National College of Ireland

Early Prediction of HDD Failures in the Cloud Using Interpretable AI Models

MSc Research Project
MSc in Cloud Computing

Ashwini Ashok Patil
Student ID: 20218745

School of Computing
National College of Ireland

Supervisor: Jitendra Kumar Sharma

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Ashwini Ashok Patil
Student ID:	20218745
Programme:	MSc in Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Jitendra Kumar Sharma
Submission Due Date:	15/08/2022
Project Title:	Title
Word Count:	6000 aprx
Page Count:	22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Ashwini Ashok Patil
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Early Prediction of HDD Failures in the Cloud Using Interpretable AI Models

Ashwini Ashok Patil
20218745

14/08/2022

Abstract

Hard Disk Drive(HDD) storage devices are essential to the cloud datacenter's dependability. The availability of cloud services can be significantly impacted by HDD failures, which can be expensive for both the cloud provider and the end user financially and in terms of reputation. To reduce the effects of failure, it is crucial to foresee HDD failure beforehand and arrange the necessary corrective actions. In this study, we offer an HDD failure prediction system that uses RUL(Remaining Useful Life) predictions to anticipate HDD failures in advance. To accurately forecast failures and enable simple user understanding of prediction findings, the suggested system uses a composite model composed of bi-directional LSTM(Long Short-Term Memory) and RFC(Random Forest Classification)/OCT(Optimal Classification Tree). The models are trained and evaluated using publicly available Backblaze dataset. The model performance of composite model was evaluated against the baseline RFC and OCT models. From the experiments, we observed the composite model made up of RFC and bi-directional LSTM had an accuracy of 99% against the baseline RFC model accuracy of 94%. This promising result requires further experimental validation.

1 Introduction

Storage devices are one of the most important components of a datacenter, and HDDs are the most commonly used storage device. Pinciroli et al. (2020) Storage device failures account for 28% of data-center failure events, according to Xu et al. (2019). HDDs, like any other piece of hardware, are prone to failure, with an estimated annual failure rate of 15% Huang et al. (2019). With data-centers housing hundreds of thousands of HDDs, it is possible that tens of HDDs will fail every day in the data-center. Failure of HDD has a noticeable impact on data-center performance, ranging from minor write issues to complete service failure Ganguly et al. (2016). Most modern data centers have fault-tolerant mechanisms in place to deal with such failures, but they are insufficient because the time between service disruption and resumption can be significant for complex services.This service outage affects the SLA (Service Level Agreement) and has a significant financial and reputational cost for both the cloud service provider and the end user. Furthermore, the data center would be operating with reduced resources until the HDDs were physically replaced, which could take anywhere from hours to weeks depending on replacement availability.

Researchers have been studying the problem of early prediction of failures in HDDs in order to effectively handle such situations in large data centers. The goal is for the cloud service provider to be able to plan a replacement strategy for the failing disk as well as a migration policy for the data and VMs running on the failing disk to a healthy diskXu et al. (2018). Device manufacturers have integrated SMART metric reporting into device hardware to aid in disk failure prediction.

Here is a summary of major objectives to be achieved by this work: (a) A composite AI model made up of OCT/RFC and Bi-directional LSTM is proposed, (b) A detailed evaluation of performance of composite model against RFC and OCT using the publicly available Backblaze dataset.

The report is organized as follows: section2 focuses on the literature review portion of the work on HDD failure prediction and defines problem statement. Section3 talks about the research methodology used. Section4 talks about overall composite model and predictor system design.Section5 mentions the implementation details.Section6 discusses the experiments and results of this work.Section7 concludes the work done so far and provides future research directions.

2 Related Work

In the section, we provide an overview of major works found in the literature to address the problem of Failure prediction of HDD's.

To monitor the internal attributes or states of individual disk drives, most modern Hard Disk Drives (HDD) report S.M.A.R.T Zeydan and Arslan (2020) metrics. Some manufacturers even provide an impending drive failure prediction feature to warn users of an impending disk failure. These approaches are based on simple SMART attribute thresholding methods Pinheiro et al. (2007); Schroeder and Gibson (2007).However, studies have found that thresholding-based methods have poor prediction performance, and to overcome this, researchers have proposed using a data-driven approaches to disk failure prediction.

In literature, failure prediction based on data-driven approach comes in two formulations: (a) binary classification problem formulation in which the output "1" indicates the presence of failure and "0" indicates the absence of failure, or (b) Remaining Useful Lifetime(RUL) estimation of a disk, which is a regression problem.

In Liu et al. (2020), tackle the prediction of HDD failure as a binary classification problem. This study was notable for taking into account both short- and long-term temporal dependencies of SMART data when predicting failure. The study created a framework for failure prediction called SHARP, which consists of an ensemble of models made up of XGBoost (Gradient Boosting Trees) and GRU (Gated Recurrent Units) and demonstrated that building an ensemble of sufficiently different models can result in better prediction performance. Based on the findings of this study, we construct a Predictor system composed of sequence models such as LSTM and interpretable models such as OCT, to improve both prediction accuracy and interpretability.

Anantharaman et al. (2018) tackle HDD failure prediction as a regression problem and use Random Forests models to predict hard disk RUL. Because RF models do not require sequence data, they are easier to train as compared to sequence-based models. Furthermore, unlike sequence models, RF models can generate predictions on a single in-

stance of SMART data. Because RF models can provide accurate estimates of RUL, they are frequently used as baseline models in the literature to evaluate model performance. In our work, we compare model performance using a classifier variant of the Random Forest model as one of the baseline models.

In Coursey et al. (2021) addresses the HDD failure prediction problem as a regression problem This paper develops a prediction pipeline based on bidirectional LSTM with a look-back period of multiple days that accounts for the long-term temporal dependencies in failure data. The bidirectional LSTM model was tested against the baseline LSTM model, and it outperformed the baseline LSTM model. The authors also looked at the Random Forest baseline and discovered that it performed worse than bi-directional LSTM. The authors reported that the bi-directional LSTM model had the highest reported accuracy of 96.46% for a 15-day look up period and is one of the most advanced models in the literature.Based on the findings of this study, we decided to incorporate the bi-directional LSTM model into our composite AI model.

Amram et al. (2021) is one of the first works in the literature to use newly improved optimal tree models to predict HDD failures. HDD failure prediction, like other major works in this field, is treated as a binary classification task. The authors employ optimal tree models such as Optimal Classification Trees (OCT) and Optimal Survival Trees (OST), On the publicly available Backblaze dataset. On test data, OCT had an accuracy of 86.1 % and OST had an accuracy of 69.2 % on a 30-day prediction window. These results for single tree-based models appear promising. Furthermore, these models are simple to interpret,making them suitable for use in critical applications. Based on the findings of this study, we decided to incorporate the OCT model into our composite AI model for HDD failure prediction system, allowing for easy interpretation of prediction results.

