Cloud Failure Prediction based on Traditional Machine Learning and Deep Learning

Tengku Nazmi Tengku Asmawi
Universiti Teknologi MARA
Azlan Ismail (azlanismail@uitm.edu.my)
Universiti Teknologi MARA
Jun Shen
University of Wollongong

Research Article

Keywords: Cloud Computing, Job and Task Failure, Failure Prediction, Deep Learning, Machine Learning

Posted Date: May 18th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1657073/v1

RESEARCH

Cloud Failure Prediction based on Traditional Machine Learning and Deep Learning

Tengku Nazmi Tengku Asmawi[2], Azlan Ismail[1,2*] and Jun Shen[3]

*Correspondence:
azlanismail@uitm.edu.my

[1]Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Kompleks Al-Khawarizmi, Universiti Teknologi MARA (UiTM), 40450 Shah Alam, Selangor, Malaysia

Full list of author information is available at the end of the article

Abstract

Cloud failure is one of the critical issues since it can cost millions to the cloud service providers in addition to the loss of productivity being suffered by the industrial users using different cloud-based applications and services. As the cloud system grows larger, the process of failure prediction is still a challenge for both practitioners and academic researchers. In this study, we tackle the challenge of predicting job and task failure using the benchmarking Google Cluster Traces Dataset. We first propose a conceptual model to prepare, construct and evaluate five traditional machine learning algorithms and three variants of the latest deep learning algorithms for predicting job and task failures. We then perform a series of experiments on two datasets; (1) Dataset A for the job failure prediction models and (2) Dataset B for the task failure prediction models. In the case of job failure prediction experiments, we observe that Extreme Gradient Boosting is the best performance with 94.35% accuracy with the F-Score score of 0.9310, 91.92% sensitivity, and 96.07% specificity. For Extreme Gradient Boosting, the disk space request and CPU request are the most important features in determining the outcome of the job prediction. In the case of task failure prediction, we observe that Decision Tree and Random Forest have achieved the highest 89.75% accuracy and 0.9145 for F-Score with 98% sensitivity and 78% specificity. The priority of the task is the most important feature for determining the task prediction outcome for both, the Decision Tree and Random Forest.

Keywords: Cloud Computing; Job and Task Failure; Failure Prediction; Deep Learning; Machine Learning

1 Introduction

Cloud computing is at the vanguard of a global digital transformation [1]. It allows a business provides an extra layer of security in term of information security and allows them to raise the level of efficiency of operation to a new level. According to Business Fortune Insight, the North America market has approximately spent a whopping $78.28 billion on cloud services. The market for cloud computing is expected to keep expanding from $219B in 2020 to $791.48B in 2028 [2].

However, there are many types of cloud failure, and each failure can result in degradation of Quality of Services (QoS), availability, and reliability that can ultimately lead to economic loss for both cloud consumers and providers [3]. Failure prediction in the cloud environment can be useful to improve its reliability since any failure

and interruption of services can greatly affect cloud users all over the world. Failure prediction in general is a very important aspect of predictive maintenance due to its ability to prevent the failure occurrence and minimize the maintenance costs.

There have been multiple studies on the cloud failure prediction as we presented in Table 8 (listed at end of the paper) that focuses on the job and task failure prediction and Table 9 that summarizes other scopes of failure prediction. As shown in these tables, there are various scopes of failure prediction addressed in the literature, such as job and task failure, node failure, VM failure, etc. In this work, we are specifically concerned with the job and task failure prediction using Google Cluster Traces. As shown in Table 8, most of the related works [4, 5, 6, 7, 8, 9, 10, 11, 12] focused on the traditional machine learning approach, whereas only a few works [13, 4, 14, 11, 12] that ever applied deep learning approach and one study [5] that considered statistical machine learning approach as well. Among these works, there are a few that addressed traditional machine learning and deep learning approaches simultaneously, namely, [4, 11, 12]. Besides that, we also reviewed the utilization of Google Cluster Traces resulting in Table 10 which has shown non-failure related prediction scope.

In relation to these works, our work focus on the comprehensive comparison between traditional machine learning and deep learning approaches using Google Cluster Traces. This direction is driven by the limited studies that addressed both approaches at the same time. In comparison to [4, 11, 12], we also tackle job and task failure prediction at the same time and implement more algorithms for the performance evaluation purpose. This evaluation is significant since it assists the community to gain more insight into each model and to select the best model for predicting job and task failure.

Therefore, in this paper, we build and train five traditional machine learning models and three different variants of deep learning models. The traditional machine learning models focused in this study are Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, and XGBoost. Meanwhile, the deep learning models refer to Single Layer Long Short Term Memory (LSTM), Bi-Layer LSTM, and Ti-Layer LSTM. We aim to analyze and interpret the performance of trained machine learning models and deep learning models in predicting job and task termination status. Thus, we are able to compare the performance of different models in predicting the failure status of cloud jobs and tasks based on the sample data of the classical Google Cluster Traces.

The contributions of this work are threefold:

1 A review of cloud failure prediction works with and without the use of Google Cluster Traces and non-failure prediction works that specifically used Google Cluster Traces dataset.

2 A conceptual model comprises of data handling, modeling and building, and analysis as well as interpretation activities.

3 The performance results and model interpretation of comparing the traditional machine learning over deep learning models.

The rest of this paper is organized as follows. Section 2 explains the used dataset and the fundamental knowledge of failures in cloud. Section 3 presents the conceptual model for the experimentation. Section 4 provides the evaluation results and findings. Section 5 summarizes the related works. Section 6 concludes the paper with future works.

2 Background

In this section, the dataset used in this study are described, followed by fundamental knowledge of failures in the cloud.

2.1 Google Cluster Traces

2.1.1 Overview

Up to date, Google has released three public traces datasets for researcher and academic usage. The first one was released in 2007 containing 7-hour workload details. The dataset only contains the basic information, which includes the time, job id, task id, the category of job, and number of cores and memory. Both the cores and the memory count have been normalized.

The second dataset contains traces of 29 days' workload from about 12,500 machines in the month of May 2011. The data consist of 672,074 jobs and around 26 million tasks that have been submitted by the user [15]. Unlike the previous dataset, this one includes more information about each task's resource utilization as well as information about the task itself, such as scheduling class and task priority.

The third dataset is the most current, documenting the use of cloud resources from eight separate clusters, with one cluster containing roughly 12,000 computers in May 2019, and it was released in early 2020 [16]. In comparison to the 2011 dataset, this dataset focuses on resource requests and utilization and contains no information about end-users. The 2019 dataset has three additions: CPU utilization information histograms for each 5 minute period, information regarding shared resources reservation by a job, and job-parent information for master/worker relationships such as MapReduce's jobs.

2.1.2 2011 Dataset

For the 2011 dataset, the machine details are described in two tables which are *machines events* and *machine attributes*. Every unique machine in both of these tables are identified by machine ID, a unique 64-bit identifier. The *machine events* table consists of a timestamp, machine ID, event type, platform ID, and the capacity of each machine. This table records every event related to a machine such as adding and removing machine in the cluster. Each record has its own timestamp indicating when the event occurs. There is also information on the machine platform representing the micro-architecture and chip-set version of the machine, and the normalized value of the machine CPU and memory capacity. In the case of *machine attributes* table, it contains the details of machine property such as kernel version, machine operation clock speed, and the external IP address that is tied to the machine.

There are four tables provided to describe about the job and task information which are *job event* table, *task event* table, *task constraint* table and *resource usage* table.

The *job event* and *task event* table are used to describe the life-cycle of the job or task. Each job is identified by a unique 64-bit identifier while each unique task is identified by the combination of job ID and task index. Every event from submission to termination is recorded in these tables. The type of event is identified by their event type column where value zero indicates the job or task submitted by the user, value one indicates the task has been scheduled to run on a machine, values two to six indicate the task has been terminated while value seven and eight indicating the task details or requirements have been updated.

Furthermore, there is also a column indicating whether a task or job has been synthesized as a replacement for a missing record. Alongside all the columns mentioned above, there is a column for scheduling class in both tables. The scheduling class is indicated by a single integer value from zero to three where zero indicates a non-production task while three indicates a latency-sensitive task. Lastly, there is one column for a username and two more columns for job names. These columns have been anonymized by combining data from several internal name fields.

In the case of *task event* table, there are six more columns in addition to all the mentioned earlier. One of the columns is machine ID which remarks on which machine the task is run. The second column is the task priority which is valued from zero as the least priority to eleven as the most important task. Next, there are three columns detailing the normalized amount of every resource requested which are CPU resources, memory resources, and disk space resources. Lastly, there is a column attributing the constraint of running on a different machine. If there is value in the column, remarking the task must be executed on a different machine than the machine that is currently running a different task from the same job.

The *task constraint* table discloses the constraints associated with each task, which may be zero or one or more. Task constraints prevent a task from running on a certain machine. Each record in the task constraint table represents exactly one task event record. Lastly, there is *resource usage* table. This table discloses the amount of computing resources such as CPU, memory, and disk space that has been utilized by every task. This usage is recorded for each five-minute or 300 seconds measurement period.

2.2 Failure in Cloud

This section explains the categories of failure and fault tolerance in the cloud environment.

2.2.1 Categories of Cloud Failure

Cloud computing, like any other computing system, is susceptible to failure. Cloud computing system fails when it fails to perform its predefined function due to hardware failures or unexpected software failures. More complex the computing system is, the higher the chance of the system failing is. There are two classifications of failures, namely, architecture-based and occurrence-based failures [17]. These failure categories are summarised in Table 1.

The architecture-based failure comprises of two types of failures, defined as service and resource failure. The service failure usually occurs due to software failures such

as unplanned reboots and cyber-attacks and scheduling failures such as the service time timeout. For this paper, we focus on scheduling failure. Meanwhile, resource failure is caused by hardware failures such as power outage, system breakdown, memory problems, and complex circuit design. The occurrence-based failure consists of two types of failures, specifically, correlated and independent failure. The correlated failure is a failure occurring due to another domain failure. For instance, cloud resources are unavailable due to a power outage that affects the cloud infrastructure. Meanwhile, the independent failure is caused by external factors such as human error and computer overheating.

