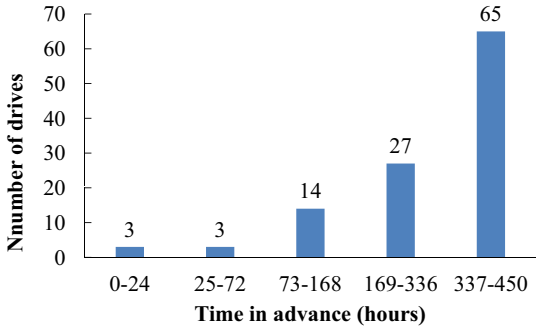

Figure 3. Distribution of time in advance of BP ANN model.

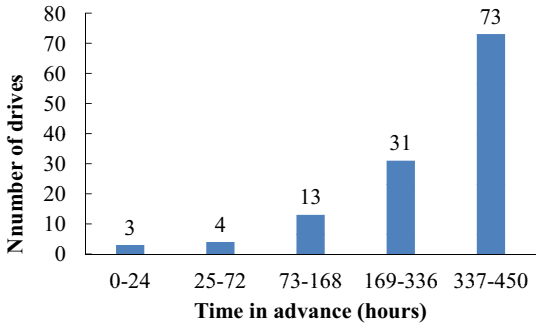

Figure 4. Distribution of time in advance of CT model.

B. Simulating Practical Use

We evaluate the CT and BP neural network models by simulating their application in real-world data centers - being used with different drive families, being used in small-scale data centers, and being updated periodically.

1) Evaluating with Different Drive Family: Different families of drives have different characteristics which may impact their reliability, even if they are made by the same manufacturers. Thereby, effectiveness with different drive families is an important metric of prediction models. However, previous works paid little attention to this partly because of the lack of appropriate datasets. We test the CT and BP ANN models with the dataset of drive family "Q", which has a much smaller volume than that of "W" dataset (see Table I). The prediction results are showed in Figure 5. The failure detection rate of CT model is varying from 100% to 93.5% with the increase of the number of voters N, meanwhile the false alarm rate is varying from 0.82% to 0.16%. And the average hours in advance is about $290 \sim 300$ hours. The prediction accuracy is not as good as that on family "W", in part because the "Q" dataset contains much less drives. However, the prediction performance is still acceptable for practical use. The BP ANN model obtains a much lower prediction accuracy than that on family "W". As a result, the performance gap between it and the CT model has remarkably widened. We are certain that our CT model is more stable in prediction performance with different drive families compared to the BP ANN model.

The experiment results on the two datasets illustrate the advantage of CT model in interpretability. By analyzing the trees, we can find out the significant attributes inducing "W" drives' failures are long power on hours "POH", high temperature or much reported uncorrectable errors "RUE". As for "Q" drives, the most common failure causes are long "POH", high temperature or high seek error rate "SER". So we can take corresponding measures to reduce the failure rate.

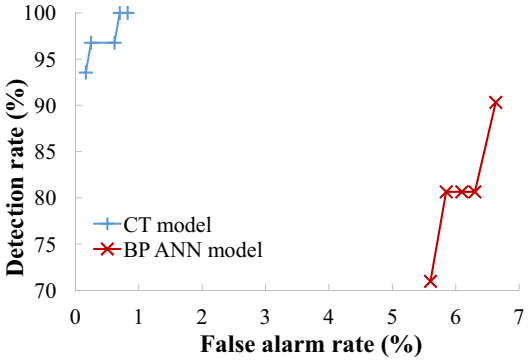

Figure 5. Prediction results of CT and BP ANN models on family "Q" using voting-based detection method. The points on each curve are achieved by setting $N = 1, 3, 5, 11$, and 17 from right to left.