2.1 Problem Formulation

We address the problem of accurately predicting HDD Remaining Useful Lifetime (RUL) using interpretable AI models such as tree-based models in this work (RFC, OCT). This work builds on and expands on the ideas and concepts presented in Zeydan and Arslan (2020),Amram et al. (2021)Coursey et al. (2021). As in Zeydan and Arslan (2020), we formulate the problem of predicting HDD failure as a multi-class classification problem, with failure states identified based on HDD RUL estimates. HDDs can be classified as Critical, Low Ideal, or High Ideal.

The main objectives of this work are as follows: (a) Composite AI model for HDD failure prediction that combines LSTMs for SOTA predictive performance and an OCT/RFC tree for ease of interpretation. According to our understanding, this is the first work that addresses the problem of failure prediction as a multi-class classification problem by combining the strengths of Deep Learning and interpretable AI models, (b) On the publicly available Backblaze dataset, compare the performance of the composite AI model to SOTA models such as RFC and OCT. We believe that the benchmarking results will provide insights into the performance of OCT against ensemble models such as RFC, which has yet to be addressed in any work, (c) Integration of this composite model into a complete HDD failure prediction system via web app which is described in section4.4.

3 Methodology

In this section, we discuss and detail the steps used to gather required data and transform it to format suitable for training AI models. We also discuss about feature selection and scaling applied on the dataset.

3.1 Dataset - Backblaze

Backblaze is an American cloud storage and data backup company, that regularly publishes the SMART metrics of all HDD's located in it's datacenter, as an opensource and freely accessible dataset on it's website. According to it's website, Backblaze hosts around 206,928 drives in it's datacenter as of December 2021. The dataset is updated on a quarterly basis and the same can be easily accessed via it's website.[1]

For the research and implementation of this project, we decided to make use of publicly available dataset opensourced by Backblaze. The reasons for selecting this dataset are as follows: (a) It is opensource and one can use it for research purposes freely without any legal obligations, (b) Prior works like Anantharaman et al. (2018); Shen et al. (2018); Aussel et al. (2017), which this project extends on, makes use of this dataset. This enables us to compare the results from our work with the prior ones, since all of them make use of this dataset as baseline. Moreover, this dataset has become standard for evaluating the performance of HDD failure prediction task as can be seen from the referred prior works above, (c) Since we don't have access to datacenter to gather raw data, the best possible source for the required data was met through this dataset.

The dataset consists of SMART metrics for all the drives captured for a particular day. Each row in the dataset corresponds to a unique drive in the datacenter and contains it's associated metric information for the day. Backblaze collates information from all drives into single csv file, denoting the snapshot of all disks for that particular day. The csv files follow the file naming format YYYY-MM-DD.csv.

In this project, we made use of Backblaze data from the period 2021 Q1 to Q3, amounting to 9 months worth of disk data available for training and evaluating the prediction models.

3.2 SMART Metrics

SMART is a disk health monitoring system that is designed for HDD, SSD and eMMC storage devices. SMART stands for Self-Monitoring, Analysis, and Reporting Technology. It's goal is to measure and report drive health status metrics denoting the reliability of drives, which can then be used to locate and report imminent drive failures Schroeder et al. (2017). Table 1 describes some of the metrics defined in SMART. There are about 255 different SMART metrics and every drive manufacturer has the freedom to select which SMART metrics will be measured and reported by his device.

[1]BackBlaze dataset: `https://www.backblaze.com/b2/hard-drive-test-data.html`

ID	Attribute	Description
07	Seek Error Rate	Rate of magnetic head seek errors
09	Power-On Hours	Total count of hours in drive power on state
240	Transfer Error Rate	Time spent in positioning drive heads
241	Total LBA's Written	Count of LBA's written
242	Total LBA's Read	Count of LBA's read

Table 1: S.M.A.R.T attributes contributors (2022)

Figure 1: Overview of Data pre-processing step

3.3 Data Pre-processing

This is the first step in a machine learning or deep learning project. The goal of this step is to clean the data and transform it into a format suitable for model training step. The output of this step is a cleaned data along with extracted informative features, that can be used to train the models. Figure 1 depicts the major sub-steps involved in the data pre-processing step.

3.3.1 Data Extraction

Before we can start with data pre-processing step, we first need to extract the data from the dataset. In it's current state, the dataset is not directly usable, since the entire data is scattered around multiple files. The dataset contains information about both failed and working disks. Since the goal of our research project is to build a model to predict the failure of HDD, we identify and only extract information on all failed disks for model ST4000DM000 similar to approach used in Coursey et al. (2021), from the entire dataset in following steps: (a) Identify all the failed disks in the dataset, (b) For each failed disk, identify it's failure date and from then onwards extract the failed disk smart metrics for the previous 60 days. The collected data was then concatanted to the dataset in the long format, similar to the approach used by Coursey et al. (2021).

At the end of failed disk extraction step, we have a dataset consisting information of failed disks 60 prior from it's date of failure. This dataset is then used for feature extraction and model training steps.

3.3.2 Data Cleaning

The input to this step is the failed disk data with estimated RUL generated in previous step from the original Backblaze dataset. Since the device manufacturers are free to

report only subset of SMART metrics, few columns in the dataset have valid values and rest of them are all either set to 0 or NaN. Since ML(Machine Learning) or DL(Deep Learning) algorithms are sensitive to missing and NaN values, we have to clean up these values before the data can be used to train the model.

The data cleaning process followed for this project is as follows: (a) Identify and remove all columns/features with all 0 or NaN values, (b) Delete columns/features like model,size, capacity_bytes, failure, date, which do not provide any intrinsic value for model training, (c) In the remaining feature/columns identify any missing values or NaN. For filling up with missing values, we used sklearn implementation of KNN Imputer algorithm. The algorithm estimates the missing value based on the values of it's neighbors. ML/DL model implementations tend to have issues with missing values in the dataset. As a result, we have to make the dataset free of missing or NaN values. The output from this step is dataset free of missing values and unnecessary columns.

3.3.3 Feature Engineering/Selection

The goal of this step is to add new features and identify relevant features in the dataset using multiple metrics. The input to this step is a cleaned dataset from the previous step. In this project, the feature engineering process is done as follows:

- A new feature called RUL(Remaining Useful Life) is added to the dataset. RUL is calculated using disk failure date as 0 and every day prior to it is incremented by 1. For a failed disk, failure day would have RUL of 0 and 60 days prior to failure will have RUL of 60. This approach is also used by Coursey et al. (2021).

- Since our composite model predicts both the disk state as well RUL, we need to add target disk state in the dataset, which is currently missing. Target disk state is estimated from RUL using a similar approach followed in Zeydan and Arslan (2020). Similar to Zeydan and Arslan (2020), we classify the disk in three possible states: Critical[0,20], Low Ideal[21,40], High Ideal[41, 60]. We add a new column called Disk State to our dataset after this step.