Table 1 Classification of Failure And Their Cause [17]

Type of Failure	Classification	Cause of Failure
Service Failure	Architecture Based	Software Failure Scheduling Failure
Resource Failure		Hardware Failure
Correlated Failure	Occurrence Based	Based On Two Temporal Or Spatial Correlation Of Two Failure
Independent Failure		Denser System Packing Human Error Heat Issue

2.2.2 Fault Tolerance in Cloud Computing

Despite the advancement in technology of cloud computing, the performance of the cloud is still hampered by its vulnerabilities to failure. Therefore, fault tolerance is one of the fundamentals requirements of cloud computing. There are two major tolerance approaches that are currently being implemented in the cloud computing [18]. The work by Shah et al. [19] has studied and summarized the related research of fault tolerance in cloud computing from the year 2013 until 2020.

The first approach is called reactive fault tolerance [20, 21]. The reactive fault tolerance approach reduces the effect of failure on the execution of an application. When a failure occurs in cloud environment, the reactive fault tolerance approach takes effect. There are several techniques used in the reactive fault tolerance approach. One of them is task replication [22]. This technique is used to duplicate tasks across multiple resources. This replication increases the likelihood that at least one task will be completed correctly. The second technique is task re-submission [23]. When a task fails, it will be re-run either with the same node or to a different resource.

The second approach is called proactive fault tolerance. This approach is implemented with the principle of preventing failure from occurring altogether. For this approach, the condition of the physical system is constantly monitored and there is a need to predict the occurrence of the failure of the system. If the probability of fault occurring is high, Cloud Service Provider (CSP) will takes pre-emptive measures such as removing hardware from the service cluster or performing corrective measures on the software.

The failure prediction can be built using the information gathered from previous cloud failures. Machine learning is an excellent tool for predicting software and hardware failure in cloud infrastructures. Failure prediction is considered a proactive fault tolerance approach if it is implemented in cloud infrastructure [24].

Existing researches have implemented the proactive tolerance approach such as predicting the failing hardware in the cloud farm [25] and predicting memory failure in computer system [26]. Another example is the work by [10] where a machine learning algorithm is constructed to predict the task failure in the cloud system. The prediction model enables the system to efficiently manage the resources available in the cloud system, ensuring that minimum failure happens potentially disrupting the availability of the cloud resources.

3 Method

This section provides a brief introduction to the conceptual model of this study and presents the machine learning techniques, which have been used to construct the failure prediction models.

3.1 Overview

Figure 1 shows the conceptual model which can be divided into three phases, namely, data handling, approaches, and analysis & interpretation. Data handling is a process of extracting the data from the data source followed by preparing two datasets (i.e. datasets A and B) for model training and testing. Table 2 shows the sample of Dataset A whilst Table 3 shows the sample of Dataset B. Our approaches would cover the process of developing and training machine learning and deep learning algorithms to predict failure in the cloud system by classifying the termination status of the job or task.

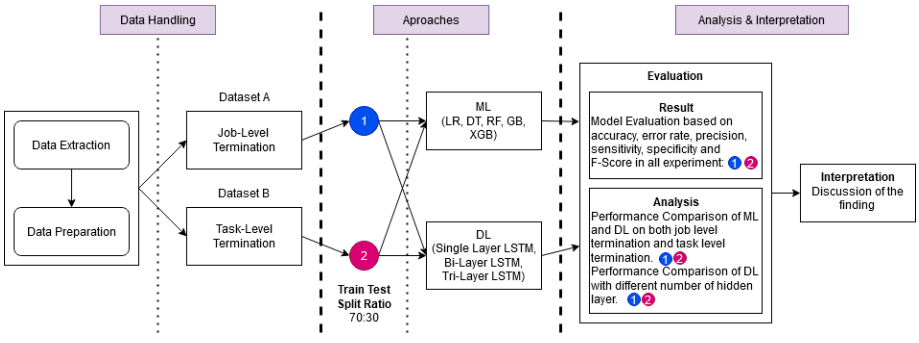

Figure 1 Conceptual Model of Cloud Failure Prediction

We proposed two approaches for this study. Approach 1 aims to classify job-level termination while approach 2 aims to classify task-level termination. The classification of both approaches is based on five traditional machine learning models and three deep learning models. The last phase, analysis & interpretation includes the performance evaluation of both approaches to conclude the best algorithm to predict failure using the Google Cluster Traces dataset.

The conceptual model was implemented and run by using Google Cloud Platform (GCP) [27]. GCP is a suite of public cloud computing services offered by Google. The platform includes a range of host services for computer storage and application development which run on hardware owned by Google. For this study, we configured a virtual machine with 4 CPU cores and 16GB of memory in addition to an NVIDIA Tesla T4 GPU.

Table 2 Dataset A

Event Type	Scheduling Class	CPU Request	Memory Request	Disk Space Request
SUCCESS	1	0.06250	0.006218	0.000045
SUCCESS	1	0.06250	0.103400	0.000038
SUCCESS	1	0.03125	0.009323	0.000154
SUCCESS	1	0.01250	0.028630	0.000077
SUCCESS	1	0.03125	0.011250	0.000011

Table 3 Dataset B

Event Type	Scheduling Class	Priority Priority	CPU Request	Memory Request	Disk Space Request
SUCCESS	0	2	0.02499	0.07959	0.000386
FAILURE	0	9	0.0625	0.006218	0.00003815
FAILURE	2	0	0.01562	0.01553	0.0002155
SUCCESS	1	9	0.04688	0.0636	0.00003815
SUCCESS	0	2	0.02499	0.07959	0.000386

3.2 Data Handling

In this section, we elaborate on two main tasks for data handling, namely, data extraction and preparation as presented in Figure 1.

3.2.1 Data Extraction

The sources of data were retrieved from Google Cloud Storage (GCS). The information about the dataset is available at [28] for the 2011 Google Cluster Traces Datasets. To automate the downloading process, a function was built (not within scope of this paper) to access all the data. We downloaded the 2011 version with a size of approximately around 400 GB. *urllib* library was used to access the data stored in the GCS.

Algorithm 1: Data Extraction

Data: Zipped File
Result: Extracted File
1 $path_S$ = source directory
2 $path_D$ = destination directory
3 **for** $i \; in \sigma$ **do**
4 $path_S = path_S + i$
5 $path_D = path_D + i$
6 Open the source file at $path_S$
7 Read data the source file
8 Store the data into a new file at $path_D$

Algorithm 1 details the process of unzipping all the compressed files and extracting the data. At first, we assign one variable for the source directory and one variable for the destination directory (lines 1 and 2). Then, a looping takes place to generate the

complete path (lines 4 and 5), followed by opening and reading the data from the source file (lines 6 and 7), the sigma (σ) symbol represents the number of partitions for each table. Following that, the read data is stored in a new file in the destination directory in Comma Separated Value (CSV) file format (line 8).

3.2.2 Data Preparation

This task involves several sub-tasks which are *data cleaning*, *data integration*, *data reduction*, *data transformation* and *class balancing*. We utilized Dask library instead of Pandas in this stage since it can parallelize the processing of a large amount of data. Technically, Dask library provides similar capability as Pandas since Dask is built on top of Pandas. It can also use static storage if dynamic storage is insufficient to handle the data. Algorithm 2 shows the general flow process of the data preparation.

Algorithm 2: Data Preparation Process

Data: Job Event Table, Task Event Table
Result: Dataset A, Dataset B
1 **if** *Missing Data Available* **then**
2 Remove rows with missing data
3 Find maximum resource request and priority for each job
4 Join ⋈ Job Event Table with Task Event Table
5 Construct Heatmap from correlation analysis
6 Remove uncorrelated column(s)
7 Reduce data using equation1 and 2
8 Group the termination status using equation 3
9 Perform SMOTE
10 Export dataset A and B

For this task, we have utilized two tables from Google cluster traces, which are job event table and task event table to produce two new datasets, namely, a job level termination dataset (i.e. dataset A) derived from combining the processed job event table and task event table, and a task level termination dataset (i.e. dataset B) derived from the processed task event table. The sample of data created for Dataset A is shown in table 2, whereas the sample of data produced for Dataset B is shown in table 3.

Data Cleaning

One of many objectives for the data cleaning is to handle the missing data in the task event table. We have resolved the missing data issue by removing those records with the use of *dropna* function in Pandas since the source data is a big amount of data. We then constructed a boxplot to observe the data distribution for the continuous data, as well as a bar and pie chart to observe the categorical data. It is important to note that we observed some outliers in the resource request columns of the task event table. However, these outliers have not been eliminated because they can contribute to the identification of the termination status for the job and task failure prediction.

Data Integration

Because the data are stored in multiple tables, data integration is performed to merge all of them into a single dataset for the modeling purpose. Hence, we integrate the job event table and task event table into a single table to produce a

job termination dataset (i.e. Dataset A as in Figure 1). Before the integration process begins, data is aggregated to determine the maximum amount of each resource requested for each job, as well as the highest priority for each work. The data integration process is carried out using *merge* function from Dask library. Dataset B is not included in the data integration process because it only contains data from the processed task event table (i.e. as introduced in Section 2.1.2 .

Data Reduction

This task involves two objectives. The first one is to reduce the number of columns, which is supported by the correlation analysis. The second objective is to reduce the number of rows by deciding the subset of the dataset. In the case of correlation analysis, we first change the value of columns with string datatype into numerical datatype. This is done by using Pandas's *astype* function. After that, we construct a Heatmap to identify the correlation between all the respective columns and to determine which columns to be removed from the dataset. In general, those columns that have weak to no correlation to the outcome of a job or task are removed. After removal, the only columns remained in the job termination dataset are scheduling requests and resource requests. Meanwhile, the task termination dataset's remaining columns are scheduling class, priority, and resource request.

To reduce the number of rows, we only selected data from the first fourteen (14) days as a sample dataset. We also removed the rows where any job or task had not yet completed its job/task life-cycle. Both of these tasks have been conducted by filtering the value inside two columns, which are timestamp and event type. Equation 1 depicts the timestamp range used for this study, where *alpha* (α) symbol representing zero and *beta* (β) symbol representing 604800600000 ms (i.e. 14 days). Equation 2 depicts the range of event types chosen for this study, where the *gamma* (γ) symbol represent value 1 and *delta* (δ) symbol representing the value 7.

$$Timestamp > \alpha \,\&\, Timestamp < \beta \tag{1}$$

$$EventType > \gamma \,\&\, EventType < \delta \tag{2}$$

Data Transformation

In this task, the focus is on re-categorizing the termination status in both datasets, A and B. Two classes were defined to update the termination status, namely, *success* and *failure*. For the *success* class, only jobs and tasks that finish normally belong to this class. The *failure* class is given to the jobs and tasks where their termination status is other than *finished*. Equation 3 shows the rules for deriving the identified

classes. The data with event type 4 is the record that terminates the job or task normally. For the event type other than 4, the job or task does not finish normally.

$$TerminationStatus \begin{cases} success & EventType = 4 \\ failure & EventType! = 4 \end{cases} \tag{3}$$

Class Balancing

This task is needed due to the number of records of the *failure* and *success* class being imbalanced. There are several techniques that can be used for balancing it. In this study, we have used Synthetic Minority Oversampling Technique (SMOTE) [29]. In general, SMOTE falicitates selecting examples that are closed in the feature space, drawing a line between the examples in the feature space, and drawing a new sample at a point along that line. The SMOTE process will increase the amount of data for minority class, and it will also aid the model construction and training, especially for deep learning models in the later stage.

