DFPE: Explaining Predictive Models for Disk Failure Prediction

Yanwen Xie[†‡], Dan Feng[*†‡], Fang Wang[†‡], Xuehai Tang[§], Jizhong Han[§], Xinyan Zhang[†‡]
[†]Wuhan National Laboratory for Optoelectronics, School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, China
[‡]Key Laboratory of Information Storage System, Engineering Research Center of data storage systems and Technology
Ministry of Education of China, Wuhan, China
[§]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
Corresponding Email: dfeng@hust.edu.cn*

***Abstract*—Recent research works on disk failure prediction achieve a high detection rate and a low false alarm rate with complex models at the cost of explainability. The lack of explainability is likely to hide bias or overfitting in the models, resulting in bad performance in real-world applications. To address the problem, we propose a new explanation method DFPE designed for disk failure prediction to explain failure predictions made by a model and infer prediction rules learned by a model. DFPE explains failure predictions by performing a series of replacement tests to find out the failure causes while it explains models by aggregating explanations for the failure predictions. A presented use case on a real-world dataset shows that compared to current explanation methods, DFPE can explain more about failure predictions and models with more accuracy. Thus it helps to target and handle the hidden bias and overfitting, measures feature importances from a new perspective and enables intelligent failure handling.**

I. Introduction

Thousands and ten thousands of servers are clustered in datacenters to store tons of data and provide the services of Internet, cloud computing, data analysis and so on. Because of the maturity and high cost-effectiveness of disk technology, a large number of disks of different ages are serving in datacenters. As a result, failure occurs frequently [1], [3] and disk failure is the principal [3]. Disk failure results in data loss when a storage system does not deploy data redundancy schemes like disk raid, replication or erasure codes. For a system with any redundancy schemes, disk failure results in great overheads for recovering the lost, including storage I/O, network I/O and CPU burst.

To reduce the impact of disk failure, many works [4]–[13] are focusing on disk failure prediction. Disk failure prediction solutions raise alarms on coming disk failures so that the storage system has ample time to migrate data and services out of high-risk disks proactively. Thus it helps to keep data and services available all the time and reduce I/O and CPU burst caused by passive failure handling.

In general, a predictive model is trustworthy when **1)** it makes correct predictions on most existing cases in tests and **2)** it gives reasonable explanations on why it makes the predictions. Recent works [7], [9]–[12] on disk failure prediction tend to focus on the former and ignore the latter. They propose to employ complex models in disk failure prediction to improve the detection rate and the false alarm rate. However, they achieve the improvements at the cost of explainability. It is hard to understand why the models predict a disk to fail in the near future. Since tests cannot cover all the possible cases, a predictive model might perform well in tests but badly in application because hidden bias or overfitting has not been exposed in the tests.

Bias, also known as machine learning bias, means a model produces systematically prejudiced results. For example, the popular Google News Word2Vec model has the gender bias because the Google News dataset is inherently bias [14]. Overfitting means a model learns the noise of the training data and corresponds too exactly to the training data. Bias and overfitting result in low prediction accuracy in application of predictive models. They can be caused by unconscious oversight in data collection and processing. It is difficult to detect and handle bias and overfitting. However, high explainability of a model can help to detect bias and overfitting when the explanations for the predictions made by the model are presented [15]. Therefore, it is important to improve the explainability of complex models on disk failure prediction.

In this paper, we propose DFPE, a **D**isk **F**ailure **P**rediction **E**xplanation method to improve the explainability of complex models on disk failure prediction. DFPE explains failure predictions made by a model by extracting related features, and infers predictive rules learned by the model by aggregating the explanations for the failure predictions and then measuring feature importances. Moreover, DFPE provides more failure-related information to enable intelligent failure handling, which takes different actions in different failure situations rather than directly discards the high-risk disks.

To sum up, we make the following contributions in this paper:

- We propose a new explanation method to improve the explainability of current complex models on disk failure prediction. To our best knowledge, it is the first to target the explainability problem of disk failure prediction.
- We provide a case on a real-world dataset to show that bias might exist in complex models on disk failure prediction, and our new method helps to detect and handle the hidden bias.

TABLE I
NOTATION LIST

Symbol	Description
I	The input of a predictive model. It contains a number of features describing the running status of a disk in various aspects. $I = I_1 I_2 ... I_n$.
I_i or Fi	The i-th feature of a disk. It is located in the i-th column of I.
P or $P(I)$	The predictive model on disk failure prediction. It takes I as input and outputs O.
O	The output of a predictive model. It presents whether a disk is going to fail or not.
$T(I_i)$	The typical value of the i-th feature of normal disks.
$C(I, i, v)$	The modified input: the value of the i-th feature is modified to v.
$CS(I, S)$	The modified input: for each column i not in the column set S, the i-th feature is modified to $T(I_i)$.
$MFCS$	Minimum Failure Cause Set.
$im(MFCS, I_i)$	The feature importance of I_i on $MFCS$ ($i \in MFCS$).
E	The explanation. It is a set of $MFCS$s.
TP,FP,FN,TN	Confusion matrix of a binary classifier.
FDR	Failure Detection Rate.
FAR	False Alarm Rate.
$imp(I_i, P)$	The feature importance of I_i on the model P.

- We show that DFPE can be applied to measure feature importances and discuss how DFPE enables intelligent failure handling.

The rest of paper will present the details of the proposed method. Section II presents the background, the problems and related work. Section III describes the design of our method and Section IV presents the evaluation. Section V concludes the paper and presents our future work.

II. Background and related work

A. Notation list

The symbols used in the paper are listed in Table I.

B. Disk failure prediction

Disk failure prediction is to predict the future status of a disk: normal or failed. However, it is more than a classification problem because it has the following characteristics:

- It is an imbalanced classification problem since disk failure cases are much rarer than normal cases. Thus, failure predictions which predict disks to fail are often much rarer than normal predictions.
- It is a time series analysis problem since the disk status changes over time.
- It is a multiple-instance learning problem [4]. For a failed disk, its final status is known, but the exact change point when the disk turns to fail is unknown.

DFPE is designed for explaining failure predictions and models on disk failure prediction: 1) DFPE exploits the first characteristic by focusing on explaining failure predictions; 2) DFPE can explain models built for time series. 3) DFPE can find out change points with given models.

C. Abstraction for disk failure predictive models

Let I be the input of a predictive model. I consists of a number of features which describe the running indicators of a disk in various aspects: $I = I_1 I_2 ... I_n$. Most of recent works on disk failure prediction build predictive models based on SMART attributes. SMART (short for Self-Monitoring, Analysis and Reporting Technology) is a monitor system to detect and report various indicators of storage drive reliability. It is well deployed in hard disk drives (HDDs), solid-state drives (SSDs) and eMMC drives. For example, the common SMART attributes of HDDs include SMART 5 (Reallocated Sectors Count), SMART 7 (Seek Error Rate), SMART 189 (High Fly Writes) and so on. Besides, some works [12] take system-level metrics into consideration, like filesystem errors, read rate, write rate, I/O queue size, I/O waiting time and I/O utilization. As the values of the features change over time, I can be the current values, or the values during a period of recent time, or the values from the deployment time to now.

Let P be the predictive model on disk failure prediction. P takes I as input and outputs whether a disk is going to fail or not: $P : I \rightarrow O$. It is complicated to build a predictive model. Many algorithms, methods and tools are proposed to address the modeling problem, like sampling, value scaling, learning, voting and so on. In this paper, we focus on the learning models. Popular learning models deployed in disk failure prediction include SVM (Support Vector Machines) [7], Decision Tree [8], and ensemble models like Random Forest [11], [12], [16], GBDT (Gradient Boosting Decision Tree) [17], and artificial neural networks like MLP (MultiLayer Perceptron) [7], RNN (Recurrent Neural Network) [9], [17] and LSTM (Long short-term memory) [16].