- The next step is to identify relevant features. This is done as follows:

  - Identify linear relationships between features and target variables in the dataset. This can be easily measured using Pearson's correlation co-efficient, also known as Pearson's r score. We make use of sklearn implementation of Pearson r-score. An r score has values between -1 and 1, where: -1 (negatively correlated), 0 (unrelated), 1(positively correlated). Pearson's r-score only captures linear relationships between features and target variables.
  - Identify non-linear relationships between features and target variables using decision trees. We made use of sklearn implementation of decision tree algorithm to fit a decision tree model to the dataset. The feature importance weights assigned to different features in the data was extracted.
  - The features having best scores from both Pearson r-score and decision tree feature importance values were selected for bi-directional LSTM model training similar to Zeydan and Arslan (2020).

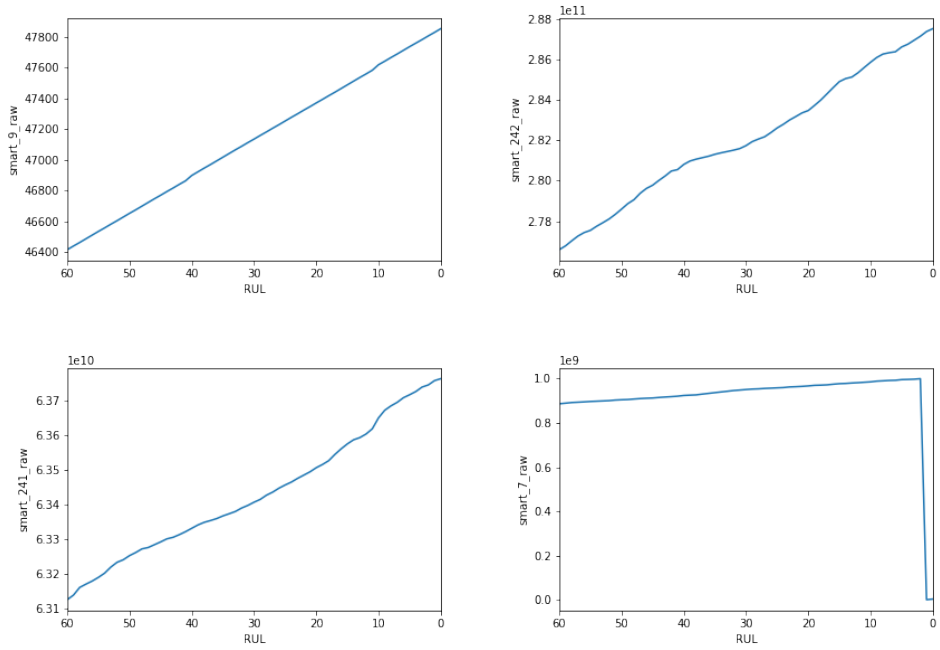


Figure 3: Relationship between selected SMART features and RUL

Figure 2 depicts the results obtained from the feature selection step. Green bar plots indicate selected features, red bar plots indicate ignored features, yellow diamond plots indicate feature importance from decision trees. X-axis denotes smart features, while Y-axis denotes Pearson r-score/ decision tree feature importance values.

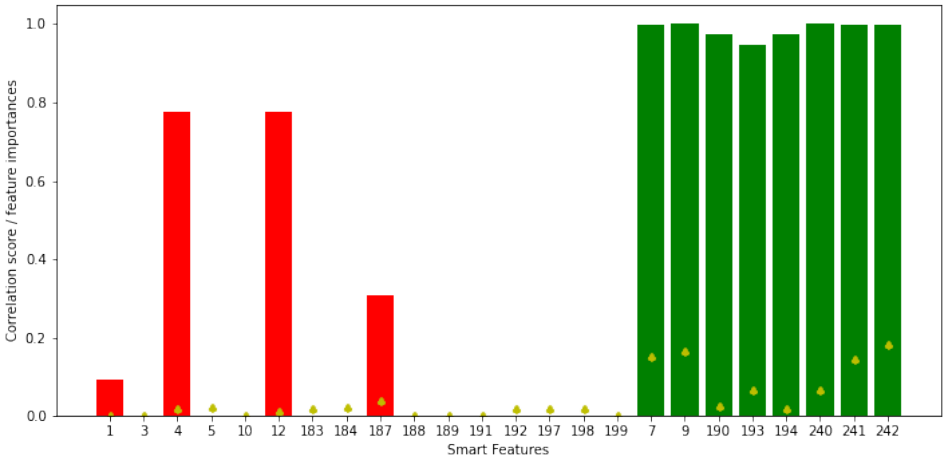


Figure 2: Results from feature selection step

From the plots in Figure 2, we can infer that SMART metrics 7,9,190,193,194,240,241,242 have the best scores and these are used for training the bi-directional LSTM models.Figure 3 depicts the relationship between some selected features and RUL.

– For training OCT and RFC models, all the features in the data are considered.

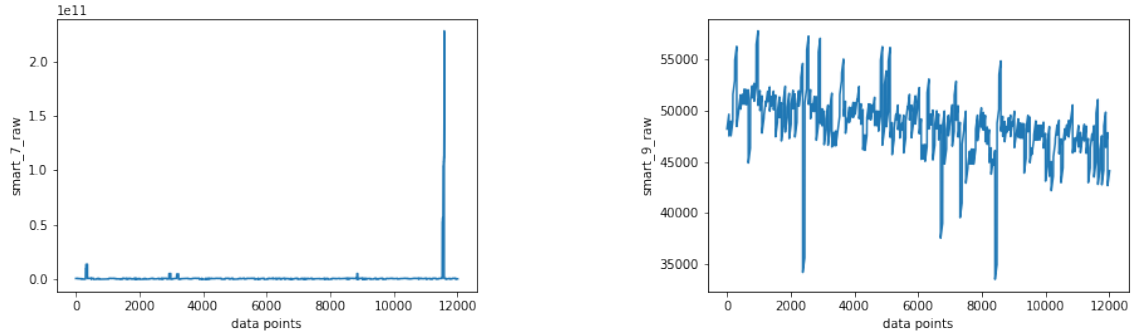


Figure 4: Result of applying scaling entire dataset to smart metrics

This approach was taken taken to ensure that our composite model does not lose out on valuable information, thereby enabling us to train better models.

The output of this step are dataset consisting only of selected features for bi-directional LSTM model training and all features dataset for RFC and OCT model training.

3.3.4 Feature Scaling

Since ML/DL models are sensitive to feature values, the values need to be scaled before the start of training step. The goal of this step is to scale the feature values in the dataset to a mean of 0 and standard deviation of 1. This process is also referred to as centering the data. We make use of standard scaler from sklearn to scale the dataset.