3.3 Our Approaches

During this phase, two types of machine learning algorithms were implemented, categorized as the traditional machine learning and deep learning algorithms. Both datasets, A and B were divided into two parts, one for training and one for validating or testing the model with the split ratio of 7:3. The details about these algorithms are explained below.

3.3.1 Traditional Machine Learning Algorithms

We have implemented three types of algorithms for addressing the classification problem, which are regression, tree, and ensemble respectively. In the case of regression, Logistic Regression [30] is chosen since it is the most investigated regression algorithm in machine learning. For the tree, we have selected Decision Tree [31] since it is the primary tree machine learning algorithm for the classification problem. Lastly, for the ensemble category, we have chosen Random Forest [32], Gradient Boosting [33], and Extreme Gradient Boosting or XGBoost [34]. These machine learning algorithms were implemented using scikit-learn [35] library, except for the XGBoost which was implemented using XGBoost [34] library. All models were built as a default model with a small change on the default arguments. However, for the Logistic Regression model, the number of maximum iterations has been changed because the default value is insufficient for converging all solvers. Solver is an optimization algorithm to calculate the loss of the Logistic Regression.

3.3.2 Deep Learning Algorithms

The last three machine learning algorithms will be three different variants of the LSTM [36] based algorithms. They are differentiated by the number of layers. The three variants of the deep learning model are Single Layer LSTM algorithm, Bi-Layer LSTM algorithm (two hidden layers), and Tri-Layer LSTM (three hidden layers) algorithm respectively. Lastly, we add a dense layer to ensure our algorithm

only produce a single value for a prediction. The epoch is set to 100 to ensure we gain the best model possible. To reduce the training times and prevent over-fitting of the model, the training is automatically stopped if there is no improvement of the validation loss value after 10 epochs.

3.4 Metrics & Interpretation

The analysis comprises the activity of validating the quality of the built models to identify the best one. Meanwhile, the interpretation is needed to gain more insight into the built models. Hence, in this section, we explain the evaluation metrics to support the analysis activity and the importance of the features to support the interpretation activity.

3.4.1 Evaluation Metrics

To measure the quality of a prediction model, it is important to specify the evaluation metrics. The goal is to design an efficient classifier to predict failure accurately and cover as many failure events as possible while generating very few false positives. The performance of training and testing have been compared by evaluating the accuracy, error rate, precision, sensitivity, specificity, and F-Score of the model. These measures can be obtained from the confusion matrix True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The details are discussed in Section 4.

Accuracy & Error Rate

Accuracy measures the overall performance on how often the classifier would correctly predict the correct termination status for a job or task in cloud services, while error rate measures how often the prediction model wrongly classified the outcome. The accuracy measure and error rate measure are calculated using the equation 4 and 5 respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$ErrorRate = 1 - Accuracy \tag{5}$$

Sensitivity & Specificity

Sensitivity measures the percentage of failed task correctly identified while specificity measure the percentage of non-faulty task correctly identified. The sensitivity and specificity are calculated using equation 6 and 7 as shown below.

$$Sensitivity = \frac{TP}{TP + FN} \tag{6}$$

$$Specificity = \frac{TN}{TN + FP} \tag{7}$$

Precision

Precision is defined as the ratio of correctly predicted failures compared to the number of all recognized failures. The precision score is calculated using equation 8.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

F-Score

F-Score values are computed from the precision score and recall score as shown in equation 9. The higher the value of the F-Score, the better the overall model performance.

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{9}$$

3.4.2 Feature Importance

The feature importance refers to the score that measures the importance of each feature in a predictive model. A high score means the feature will have high priority in determining the outcome of the prediction. Thus, the feature importance score helps in the data understanding, model improvement, and model interpretability. In this study, the feature importance scores are calculated using *Dalex* library [37].

4 Analysis and Results

This section explains the analysis of the datasets and the performance results of the predictive models.

4.1 Result of Exploratory Data Analysis

In this section, we explain our findings on the two tables, which are the job event and task event table. The data in these tables had been partially cleansed before they were made public.

4.1.1 Job Event Table

The job event table contains eight columns and 2,012,242 rows. From eight columns, there are three hash string columns and five integer columns. From all of the rows, there are only 672,074 unique job id. Because every event such as job submission, job scheduling, and job termination are recorded, a single job may have at least three occurrences in the dataset. A total of 19 records of job events were synthesized. The data is synthesized because the transition data is not recorded or missing from the traces record. Figure 2 shows the distribution of scheduling classes in the job event table while Figure 3 shows the job termination status categorized by their scheduling class.

There are four scheduling classes whereas scheduling class 0 indicates non-production jobs such as non-business critical analysis and scheduling class 3 represents latency-sensitive jobs such as serving revenue-generating user requests. Based

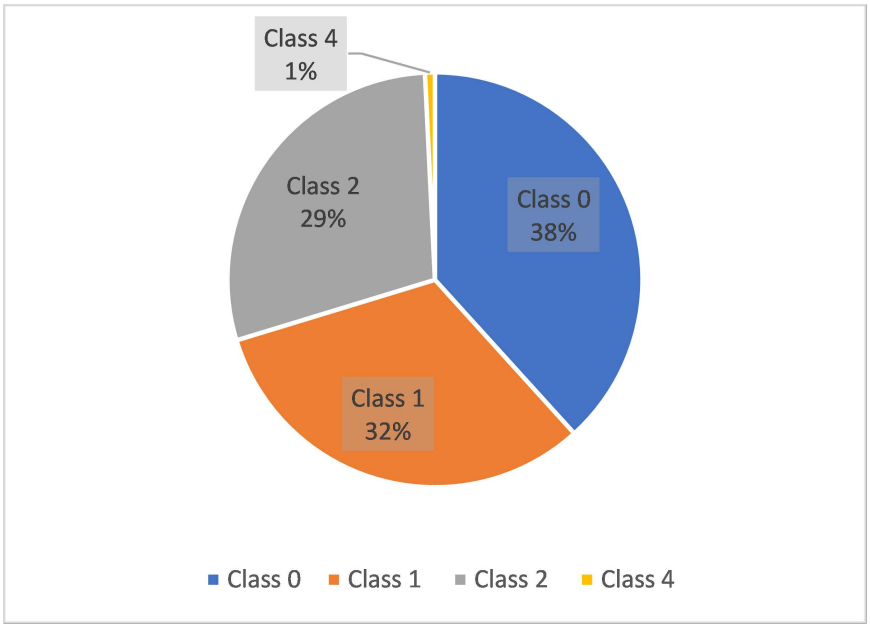

Figure 2 Distribution of Job Scheduling Class

on Figure 2, almost half of the jobs in the dataset are class 0. There are around 5,000 jobs designated as class 3. There are 215,000 jobs in class 1 and 194,000 jobs in class 2. From Figure 3, about 385,582 jobs finished normally, and 274,341 jobs were killed due to interruption from the user, or their dependent job died. About 10,000 jobs were failed where the job was terminated early due to task failures. Lastly, about 22 jobs were evicted. The reasons for the jobs to be evicted might be because of a higher priority task or job, the scheduler over-committed, the actual demands exceeded the machine capacity, the machines which the jobs were running became unusable, or because the disks holding the task's data related to the job were lost.

Figure 4 shows the heatmap of correlation analysis of the job event table. Based on the figure, we can observe that the job event type is highly correlated with the missing info column. We also can observe that other columns have weak to no correlation with the event type column. The user name and job name columns are not considered for correlation analysis as the data in the columns only represent the anonymized values.

4.1.2 Task Event Table

The task event table contains 13 columns and 144,648,288 rows. In the table, there are eight integer columns, three float columns, and one string column. Every unique task is identified by the combination of job id and task index. In total, there are 25,424,731 unique tasks in the table. Similar job event table, a single task may contain at least three occurrences in the dataset because every event such as task submission, task scheduling, and task termination are recorded. Out of 25,424,731 rows, 18,375 rows of data were synthesized, which was around 7% of the total

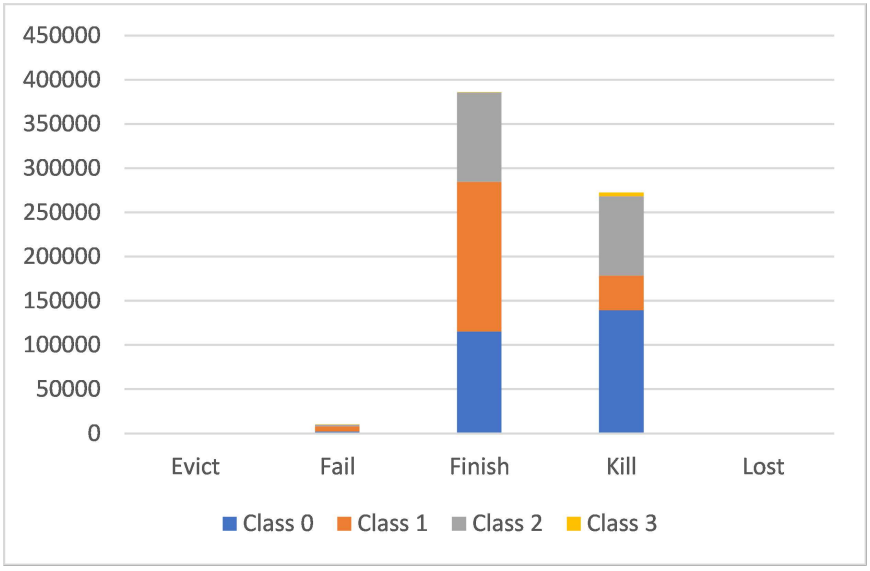

Figure 3 Job Termination Status Categorized by Scheduling Class

records. Figure 5 shows the distribution of task priority in the dataset while figure 6 shows the termination status of tasks categorized by their priority.