Let O be the output of a predictive model. It infers whether a disk is going to fail or not in the near future. It can be a Boolean value (fail or not), a float value (how close to failure) [8] or an integer value (different levels of urgency) [9]. In this paper, O is default to a Boolean value and the other types of O can be converted to Boolean easily.

D. Explainability

With the rapid development of artificial intelligence, more and more complex models are proposed to improve the accuracy. The proposed models often have hundreds and thousands of parameters. It is really impossible for human to understand what exactly each parameter means and how the model infers the result from the input. In other words, explainability is sacrificed to improve the accuracy.

However, as the requirements for security and believability have grown stronger, more and more research works [15], [18]–[21] are focusing on improving the explainability of complex models. Explainability requires the model does not only output the result, but also explains why it infers the result and what rules it has learned. Although it is unrealistic to understand what exactly all the parameters inside a complex model mean, explainability just requires the model to qualitatively explain the relationship between the input and the output or to quantify how much each feature of the input contributes

to the output. Such explanations provide insights into the model so that one can decide whether the output and the model are trustworthy or not. The advantages of the emphasis of explainability are that 1) it helps to improve the believability of a model; 2) it helps to detect bias or overfitting in a model when the model gives out an unbelievable explanation; 3) it gives more detail about the result to support more intelligent decisions subsequently.

There are two popular methods to achieve high explainability: 1) Employ an explainable model. For example, decision tree models are easy to understand by checking the parameters to infer the rules. Complex explainable models with a high level of learning capacity have been being developed [18]. 2) Employ an explanation method to improve the explainability of complex models. The explanation method does a series of tests on the model, like random permutation tests [22], to infer the relationship between the input and the output and conclude the learned rules. Its advantage is that it works well with existing learning models which often achieve better learning performance than explainable models.

Recently, more and more complex models are deployed in disk failure prediction and they achieve really good learning performance (high detection rate and low false alarm rate), like Random Forest [11], [12], [16], GBDT [17], MLP [7], RNN [9], [17] and LSTM [16]. Therefore, we propose a new explanation method to improve the explainability of existing models on disk failure prediction without any degradation of the prediction accuracy.

E. Related work

1) Disk failure prediction: Disk failure prediction is well studied in recent years. Many machine learning methods are deployed to build high-quality predictive models with both a high detection rate and a low false alarm rate.

On one hand, some works build simple predictive models of high explainability. For example, Murray et al. [4] employ Naive Bayes. Pitakrat et al. [5] find out that Nearest Neighbors Classifiers get the best prediction quality among 21 classification algorithms. Li et al. [8] employ Decision Tree. Ma et al. [23] shows that the accumulation of reallocated sectors indicates the deterioration degree of a disk, so a predictive model can be built from the metric. For these models, how a prediction is made and what prediction rules are learned can be easily traced by checking the parameters inside the models. The simple models make predictions so fast that they are well deployed in the scenarios that emphasize low overheads.

On the other hand, some works achieve better prediction accuracy at the cost of explainability. The built models are complex. First, the models contain so many parameters that it is difficult to understand every parameter. For example, Zhu et al. [7] explore the ability of Backpropagation neural network (MLP models) and develop an improved SVM. Xu et al. [9] deploy RNN to measure the health levels of hard disks. Second, ensemble learning methods combine many basic models together to get a better performance. The more basic models an ensemble model contains, the lower explainability it has. For example, Botezatu et al. [10] employ Regularized Greedy Forests. Mahdisoltani et al. [12] find out that classifiers based on Random Forests can predict sector errors accurately. Xiao et al. [11] employ Online Random Forests which can update the predictive model online over time.

Our paper is targeted to improve the explainability of current complex models on disk failure prediction. With our new method, current complex models cannot only keep their high prediction accuracy, but also get the advantages of high explainability.

2) Explainability: With the development of artificial intelligence, more and more complex modeling methods are proposed and the demand of explainability is increasing steadily. Many explanation methods are proposed to improve the explainability of current complex models. According to explanation targets, explanation methods can be classified to local and global explanation methods.

Local explanation methods attempt to explain predictions made by models. Robnik-Šikonja et al. [21] propose to measure the importance of each feature on the prediction by calculating the difference between the original prediction and the one made with omitting the feature. Baehrens et al. [20] propose to explain the prediction result by measuring the local gradients that characterize how the input has to be adjusted to change the prediction result. Ribeiro et al. [15] propose LIME to explain the predictions of any classifier by learning the model locally around the prediction.

Global explanation methods attempt to explain models. Mean Decrease in Impurity (MDI) and Mean Decrease in Accuracy (MDA) [22], [24] are two popular methods to explain tree models by measuring the importances of features. MDI counts the number of node splits including the feature, weighted by the number of samples it splits while MDA calculates the mean increase error of the model when the values of the feature are randomly permuted. Lakkaraju et al. [19] propose BETA, a model agnostic framework to produce global explanations by optimizing for fidelity to the original model and interpretability of the explanation.

These above explanation methods are designed for any applications while our new method is designed for disk failure prediction. Because of the listed characteristics of disk failure prediction, our new method can explain more and better than current explanation methods.

Besides, explainable models with high learning capacity are being developed [18]. It is an another solution to provide models on disk failure prediction with both high prediction accuracy and high explainability.

III. The proposed explanation method

Learning from previous works [20], [21], we develop DFPE, a new explanation method which performs a series of replacement tests to make explanations for disk failure prediction.

A. Replacement tests

For the input I of a disk, replace the i-th feature with the value v and we get the modified input $C(I, i, v)$. Let $T(I_i)$

be the typical value of the i-th feature of normal disks. It can be the mean or median value of the feature of normal disks. Thus $C(I, i, T(I_i))$ means to omit the i-th feature. For each column i not in the column set S, replace the i-th feature with $T(I_i)$ and finally we get the modified input $CS(I, S)$. $CS(I, S)$ means to omit all features except those in S.

For disk failure prediction, a predictive model is supposed to make many more normal predictions than failure predictions. However, the failure predictions are much more important than the normal predictions. Thus the purpose of replacement tests is to test whether a failure prediction is caused by a given feature set. For a feature set S, a replacement test will test whether both $P(I)$ and $P(C(I, S))$ predict a disk to fail. If so, that means the model makes the same failure prediction on the disk even when all of the features outside S are omitted. Thus, the features in the set S cause the failure prediction for the disk.

B. Definition of MFCS

Definition 1: For the input I of a disk which is predicted to fail $P(I) = true$, when a column set S meets the following conditions, S is defined to be a Minimum Failure Cause Set ($MFCS$) of the disk:

1) $P(CS(I, S)) = true$
2) $\nexists S' \subset S : P(CS(I, S')) = true$

The first condition shows that even when all of the features not in the $MFCS$ are omitted from the input I, the predictive model P still predict the disk to fail. In other words, for the disk, the features in the $MFCS$ convince the model P that the disk would fail in the near future. The second condition shows the minimality. There are no proper subsets of $MFCS$ that will convince P that the disk would fail. Any feature in the $MFCS$ is essential to support the failure prediction.

$MFCS$ is defined to explain failure predictions. An $MFCS$ tells which features get the model to make the failure prediction. Moreover, a learned rule can be inferred from an $MFCS$ as the model has learned the relationship between the failure and the features in the $MFCS$.