Figure 4 depicts the results of applying scaling to the entire dataset as a whole. From the plots, we can observe that, even after applying scaling, we see some extreme values in the dataset. Regression models like LSTM's are sensitive to such noise in the data and this can affect the learned model accuracy on unseen data. This problem can be solved in two ways: (a) Drop the entries having extreme values after scaling the dataset. The drawback of this approach is that we lose some valuable information, which might hamper the performance of the trained model, (b) Instead of scaling for all disks at once, scale the data individually for each disk. The advantage of this approach is that we do not lose any information from the dataset, thereby enabling us to train better models. This approach was followed by Coursey et al. (2021) for training LSTM models.

In this project, we follow the approach of scaling data for each disk separately. The output from this step is properly scaled dataset that can now be used for training the models.

4 Design Specification

In this section we provide a detailed description of the design of proposed system consisting of composite AI model. Figure 5 depicts the overall architecture of HDD failure predictor system and it's associated components.

4.1 Data Logger

The function of data logger is to capture and aggregate SMART metrics from all the drives located in the datacenter or cloud and store into a database for further processing

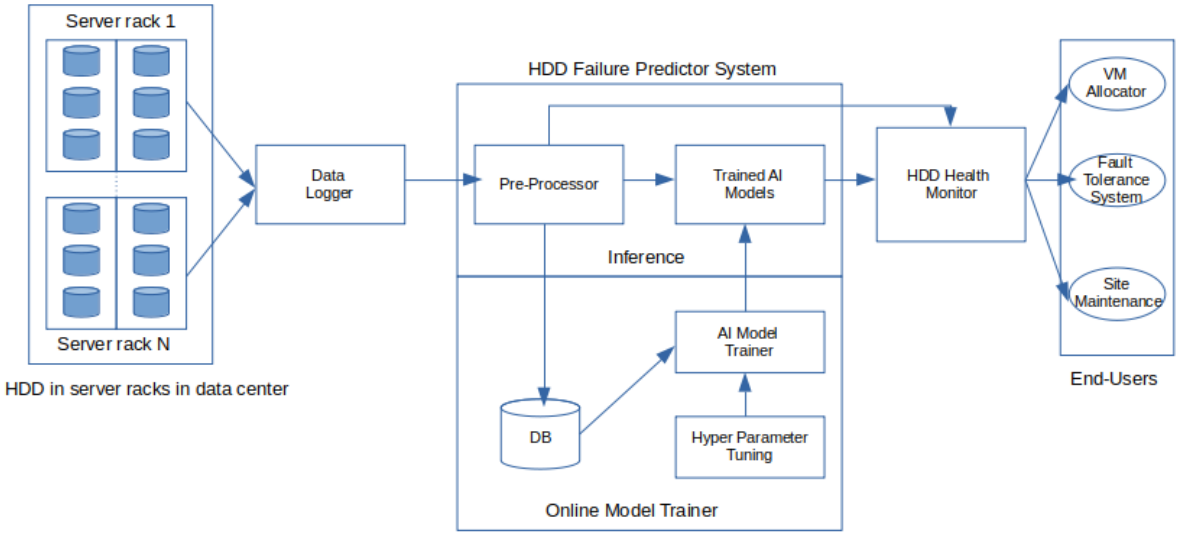


Figure 5: HDD failure prediction system for cloud using composite AI models

by the services like HDD failure prediction system.

In this project, since we don't have access to physical cloud to capture the data, we make use of data published by Backblaze as input to our project. Backblaze would make use of service similar to data logger in their datacenter to capture the SMART metrics. More information on the dataset can be found in section 3.1.

4.2 HDD Failure Predictor System

The function of this component is to predict RUL and Disk state for given input sequence of SMART data. In order to achieve this, the component makes use of composite AI model for prediction estimates. This component is responsible for training, updating the AI models and offering inference service to end-users.

4.2.1 Training Mode

In this mode, the components goal is to train new composite AI models using the data stored in the system database. The intention is to keep models updated on recent data that might contain new trends in the disk behavior, which might hamper the failure prediction performance of the model if model is not updated.

Figure 6 depicts sequence of steps involved in training composite AI models:

- Data Pre-processing: In this step raw data is processed and transformed to format suitable for training the AI models. section 3.3 discusses in detail the pre-processing steps used for training the composite AI models in this project.
- Data Sampling: This step is already done during data extraction step described in section 3.3.1, since we capture last 60 days of data of all failed disks in the dataset. The dataset thus extracted is well-balanced.
- Model Training: In this step, the dataset obtained from the pre-processing step is made use of. This dataset is split into two halves in the ratio 90:10, where the

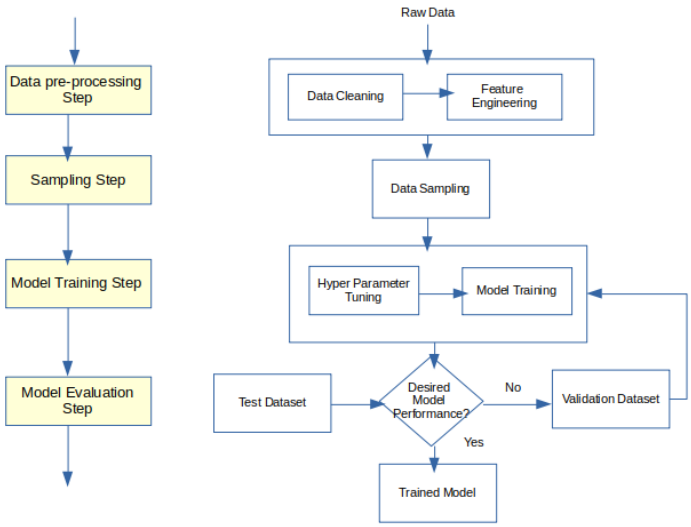


Figure 6: Interpretable HDD failure prediction system for cloud

90% of samples in the data are used for model training and remaining 10% of the samples are used for evaluating the accuracy of the trained model. In this work, This train-test split strategy is used for training tree-based models like RFC and OCT. For training bi-directional LSTM models, We further split the train set in the ratio 80:20, where 80% of earlier train set samples are used for training and the remaining 20% of train samples are used as dev set, which is used for tuning the model hyper parameters. The output from this step are trained tree-models like OCT and RFC along with bi-directional LSTM.

- Model Evaluation: This is the final step in the training mode, where the newly trained models are measured for performance against the existing models and baseline models. If the new models match or exceed the performance criteria, then these models are integrated into the system, thereby replacing earlier models.

In this work, the model training process is currently implemented as offline step. The work to extend this to online mode will taken up as part of future work.

4.2.2 Inference Mode

In this mode, the component offers disk failure prediction service. For given data, it makes use of trained composite AI models for predictions.

Figure 7 depicts the steps involved in the inference process. The steps are as follows: (a) Pre-processing- Here, raw data is transformed to format suitable for making predictions using composite AI model. The detailed processing strategy is already discussed in section3.3, (b) Composite AI model prediction- The prep-processed data is then fed to the composite AI model. The model outputs RUL estimates along with disk states. These are then passed onto the end-users. In addition, the system stores a copy of input data along with it's predictions to database as shown in Figure 2for further use.