From Figure 5, we can see that more than half of the task priority in the dataset is level 0. According to the schema, the priority in the dataset ranges from zero to eleven, where zero is the least priority task while eleven is the highest priority task. Unlike job, task priority plays a more significant role in determining the resources access for each task. From Figure 6, we can see that the ratio of failed tasks to finished tasks is 3 to 4. This means that from the total of failed tasks and finished tasks, three out of four tasks did not finish properly. This ratio is abnormal since a cloud service provider usually aims for 99.9% service availability. Hence, we can assume that there were a series of outages on the site while monitoring the traces.

The distribution of resource requests is depicted in Figure 7. According to the graph, the average value for CPU requested per task is around 0.15% to 6% of the total CPU resources in a machine. The quantity of memory required is often in the range of 1.5% to 3%, while the disk space is in the range of 0.01% to 0.04%. The values of all resources have been normalized between 0 to 1.

Figure 8 shows correlation heatmap of the task event table. From the figure, we can see that the event type has no strong correlation with any other columns. Other than that, we can see that there is a positive correlation between resource usage and priority. Scheduling class does not seem to affect the amount of resources requested and task status. Although it is a weak correlation, the missing info column is adversely correlated with the resource request column.

4.2 Results of Model Analysis and Interpretation

The jobs or tasks that do not finish successfully are identified as failure, otherwise, they are classified as success. Both datasets A and B were divided into two groups

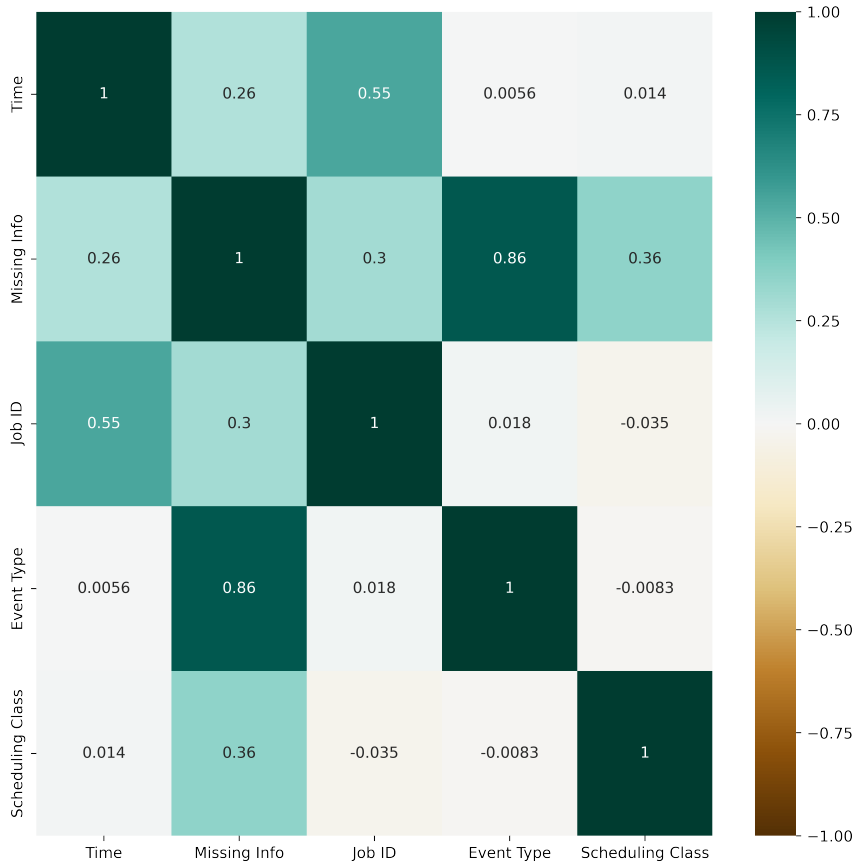

Figure 4 Job Event Table Correlation

with a ratio of 70 to 30 where 70% of the dataset was used for training the model and the remaining 30% was used for testing the model. The performances of the models are explained in the following subsections.

4.2.1 Job-Level Failure Prediction

Predictor importance assists to understand the relative importance of each predictor in estimating the models. Figure 9 shows the predictors that matter the most. Predictor importance does not relate to the model accuracy. It only relates to the importance of each predictor in making a prediction. The predictor importance is calculated by predicting the training data.

Figure 9 shows that for all machine learning models with the exception of Gradient Boosting and Extreme Gradient Boosting, in overall, scheduling class and CPU request are the most significant features in determining the outcome of the termination status of the job. These two features score around 30 to 35% for their importance in the cloud job termination status. For LSTM, it seems that disk space request is insignificant in determining the termination status of cloud job. Other than that, the Logistic Regression algorithm used memory request as the determining feature in order to evaluate a job termination status.

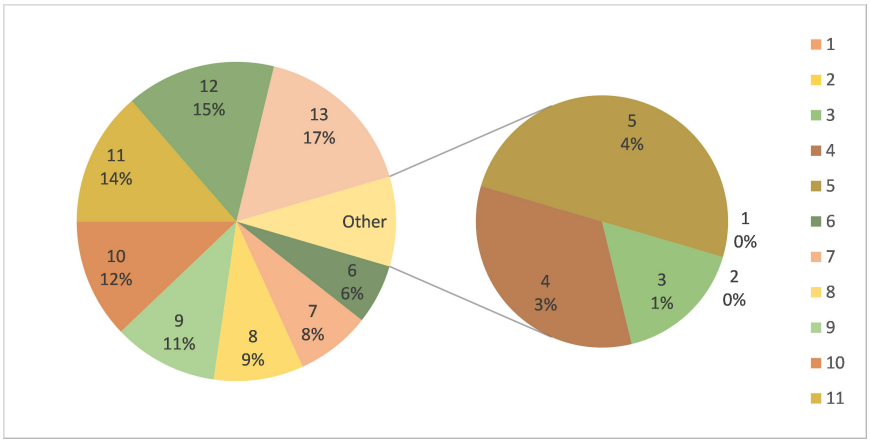

Figure 5 Distribution Of Task Priority

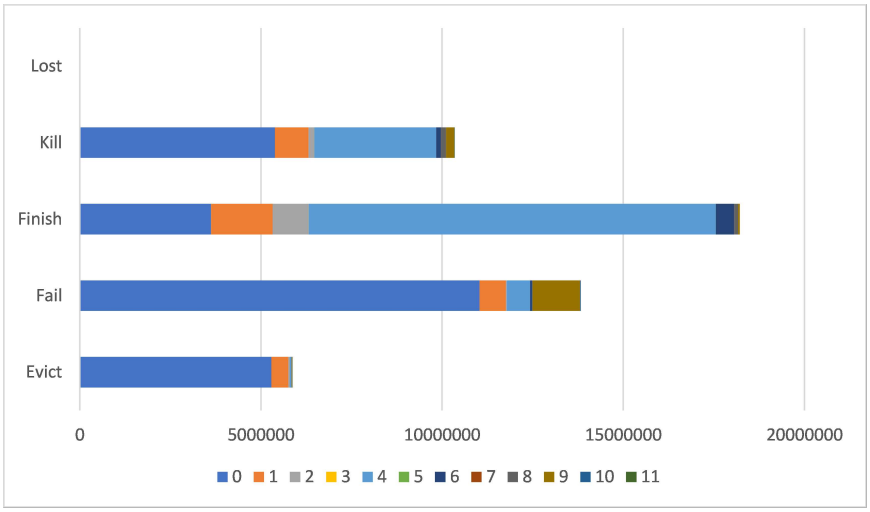

Figure 6 Task Termination Status Categorized by Task Priority

For the job level failure prediction, the amount of data used was around one million rows of data. We can observe the performance of all LSTM models are lower than the performance of traditional machine learning models with the exception of Logistic Regression model. Interestingly, this might be happening because the deep learning models do not understand the complexities of the dataset. Table 4 shows the performance of every model using the training dataset while Table 5 shows the performance of all models trained in classifying the termination status of the test dataset.

From Table 4 and 5, we can observe that Extreme Gradient Boosting or XGBoost has the best accuracy at 93.25% end 93.10%. The F-Score score of 0.9325 and 0.9310 further demonstrated that XGBoost is the best model out of all models generated in this experiment. XGBoost also demonstrated the highest precision, correctly identifying 94.31% of job termination statuses. Furthermore, XGBoost's sensitivity

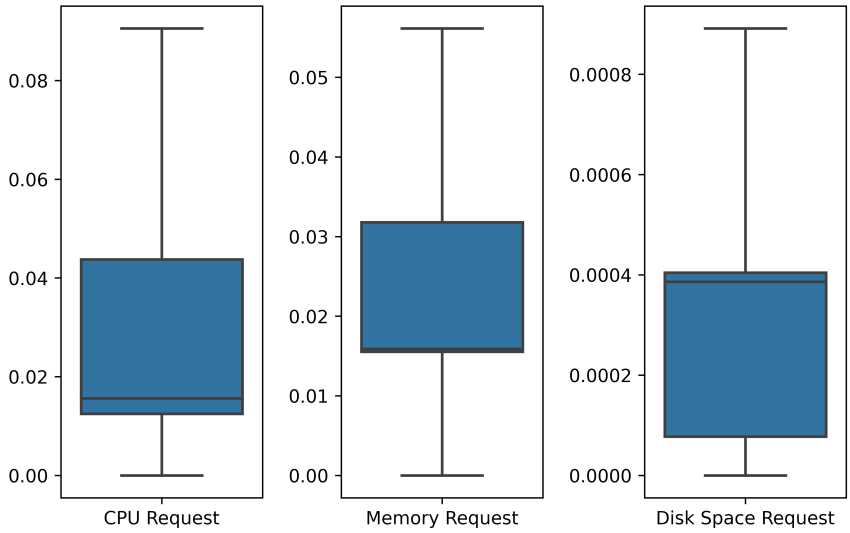

Figure 7 Resource Request Distribution

Table 4 Training Performance Evaluation

Model	Accuracy(%)	Error Rate(%)	Precision(%)	Sensitivity(%)	Specificity(%)	F-Score
Logistic Regression	63.11	34.66	41.44	55.65	66.30	0.4751
Decision Tree	93.43	8.08	92.19	91.56	94.70	0.9187
Random Forest	93.27	7.63	91.47	91.80	94.26	0.9163
Gradient Boosting	90.72	9.43	91.40	86.35	93.96	0.8098
XGBoost	**94.49**	**6.27**	**94.46**	**92.08**	**96.19**	**0.9325**
Single Layer LSTM	88.83	12.27	85.75	86.42	90.44	0.8609
Bi-Layer LSTM	89.91	11.42	86.42	88.29	90.97	0.8734
Ti-Layer LSTM	85.10	14.77	81.75	81.34	87.65	0.8155

and specificity scores are at 91.92% and 96.07% respectively, indicating that it can successfully predict job termination status.