Let $im(MFCS, I_i)$ be the importance of the feature I_i for the $MFCS$ ($i \in MFCS$). Learning from [20], DFPE measures $im(MFCS, I_i)$ by measuring how the feature can be adjusted to change the prediction result as shown in Algorithm 1. Since most features about disk reliability have the increasing or decreasing trends [4], DFPE calculates the change point for the feature by Binary Search with limited steps ($Step_{max}$). Then DFPE normalizes the distance between the change point and the value of the feature. To be noticed, I_i can be a single value or a time series, but Algorithm 1 can work for both cases.

As a disk may be predicted to fail because of more than one rule, there may be more than one $MFCS$ for a failure prediction. Let E be the explanation for a failure prediction on a disk. E is a set of $MFCS$s and $E = \{MFCS_1, MFCS_2 ... MFCS_m\}$.

Algorithm 1: Measuring $im(MFCS, I_i)$, the importance of the feature I_i on the $MFCS$.

Input: I: the input of a disk, I_i: the i-th feature, P: the predictive model, $MFCS$, and $T(I_i)$: the typical value of the i-th feature of normal disks.
Output: $impact$: the importance value.

```
curI = CS(I, MFCS);
left = T(I_i), right = I_i;
for k = 1 to Step_max do
    cur = 0.5 (left + right);
    if P(C(curI, i, cur)) then
        right = cur;
    else
        left = cur;
    end
end
changepoint = 0.5 (left + right);
impact = |changepoint − left| / |right − left|;
return impact;
```

C. Explaining a failure prediction

To explain a failure prediction on a disk is now transformed to find out all $MFCS$s or E for the disk. For a disk with n features, the complexity of testing all possible $MFCS$s is $O(2^n)$. It will cost so much time. Therefore, rather than tests all possible $MFCS$s, DFPE employs a two-step method to find out as many $MFCS$s as possible.

In the first step, DFPE does some replacement tests to look for potential $MFCS$s as shown in Algorithm 2. Algorithm 2 contains two nested loops. The inner loop (line $5 \sim 15$) attempts to find out an $MFCS$ by replacing the features one by one to test whether the feature influences the prediction result. If so, DFPE will roll back the replacement and add the feature to the current $MFCS$. If not, DFPE will keep the replacement and continue the iteration. After finding an $MFCS$, DFPE omits all features in the $MFCS$ and tests whether there are more $MFCS$s in the outer loop. If so, DFPE attempts to find out more $MFCS$s with the inner loop again. If not, DFPE returns all the found $MFCS$s.

Algorithm 2 can only find out $MFCS$s without overlaps. For example, it can find out $\{1, 3\}$ and $\{2, 4\}$. However, it cannot find out all $MFCS$s when there are overlaps. For example, when the $MFCS$s are $\{1, 3\}$ and $\{2, 3\}$, Algorithm 2 can only find out $\{2, 3\}$ because the feature I_3 is omitted after $\{2, 3\}$ is found. In order to find out more $MFCS$s for a failure prediction, DFPE takes the second step.

In the second step, DFPE maintains an $MFCS$ set, called $knownMFCSs$, for a predictive model. $knownMFCSs$ contains all known $MFCS$s found by Algorithm 2 for historical failure predictions. In general, DFPE builds $knownMFCSs$ with the training data. As Algorithm 3 shows, DFPE checks every element in $knownMFCSs$ to find out more $MFCS$s for a failure prediction. In order to reduce the number of checks, DFPE first sorts the elements in

Algorithm 2: Step 1: Looking for potential $MFCS$s.

Input: I: the input with n features of a disk, and P: the predictive model
Output: E: the explanation

```
1  E = ∅;
2  curI = I;
3  curS = fullS = {1, 2..., n};
4  while P(curI) do
5      MFCS = ∅;
6      foreach i ∈ curS do
7          if P(C(curI, i, T(I_i))) then
8              curI = C(curI, i, T(I_i));
9          else
10             MFCS = MFCS ∪ {i};
11             curS = curS − {i};
12         end
13     end
14     E = E ∪ {MFCS};
15     curI = CS(I, fullS − MFCS);
16 end
17 return E;
```

Algorithm 3: Step 2: Checking known $MFCS$s.

Input: I: the input with n features of a disk, P: the predictive model, $knownMFCSs$: known $MFCS$s, and E: the explanation
Output: E: the explanation

```
1  Sort knownMFCSs;
2  foreach KS ∈ knownMFCSs do
3      if ∄S ∈ E : (S ⊆ KS) ∨ (KS ⊆ S) then
4          if ∃S ∈ E : S ∩ KS ≠ ∅ then
5              if P(CS(I, KS)) then
6                  E = E ∪ {KS};
7              end
8          end
9      end
10 end
11 return E;
```

$knownMFCSs$ in ascending order of size. Then for each element KS in $knownMFCSs$, DFPE checks whether there is any element in E which is a subset of KS or a superset of KS (See Line 3). If so, KS cannot be an $MFCS$ of the disk because either of the conditions in the definition of $MFCS$ cannot be met. If KS is a subset of an element in E, $P(CS(I, KS))$ equals to false, so the first condition cannot be met. If KS is a superset of an element in E, the second condition cannot be met. If the test at Line 3 is passed, DFPE further tests whether there is any found $MFCS$ that shares features with KS (See Line 4). Because Algorithm 2 has found all $MFCS$s without overlaps, Algorithm 3 can only find out $MFCS$s overlapping with found $MFCS$s. Finally, if the check cannot rule out KS, DFPE will perform a replacement test to check whether KS is really an $MFCS$ of the disk.

DFPE makes no guarantee that it will find out all $MFCS$s. Suppose an $MFCS$ does exist and hides from the above two steps. According to the first step, the $MFCS$ appears together with another $MFCS'$. According to the second step, there are two cases. First, the $MFCS$ has never appeared before. In this case, it is hard to find it out heuristically without traversing through all possibilities. Second, the $MFCS$ has never appeared alone before. In this case, it probably implies that there are redundant features that just cause redundant explanations. Therefore, DFPE has not employed more steps to dig hidden $MFCS$s out.

D. Explaining a predictive model

To explain a predictive model, DFPE aggregates all historical explanations for failure predictions to infer prediction rules a predictive model has learned. In general, DFPE explains models with the training data or validation data. DFPE treats an $MFCS$ as a prediction rule. For each $MFCS$, DFPE counts how many failure predictions caused by the $MFCS$ are correct and incorrect respectively (labeled as TP_{MFCS} and FP_{MFCS}). Let FN_{MFCS} be the number of failed disks that do not have the $MFCS$. Let TN_{MFCS} be the number of normal disks that do not have the $MFCS$. Then DFPE further calculates how many percent of correct failure predictions are caused by the $MFCS$ and how much its false alarm rate is (labeled as FDR_{MFCS} and FAR_{MFCS}) according to Equations (1) and (2) respectively.

$$FDR_{MFCS} = \frac{TP_{MFCS}}{TP_{MFCS} + FN_{MFCS}} \tag{1}$$

$$FAR_{MFCS} = \frac{FP_{MFCS}}{FP_{MFCS} + TN_{MFCS}} \tag{2}$$

For an $MFCS$, FDR_{MFCS} describes its popularity and importance while FAR_{MFCS} describes its believability. DFPE sorts all the $MFCS$s according to their FDR_{MFCS}s to present the most popular failure causes. Meanwhile, DFPE sorts all the $MFCS$s according to their FAR_{MFCS}s to present the most suspected rules which alarm the administrator to handle related failure predictions carefully.