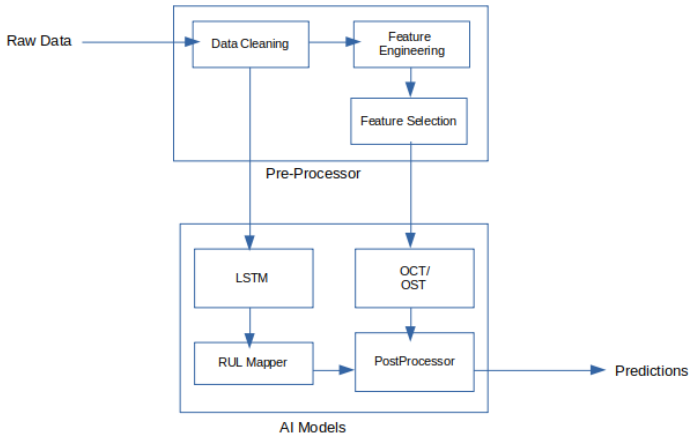


Figure 7: Detailed system design for inference mode

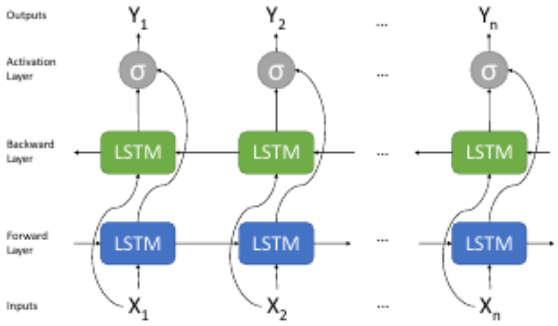


Figure 8: Bi-directional LSTM Coursey et al. (2021)

4.3 Composite model

Composite AI model is the core idea of this work. As discussed in section 2.1, the idea behind composite models is to combine of the predictive power of bi-directional LSTM models on temporal data, along with predictive power and ease of interpret-ability of tree based models like RFC or OCT for failure prediction. We now provide a brief overview of the workings of LSTM, RFC and OCT models.

4.3.1 bi-directional LSTM

For tasks involving temporal or sequence data, Recurrent Neural Networks(RNN) and it's variants like LSTM, GRU and bi-directional have demonstrated SOTA performance and these are the most widely used models in the literature for tasks involving this kind of data Tokgöz and Ünal (2018),Tokgöz and Ünal (2018),Hochreiter and Schmidhuber (1997).

In this work, we make use of bi-directional LSTM which is a variant of RNN. Figure 8 depicts the structure of bi-directional LSTM model.

bi-directional LSTM consists of two LSTM's, one running on forward input sequence and the other running backwards Guo et al. (2019). The idea behind using two LSTM's is to better capture the relationships between the features in sequence or temporal data. Bidirectional LSTM models were first applied to HDD failure prediction proiblem by Coursey et al. (2021). The reasoning behind this is that bi-directional LSTM's are better equipped to capture the relationship between input features and the RUL target variable in the data, since one LSTM is running in forward direction i.e. towards the failure and the other is running backwards i.e. away from the failure. These models were found to be best-performing in the literature for the given task and hence were chosen for this work as well.

4.3.2 OCT

Optimal tree models like OCT, OST and other are designed to address the problem of constructing small trees that deliver SOTA performance. Trincavelli (2021) Optimal tree models are able to construct small trees in a single step by taking advantage of global optimization methods instead of greedy methods. The global optimization methods leverage the advances made in solvers in last 25 years to train smaller trees in a single step. This approach has following advantages: (a) Optimal trees model are small, (b) Impose Global constraints, (c) Generalization.

4.3.3 RFC

Education (2020)RFC are an extension of decision trees. Instead of single decision tree. RFC trains several decision trees by randomly sampling from input data. The algorithm works as follows: For given input data, randomly select the samples, Individual trees are constructed for each selected sampled subset,Each decision tree will generate an output, The results from all trees are put to majority voting scheme. The result that gets major votes is selected as final output. The benefits for using RFC are: (a) reduce the risk of over-fitting (b)Easy to determine feature importances.

4.3.4 Composite Model

The internal design of composite model is depicted in the Figure 7. The working of individual blocks is described below:

- bi-directional LSTM: The best performing bi-directional LSTM model from the train step is integrated into the composite model. This model ingests both spatial and temporal features present in the input data fetched from the pre-processor and outputs RUL estimates.
- OCT: The best performing OCT model from the train step is integrated into the composite model. This model ingests the input data from the pre-processor and outputs the disk state i.e. Critical, Low Ideal, High Ideal. Definition of these states can be found in section 3.3.3.
- RFC: The best performing RFC model from the train step is integrated into the composite model. This model ingests the input data from the pre-processor and outputs the disk state i.e. Critical, Low Ideal, High Ideal.

- RUL mapper: The output from bi-directional LSTM are RUL estimates. These estimates are then mapped to their corresponding disk states using RUL mapper function. This is done to enable easy merging of outputs from both the models into single disk state estimate.

- Post-processor: It's function is to merge the disk state outputs from both the LSTM and tree-based models into one unified disk state estimate.The merging is done as follows: (a) if the disk state estimates are same from both the models, it is added to the final output, (b) if disk state estimates are different, then the final state output is decided based on confidence of disk estimates from tree-based models.if tree-based models have high confidence, their estimates are used a final state estimates otherwise LSTM estimates are used.

The final output from composite AI models are RUL estimate and Disk state for the given input data.

4.4 HDD Health Monitor App

This component is responsible to keep track RUL and disk state for all HDD's located in the datacenter and make necessary statistics available to different stakeholders.

As part of this work, a subset of functionality of this app is implemented, where in it takes input data from user and outputs the RUL estimates and disk state predictions from the model, along with pie-chart visualization of disk state statistics i.e. Critical, Low Ideal and High Ideal count for the given data.

5 Implementation

In this section we provide details about the implementation of composite AI model and the HDD failure prediction system. We also provide short overview of tools and frameworks used for the implementation. The entire work is implemented using Python programming language using native python tools and frameworks.

As discussed in section4, the HDD failure prediction systems consists of multiple steps like data pre-processing, composite model training, model inference and web app to display the prediction results. Backblaze dataset from year 2021 Q1-Q4 was used for model training and evaluation [2]

5.1 Data pre-processing

In this step, the raw data is cleaned, processed and transformed into a format suitable for model training and evaluation. The code for this step can be found in *utils/data_preprocessing.py*. The data pre-processing pipeline is implemented as follows:

- Data extraction: It's function is to identify and extract list of failed disks from the entire Backblaze dataset. $ClassExtractFailedDisk$ implements the data extraction functionality. This class takes as input: location of dataset stored, disk model name, dataset year, no:of months of data present in dataset and max days for each month. The output from this class is list of failed disks for the given model from the dataset, the output list is stored as failed_disk.csv file in the *data/* folder.