The second highest accuracy is recorded by Decision Tree in overall algorithms. The result for XGBoost and Decision Tree only show slight difference in terms of performance of accuracy, precision, and specificity during the testing and training phase. The Decision Tree achieved 93.27% accuracy during training and 93.23% accuracy during testing, with a precision of 92.19% during training and 91.95% during testing. Similar to XGBoost, Decision Tree also obtained a high score for sensitivity and specificity during the training and testing phase which demonstrates that the model is able to correctly classify the job termination status.

Gradient Boosting has the third-highest accuracy at 90.72% during the testing phase compared to Random Forest at 89.65%. But during the training phase, Gradient Boosting shows lower accuracy at 90.72% as compared to Random Forest at 93.27%. Gradient Boosting has shown better performance during the testing phase where its F-Score is 0.8874 during the testing phase while the F-Score in the training phase is 0.8098. Thus, Random Forest is the fourth performing model to predict job failure prediction although the accuracy is slightly lower during testing which is 89.65%.

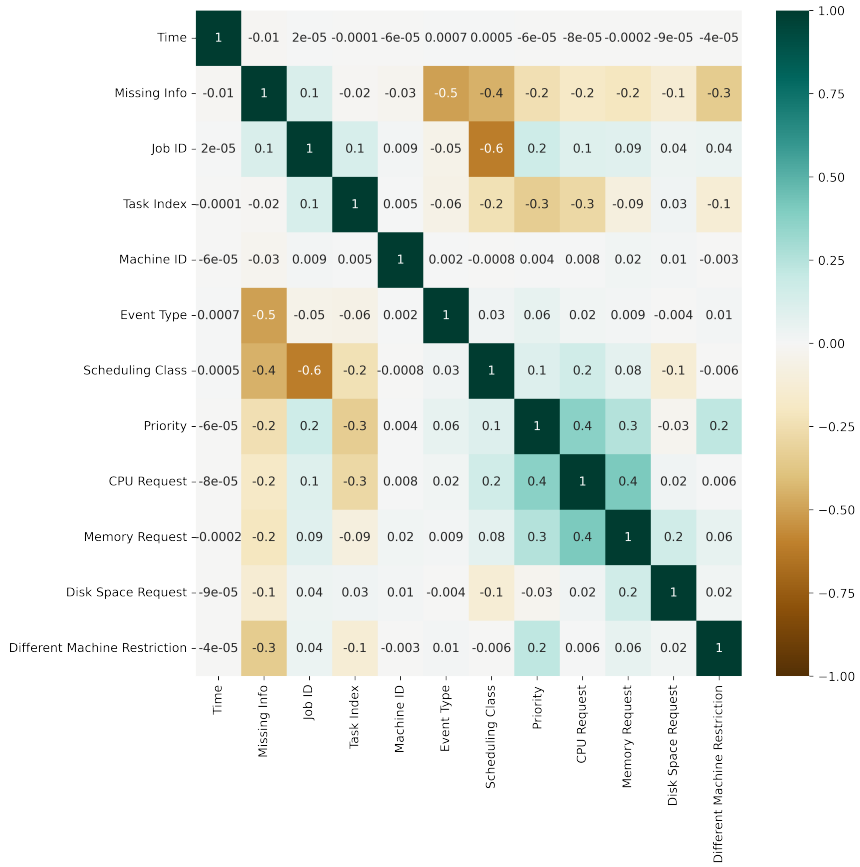

Figure 8 Task Event Table Correlation Heatmap

Logistic Regression has the least accuracy compared to another model. This model has the lowest F-Score and accuracy among the other traditional machine learning model for this experiment. Logistic Regression managed to obtain the accuracy score of 65% and wrongly classify the outcome of 35% of the jobs in the dataset.

Finally, with the exception of Logistic Regression, the three LSTM models developed for this experiment performed somewhat worse than other traditional machine learning algorithms. This may be caused by underfitting the models where there is not large enough data for the model to interpret complexity of the data in order to make the correct classification. We also observed that there was not much difference in performance despite their difference in the number of hidden layers. We also can conclude the amount of LSTM hidden layer suitable for this experiment is two as we can see there is a dip in performance for Tri-Layer LSTM compared to Bi-Layer LSTM.

4.2.2 Task-Level Failure Prediction

We also investigated the feature importance for task-level prediction in order to identify which features have a bigger impact on the decisions made by the machine learning models in predicting failure. All feature importance values were derived using the training dataset.

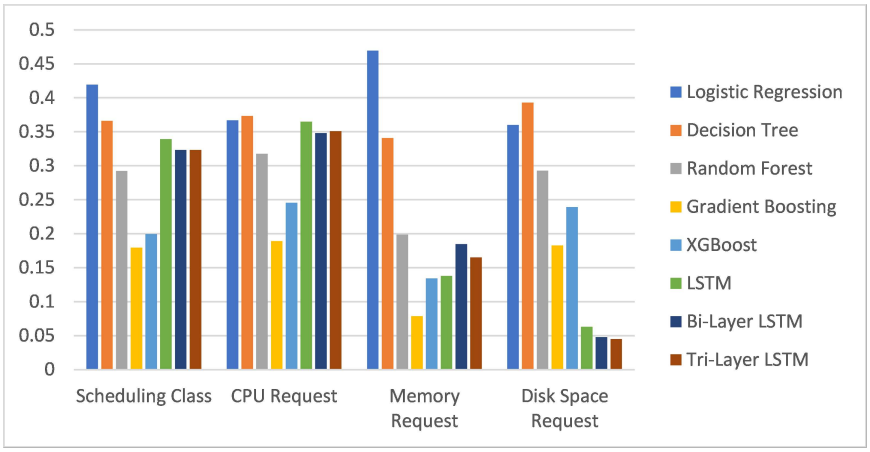

Figure 9 Job-Level Prediction Model Feature Importance

Table 5 Testing Performance Evaluation

Model	Accuracy(%)	Error Rate(%)	Precision(%)	Sensitivity(%)	Specificity(%)	F-Score
Logistic Regression	63.13	36.87	41.36	55.90	66.21	0.5755
Decision Tree	93.23	6.77	91.95	91.34	94.52	0.9165
Random Forest	89.65	10.35	50.97	52.21	94.07	0.5158
Gradient Boosting	90.65	9.35	91.25	86.37	93.83	0.8874
XGBoost	**94.35**	**5.65**	**94.31**	**91.92**	**96.07**	**0.9310**
Single Layer LSTM	88.78	11.22	85.64	86.47	90.33	0.8605
Bi-Layer LSTM	89.78	10.22	56.26	90.82	88.19	0.8722
Ti-Layer LSTM	85.14	14.86	81.57	81.64	57.52	0.8161

As shown in Figure 10, we can see that the priority is considered the most important feature in evaluating the outcome of the task termination status. For the traditional machine learning algorithms with the exception of Gradient Boosting, the priority importance is leveled at 0.2, meaning it became the main factor in determining 20% of the dataset outcome is determined by the priority. For resource requests, only Logistic Regression has a higher score in feature importance compared to the rest of the traditional machine learning algorithms. For the deep learning algorithms, all three LSTM models considered all features as important as we can see in Figure 10. When compared to traditional machine learning models that we have trained, the feature importance score of all features for deep learning models is greater than the feature importance score of traditional machine learning models.

For task-level failure prediction, the amount of data selected is around fourteen million rows of data. Table 6 shows the performance of every model using the training dataset while Table 7 shows the performance of all models trained in predicting the outcome of the test dataset.

From Table 6 and 7, the best performing model in predicting task level failure prediction is tied between Random Forest and Decision Tree. Both of them managed to obtain an accuracy score of 89.75% during the testing phase. However, the Random Forest model gains a slight edge during the training phase, where the Random Forest gained an accuracy score of 92.47% while the decision tree obtained an accuracy

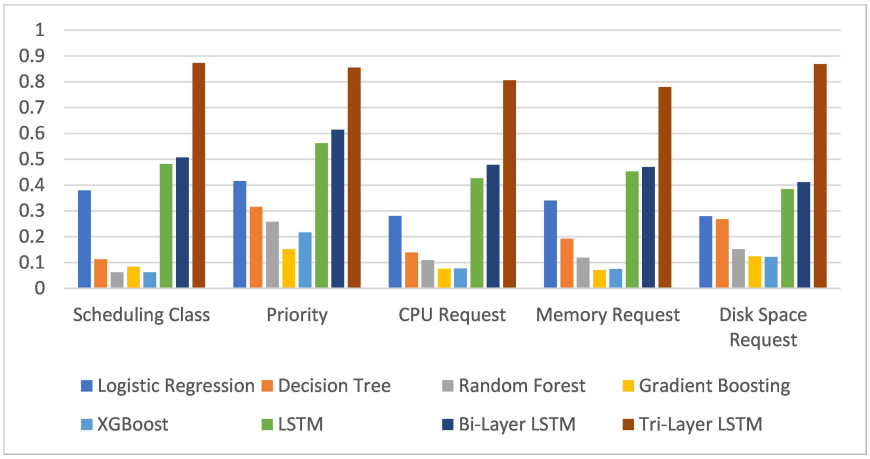

Figure 10 Task-Level Prediction Model Feature Importance

Table 6 Training Performance Evaluation

Model	Accuracy(%)	Error Rate(%)	Precision(%)	Sensitivity(%)	Specificity(%)	F-Score
Logistic Regression	69.68	30.32	71.09	79.95	55.67	0.7526
Decision Tree	89.75	10.25	85.44	98.56	78.42	0.9153
Random Forest	**92.47**	**7.53**	**90.66**	**99.13**	**78.86**	**0.9471**
Gradient Boosting	87.81	12.19	84.81	95.92	76.86	0.9003
XGBoost	89.34	10.66	85.04	98.30	77.89	0.9119
Single Layer LSTM	87.74	12.26	83.94	96.85	75.87	0.8994
Bi-Layer LSTM	86.93	13.07	81.36	98.18	73.84	0.8898
Ti-Layer LSTM	87.54	12.46	82.90	79.53	75.27	0.8962

score of 89.75%. Both Decision Tree and Random Forest only managed to correctly classify 78.4% of the task that finished normally. This score may be the result of the cloud outage mentioned earlier. This trend however can be seen in all machine learning models.