At last, DFPE measures the importance of every feature on the predictive model (labeled as $imp(I_i, P)$). For each feature I_i, DFPE counts how many failed disks are predicted successfully, for which the explanation contains the feature. The counter is labeled as TP_{I_i}. Then DFPE calculates $imp(I_i, P)$ by normalizing TP_{I_i} according to Equation (3). The $imp(I_i, P)$ values represent the importances of the features in the model. The bigger the value of $imp(I_i, P)$ is, the more important the feature is. The values are useful in feature engineering, such as feature selection where the features with the top $imp(I_i, P)$ values can be selected to build a more concise model.

$$imp(I_i, P) = \frac{TP_{I_i}}{\sum_{j=1}^{n} TP_{I_j}} \tag{3}$$

E. Complexity and overhead analysis

The most time-consuming operation of DFPE is to calculate $P(I)$ to make a prediction. Thus the time complexity is measured by the calculation times of $P(I)$. Current methods for disk failure prediction without explanation need calculate $P(I)$ just once, so their time complexities are $O(1)$. DFPE need calculate $P(I)$ many more times so that it can make explanations on failure predictions and provide more details about the coming failures. We believe it's acceptable because of the advantages of high explainability.

Algorithm 2 attempts to find out all $MFCS$s without overlaps. Its complexity is determined by how many $MFCS$s can be found. Let n be the number of features in the dataset. In the best case, there is only one $MFCS$ and DFPE needs to make replacement tests on every feature, so the tight lower bound of the complexity is $\Omega(n)$. In the worst case, every feature comes to be an $MFCS$ and DFPE can find out only one $MFCS$ for a loop of replacement tests on all features. Therefore, the tight upper bound of the complexity is $O(n^2)$. In the tests, we found that the mean number of $MFCS$s for a disk is often much less than the number of features, so the actual cost will be much less than that in the worst case.

Algorithm 3 attempts to find out more $MFCS$s by checking known $MFCS$s. The complexity is $O(|knownMFCSs|)$. In the worst case, $|knownMFCSs| = 2^n$. However, the worst case is rare because the number of $|knownMFCSs|$ is often much smaller than 2^n. Moreover, DFPE skips a large part of known $MFCS$s by checking their relationships with found $MFCS$s. Thus, the actual cost will be much less than that in the worst case.

Algorithm 1 performs binary search to measure the importance of a feature in limited steps ($Step_{max}$), so its time complexity is $O(Step_{max})$. Because $Step_{max}$ is a specified constant parameter, $O(Step_{max}) = O(1)$. An explanation for a failure prediction cannot have more than 2^n $MFCS$s and an $MFCS$ cannot have more than n features. Therefore, the upper bound complexity of measuring feature importances for an explanation is $o(n * 2^n)$. However, we found in the evaluation that the mean number of $MFCS$s for an explanation is often much less than the number of features, so the actual cost will be much less than that in the worst case.

To explain a failure prediction, DFPE finds out $MFCS$s according to Algorithms 2 and 3 and measures the feature importances of found $MFCS$s. Therefore, The complexity is $O(n^2)+O(2^n)+o(n*2^n) = o(n*2^n)$. It would cost so much time in the worst case. However, the above description infers that although the worst case is really bad, the actual cost is often much less than the cost in the worst case. In Section IV, we will show that the cost is acceptable.

Finally, DFPE infers predictive rules to explain a predictive model by gathering explanations for failure predictions. The inference method is simple since it just does some counting works and calculates FDR_{MFCS}, FAR_{MFCS} and $imp(I_i, P)$ with the counters. Therefore, the dominant cost of explaining a model lies in explaining failure predictions.

To reduce the overheads, DFPE can be customized by disabling some functions. For example, Algorithm 3 can be omitted, which would reduce the number of found $MFCS$s. Explaining a model can also be optional. DFPE can just explain predictions just like other local explanation methods. Moreover, sampling technology can be employed to reduce the overheads further. For example, when explaining a model, DFPE can sample the dataset by limiting the length of time series for each disk.

IV. Evaluation

A. Setup

We ran the evaluation in a server with two Intel Xeon E5-2620 CPUs, 128 GB memory and Ubuntu x86_64 16.04 LTS with Linux kernel 4.4.0. We implemented DFPE and related methods based on scikit-learn [25] 0.19.1 and PyTorch [26] 0.4.0. The datasets involved in the evaluation are presented in Table II. All the datasets in Table II are formed of records of SMART attributes. Dataset D0 has been used in [7]–[9], [17], D1, D3, D4 and D8 in [10], and D1 and D2 in [11].

For each disk series, the dataset was split into training set and test set according to the ratio $7:3$. The training set is used to train, tune and then explain the predictive models. With the training sets, we built models and tuned them by iterating small ranges of training parameters. The built models are not guaranteed to be optimal. One may get better models with further tuning or new modeling methods. How to build a high-quality model on disk failure prediction is not the main topic of this paper. The test sets are used to evaluate the performance of the models and to present the explanations for failure predictions.

In the evaluation, we first focused on dataset D0 and compared the explanation made by DFPE and other explanation methods in detail, and then we expanded the evaluation over the other datasets to show the usages and the overheads of DFPE. $Step_{max}$ is set to 10, so the precision of $im(MFCS, I_i)$ equals to $2^{-10} \approx 0.001$.

B. Evaluation on D0

The SMART attributes selected to build predictive models for D0 are listed in Table III. The predictive models employed in this paper were Random Forest models.

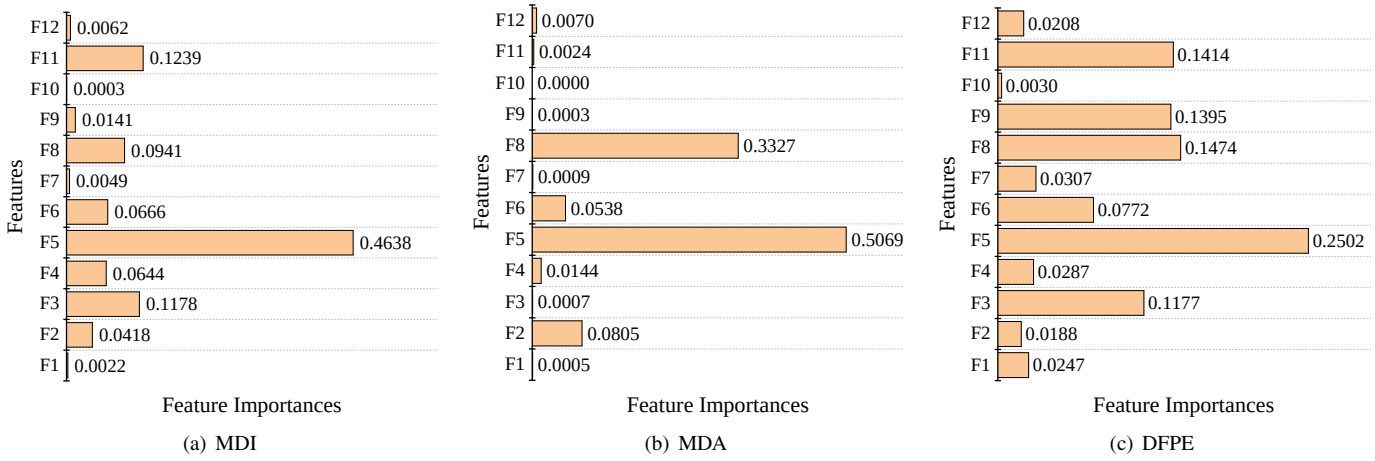
1) Explaining a predictive model: MDI and MDA are two popular methods to explain a Random Forest model by measuring feature importances. Figures 1(a) and 1(b) shows the explanations made by MDI and MDA respectively. Compared with them, DFPE not only measures the feature importances as shown in Figure 1(c), but also infers prediction rules and calculates the metrics of the rules as shown in Tables IV and V.