2) Evaluating with Fewer Drives: In Section V-A, we use a very large dataset containing 23,224 drives, which is collected from a big data center. In the real world, however, prediction models will most likely be used in small and medium-sized data centers. To evaluate the effectiveness of prediction models applying to small and medium-size data centers, we test them with synthesized datasets containing fewer drives. We create four small datasets (named *A*, *B*, *C* and *D*) by randomly choosing 10%, 25%, 50% and 75% of all the good and failed drives respectively from the "W" dataset. So the smallest dataset *D* contains only 2,790 good drives and 43 failed drives. Table V shows the prediction performance of the CT and BP ANN models with these datasets. The voting-based detection algorithm with 11 voters is used. As expected, as the size of dataset decreases, both CT and BP ANN models suffer performance degradation. However, even with the dataset that is one order of magnitude smaller than the original dataset, both models obtain acceptable FDR and FAR. Especially, CT model remains reasonably low FAR. Moreover, both models keep an average TIA about two weeks. Notice that the dataset used in the last subsection is actually a small-sized dataset, and the results on it agree with the results in this subsection. Since our CT model obtains acceptable prediction performance with both real-world and synthesized small-sized dataset, we believe it can predict hard drive failure effectively in small and medium-sized data centers.

Table V
PREDICTION PERFORMANCE ON SMALL-SIZED DATASETS.

Model	Dataset	FAR (%)	FDR (%)	TIA (hours)
BP ANN	A	2.93	88.24	329.6
	B	1.10	90.63	345.3
	C	0.16	84.38	338.0
	D	0.03	81.82	350.6
CT	A	0.22	82.35	345.7
	B	0.07	90.63	346.6
	C	0.11	90.63	343.3
	D	0.09	91.82	341.7

3) Model Updating Strategies: Another critical problem neglected by previous works is model aging. So it is reasonable to assume that they use a "train once, use forever" strategy, that is, a prediction model is built and then remains unchanged. However, as time goes on, drives' SMART attributes will change, so models may gradually lose their effectiveness. We actually observe significant drop of prediction accuracy when we simulate long-term use of prediction models.

We explore three different model updating strategies to keep prediction accuracy. The first is the *fixed* strategy which remains a prediction model unchanged over time after building it, that is, no updating. The second strategy is to update the model periodically, say, once a week, using samples collected by the last week, and then to apply it

to the current week. We name it the *accumulation* strategy. The third strategy is also to update the model periodically. However, only samples collected within the last cycle are used to build a new model, and this model is used to predicting failures during the current cycle. So we name it the *replacing* strategy. Since the last strategy uses drives' latest status to update models, we expect it to achieve the best prediction accuracy.

To evaluate the ability of these strategies to keep prediction accuracy as time goes on, we use all of good samples spanning eight weeks rather than a single week to simulate long-term use of prediction models. More specifically, for the fixed strategy, we train models using samples within the first week and then apply them unchangeably to test the samples respectively collected within the second week, the third week, until the eighth weeks. The ratio of the size of the training set to the size of each test set is still 7 to 3. Since failed samples are much less than good samples and they are collected at different times, we still divide all of failed drives randomly in a 7 to 3 ratio for training and test respectively. For the accumulation strategy, we train models using good samples collected from the first week to the i-th week, and then apply it to the $(i+1)$-th week, where $i = 1, 2, \ldots, 7$. For the replacing strategy, we train a new model using good samples collected from the $((i-1)\times c+1)$-th week to the $(i \times c)$-th week, and then use it to predict drive failures during next c weeks. c denotes the cycle length, and we tried $c = 1, 2$ and 3.

We test the CT and BP ANN models with three updating strategies on the drive families "W" and "Q". The voting-based detection algorithm is used, and 11 voters are counted for each time point. In this part, we show only false alarm rate and omit failure detection rate, in part due to space limitations, and in part because we use the same failed sample set in all experiments which implies that the false alarm rate is more important. Figures 6 - 9 show the false alarm rates of the two models with different updating strategies as time goes on. In general, the fixed strategy performs worst. As time goes on, its FAR increases gradually. After the sixth week, since the distance between the training and test samples are quite far, the up trend becomes very steep. Finally, its FAR rises to $10\% - 20\%$ which implies that the false alarms will be hundreds of times more than the correct detections. This will cause unacceptable processing cost. Apparently, it is impolitic to remain prediction models unchanged. The accumulation strategy performs better because more recent good samples are used to train prediction models. However, its FAR also increases apparently in the last weeks and rises to an unacceptable high value. As expected, the replacing strategy performs best. Since only the latest drive status is used for training, this strategy can maintain a reasonably low FAR. Among the three different cycle lengthes, 1-week updating achieves the lowest FAR. We think the results are broadly in line with what we expected. We can conclude that it is necessary to update prediction models for a long-term practical use.