Extreme Gradient Boosting has the third-highest accuracy at 89.35% during the testing phase compared to gradient boosting at 87.87%. The same is true for the training phase where the accuracy for Gradient Boosting is at 87.81% and the accuracy for XGBoost at 89.34%. F-Score for XGBoost is at 0.9120 and 0.9119 while the F-Score for Gradient Boosting is at 0.9003. Logistic Regression has the least accuracy compared to another model. This model has the lowest F-Score and accuracy among the other traditional machine learning models for this experiment. Logistic Regression managed to obtain the accuracy score of 70% and wrongly classify the outcome of 30% of the task in the dataset.

Finally, similar to the job level failure prediction, three LSTM models constructed for this experiment performed marginally worse than all traditional machine learning models built in this experiment, with the exception of Logistic Regression. The accuracy score for all LSTM models are between 86% to 87%. We also observed that there is not much difference in terms of performance between the three LSTM models despite their difference in the number of hidden layers.

Table 7 Testing Performance Evaluation

Model	Accuracy(%)	Error Rate(%)	Precision(%)	Sensitivity(%)	Specificity(%)	F-Score
Logistic Regression	69.69	30.31	85.45	98.57	78.42	0.9154
Decision Tree	**89.75**	**10.25**	**85.45**	98.57	**78.42**	**0.9154**
Random Forest	**89.75**	**10.25**	85.43	**98.58**	78.40	**0.9154**
Gradient Boosting	87.87	12.13	84.80	95.94	76.88	0.9003
XGBoost	89.35	10.65	85.05	98.31	77.89	0.9120
Single Layer LSTM	87.82	12.18	83.39	96.86	76.20	0.8994
Bi-Layer LSTM	86.95	13.05	81.39	98.18	13.87	0.8900
Ti-Layer LSTM	87.55	12.45	82.92	97.53	75.28	0.8963

5 Related Works

In this section, we discuss the related works based on three aspects. Firstly, we review the works that have addressed job and task failure prediction using Google Cluster Traces. Secondly, we review the works that addressed different types of failure prediction. Thirdly, we review other types of predictions that have made use of Google Cluster Traces.

5.1 Job and/or Task Failure Prediction

Table 8 job and task Failure Prediction using Google Cluster Traces [Note: SML - Statistical Machine Learning Algorithms, TML - Traditional Machine Learning Algorithms, DL - Deep Learning Algorithms]

Article	Prediction Scope	Feature Studied	SML	TML	DL
Chen et al. [13]	Job and Task Failure	Job Priority, Resource Requested	-	-	RNN
Soualhia et al. [4]	Task Failure	Waiting Time, Serving Time, Scheduling Class, Priority, Resource Request, Resource Usage	-	Tree, Boost, GLM, CT, RF	NN
Rosa et al. [5]	Job Failure	Task Priority, Resource Request, Scheduling Class, Job Size	LDA, ELDA, QDA	LR	-
Tan et al. [6]	Task Failure	Scheduling Class, Priority, Task Duration, Hourly Failure Frequency, Resource Usage	-	KNN, Clustering	-
Islam and Manivannan [14]	Job and Task Failure	Resource Usage, Priority, Scheduling Class, Job Duration, Number Of Task Re-Submission, Scheduling Delay	-	-	LSTM
Liu et al. [11]	Job Failure	Scheduling Time, Scheduling Class, Job Size, Task Priority, Resource Request	-	SVM, OS-SVM	ELM, OS-ELM
El-Sayed et al. [7]	Job Failure	Job Priority, Sheduling Class Job Size, Resource Request, Resource Usage	-	RF	-
Jassas and Mahmoud [8]	Job Failure	Resource Request, Scheduling Class, Priority	-	DT, RF	-
Shetty et al. [9]	Task Failure	Resource Usage, Job Duration	-	XGBoost	-
Gao et al. [10]	Job and Task Failure	Task Priority, Task Re-Submission, Scheduling Delay, Resource Usage	-	DT, RF	-
Jassas and Mahmoud [12]	Job Failure	Resource Request, Scheduling Class, Priority	-	DT, RF	ANN

The summary of related works regarding job and task failure is presented in Table 8.

Most studies [13, 5, 6, 7, 8, 9, 10, 4, 14, 11] have utilized the data from three tables, which are job event, task event, and task usage. The termination status of a job or task was sorted into two groups which were job or task that has finished successfully and job or task that had failed to finish its intended duty. Due to time and computing power constraints, some of the works opted to make use of a subset of data sampled from the overall dataset.

5.1.1 Statistical Machine Learning Approach

There is a limited number of studies that applied statistical machine learning algorithms. To the best of our knowledge, Rosa et al. [5] is the only study that utilized three statistical machine learning algorithms namely, Linear Discriminant Analysis (LDA), Expanded Linear Discriminant Analysis (ELDA), and Quadratic Linear Discriminant (QDA) to predict Job Failure using Google Cluster Traces. They observed that ELDA is the best statistical machine learning model to predict *job failure* with the accuracy score of 72%.

5.1.2 Traditional Machine Learning Approach

Most of the related works have applied traditional machine learning approach. Soualhia et al. [4] constructed multiple traditional machine learning algorithms namely Tree, Boost, General Linear Model (GLM), Conditional Tree (CT), and Random Forest (RF). Out of all models, RF is the best model for predicting *task failure* in Google Cluster Traces. Meanwhile, Jassas and Mahmoud [8] developed two machine learning models, RF and Decision Tree (DT), to predict *task failure* in Google Cluster Traces. Both models achieved the highest score of 99% accuracy. Gao et al. [10] also performed similar experiment as Jassas and Mahmoud [8] and the only difference between two studies is Gao et al. [10] have studied the effectiveness of both models for the *job* and *task failure* prediction.

Liu et al. [11] built two machine learning models that are based on Support Vector Machine (SVM) to predict job failure. One model is a normal model while another one is an online sequential model. Online sequential is a faster way of training a model. Tang et al. [6] built two task failure prediction models named K-HUNTER which is based on K-Nearest Neighbour (KNN) and C-HUNTER which is based on the clustering algorithm. They observed that both models can effectively detect tasks that are prone to failure at an early stage with high precision.

El-Sayed et al. [7] has built one model based on RF algorithm to predict *job failure* based on scheduling class, job size, and resource requested. Shetty et al. [9], on the other hand, built an Extreme Gradient Boosting Model (XGBoost) to predict *task failure* in the cloud.

5.1.3 Deep Learning Approaches

Islam and Manivannan [14] applied Long-Short Term Memory (LSTM) algorithm for predicting *job* and *task failure* in cloud. For the *job failure*, LSTM achieved an accuracy score of 93%. For the *task failure*, LSTM achieved an accuracy score of 87%. Meanwhile, Chen et al. [13] constructed an ensemble of Recurrent Neural Network (RNN) to predict *job* and *task failure* in cloud. Liu et al. [11] developed

an online learning model which was based on Online Sequential Extreme Learning Model (OS-ELM). The model was then compared to Extreme Machine Learning (ELM), SVM, and Online Sequential Support Vector Machine (OS-SVM). OS-ELM was determined as the best model with the highest accuracy and precision, and better false-positive performance.

Soualhia et al. [4] has constructed a Neural Network (NN) to be compared with multiple traditional machine learning models. They observe NN is not the best model out of all models has been studied. Jassas and Mahmoud [12] also studied the difference between deep learning model performance with traditional machine learning model performance. They built an Artificial Neural Network (ANN) model and compared it to models they have built-in previous studies [8]. They observe ANN achieves its highest performance with 29-days of data from Google Cluster Traces.

5.2 Failure Prediction Using Other Datasets

Table 9 Other Scopes of Failure Prediction [Note: SML - Statistical Machine Learning Algorithms, TML - Traditional Machine Learning Algorithms, DL - Deep Learning Algorithms]

Article	Data Source	Prediction Scope	Feature Studied	SML	TML	DL
Guan et al. [38]	Private Data	System Failure	CPU usage, Memory Usage, Swap Space Utilization Page Faults, Interrupts, Network Activity, I/O and Data Transfer, Power Management	-	Bayesian Network, DT	-
Adamu et al. [39]	NERSC	Component Failure	Failed Component, Failure Time	-	LR, SVM	-
Pitakrat et al. [40]	Private Data	System Failure	Resource Usage, Failure Information, Failure Time	-	RF	-
Zhang et al. [41]	Public Dataset	Switch Failure	Message Template Sequence, Frequency, Seasonality, Surge	HSMM	RF, SKSVM	-
Lin et al. [42]	Private Cloud	Node Failure	Resource Usage, Group Policy, Domain Group, Rack Location	-	LR, SVM, RF	LSTM
Han et al. [43]	Alibaba's Cloud, Backblaze SMART	Disk Failure	SMART Log, SysLog, Trouble Ticket	-	LGBM	
Mohammed et al. [44]	NERSC	Component Failure and Service Failure	Multiple Sources of Failure	ARIMA	LDA, CART, RF, SVM, KNN	-
Chen et al. [45]	Microsoft Cloud System	Outage Prediction	Storage Location, Physical Networking, Storage Streaming Component	-	SVM, PLR, Bayesian Network, XGBoost	-
Li et al. [25]	System X	Node Failure	Resource Usage, Group Policy, Domain Group, Rack Location	-	RF	LSTM
Rawat et al. [46]	Private Cloud	VM Failure	Number of VM Used	ARIMA	-	-
Yu et al. [47]	Alibaba Cloud System	DRAM Failure	System Kernel Log, MCA Data Log	-	XGBoost	-

Besides Google Cluster Traces, there are other datasets used in the literature to study failure prediction in the cloud environment. Table 9 shows the summary of the related works.

5.2.1 Statistical Machine Learning Approach

There are three main works identified under this approach. Firstly, the work by Rawat et al. [46] who has built a model based on Auto-Regressive Integrated Machine Learning (ARIMA) algorithm trying to forecast Virtual Machine (VM) failures based on the analysis of VM usage patterns. The model can perform well and manage to obtain the root mean square error (RMSE) score of 0.046. Secondly, the work by Mohammed et al. [44] also applied ARIMA to forecast the future trend of a system. The model then was implemented together with a traditional machine learning classifier to predict failure in a cloud data center. Thirdly, the work by Zhang et al. [41], which used Hidden Semi-Markov Model (HSMM) as a baseline model, can be compared to a new model for predicting a switch failure.