Figure 1 shows the comparison among the three methods on measuring feature importances. On one hand, some differences are observed. It shows that the three methods have large differences in the importances of F11, F9, F7 and F3. For example, MDI and MDA believe F9 is not important while DFPE believes F9 is important. On the other hand, there are more similarities. For example, F5 is the most important

TABLE II
DATASETS OF SEVERAL DISK SERIES

Label	Disk Series	Collected From	Download	Normal disks	Failed disks	Sampling Interval	Total time
D0	Seagate ST31000524NS	Baidu Company	[7]	22962	433	1 hour	1 week or 20 days[1]
D1	Seagate ST4000DM000	Backblaze Company	[27]	34295	2502	1 day	Feb 2014 ∼ Sep 2017
D2	Seagate ST3000DM001			2898	1006		Feb 2014 ∼ Nov 2015
D3	Seagate ST31500541AS			1679	238		Feb 2014 ∼ Sep 2017
D4	Hitachi HDS722020ALA330			4535	193		Feb 2014 ∼ Sep 2017
D5	WDC WD30EFRX			1161	152		Feb 2014 ∼ Sep 2017
D6	HGST HMS5C4040ALE640			8569	126		Feb 2014 ∼ Sep 2017
D7	HGST HMS5C4040BLE640			16181	120		Mar 2014 ∼ Sep 2017
D8	Hitachi HDS5C3030ALA630			4512	116		Feb 2014 ∼ Sep 2017
D9	Seagate ST8000DM002			9882	110		May 2016 ∼ Sep 2017

[1] For D0, 1 week samples are collected for normal disks and 20 days before the failure for failed disks



(a) MDI (b) MDA (c) DFPE

Fig. 1. **Feature importances measured by MDI, MDA and DFPE respectively**. Similarities can be observed in the features F5, F6, F8 and F10 while differences in F3, F7, F9 and F11.

TABLE III
SELECTED SMART ATTRIBUTES AND THEIR LABELS IN THIS PAPER

ID	Attribute Name	Value[1]	Raw_Value[2]
1	Read Error Rate	F1	
3	Spin Up Time	F2	
5	Reallocated Sectors Count	F3	F11
7	Seek Error Rate	F4	
9	Power On Hours	F5	
187	Reported Uncorrectable Errors	F6	
189	High Fly Writes	F7	
194	Temperature Celsius	F8	
195	Hardware ECC Recovered	F9	
197	Current Pending Sector Count	F10	F12

[1] Value: Normalized value of the raw counter ranging from 0 to 255. The normalization methods are defined by the vendors.
[2] Raw_Value: Raw counter.

feature, F8 and F6 are somewhat important, and F10 is the least important. DFPE is designed for disk failure prediction. Since disk failure prediction is an imbalanced classification problem, DFPE only analyzes the failure predictions. It is the key difference between DFPE and the other two methods. Thus the results from DFPE is supposed to be more accurate than those from the other two methods. Moreover, due to the difference, the importance values measured by DFPE distribute more evenly, which allows better comparison between features.

Feature selection can be done after measuring feature importances. We built models for the 10 datasets after employing feature selection based on the three methods respectively. Because of the similarities, many common features were selected by the three methods. As a result, no obvious improvements made by DFPE were observed. Therefore, DFPE just provides another perspective for measuring feature importances in a model and another option for feature selection.

Compared to MDI and MDA, DFPE can explain more about a predictive model by inferring prediction rules. DFPE sorts the rules according to their detection rates to show the most popular and important rules intuitively. Table IV shows the top 10 important rules the Random Forest model had learned. It can be seen that most failures can be detected by only a few of rules. Moreover, DFPE sorts the rules according to their false alarm rates to show the most suspected rules intuitively. Table V shows the top 10 suspected rules the Random Forest model had learned. It can be seen that most false alarms are caused by only a few of rules. It also shows that the false alarm rates of the rules are really small, which means the Random Forest model had corresponded to the training data

TABLE IV
INFERRED RULES BY DFPE SORTED BY THEIR FDR_{MFCS}S

Inferred Rule ($MFCS$)	$FDR_{MFCS} \downarrow$ [1]	FAR_{MFCS}
$\{5\}$	0.7822	0.00019
$\{8\}$	0.3201	0.00075
$\{3, 9, 11\}$	0.3168	0.00137
$\{6\}$	0.1518	0.00000
$\{8, 9, 11\}$	0.0495	0.00006
$\{1, 3, 6\}$	0.0462	0.00000
$\{6, 11\}$	0.0396	0.00000
$\{3, 5\}$	0.0330	0.00093
$\{4, 8, 12\}$	0.0330	0.00006
$\{2, 7, 8, 11\}$	0.0330	0.00006
...	...	...

[1] A failure prediction may be cause by several $MFCS$s, so the sum of FDR_{MFCS} is larger than 1.

TABLE V
INFERRED RULES BY DFPE SORTED BY THEIR FAR_{MFCS}S

Inferred Rule ($MFCS$)	FDR_{MFCS}	$FAR_{MFCS} \downarrow$ [1]
$\{3, 9, 11\}$	0.3168	0.00137
$\{3, 5\}$	0.0330	0.00093
$\{8\}$	0.3201	0.00075
$\{5\}$	0.7822	0.00019
$\{3, 4, 8, 9\}$	0.0198	0.00012
$\{1, 8\}$	0.0033	0.00006
$\{4, 8, 12\}$	0.0330	0.00006
$\{6, 8, 11\}$	0.0231	0.00006
$\{8, 9, 11\}$	0.0495	0.00006
$\{4, 8, 11\}$	0.0099	0.00006
...	...	...

[1] A false alarm may be cause by several $MFCS$s, so the sum of FAR_{MFCS} is often larger than the false alarm rate of the model.

very well. In general, rules with high false alarm rates or low detection rates can be excluded without rebuilding to improve the quality of a model.

2) Explaining a failure prediction: We randomly selected a failed disk in the test set to show the explanation made by DFPE. For comparison, we also employed LIME to explain the same failure prediction. The explanation of LIME in Figure 2 shows that F5, F2, F12, F11, F7, F9 and F1 attempts to convince the model that the disk would fail in different degrees while F4 and F8 attempts to convince the model that the disk is working normally. It also shows that F5 is the dominant factor resulting in the failure prediction.

Compared to LIME, DFPE can explain more about the failure prediction and the explanation is more accurate. Table VI shows that DFPE found out three $MFCS$s for the failure prediction. Each $MFCS$ could convince the model to make the failure prediction individually. According to Table IV, the three $MFCS$s belong to the top 4 important $MFCS$s, which means the three rules have worked correctly for many failed disks. Among them, $\{6\}$ has the smallest FAR_{MFCS} that equals to 0, which means the failure prediction is highly believable. DFPE also measured the importances of features on the failure prediction as shown in Figure 3. It shows that F6, F5, F9, F3 and F11 are important for the failure prediction, which is quite different from the explanation of LIME. The

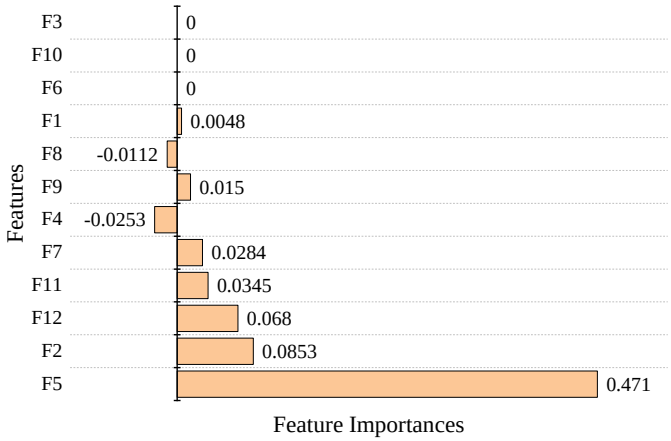


Fig. 2. **An explanation made by LIME for a failure prediction**. LIME explains a prediction by measuring the feature importances. It shows that for the failure prediction, F5, F2, F12, F11, F7, F9 and F1 results in the failure prediction and F5 is the dominant feature.