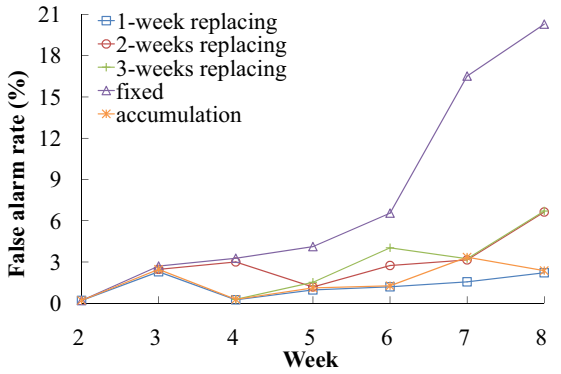

Figure 6. FAR of CT with updating on "W".

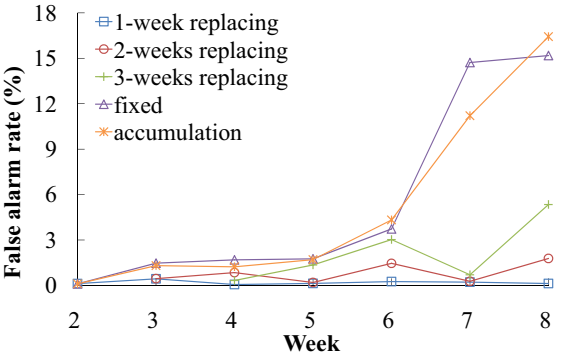

Figure 7. FAR of BP ANN with updating on "W".

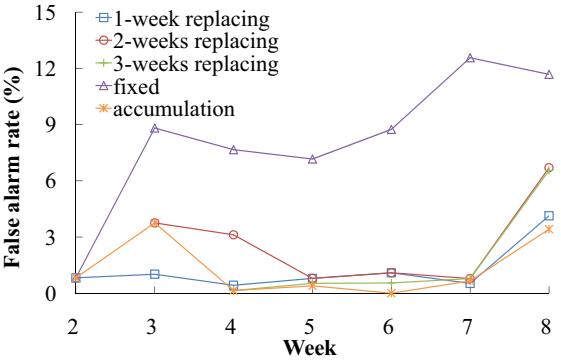

Figure 8. FAR of CT with updating on "Q".

Combining with the replacing strategy, both CT and BP ANN models maintain a low false alarm rate for a long time on both "W" and "Q" families. However, they perform quite differently in failure detection rate. The CT model maintains a FDR above 90% with all of the updating strategies and both drive families. While the FDR of BP ANN model is

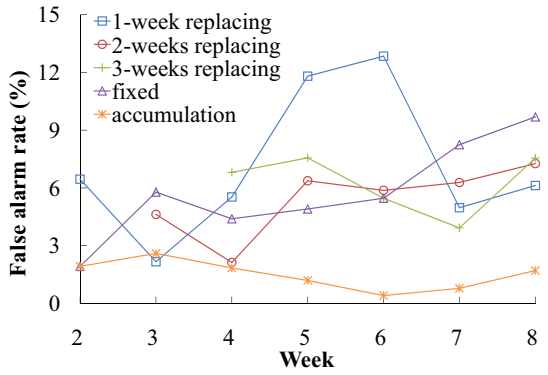

Figure 9. FAR of BP ANN with updating on "Q".

about ten percentage points lower than that of the CT model on average and fluctuates wildly. Moreover, BP ANN model shows quite different FDRs and FARs with the two different drive families, while CT model shows close performance trend with different drive families. This further verifies the advantage of the CT model over the BP ANN model in prediction performance as well as stability.

In this subsection, we simulate three practical application scenarios of prediction models. The experimental results show that our classification tree model maintains high failure detection rate and low false alarm rate as well as good stability in these scenarios.

C. Evaluating Health Degree Model

In this subsection, we test the health degree model based on regression tree. We first train a CT model using the training set, and then use it to determine TIA for each failed drive. The target value of each failed sample is then set by formula 6. If a failed drive is missed by the CT model, the target values of its samples are set by formula 5 and the deterioration window is set to 24 hours. We do not use all samples within the deterioration window to train the RT model. Instead, we choose 12 samples evenly within the window for each failed drive. To evaluate the effectiveness of the health degree model, we train another RT model for comparison. The target values are set to $+1$ and -1 respectively for good and failed samples.