5.2.2 Traditional Machine Learning Approach

Most of the related works have implemented this approach. Lin et al. [42] constructed MING, a combination of LSTM and RF to predict node failure in a cloud data center. MING has an excellent performance in node failure prediction compared to other existing algorithms such as Logistic Regression (LR), SVM, RF, and LSTM. Li et al. [25], on the other hand, utilizes MING algorithm alongside RF and LSTM to compare different sampling methods for training data to the model performance. RF is proven to be the most suitable model to be fitted with the new sampling method.

Yu et al. [47] built an XGBoost model to predict the DRAM failure in cloud data centers. The training data are from multi-source, including more than three million kernel, data, and mcelogs provided by Alibaba Cloud through PAKDD 2021 competition. The study utilized seven machine learning models, SVM, LR, RF, DT, Gradient Boosting Decision Tree (GBDT), XGBoost, and Light Gradient Boosting Machine (LGBM). Both XGBoost and LGBM obtained superior performance compared to other classifiers used in this study.

Mohammed et al. [44] and Adamu et al. [39] use National Energy Research Scientific Computing Center (NERSC) data obtained from Computer Failure Data Repository (CDFR) website. The data contains information about failures occurring at NERSC from the year 2001 to the year 2006. Adamu et al. [39] used two machine learning models, LR and SVM to predict any hardware or component (CPU, memory, disk, etc) failure in a data center. Mohammed et al. [44] on the other hand used the NERSC data to predict system failure as a whole. The algorithm considered for system failure prediction are SVM, RF, K-Nearest Neighbour (KNN), Classification and Regression Tree (CART), and LDA. SVM is the best model to predict failure with the accuracy score of 90%. SVM algorithm is fitted alongside ARIMA to produce a model that can forecast system failure in a data center.

Chen et al. [45] shows the case study of cloud outage prediction using data obtained from Microsoft Cloud System. Outage prediction helps Cloud Service Provider to

reduce cloud downtime and increase availability. The models utilized in this experiment are Simple Spike (Rule-Based Prediction), SVM, Penalize Logistic Regression (PLR), AirAlert Related (Bayesian Network), and AirAlert Full (XBoost). For component level outage, all the models score at a similar level. In some cases, Simple Spike shows the recall rate of 100% indicating it is the best model for component level outage prediction. For service-level outage prediction, all the models' performance varies. However, Simple Spike, which shows the best performance for component level outage prediction, fails miserably for service level outage prediction. This shows rule-based prediction is not suitable for complex cases.

Guan et al. [38] used an ensemble of Bayesian network and DT to predict cloud failure. Health-related data such as performance information data are used as the feature submitted to predict cloud failure. Another cloud failure research was carried out by Zhang et al. [41], which has built PreFix, a switch failure predictor. PreFix consists of an offline training component that utilised RF and an online prediction component. The training component learns and analyzes the Syslog pattern from past failure and the prediction component use the current Syslog to find any pattern or sign that leads to switch failure. Pitakrat et al. [40] had built HORA, a combination of failure predictor component and architecture knowledge component. For failure prediction, HORA utilizes RF as the base machine learning algorithm.

Han et al. [43], on the other hand, uses RODMAN, which is based on LGBM, Regularized Greedy Forest (RGF), and RF to predict disk failure in a cloud environment. Data collected from Alibaba Cloud and Backblaze SMART are used as the training data. SMART logs contain daily logs of disks performance and reliability statistics. Syslog is a logging system installed with Linux software that records every event hourly. A trouble ticket is a system-generated ticket when abnormal behavior is detected. RODMAN is proven to improve accuracy by its data pre-processing technique compared to the base machine learning algorithm.

5.2.3 Deep Learning Approach

There are only two works that we have identified under this approach category. Lin et al. [42] which has been mentioned earlier, developed MING that illustrates the benefit of integrating traditional machine learning and deep learning algorithms. Their experiment has shown that MING outperformed all the other models that were used as the baselines including LR, SVM, RF, and LSTM. However, Li et al. [25] determined that Random Forest (RF) outperformed both MING and LSTM when it was trained using a new oversampling method, which enriched data by collecting both failing and regular activity for eventually failed nodes, as well as normal node behavior.

5.3 Other Prediction using Google Cluster Traces

Table 10 presents the summary of other prediction approaches using Google Cluster Traces. Most of them focused on the workload prediction, namely, Rasheduzzaman et al. [48], Gao et al. [54], Chen et al. [53], Liu et al. [49], and Zhang et al. [50]. There is one study by Zhang et al. [52] that addressed the resource request prediction and another study by Hemmat and Hafid [51] that tackled the SLA violation prediction.

Table 10 Other Prediction using Google Cluster Traces [Note: SML - Statistical Machine Learning Algorithms, TML - Traditional Machine Learning Algorithms, DL - Deep Learning Algorithms]

Article	Prediction Scope	Feature Studied	SML	TML	DL
Rasheduzzaman et al. [48]	Workload Prediction	Resource Usage, Resource Request	ANFIS, NARX, ARIMA	-	-
Liu et al. [49]	Workload Prediction	Job Duration, Job Waiting Time, Job Status, Machine Availability	MA, WMA	MR	NN
Zhang et al. [50]	Workload Prediction	Resource Request	-	-	RNN
Hemmat and Hafid [51]	SLA Violation	Resource Request, Resource Usage, Resource Availability	-	DT, RF	-
Zhang et al. [52]	Request Prediction	Resource Request	-	-	DBN
Chen et al. [53]	Workload Prediction	Resource Usage	-	SA	RNN, LSTM, GRU, ESN
Gao et al. [54]	Workload Prediction	Resource Usage, Resource Request	ARIMA	BRR, SVR	LSTM
Di et al. [55]	Workload Prediction	Resource Usage	SMA, LWMA, EMA, AR	Bayesian Network	-

5.3.1 Statistical Machine Learning Approach

The work by Rasheduzzaman et al. [48] used Adaptive Neuro-Fuzzy Inference System (ANFIS), Non-Linear Auto-Regressive Network With Exogenous input (NARX), and Auto-Regressive Integrated Moving Average (ARIMA). They observed that NARX outperformed both ANFIS and ARIMA in the workload prediction. Meanwhile, the work by Gao et al. [54] has built an Auto-Regressive Integrated Moving Average (ARIMA) algorithm and compared it with two machine learning models and one deep learning model. All models showed the same level of performance in predicting workload in the cloud.

Liu et al. [49] have experimented with Moving Average (MA) algorithm and Weighted Moving Average (WMA) algorithm together with one traditional machine learning and one deep learning algorithm. All the algorithms then combine to produce EnWoP, an ensemble of all four algorithms. Di et al. [55] used several statistical machine learning algorithms namely, Simple Moving Average (SMA), Linear Weighted Moving Average (LWMA), Exponential Moving Average (EMA), and Auto-Regressive (AR) as a baseline model to be compared to a new model which is based on Bayesian model. The Bayesian model outperformed all the baseline models by up to 50% for long-term prediction.

5.3.2 Traditional Machine Learning Approach

Gao et al. [54] utilizes Bayesian Ridge Regression (BRR) and SVM to analyze performance differences between statistical and traditional machine learning as well as deep learning to predict workload patterns. All models built in this experiment perform at the same level. Liu et al. [49] has used Multi-linear Regression (MR) with statistical machine learning algorithm and deep learning algorithm to produce a model with an ensemble of algorithms. Zhang et al. [50] has used Sparse Autoencoder (SA) to produce an ensemble of SA and RNN to create a new workload prediction model.

Di et al. [55] has constructed a new model based on Bayesian Network to predict the workload of a cloud data center. The model managed to outperform all the baseline statistical models. Hemmat and Hafid [51] built two models which were based on RF and DT. This was made possible by analyzing the amount of available resources, amount of resources requested, and the amount of resource usage. They also studied several sampling methods. They discovered that when SMOTE and Edited Nearest Neighbour (ENN) sampling techniques were applied, the RF performed best.

5.3.3 Deep Learning Approach

The work by Gao et al. [54] built a new model with LSTM algorithm to analyze the performance difference between the statistical machine learning approach, traditional machine learning, and deep learning algorithms. All the models perform at the same level. Both Zhang et al. [50] and Chen et al. [53] built an RNN to predict the workload in the cloud using Google Cluster Traces. Zhang et al. [50] has observed that RNN predicts short term time sequence whereas it is not possible for long term time sequence prediction. Chen et al. [53] has combined Sparse Autoencoder (SA) with RNN to build a workload prediction model. The model outperforms classic RNN, LSTM, Gated Recurrent Unit (GRU), and Echo State Network (ESN). Zhang et al. [52] constructed a Deep Belief Network (DBN) to predict the amount of resource requested using the Google Cluster Traces dataset. Lastly, Liu et al. [49] used NN as a part of an ensemble of multiple algorithms to produce EnWoP to perform workload prediction in the cloud.

6 Conclusion and Future Work

In the cloud environment, reliability and availability are among the key attributes of Quality of Services. In this paper, we have constructed five machine learning models and three deep learning models and compared their performance in predicting job and task failure using Google Cluster Traces dataset. For the job level failure prediction, the best performing model is the XGBoost classifier which has achieved the accuracy score of 94.35% and the F score of 0.9310. For the task level prediction, there are two best performing models, which are Decision Tree and Random Forest. Both models have obtained the accuracy score of 89.75% and an F-Score of 0.9154. This study also has shown that the traditional machine learning models perform slightly better than the deep learning models in classifying job and task termination status in the cloud.

There are several recommendations for future work. Firstly, it is recommended to obtain the raw data to further understand the cause of failures. With these data, the values are not anonymized, hence will aid to gain more insights into the context of cloud job and task failure. Secondly, this study can be replicated to deal with big data with multidimensional features to further assess the performance of the models in categorizing job and task termination status. Lastly, future work may consider time-series data to address the regression problem, especially from the latest version of the Google Cluster Traces dataset.

Acknowledgements
Azlan Ismail acknowledges the support of the Fundamental Research Grant Scheme, FRGS/1/2018/ICT01/UITM/02/3, funded by Ministry of Education Malaysia.

Funding
Similar as in the acknowledgement.

Availability of data and materials
Not applicable.

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Authors' contributions
T.N. conducted the experiments, analyzed the data and wrote the paper; A.I. proposed the idea, reviewed the experiments and revised the paper; J.S revised the paper. All authors have read and agreed to the published version of the manuscript.

Author details
[1]Institute for Big Data Analytics and Artificial Intelligence (IBDAAI), Kompleks Al-Khawarizmi, Universiti Teknologi MARA (UiTM), 40450 Shah Alam, Selangor, Malaysia. [2]Faculty of Computer and Mathematical Sciences (FSKM), Universiti Teknologi MARA (UiTM), 40450 Shah Alam, Selangor, Malaysia. [3]Faculty of Engineering and Information Sciences, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522 AUstralia.

