Tengku Asmawi *et al.*
Journal of Cloud Computing (2022) 11:47
https://doi.org/10.1186/s13677-022-00327-0

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Cloud failure prediction based on traditional machine learning and deep learning

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

Abstract

Cloud failure is one of the critical issues since it can cost millions of dollars to cloud service providers, in addition to the loss of productivity suffered by industrial users. Fault tolerance management is the key approach to address this issue, and failure prediction is one of the techniques to prevent the occurrence of a failure. One of the main challenges in performing failure prediction is to produce a highly accurate predictive model. Although some work on failure prediction models has been proposed, there is still a lack of a comprehensive evaluation of models based on different types of machine learning algorithms. Therefore, in this paper, we propose a comprehensive comparison and model evaluation for predictive models for job and task failure. These models are built and trained using five traditional machine learning algorithms and three variants of deep learning algorithms. We use a benchmark dataset, called Google Cloud Traces, for training and testing the models. We evaluated the performance of models using multiple metrics and determined their important features, as well as measured their scalability. Our analysis resulted in the following findings. Firstly, in the case of job failure prediction, we found that Extreme Gradient Boosting produces the best model where the disk space request and CPU request are the most important features that influence the prediction. Second, for task failure prediction, we found that Decision Tree and Random Forest produce the best models where the priority of the task is the most important feature for both models. Our scalability analysis has determined that the Logistic Regression model is the most scalable as compared to others.

Keywords: Cloud computing, Job and task failure, Failure prediction, Deep learning, Machine learning

Introduction

Cloud computing is at the forefront of a global digital transformation [1]. It allows a business to provide an extra layer of security in terms 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 American market has spent approximately 78.28 billion dollars on cloud services. The cloud computing market is expected to continue to expand from $219B in 2020 to $791.48B in 2028 [2].

The implementation and deployment of the cloud system opens up to deal with different types of cloud failure [3]. Failing to handle these failures will result in degradation of quality of service (QoS), availability, and reliability. It will ultimately lead to an economic loss for both cloud consumers and providers [4]. This challenge is commonly addressed with fault tolerance management, which offers the ability to detect, identify, and handle faults without damaging the final result of cloud computing [5]. There are several categories of fault tolerance techniques that include redundancy techniques, fault-aware policies (i.e., reactive and proactive policies), and load balance [5]. In this paper, we focus on proactive policies that can be implemented with a failure prediction technique trained using machine learning algorithms. Failure prediction is significant in preventing the occurrence of failure and in minimizing the maintenance costs

*Correspondence: azlanismail@uitm.edu.my

[2] 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

of fault tolerance management. As there are different types of cloud failure, we pay particular attention to the prediction of job and task failure. Both failures are interconnected (i.e., a job contains one or more tasks) and should be tackled simultaneously.

Therefore, in this paper, our aim is to build and evaluate a set of trained models to predict the job and task termination status (i.e. failure or success). For this reason, we have chosen five traditional machine learning algorithms (TML) and three variants of deep learning algorithms (DL). TML algorithms include logistic regression (LR), decision tree (DT), random forest (RF), gradient boost (GB), and extreme gradient boost (XGBoost). Meanwhile, the DL algorithms refer to single-layer long-short-term memory (LSTM), two-layer (bi-layer) LSTM, and three-layer (tri-layer) LSTM. We used the benchmark dataset, Google Cluster Traces (GCT), published in 2011, to train and test the models. We then perform a series of evaluations to find the best models. Therefore, this work contributes threefold. First, an approach to comprehensively produce and evaluate predictive failure models (Section 3). Second, the results and findings of four types of analyzes, namely exploratory data analysis, feature analysis, performance analysis, and scalability analysis (Section 4). Third, a review of cloud failure prediction and machine learning approaches specifically related to GCT, as well as other datasets (Section 5).

The remainder of this paper is organized as follows. Section 2 explains the dataset used and the fundamental background of cloud failures. Section 3 presents the approach to conducting this study. Section 4 provides the results and findings of the analysis. Section 5 summarizes the related works. Section 6 concludes the paper with several future works.

Background

In this section, the dataset used in this study is introduced, followed by a fundamental understanding of cloud failures.

Google Cluster Traces

Overview

Up to date, Google has released three public trace datasets for research and academic use. The first one was released in 2007 and contains 7-hour workload details. The dataset contains only basic information, including time, job id, task id, job category, 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 [6]. 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 was released in early 2020 [7]. Compared 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.

2011 Dataset

The data provided in this version can generally be divided into machine details, and job and task details.

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

Four tables are provided to describe the details of the job and tasks, which are table *job event*, table *task event*, table *task constraint*, and table *resource usage*.

Tables *job event* and *task event* are used to describe the life cycle of the job or the 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 the value zero indicates the job or task submitted by the user, the value one indicates that the task has been scheduled to run on a machine, values two to six indicate that the task has been terminated, while values seven and eight indicate that the task details or requirements have been updated.