TABLE VI
AN EXPLANATION MADE BY DFPE FOR A FAILURE PREDICTION

$MFCS$	FDR_{MFCS}	FAR_{MFCS}
$\{6\}$	0.1518	0.00000
$\{5\}$	0.7822	0.00019
$\{3, 9, 11\}$	0.3168	0.00137

reason is that DFPE, different from LIME, measures the importances of features in an $MFCS$ by omitting the other features outside the $MFCS$. For this example, the model can make the failure prediction simply because of just F5, but omitting F5 would not change the failure prediction because of F6, F3, F9 and F11. Thus the importances of F6, F3, F9 and F11 cannot be exposed without excluding the influence of F5. Therefore, the explanation of DFPE is more accurate than that of LIME in the application of disk failure prediction.

3) Detecting and handling bias: From the explanations of MDA, MDI and LIME as shown in Figures 1(a), 1(b) and 2, it can be seen that F5, the power-on time of a disk, is the most important feature in disk failure prediction. It is reasonable because the longer a disk has run, the more likely it would be to fail. However, from the explanations of DFPE as shown in Figures 1(c) and 3 and Tables IV to VI, it can be seen that the model has learned that F5 can determine the failure prediction alone. It means that whenever the power-on time of a disk expires a certain threshold, the model would predict the disk to fail definitely. The learned rule is unreasonable as it would cause lots of false alarms after a certain period of usage of disks. In short, there is bias in the model and DFPE helps to expose it.

Because the false alarm rate of the rule $\{5\}$ is really low as shown in Table V, the bias is likely to be caused by the dataset rather than the modeling method. To confirm that the bias is not caused by the modeling method, we built several predictive models from other learning methods: GBDT, XGBoost [28], SVM, MLP and LSTM. Table VII shows that all of these

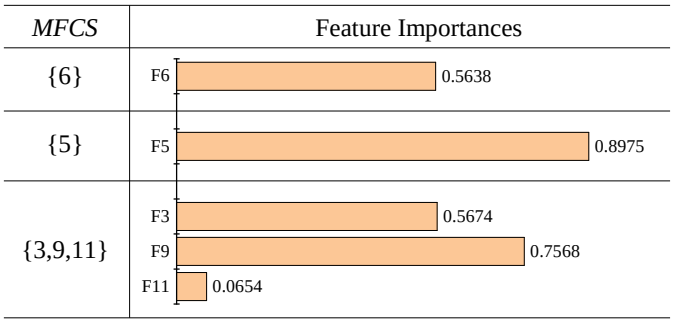


Fig. 3. **An explanation made by DFPE for a failure prediction**. It shows that the explanation contains three $MFCS$s and DFPE measures the feature importances for every $MFCS$ separately.

TABLE VII
DIFFERENT LEARNING MODELS LEARNED THE SAME RULE $\{5\}$ FOR D0

Modeling	Rank[1]	FDR_{MFCS}	FAR_{MFCS}
Random Forest	1	0.7822	0.00019
GBDT	1	0.7096	0.00000
XGBoost	1	0.6469	0.00000
SVM	1	0.5050	0.00013
MLP	1	0.7657	0.00065
LSTM	1	0.6337	0.00000

[1] Ranked by FDR_{MFCS} which indicates the importance of a rule.

models learned the rule $\{5\}$ and it is the most important rule with the highest detection rate and a really low false alarm rate. It means the bias exists no matter which modeling method is employed. To target the bias in the dataset, Figure 4 shows the value distribution of F5 of normal disks and failed disks respectively. The original F5 values range from 0 to 255 and they are scaled linearly to the range $[-1, 1]$ in pre-process. The following characteristic remains: the bigger value, the less power-on time. It can be seen that F5 values of normal disks range from -0.08 to 1 while F5 values of failed disks range from -1 to 1. Missing samples of normal disks with F5 values ranging in $[-1, -0.08]$ leads to a model easily learning the rule that a disk would fail if its power-on time expires a certain threshold. It means the dataset has the age bias. The age bias is probably caused by a data collection method that one-week samples of all disks are exported at a certain time and after that time, only samples from failed disks are updated.

There are two simple methods to handle the data bias: 1) Let the model predict a disk to fail only when there exists any $MFCS$ not equal to $\{5\}$ in the explanation. 2) Rebuild a model without the feature F5. The evaluation results of the two methods based on the Random Forest model are shown in Table VIII. It shows that the former method has a better FAR while the latter has a better FDR. Both methods perform worse than the original model because they have not exploited the data bias. However, both methods are supposed to perform better in the real world than the original model because they don't have the unsound rule.

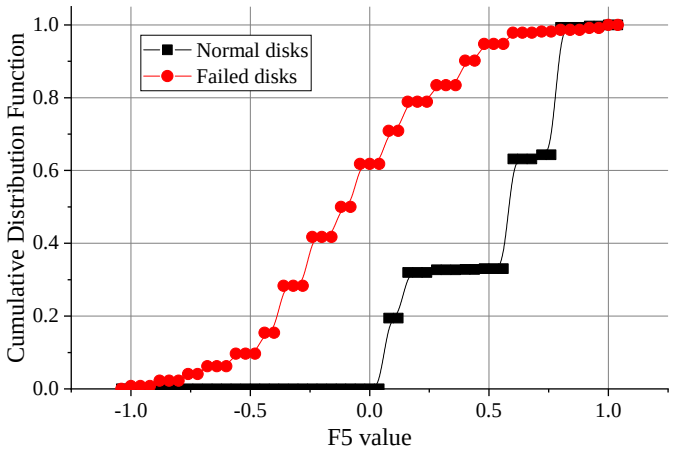


Fig. 4. **Cumulative Distribution Function of F5 values**. Data bias can be observed obviously. The F5 values of normal disks distribute in a narrow range while those of failed disks in a wide range.

TABLE VIII
PREDICTION METRICS AFTER HANDLING THE DATA BIAS FOR D0

Method	FDR	FAR
Original	0.8769	0.0033
1	0.6385	0.0029
2	0.6769	0.0109

4) Summary: The evaluation on D0 has shown that compared to current methods, DFPE explains more about a model and more about failure predictions made by the model, and the explanations of DFPE are more accurate. Thus DFPE helps to detect and handle the bias intuitively.

C. The overheads

We expanded the evaluation to the ten datasets to show the overheads of DFPE. For each dataset, a Random Forest model for evaluation is built on selected 12 most related features. SMART attributes listed in the Table IX are selected for one or more datasets beside those listed in Table III. As some SMART attributes suggest the age of a disk, a model might have the mentioned age bias, so it is helpful to deploy DFPE to detect and handle the bias. The overheads are measured by related time costs. The presented time costs are normalized by dividing them by the mean time cost of a model making a prediction and then multiplying them by 10^{-6} seconds to exclude the influence of the complexity of a model. The reason is that the magnitude of the time cost of a model making a prediction is 10^{-6} seconds in the evaluation.