When we build the RT models, split conditions (MinimumSplit, MinimumBuketSize) and pruning parameter (ComplexityParameter) are set to the same values as in training the CT model. We test the health degree model and the control group with the "W" dataset. We use a new voting-based detection algorithm. For each drive in test, if the average output of the last N samples is lower than the threshold, the drive is predicted to be failed, and the next time point is tested otherwise. Figure 10 plots the ROC curves of the health degree model and the control group. N is set to 11. Each data point is obtained by a unique threshold. The health degree model achieves a maximum FDR above 96%. And its ROC curve is closer to the upper left corner of the quadrant than that of the control group, which means higher FDR and lower FAR. The health degree model has another advantage over binary classifiers, such as the CT model. Since it outputs real values, we can trade off between the FDR and the FAR finely by simply applying the model with different detection thresholds, while the CT model can only make trade-off coarsely by tuning some training and detection parameters. We can reach a very high FDR, or reach a very low FAR, or select a pair in the middle. In a word, the health degree model provides a way to distinguish detections more finely as well as additional flexibility in performance adjusting.

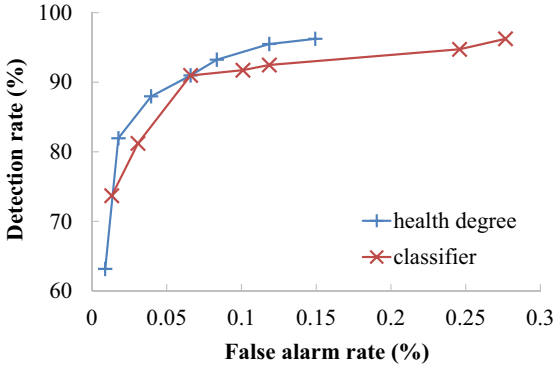

Figure 10. ROC curves of RT models. The points on curve of health degree model are obtained by setting threshold $= -0.5, -0.37, -0.3, -0.2, -0.1, -0.02$ and 0.0 from left to right. The points on curve of classifier are obtained by setting threshold $= -0.94, -0.86, -0.6, -0.4, -0.2, -0.05$ and 0.0 from left to right.

VI. Reliability Analysis

We use several Markov models to evaluate the benefits of our decision tree models on reliability.

Eckart *et al.* [17] deduced an formula to approximate the Mean Time To Data Loss (MTTDL) of a single hard drive with failure prediction:

$$MTTDL_{drive} \approx \frac{MTTF}{1 - \frac{k\mu}{\mu+\gamma}} \tag{7}$$

where MTTF means Mean Time To Failure of a single drive. k denotes FDR and γ is the inverse of TIA. μ is the inverse of the Mean Time To Repair (MTTR). This formula demonstrates the major relationship between the MTTDL of a single drive and the failure prediction accuracy.

Table VI shows the MTTDL of a single drive with different prediction models. The parameters come from real-world data centers and our experiments. All three prediction models improve the MTTDL by an order of magnitude even for the simplest single hard drive configuration. Moreover, although the prediction performance advantage of our decision tree models over the BP ANN model is small, it

Table VI
IMPACT OF FAILURE PREDICTION ON MTTDL. MTTF = 1,390,000 HOURS, MTTR = 8 HOURS. FOR CT MODEL, $k = 0.9549$, $\gamma = 1/355hours$; FOR RT MODEL, $k = 0.9624$, $\gamma = 1/351hours$, AND FOR BP ANN MODEL, $k = 0.9098$, $\gamma = 1/343hours$.

Model	MTTDL (years)	% increase
No prediction	158.67	0.00
BP ANN	1430.33	801.42
CT	2398.92	1411.84
RT	2687.31	1593.59

makes the twofold gap of MTTDL because of the superlinear growth of MTTDL as the prediction accuracy increases. This implies that even a small improvement in prediction accuracy is worthwhile.