References

1. Stein, M., Campitelli, V., Mezzio, S.: Managing the impact of cloud computing. The CPA Journal; New York **90**(6), 20–27 (2020)
2. Fortune Business Insight: Cloud computing market size, share & covid-19 impact analysis, by type (public cloud, private cloud, hybrid cloud), by service (infrastructure as a service (iaas), platform as a service (paas), software as a service (saas)), by industry (banking, financial services, and insurance (bfsi), it and telecomunications, goverment, consumer goods, and retai, healthcare, manufacturing, others), and regional forecast, 2021-2028. Technical report, Fortune Business Insight (2021)
3. Press Association: British airways it failure caused by 'uncontrolled return of power'. The Guardian (2017)
4. Soualhia, M., Khomh, F., Tahar, S.: Predicting scheduling failures in the cloud: A case study with google clusters and hadoop on amazon emr. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, pp. 58–65 (2015). IEEE
5. Rosa, A., Chen, L.Y., Binder, W.: Predicting and mitigating jobs failures in big data clusters. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 221–230 (2015). IEEE
6. Tang, H., Li, Y., Jia, T., Wu, Z.: Hunting killer tasks for cloud system through machine learning: A google cluster case study. In: 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS), pp. 1–12 (2016). IEEE
7. El-Sayed, N., Zhu, H., Schroeder, B.: Learning from failure across multiple clusters: A trace-driven approach to understanding, predicting, and mitigating job terminations. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1333–1344 (2017). IEEE
8. Jassas, M.S., Mahmoud, Q.H.: Failure characterization and prediction of scheduling jobs in google cluster traces. In: 2019 IEEE 10th GCC Conference & Exhibition (GCC), pp. 1–7 (2019). IEEE
9. Shetty, J., Sajjan, R., Shobha, G.: Task resource usage analysis and failure prediction in cloud. In: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), pp. 342–348 (2019). IEEE
10. Gao, J., Wang, H., Shen, H.: Task failure prediction in cloud data centers using deep learning. IEEE transactions on services computing (2020)
11. Liu, C., Han, J., Shang, Y., Liu, C., Cheng, B., Chen, J.: Predicting of job failure in compute cloud based on online extreme learning machine: a comparative study. IEEE Access **5**, 9359–9368 (2017)
12. Jassas, M.S., Mahmoud, Q.H.: A failure prediction model for large scale cloud applications using deep learning. In: 2021 IEEE International Systems Conference (SysCon), pp. 1–8 (2021). IEEE
13. Chen, X., Lu, C.-D., Pattabiraman, K.: Failure prediction of jobs in compute clouds: A google cluster case study. In: 2014 IEEE International Symposium on Software Reliability Engineering Workshops, pp. 341–346 (2014). IEEE
14. Islam, T., Manivannan, D.: Predicting application failure in cloud: A machine learning approach. In: 2017 IEEE International Conference on Cognitive Computing (ICCC), pp. 24–31 (2017). IEEE

15. Abdul-Rahman, O.A., Aida, K.: Towards understanding the usage behavior of google cloud users: the mice and elephants phenomenon. In: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, pp. 272–277 (2014). IEEE

16. Verma, A., Pedrosa, L., Korupolu, M.R., Oppenheimer, D., Tune, E., Wilkes, J.: Large-scale cluster management at Google with Borg. In: Proceedings of the European Conference on Computer Systems (EuroSys), Bordeaux, France, pp. 1–17 (2015)

17. Gill, S.S., Buyya, R.: Failure management for reliable cloud computing: a taxonomy, model, and future directions. Computing in Science & Engineering **22**(3), 52–63 (2018)

18. Bala, A., Chana, I.: Fault tolerance-challenges, techniques and implementation in cloud computing. International Journal of Computer Science Issues (IJCSI) **9**(1), 288 (2012)

19. Shahid, M.A., Islam, N., Alam, M.M., Mazliham, M., Musa, S.: Towards resilient method: An exhaustive survey of fault tolerance methods in the cloud computing environment. Computer Science Review **40**, 100398 (2021)

20. AbdElfattah, E., Elkawkagy, M., El-Sisi, A.: A reactive fault tolerance approach for cloud computing. In: 2017 13th International Computer Engineering Conference (ICENCO), pp. 190–194 (2017). IEEE

21. Asghar, H., Nazir, B.: Analysis and implementation of reactive fault tolerance techniques in hadoop: a comparative study. The Journal of Supercomputing **77**(7), 7184–7210 (2021)

22. Setlur, A.R., Nirmala, S.J., Singh, H.S., Khoriya, S.: An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. Journal of Parallel and Distributed Computing **136**, 14–28 (2020)

23. Kochhar, D., Jabanjalin, H.: An approach for fault tolerance in cloud computing using machine learning technique. International Journal of Pure and Applied Mathematics **117**(22), 345–351 (2017)

24. Mukwevho, M.A., Celik, T.: Toward a smart cloud: A review of fault-tolerance methods in cloud systems. IEEE Transactions on Services Computing **14**(2), 589–605 (2018)

25. Li, Y., Jiang, Z.M., Li, H., Hassan, A.E., He, C., Huang, R., Zeng, Z., Wang, M., Chen, P.: Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. ACM Transactions on Software Engineering and Methodology (TOSEM) **29**(2), 1–24 (2020)

26. Costa, C.H., Park, Y., Rosenburg, B.S., Cher, C.-Y., Ryu, K.D.: A system software approach to proactive memory-error avoidance. In: SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 707–718 (2014). IEEE

27. Bisong, E.: An overview of google cloud platform services. Building Machine Learning and Deep Learning Models on Google Cloud Platform, 7–10 (2019)

28. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA (November 2011). Revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data

29. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)

30. Tolles, J., Meurer, W.J.: Logistic regression: relating patient characteristics to outcomes. Jama **316**(5), 533–534 (2016)

31. Myles, A.J., Feudale, R.N., Liu, Y., Woody, N.A., Brown, S.D.: An introduction to decision tree modeling. Journal of Chemometrics: A Journal of the Chemometrics Society **18**(6), 275–285 (2004)

32. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

33. Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial. Frontiers in neurorobotics **7**, 21 (2013)

34. Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., *et al.*: Xgboost: extreme gradient boosting. R package version 0.4-2 **1**(4), 1–4 (2015)

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

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

37. Baniecki, H., Kretowicz, W., Piatyszek, P., Wisniewski, J., Biecek, P.: dalex: Responsible machine learning with interactive explainability and fairness in python. arXiv preprint arXiv:2012.14406 (2020). 2012.14406

38. Guan, Q., Zhang, Z., Fu, S.: Ensemble of bayesian predictors and decision trees for proactive failure management in cloud computing systems. J. Commun. **7**(1), 52–61 (2012)

39. Adamu, H., Mohammed, B., Maina, A.B., Cullen, A., Ugail, H., Awan, I.: An approach to failure prediction in a cloud based environment. In: 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 191–197 (2017). IEEE

40. Pitakrat, T., Okanović, D., van Hoorn, A., Grunske, L.: Hora: Architecture-aware online failure prediction. Journal of Systems and Software **137**, 669–685 (2018)

41. Zhang, S., Liu, Y., Meng, W., Luo, Z., Bu, J., Yang, S., Liang, P., Pei, D., Xu, J., Zhang, Y., *et al.*: Prefix: Switch failure prediction in datacenter networks. Proceedings of the ACM on Measurement and Analysis of Computing Systems **2**(1), 1–29 (2018)

42. Lin, Q., Hsieh, K., Dang, Y., Zhang, H., Sui, K., Xu, Y., Lou, J.-G., Li, C., Wu, Y., Yao, R., *et al.*: Predicting node failure in cloud service systems. In: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 480–490 (2018)

43. Han, S., Wu, J., Xu, E., He, C., Lee, P.P., Qiang, Y., Zheng, Q., Huang, T., Huang, Z., Li, R.: Robust data preprocessing for machine-learning-based disk failure prediction in cloud production environments. arXiv preprint arXiv:1912.09722 (2019)

44. Mohammed, B., Awan, I., Ugail, H., Younas, M.: Failure prediction using machine learning in a virtualised hpc system and application. Cluster Computing **22**(2), 471–485 (2019)

45. Chen, Y., Yang, X., Lin, Q., Zhang, H., Gao, F., Xu, Z., Dang, Y., Zhang, D., Dong, H., Xu, Y., *et al.*: Outage prediction and diagnosis for cloud service systems. In: The World Wide Web Conference, pp. 2659–2665 (2019)

46. Rawat, A., Sushil, R., Agarwal, A., Sikander, A.: A new approach for vm failure prediction using stochastic model in cloud. IETE Journal of research **67**(2), 165–172 (2021)

47. Yu, F., Xu, H., Jian, S., Huang, C., Wang, Y., Wu, Z.: Dram failure prediction in large-scale data centers. In: 2021 IEEE International Conference on Joint Cloud Computing (JCC), pp. 1–8 (2021). IEEE

48. Rasheduzzaman, M., Islam, M.A., Islam, T., Hossain, T., Rahman, R.M.: Study of different forecasting models on google cluster trace. In: 16th Int'l Conf. Computer and Information Technology, pp. 414–419 (2014). IEEE

49. Liu, B., Lin, Y., Chen, Y.: Quantitative workload analysis and prediction using google cluster traces. In: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 935–940 (2016). IEEE

50. Zhang, W., Li, B., Zhao, D., Gong, F., Lu, Q.: Workload prediction for cloud cluster using a recurrent neural network. In: 2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI), pp. 104–109 (2016). IEEE

51. Hemmat, R.A., Hafid, A.: Sla violation prediction in cloud computing: A machine learning perspective. arXiv preprint arXiv:1611.10338 (2016)

52. Zhang, W., Duan, P., Yang, L.T., Xia, F., Li, Z., Lu, Q., Gong, W., Yang, S.: Resource requests prediction in the cloud computing environment with a deep belief network. Software: Practice and Experience **47**(3), 473–488 (2017)

53. Chen, Z., Hu, J., Min, G., Zomaya, A.Y., El-Ghazawi, T.: Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Transactions on Parallel and Distributed Systems **31**(4), 923–934 (2019)

54. Gao, J., Wang, H., Shen, H.: Machine learning based workload prediction in cloud computing. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–9 (2020). IEEE

55. Di, S., Kondo, D., Cirne, W.: Host load prediction in a google compute cloud with a bayesian model. In: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, pp. 1–11 (2012). IEEE