The overheads of explaining a model are measured by the time cost of explaining the model with the training set. MDI measures feature importances in the modeling period, so it requires no more costs. Thus we only compare the time costs of MDA and DFPE as shown in Figure 5. Compared to MDA, DFPE requires 89X time for D2 and 1.2X $\sim$ 12X time for the other datasets. The overheads of DFPE is much more than those of MDA because DFPE explains every failure prediction in detail by a lot of replacement tests to extract

TABLE IX
ADDITIONAL SELECTED SMART ATTRIBUTES FOR D1∼D9

ID	Attribute Name
2	Throughput Performance
4	Start/Stop Count
10	Spin Retry Count
12	Power Cycle Count
188	Command Timeout
191	G-sense Error Rate
192	Power-off Retract Count
193	Load Cycle Count
196	Reallocation Event Count
198	Uncorrectable Sector Count
241	Total LBAs Written
242	Total LBAs Read

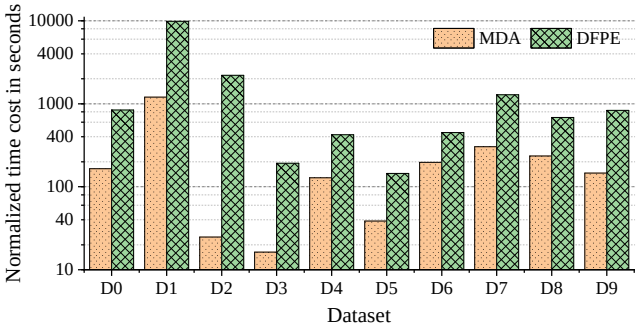


Fig. 5. **The normalized time cost of explaining a model**. It shows that DFPE requires much more time than MDA because DFPE attempts to make an explanation in more details.

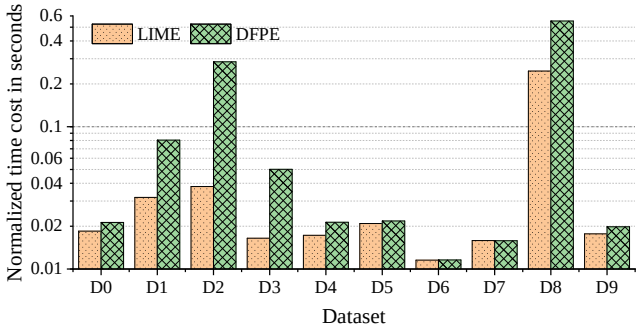


Fig. 6. **The mean time cost of explaining a failure prediction**. It shows that DFPE requires 1.1X $\sim$ 8.0X time compared to LIME because DFPE attempts to find out more causes while LIME is focusing on the main cause.

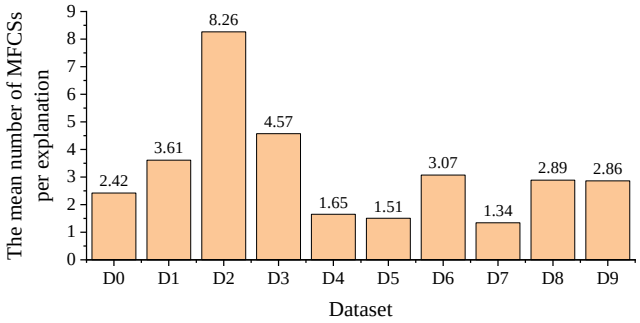


Fig. 7. **The mean number of $MFCS$s per explanation**. It shows that the values are less than 12, the number of features. Therefore, the actual costs are much less than the theoretical costs in the worst case.

more information about the model. Besides, given the same number of features, the time cost of MDA is determined by the size of the training set while that of DFPE is determined mostly by the number of failure predictions for the training set. The dataset D2 is relatively small but it has many failure predictions, so DFPE requires much more time than MDA. Explaining a model is an offline task of data analysis, so the time cost of DFPE is acceptable considering the benefits of high explainability.

The overheads of explaining predictions are measured by the mean time cost of explaining a prediction for a disk in the test set. Let LIME and DFPE only explain failure predictions. Figure 6 shows the time cost of LIME and DFPE respectively. It shows that compared to LIME, DFPE requires 1.1X $\sim$ 8.0X time for the ten datasets. The overheads of DFPE are more than those of LIME because DFPE attempts to explain more by looking for more failure causes and measuring feature importances separately. Explaining a prediction is an online task following after making a prediction online. The faster the better. However, when a model makes a failure prediction, it can't be too cautious to make an explanation because failure handling would cost much more than making an explanation.

We further look into the cost of DFPE. Figure 7 shows that the mean numbers of $MFCS$s per explanation for the ten datasets range from 1.3 to 8.2. They are less than 12, the number of features, so the actual costs of looking up $MFCS$s according to Algorithm 2 and measuring feature importances according to Algorithm 1 are much less than the theoretical costs in the worst case. Figure 8 shows that the $|knownMFCSs|$ values for the ten datasets range from 11 to 192. They are much less than 2^{12}, the theoretical maximum size of $knownMFCSs$, so the actual costs of checking known $MFCS$s according to Algorithm 3 are much less than the theoretical costs in the worst case. Figure 8 also shows that the mean numbers of checking known $MFCSs$ per explanation are about $8\% \sim 46\%$ of the size values. That means the filter in Algorithm 3 helps to reduce $54\% \sim 92\%$ of the overheads of Algorithm 3.

To show the relationship between the overheads of DFPE and the number of selected features, models over the dataset D4 with different numbers of features were built and explained. To save time, only D4 was chosen since the models on D4 were explained quickly. Figure 9 shows that with the increasing number of selected features, the overheads tend to grow slowly with considerable oscillations. The reason of the slow growth is that only a small part of features are related to disk failure and most failures can be predicted with even less features. Due to the slow growth, DFPE would have a great scalability in application. The reason of the oscillations is that a model might learn the noise in the training data when too few features or unrelated features are involved, which results in uncertainly increasing time costs of making explanations.

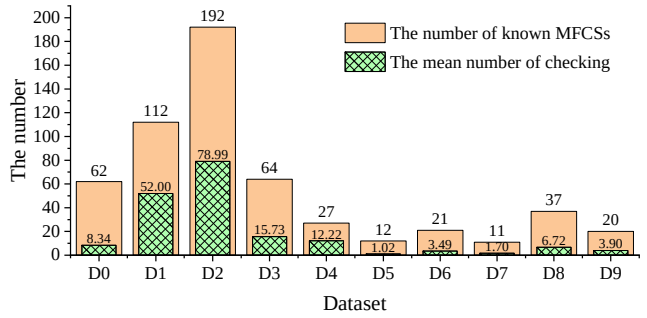
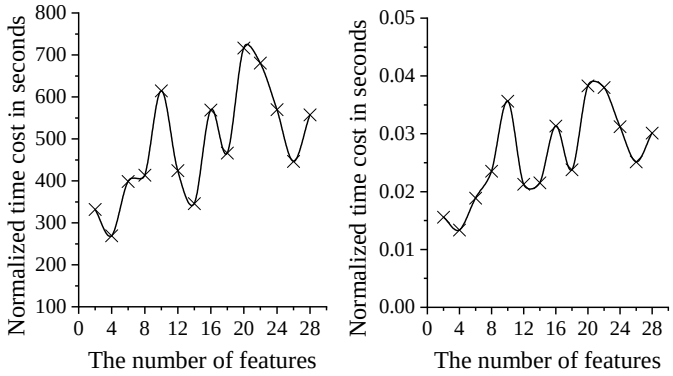


Fig. 8. **The number of known $MFCS$s and the mean number of checking**. It shows the mean number of checking is less than half of the number of known $MFCS$s.



(a) Explaining a model (b) Explaining a prediction

Fig. 9. **The relationship between the overheads and the number of features**. It shows that with the increasing number of features, the normalized time costs increase slowly with considerable oscillations.