[2]Backblaze dataset: `https://www.backblaze.com/b2/hard-drive-test-data.html`

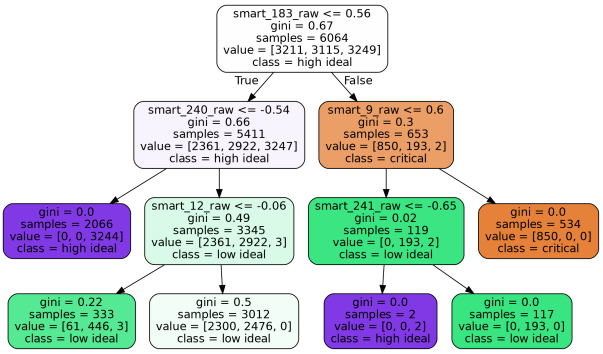


Figure 9: Snapshot of learned estimator from RFC

- Data Sampling: It's function is to extract last 60 days worth of disk data for every failed disk found in the previous stage. *ClassCreateDataset* implements the required functionality as described. The input to this class are:location of dataset stored, year, csv file containing failed disk info, duration of previous disk data to be extracted for each failed disk. The output from this class is sampled dataset consisting of disk data for all failed disks from last 60 days up-to the date of failure.

- Data Pre-processing: It's function is to transform the raw disk data extracted in previous step to format suitable for ML/DL model training. *ClassPreprocessData* implements the required functionality as described in section 3.3. The class takes as input: file path for data extracted in previous step. The output from is this class are csv files containing transformed data in a format suitable for training ML/DL models.

The following python packages were used for implementing the above pipeline: Pandas,Numpy, Scikit-Learn, Matplotlib.

5.2 RFC model

In this step, an RFC model is trained from the dataset obtained from the pre-processing step. The dataset was split into train and test sets in the ratio 90:10. The code to train RFC model can be found in $rfc_model/$. The implementation makes use of RandomForestClassifier implementation from scikit-learn package. The model was trained with following parameters: (a) n_estimators: 100, (b) max_depth: [2,3,4,5]. The best-performing RFC model on test set was stored for future-use. The results will be discussed in the evaluation section 6. Figure 9 depicts one of the learned decision tree estimator from trained RFC.

5.3 OCT model

In this step, an OCT model is trained from the dataset obtained from the pre-processing step. The dataset was split into train and test sets in the ratio 90:10. The code to train OCT model can be found in $oct_model/$. For OCT model training:

- Interpretable AI [3] OCT model implementation from this package was initially considered. Due to package installation complexity and license issues as it is proprietary

[3]Interpretable AI: `https://docs.interpretable.ai/stable/`

software, this was not preferred

- Python OCT implementationTang (2021): Alternatively an open-sourced python implementation for OCT was found. The OCT implementation from this package was used for OCT model training. This implementation makes use of gurobi solver. An academic license for the same can be easily obtained from gurobi solver website.

The model was trained with following parameters: (a) alpha: 0, (b) max_depth: [2,3,4,5]. The best-performing OCT model on test set was stored for future-use. The results will be discussed in the evaluation section 6.

5.4 bi-directional LSTM model

In this step, bi-directional LSTM model was trained from the dataset obtained from the pre-processing step. The dataset was split into train,dev and test sets in the ratio 72:18:10. The code to train the model can be found in $bi_lstm_model/$. The code train LSTM is implemented as a metaflow pipeline:

- Data Loader: $ClassBackblazeDataset$ implements the functionality to read data from csv files into pytorch data format.

- Vanilla LSTM model: $ClassHDDRULPredictorSystem$ implements the vanilla LSTM model for HDD failure prediction. It makes use of pytorch Lightning framework for model implementation. It takes dataloader object as input and outputs a trained model and results of model on test set.

- bi-directional LSTM model: $ClassHDDRULPredictorSystemBiLstm$ implements the bi-directional LSTM model for HDD failure prediction. It makes use of pytorch Lightning framework for model implementation. It takes dataloader object as input and outputs a trained model and results of model on test set.

The following frameworks and tools were used for implementation: (a)Pytorch Lightning- It's user-friendly DL framework built on top of pytorch, (b) Metaflow- It is python framework that streamlines the process of creating ML/DL projects from design to deployment stage. In this work, metaflow was primarily used to streamline model training and evaluation.

Figure 10 denotes train/val loss curves for one of training runs. These curves can be used to debug if model implementation is correct and model is learning.

The model was trained with following parameters: (a) input_size: 5 (i.e. input feature size), (b) hidden_size: 32 (i.e no:of cells in each hidden layer), (c) num_layers: 1 (i.e. no:of hidden layers), (d) timesteps: [5, 10, 15, 30] (i.e. look back period), (e) loss: MAE (Mean Absolute Error) - More robust to outliers

The best-performing bi-directional LSTM model on test set was stored for future-use. The results will be discussed in the evaluation section 6.

5.5 Composite Model

In step, we implement a composite model, that integrates the best performing RFC/OCT model and bi-directional LSTM model into a unified entity. For given input dataset, the function of this model is to predict both RUL estimates and Disk state. The code for composite model implementation can be found in $compoiste_model/$ folder.

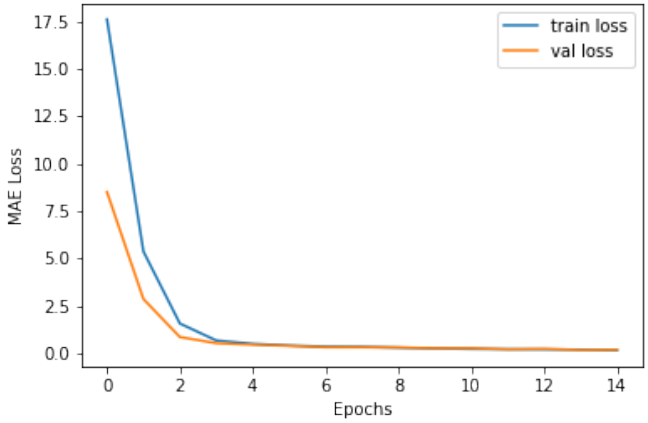


Figure 10: Train/Val loss curves for Bi-LSTM

5.6 HDD Health Monitor App

This step implements the functionality described in section 4.4 as web application using python flask framework. The app integrates the composite_model built in previous step, and uses it to make predictions for the given input data. The app also visualizes the disk state predictions as a pie chart. Figure11 depicts a snapshot of the web app for given input.