To evaluate the benefits of drive failure prediction on large-scale storage systems, we build a Markov model showed in Figure 11 describing reliability of RAID-6 systems with proactive fault tolerance. For any N disks configuration, there are $3N+1$ states: $N+1$ prediction states, P_i where i represents the number of drives that are currently predicted to fail; N single-erasure states, SP_i where one drive has already failed and i drives are currently predicted to fail; $N-1$ double-erasure states DP_i; and the absorbing state F where data loss occurs.

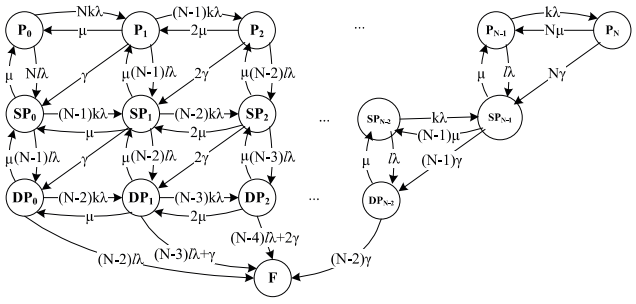

Figure 11. Reliability model for RAID-6 system with failure prediction. l denotes the missed alarm rate (i.e. $l = 1-k$).

Due to the pressure of construction and maintenance cost, the commercial companies are preferring consumer SATA hard drives which are much cheaper but less reliable than enterprise-class SAS drives to construct data centers. Therefore, we compare four RAID systems: two RAID-6 systems respectively composed of SAS and SATA drives, another SATA RAID-6 system, and a SATA RAID-5 system. The last two systems employ our CT model, and the first two do not use any drive failure prediction model. Figure 12 plots the systems' MTTDL as their size increases. We calculate the first two systems' MTTDL using Formula 8 [18], and the last two systems' MTTDL by the Markov models showed respectively in Figure 11 and [17].

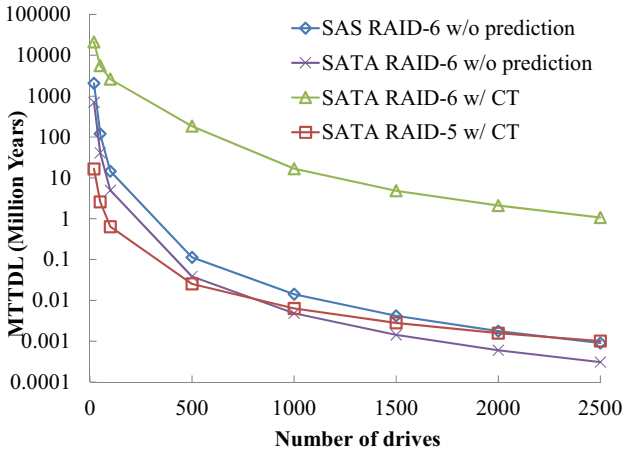

Figure 12. MTTDL of RAID systems. We use CT model whose $k = 0.9549$ and $\gamma = 1/355hours$. MTTF of SAS drive is set to 1,990,000 hours, and MTTF of SATA drive is set to 1,390,000 hours.

$$MTTDL_{RAID-6} \approx \frac{MTTF^3}{N(N-1)(N-2)MTTR^2} \quad (8)$$

As the figure shows, although the annual repair rate of SATA drives is almost 40% higher than that of SAS drives, the SATA RAID-6 system with CT model achieves MTTDL several orders of magnitude higher than that of the RAID-6 system composed of SAS drives but without drive failure prediction. This means that, with the help of our CT model, we indeed improve reliability of storage system constructed with cheap drives dramatically. The curves of the other three systems are close, especially when the systems are large. This indicates that, by employing our prediction model, you can keep similar or higher reliability while significantly reduce the construction and maintenance costs by reducing redundancy and/or using cheaper hard drives.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose new hard drive failure prediction models based on the classification and regression trees. Compared with the state-of the-art model, the Backpropagation artificial neural network model, our classification tree model performs better in prediction accuracy as well as stability and interpretability. We test prediction models with different drive families and with fewer number of drives, and our CT model still shows steady and good prediction performance and advantage over the BP ANN model. We present two model updating strategies to prevent model aging. To evaluate their effectiveness, we simulate long-term use of prediction models. Compared with the strategy remaining models unchanged, our updating strategies indeed keep smoother prediction accuracy and the 1-week replacing strategy performs best. These experimental results indicate that our new model is suitable for practical use. We also

propose a health degree model based on the regression tree. Compared with previous binary classifiers, such as our CT model, the RT model can give a drive in test a real value representing its health degree rather than a simple binary classification result. This value can serve for the priorities of detections to determine their processing order. Moreover, this model provides a easy way to tune the detection rate and the false alarm rate finely by adjusting the detection threshold. To evaluate the benefits of our models on the reliability, we present a Markov model describing RAID-6 systems with failure prediction. The simulation results show that our models can significantly improve reliability and/or reduce cost.