To sum up, DFPE requires more overheads to explain more about the models and failure predictions compared to current explanation methods. Because of the benefits of high explainability and the high costs of failure handling, we argue that the overheads are acceptable.

D. Discuss: intelligent disk failure handling

Currently, proactive disk failure handling is to migrate the data and replace the disks predicted to fail. Discarding the disks would result in a huge waste of storage resources when the disks can be fixed or deployed in another application scenario. For example, when a disk is predicted to fail because of its temperature (SMART 194), the disk might be fixed by cleaning the dirt. When a disk is predicted to fail because of its Reallocated Sectors Count (SMART 5), the data in the disk still can serve normally with a relatively high error rate. In this case, the disk can be employed in the application scenario that is not sensitive to tail latency and the storage systems can employ disk scrubbing to find the error and store redundant data to recover the lost. The handling action is also known as degrading the usage. To reduce the waste of storage resources, it is important to employ intelligent disk failure handling which means to handle predicted disk failures intelligently according to failure causes.

DFPE enables intelligent disk failure handling by providing failure causes in the form of $MFCS$s. Here, we introduce a simple intelligent disk failure solution, called SIDF. For every feature, SIDF provides a corresponding action to handle predicted failures caused by the feature. When an $MFCS$ is in an explanation for a failure prediction, one of the handling actions corresponding to the features in the $MFCS$ must be taken to handle the failure. SIDF can choose the action which can allow the best use of the disk rather than discard the disk. When an explanation has several $MFCS$s, the same number of corresponding handling actions must be taken. When one handling action can handle several $MFCS$s, SIDF takes the action with high priority. However, when one of the action is to discard the disk, it is unnecessary to take the other actions.

Take the disk series of dataset D0 as an example. SIDF can provide three handling actions: lowering the temperature, degrading the usage and discarding the disk. Lowering the temperature is the handling action for the feature F8. Degrading the usage of the disk is to handle failures caused by F1, F3, F6, F9, F10, F11 or F12 because abnormality of these features may only result in a high data error rate. Discarding the disk is for F2, F4 and F7 because abnormality of these features may be caused by the damage of some mechanical parts. When a disk is predicted to fail with an explanation $\{\{8, 9, 11\}\}$, SIDF should take one of the handling actions: lowering the temperature or degrading the usage. When the explanation is $\{\{4, 8, 12\}, \{3, 5\}\}$, SIDF would take the action of degrading the usage because the action is for both F12 and F3. When the explanation is $\{\{4\}, \{1, 3, 6\}\}$, SIDF should take both actions: degrading the usage and discarding the disk. However, the disk is supposed to be discarded, so SIDF need not take the action of degrading the usage.

V. Conclusion and Future Work

In this paper, we propose an explanation method DFPE to improve the explainability of complex models on disk failure prediction. The evaluation on the real-work datasets shows that DFPE can explain failure predictions made by a model and infer prediction rules learned by the model. Compared to current explanation methods, DFPE can explain more about failure predictions and models more accurately. Thus DFPE helps to detect and handle bias and overfitting in the models, provides another perspective for measuring feature importances and enables intelligent disk failure handling.

Our future work is to improve DFPE by reducing the overheads, to explore more suitable applications of DFPE, and to test the practicability of intelligent disk failure handling.

Acknowledgment

This work was supported in part by NSFC No.61832020, National Key R&D Program of China NO.2018YFB10033005, Hubei Province Technical Innovation Special Project (2017AAA129), Wuhan Application Basic Research Project (2017010201010103), Fundamental Research Funds for the Central Universities.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, Oct. 2003.

[2] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster," in *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, ser. HotStorage'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 8–8.

[3] G. Wang, L. Zhang, and W. Xu, "What can we learn from four years of data center hardware failures?" in *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN'17, June 2017, pp. 25–36.

[4] 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, Dec. 2005.

[5] T. Pitakrat, A. van Hoorn, and L. Grunske, "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*, ser. ISARCS '13. New York, NY, USA: ACM, 2013, pp. 1–10.

[6] L. P. Queiroz, F. C. M. Rodrigues, J. P. P. Gomes, F. T. Brito, I. C. Chaves, M. R. P. Paula, M. R. Salvador, and J. C. Machado, "A fault detection method for hard disk drives based on mixture of gaussians and nonparametric statistics," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 542–550, April 2017.

[7] B. Zhu, G. Wang, X. Liu, D. Hu, S. Lin, and J. Ma, "Proactive drive failure prediction for large scale storage systems," in *IEEE 29th Symposium on Mass Storage Systems and Technologies*, ser. MSST '13, May 2013, pp. 1–5.

[8] J. Li, R. J. Stones, G. Wang, X. Liu, Z. Li, and M. Xu, "Hard drive failure prediction using decision trees," *Reliability Engineering & System Safety*, vol. 164, pp. 55–65, 2017.

[9] C. Xu, G. Wang, X. Liu, D. Guo, and T.-Y. Liu, "Health status assessment and failure prediction for hard drives with recurrent neural networks," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3502–3508, Nov. 2016.

[10] M. M. Botezatu, I. Giurgiu, J. Bogojeska, and D. Wiesmann, "Predicting disk replacement towards reliable data centers," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 39–48.

[11] J. Xiao, Z. Xiong, S. Wu, Y. Yi, H. Jin, and K. Hu, "Disk failure prediction in data centers via online learning," in *Proceedings of the 47th International Conference on Parallel Processing*, ser. ICPP 2018. New York, NY, USA: ACM, 2018, pp. 35:1–35:10.

[12] F. Mahdisoltani, I. Stefanovici, and B. Schroeder, "Improving storage system reliability with proactive error prediction," in *Proceedings of the 2017 USENIX Conference on Usenix Annual Technical Conference*, ser. USENIX ATC '17. Berkeley, CA, USA: USENIX Association, 2017, pp. 391–402.

[13] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population." in *FAST*, vol. 7, no. 1, 2007, pp. 17–23.

[14] T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. USA: Curran Associates Inc., 2016, pp. 4356–4364.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "'why should i trust you?': Explaining the predictions of any classifier," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 1135–1144.

[16] P. Anantharaman, M. Qiao, and D. Jadav, "Large scale predictive analytics for hard disk remaining useful life estimation," in *2018 IEEE International Congress on Big Data (BigData Congress)*, July 2018, pp. 251–254.

[17] J. Li, R. J. Stones, G. Wang, Z. Li, X. Liu, and K. Xiao, "Being accurate is not enough: New metrics for disk failure prediction," in *2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*, Sep. 2016, pp. 71–80.

[18] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2017.

[19] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & explorable approximations of black box models," *CoRR*, vol. abs/1707.01154, 2017.

[20] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller, "How to explain individual classification decisions," *J. Mach. Learn. Res.*, vol. 11, pp. 1803–1831, Aug. 2010.

[21] M. Robnik-Šikonja and I. Kononenko, "Explaining classifications for individual instances," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 5, pp. 589–600, May 2008.

[22] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001.

[23] A. Ma, R. Traylor, F. Douglis, M. Chamness, G. Lu, D. Sawyer, S. Chandra, and W. Hsu, "Raidshield: Characterizing, monitoring, and proactively protecting against disk failures," *ACM Trans. Storage*, vol. 11, no. 4, pp. 17:1–17:28, Nov. 2015.

[24] G. Louppe, "Understanding random forests: From theory to practice," *arXiv preprint arXiv:1407.7502*, 2014.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[27] Backblaze, "Hard drive data and stats," Dec. 2017, https://www.backblaze.com/b2/hard-drive-test-data.html.

[28] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794.