6 Evaluation

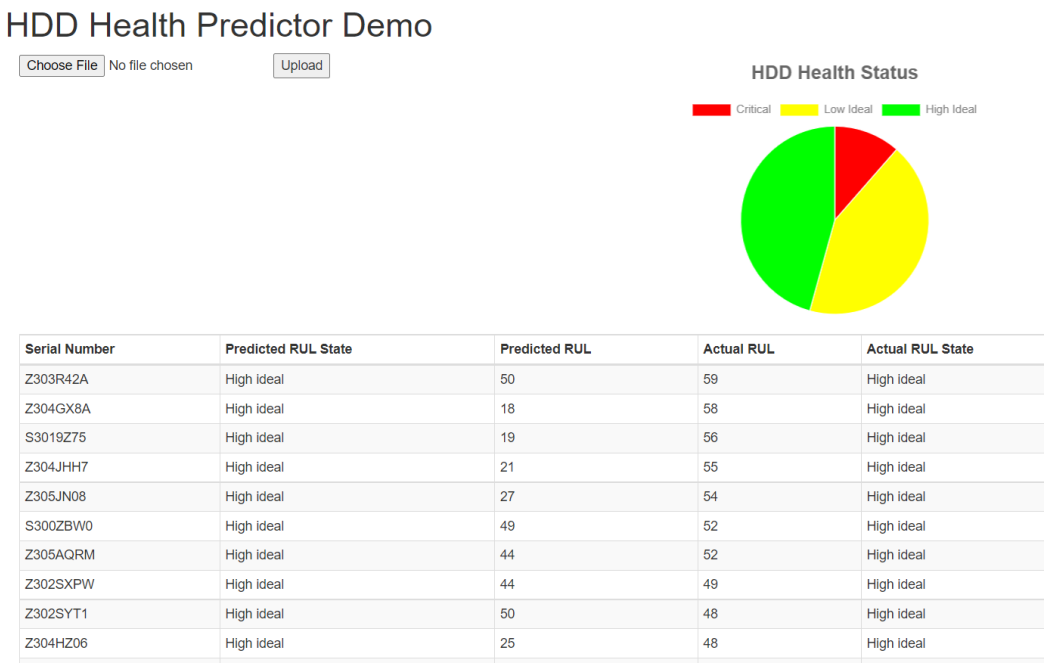
In this section we discuss the train and test results for models trained. We will also compare and contrast the performance of composite model against tree-based models like OCT and RFC which are considered as baseline models due to their SOTA performance Zeydan and Arslan (2020), Amram et al. (2021).

6.0.1 Classification Metrics

Accuracy: It is defined as the fraction of correct predictions(TP + TN) over all the predictions(TP + TN + FP + FN) made by the model.
Precision: It is defined as the fraction of predicted true positives(TP) over total number of positive predictions(TP + FP).
Recall: It is the fraction of true positives(TP) labels predicted overall the total actual positive(TP + FN) labels in the dataset.
F1-score: It is computed via harmonic mean between Precision and Recall scores. F1-score lies in the range $[0, 1]$. higher the F1-score the better, Where TP,FP,TN,FN stands for True Positive, False Positive, True Negative, False Negative respectively.

6.0.2 Regression Metrics

Mean Absolute Error(MAE): Average difference between the expected and predicted values.
R2 score: Measures the goodness of fit. If the regression model perfectly captures the



Serial Number	Predicted RUL State	Predicted RUL	Actual RUL	Actual RUL State
Z303R42A	High ideal	50	59	High ideal
Z304GX8A	High ideal	18	58	High ideal
S3019Z75	High ideal	19	56	High ideal
Z304JHH7	High ideal	21	55	High ideal
Z305JN08	High ideal	27	54	High ideal
S300ZBW0	High ideal	49	52	High ideal
Z305AQRM	High ideal	44	52	High ideal
Z302SXPW	High ideal	44	49	High ideal
Z302SYT1	High ideal	50	48	High ideal
Z304HZ06	High ideal	25	48	High ideal

Figure 11: Web App- Prediction results

Depth	Train Accuracy (%)	Test Accuracy(%)
2	99.69	99.71
3	99.93	1.0
4	99.93	1.0
5	99.94	1.0

Table 2: RFC train and test results

variance in the target variable, R2 score is closer to 1.0 (ideal). If the regression model fails to capture the variance in the target variable, R2 score is closer to 0.0 (bad-fit).

6.1 RFC model results

For our work, we trained RFC models for different depths [2,3,4,5] and the model with highest accuracy score on test set was stored for integration with composite model. Table 2 denotes the performance of RFC models on test set for different depths. We can observe that RFC models with differing depths have similar performance on the test set. For our work, we chose to integrate RFC model with depth of 2 into the composite model.

6.2 OCT model results

For our work, we trained OCT models for alpha: 0, depths [2,3,4,5], and the model with highest accuracy score on test set was stored for integration with composite model. Table 3 denotes the performance of OCT models on test set for different depths. From the table

Depth	Train Accuracy (%)	Test Accuracy(%)
2	77.24	77.86
3	67.13	66.93
4	67.13	66.93
5	67.13	66.93

Table 3: OCT train and test results

Timesteps	MAE loss	R2-score
5	0.1014	0.9994
10	0.1003	0.9994
15	0.0855	0.9995
30	0.1212	0.9971

Table 4: Bi-directional LSTM test results

we infer that the model accuracy decreases slightly for depths 3,4 and 5 as compared to depth of 2.

The model was also trained for different values of alpha [0,0.05, 0.5]. The OCT open-source implementation had crash issues while training for alpha's 0.05 and 0.5 for depth greater than 3, so no results are reported for the same.

6.3 bi-directional LSTM model results

For our work, we trained bi-directional LSTM models for multiple look-back periods [5,10, 15, 30]. We compare the performance different models using their MAE loss and r2-score. From the table 4, bi-directional LSTM model with look back of 15 timesteps has the lowest MAE loss and r2-score closer to 1.0 on the test set among all the models. Hence, this bi-directional model is chosen for integration with the composite model.

6.4 Composite Model results

The best performing tree based models RFC/OCT and bi-directional LSTM model were then integrated into the composite model. The performance of this composite model was then compared against baseline models from Zeydan and Arslan (2020), Amram et al. (2021).

6.5 Experiment 1

Integrate the best performing model of RFC and bi-directional LSTM as a composite model and evaluate it's performance on the test set. Table5 presents the results of composite model performance against baseline RFC model. Clearly the composite model appears to be the better model among the two.

Model	Accuracy (%)	F1-score	Precision	Recall
RFC	94	0.94	0.94	0.94
Composite Model	99.71	0.9970	0.9971	0.9971

Table 5: Composite model vs baseline RFC model results

Model	Accuracy (%)	F1-score	Precision	Recall
OCT	86.1			
Composite Model	77.87	0.74374	0.9970	0.9971

Table 6: Composite model vs baseline OCT model results

6.6 Experiment 2

Integrate the best performing model of OCT and bi-directional LSTM as a composite model and evaluate it's performance on the test set. Table6 presents the results of composite model performance against baseline OCT model. Clearly the stand-alone OCT model appears to be the better model among the two.