The health degree model does not completely outperform binary classifier models, such as our CT model. It is worthwhile to study other methods to build more effective health degree models. Moreover, we will try other statistical and machine learning methods, such as random forest, to boost the prediction performance. Finally, introducing prediction models into real-world distributed storage systems is the most important task in the near future.

ACKNOWLEDGMENT

This paper is partially supported by NSF of China (61373018, 11301288), Program for New Century Excellent Talents in University (NCET130301) and the Fundamental Research Funds for the Central Universities (65141021). Finally, We would like to thank the referees for their time and appreciate the valuable feedback.

REFERENCES

[1] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability," in *Proceedings of the 1st ACM symposium on Cloud computing*, Indianapolis, IN, June 2010, pp. 193–204.

[2] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?" in *Proceedings of 5th USENIX Conference on File and Storage Technologies (FAST)*, San Jose, CA, Feb 2007.

[3] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, "An analysis of latent sector errors in disk drives," in *Proceedings of the International Conference on Measurements and Modeling of Computer Systems*, New York, June 2007, pp. 289–300.

[4] L. N. Bairavasundaram, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, G. R. Goodson, and B. Schroeder, "An analysis of data corruption in the storage stack," *ACM Transactions on Storage (TOS)*, vol. 4(3), 2008.

[5] B. Allen, "Monitoring hard disks with SMART," *Linux Journal*, no. 117, Jan 2004.

[6] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application," *Journal of Machine Learning Research*, vol. 6, pp. 783–816, May 2005.

[7] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *Proceedings of the 18th International Conference on Machine Learning*, San Francisco, CA, June 2001, pp. 202–209.

[8] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved disk-drive failure warnings," *IEEE Transactions on Reliability*, vol. 51, no. 3, pp. 350–357, Sep 2002.

[9] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Hard drive failure prediction using non-parametric statistical methods," in *Proceedings of the International Conference on Artificial Neural Networks*, June 2003.

[10] Y. Zhao, X. Liu, S. Gan, and W. Zheng, "Predicting disk failures with HMM- and HSMM-based approaches," in *Proceedings of the 10th industrial conference on Advances in data mining: applications and theoretical aspects*, July 2010, pp. 390–404.

[11] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, "Proactive drive failure prediction for large scale storage systems," in *29th IEEE Conference on Massive Storage Systems and Technologies (MSST)*, Long Beach, CA, 2013, pp. 1–5.

[12] Y. Wang, Q. Miao, and M. Pecht, "Health monitoring of hard disk drive based on mahalanobis distance," in *Prognostics and System Health Management Conference (PHM-Shenzhen)*, Shenzhen, China, May 2011, pp. 1–8.

[13] Y. Wang, Q. Miao, E. W. Ma, K.-L. Tsui, and M. G. Pecht, "Online anomaly detection for hard disk drives based on mahalanobis distance," *IEEE Transactions on Reliability*, vol. 62, no. 1, pp. 136–145, March 2013.

[14] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Belmont, CA: Wadsworth, 1984.

[15] G. Williams, *Use R: Data Mining with Rattle and R: the Art of Excavating Data for Knowledge Discovery*. Springer, 2011.

[16] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proceedings of 5th USENIX Conference on File and Storage Technologies(FAST)*, San Jose, CA, Feb 2007, pp. 17–29.

[17] B. Eckart, X. Chen, X. He, and S. L. Scott, "Failure prediction models for proactive fault tolerance within storage systems," in *IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, Baltimore, MD, Sep 2008, pp. 1–8.

[18] G. A. Gibson and D. A. Patterson, "Designing disk arrays for high data reliability," *Journal of Parallel and Distributed Computing*, vol. 17, no. 1, pp. 4–27, Jan 1993.