6.7 Discussion

In experiment 1, we compared the performance of our composite model consisting of RFC and bi-directional model against the baseline RFC model. On the surface, it appears that our composite model might be better of the two. But, it needs some more experiments on the composite model to confirm the improved performance. More

In experiment2, we compared the performance of our composite model consisting of OCT and bi-directional model against the baseline OCT model.Our composite model has lower accuracy than the baseline OCT model. The difference in performance can be due to multiple factors like: (1) Implementation issues in open source OCT implementation. The suggested approach is to train our OCT model using the Interpretable AI package that is written by original authors of the algorithm, (2) The current OCT implementation does not provide confidence values for predictions, hence the disk state predictions obtained from LSTM are not take into account. If LSTM disk state predictions are taken into account, there should further improvement in model performance.

6.7.1 Improvements

The following improvements are suggested: (a) Re-classify the dataset and retrain bi-directional LSTM model training as discussed in Zeydan and Arslan (2020). The current RUL predicted estimates are still not accurate as the baseline model estimates. Also, experiment by increasing more hidden layers from the existing 1 layer to improve the overall LSTM model performance, (b) Setup the interpretable AI package from original authors and retrain the OCT model for better model performance comparison with baseline models.

7 Conclusion and Future Work

In this work, we tackled the problem of HDD failure prediction as multi-class classification problem, where in the HDD can be in one of the following three states: Critical, Low Ideal and High Ideal based on RUL estimatesZeydan and Arslan (2020). The objective of this work was to make use of combination of tree-based models like RFC/OCT and along with black box Deep Learning models like bi-directional LSTM to achieve both the ease of interpretability of results and still achieve SOTA performance. The RFC, OCT and bi-directional LSTM models were trained using publicly available backblaze dataset. The best performing models on test set were then integrated into the composite model. Two composite models were built, one with RFC and bi-directional LSTM and the other with OCT and bi-directional LSTM. The performance of these two composite models were compared against the baseline models RFC and OCT respectively. The composite model consisting of OCT and bi-directional LSTM had an accuracy of 78% against the baseline OCT model accuracy of 86%. The OCT model code used for training had some implementation issues, which needs further investigation. The OCT model has to be re-implemented using interpretable AI package and this will be taken up as part of future work. The composite model consisting of RFC and bi-directional LSTM had an accuracy of 99% against the baseline RFC model accuracy of 94%. Though the composite model shows promising result, it needs further experiments to come to a conclusion.

7.1 Future Work

The following topics will be taken up as part of future work: (a) Retrain OCT model using stable implementation of OCT trees from interpretable AI package from the original authors, (b) Benchmark the performance OCT trees against baseline RFC trees, (c) Perform more experiments on the composite model consisting of RFC and bi-directional LSTM model to validate the improved model performance against the baseline RFC model, (d) Integrate the online model training feature to the web app from the existing offline procedure.

References

Amram, M., Dunn, J., Toledano, J. J. and Zhuo, Y. D. (2021). Interpretable predictive maintenance for hard drives, *Machine Learning with Applications* **5**: 100042.

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

Aussel, N., Jaulin, S., Gandon, G., Petetin, Y., Fazli, E. and Chabridon, S. (2017). Predictive models of hard drive failures based on operational data, *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, pp. 619–625. CORE2021 Rank: C.

contributors, W. (2022). S.m.a.r.t, `https://en.wikipedia.org/w/index.php?title=S.M.A.R.T.&oldid=1078605550`.

Coursey, A., Nath, G., Prabhu, S. and Sengupta, S. (2021). Remaining useful life estimation of hard disk drives using bidirectional lstm networks, *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 4832–4841. CORE2021 Rank: B.

Education, I. C. (2020). Random forest, `https://www.ibm.com/cloud/learn/random-forest`.

Ganguly, S., Consul, A., Khan, A., Bussone, B., Richards, J. and Miguel, A. (2016). A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters, *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, IEEE, pp. 105–116.

Guo, G., Wang, C., Chen, J., Ge, P. and Chen, W. (2019). Who is answering whom? finding "reply-to" relations in group chats with deep bidirectional lstm networks, *Cluster Computing* **22**(1): 2089–2100.

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

Huang, S., Liang, S., Fu, S., Shi, W., Tiwari, D. and Chen, H.-b. (2019). Characterizing disk health degradation and proactively protecting against disk failures for reliable storage systems, *2019 IEEE International Conference on Autonomic Computing (ICAC)*, IEEE, pp. 157–166. CORE2021 Rank: B.

Liu, W., Xue, Y. and Liu, P. (2020). Sharp: Smart hdd anomaly risk prediction, *AI Ops Competition*, Springer, pp. 74–84.

Pinciroli, R., Yang, L., Alter, J. and Smirni, E. (2020). The life and death of ssds and hdds: Similarities, differences, and prediction models, *arXiv preprint arXiv:2012.12373* .

Pinheiro, E., Weber, W.-D. and Barroso, L. A. (2007). Failure trends in a large disk drive population.

Schroeder, B. and Gibson, G. A. (2007). Understanding disk failure rates: What does an mttf of 1,000,000 hours mean to you?, *ACM Transactions on Storage (TOS)* **3**(3): 8–es.

Schroeder, B., Merchant, A. and Lagisetty, R. (2017). Reliability of nand-based ssds: What field studies tell us, *Proceedings of the IEEE* **105**(9): 1751–1769. JCR Impact Factor 2021: 10.961.

Shen, J., Wan, J., Lim, S.-J. and Yu, L. (2018). Random-forest-based failure prediction for hard disk drives, *International Journal of Distributed Sensor Networks* **14**(11): 1550147718806480. JCR Impact Factor 2021: 2.03.

Tang, L. B. (2021). Optimal$_c lassification_t rees$, .

Tokgöz, A. and Ünal, G. (2018). A rnn based time series approach for forecasting turkish electricity load, *2018 26th Signal Processing and Communications Applications Conference (SIU)*, IEEE, pp. 1–4.

Trincavelli, M. (2021). Optimal decision trees - mlearning.ai - medium, `https://medium.com/mlearning-ai/optimal-decision-trees-dbd16dfca427`.

Xu, E., Zheng, M., Qin, F., Xu, Y. and Wu, J. (2019). Lessons and actions: What we learned from 10k SSD-Related storage system failures, *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, USENIX Association, Renton, WA, pp. 961–976. CORE2021 Rank: A.
URL: *https://www.usenix.org/conference/atc19/presentation/xu*

Xu, Y., Sui, K., Yao, R., Zhang, H., Lin, Q., Dang, Y., Li, P., Jiang, K., Zhang, W., Lou, J.-G. et al. (2018). Improving service availability of cloud systems by predicting disk error, *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pp. 481–494. CORE2021 Rank: A.
URL: *https://www.usenix.org/conference/atc18/presentation/xu-yong*

Zeydan, E. and Arslan, S. S. (2020). Cloud 2 hdd: large-scale hdd data analysis on cloud for cloud datacenters, *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, IEEE, pp. 243–249. CORE2021 Rank: National: France.