Furthermore, there is also a column that indicates whether a task or a 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 nonproduction 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 the table *task event*, there are six more columns in addition to all the ones mentioned above. One of the columns is the machine ID, which determines which machine the task is run on. The second column is the priority of the task, which is valued from zero as the least priority to 11 as the most important task. Next, there are three columns detailing the normalized amount of every requested resource, which are CPU resources, memory resources, and disk space resources. Lastly, there is a column that attributes the constraint of running on a different machine. If there is a value in the column, we note that the task must be executed on a machine different from the machine currently running a different task from the same job.

The table *task constraint* 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. Finally, there is a table *resource usage*. This table discloses the amount of computing resources, such as CPU, memory, and disk space, that have been used for each task. This usage is recorded for each 5-minute or 300-second measurement period.

Failure in Cloud

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

Categories of Cloud Failure

The 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. The more complex the computing system, the higher the probability that the system will fail. There are two classifications of failures, namely architecture-based and occurrence-based failures [3]. These failure categories are summarized in Table 1.

Architecture-based failures include two types of failure, defined as service and resource failures. Service failure usually occurs due to software failure, such as unplanned reboots and cyberattacks, or scheduling failure, such as service timeout. Meanwhile, resource failure is caused by hardware failure, such as power outages, system breakdown, memory problems, and complex circuit design. The occurrence-based failure consists of two types of failure, specifically correlated and independent failures. Correlated failure is a failure that occurs due to another domain of failure. For example, a cloud resource is unavailable due to a power outage that affects the cloud infrastructure. Meanwhile, independent failure is caused by external factors, such as human error and computer overheating.

Table 1 Classification of failure and its cause [3]

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

As we are concerned about the termination status of the job and task, our work can be associated with service failure, where the main cause is driven by scheduling failure.

Fault Tolerance in Cloud Computing

Despite the advancement of cloud computing technology, cloud computing performance is still hampered by its vulnerability to failure. Therefore, fault tolerance is one of the fundamental requirements of cloud computing. There are different ways of classifying fault tolerance approaches [5, 8, 9]. Here, we highlight two major tolerance approaches, namely the reactive and proactive approach.

The reactive fault tolerance approach reduces the effect of failure on the execution of an application. When a failure occurs in the 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 [10]. This technique is used to duplicate tasks on multiple resources. This replication increases the likelihood that at least one task will be completed correctly. The second technique is the re-submission of tasks [11]. When a task fails, it will be rerun with the same node or with a different resource.

The proactive fault tolerance approach is based on the principle of preventing the entire failure from occurring. For this approach, the condition of the physical system is constantly monitored, and the occurrence of a system failure must be predicted. If the probability of a

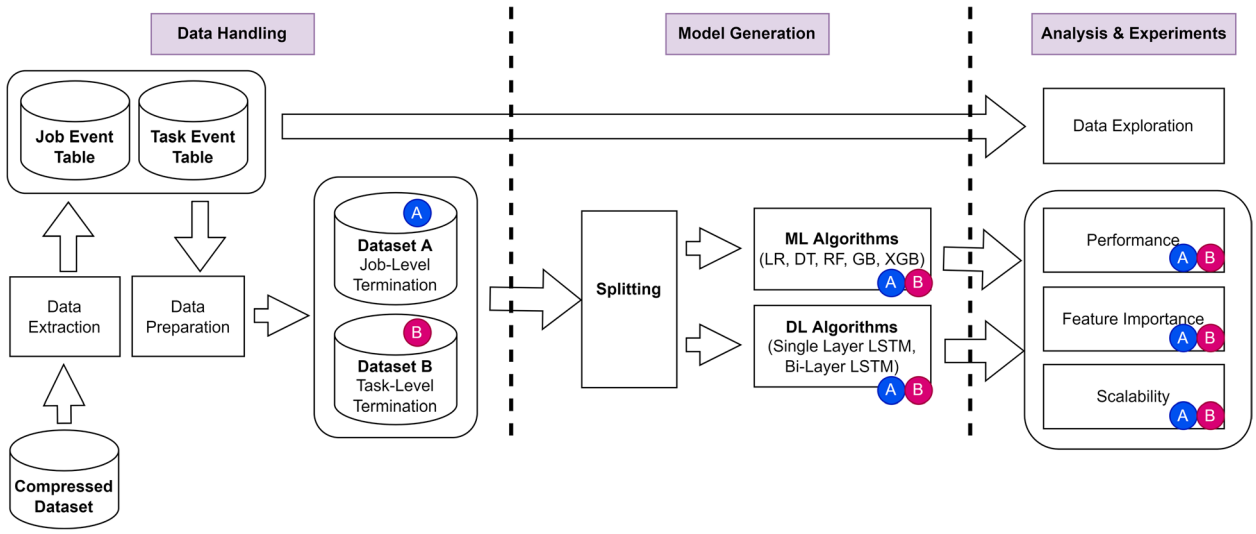

Fig. 1 This figure shows the approach to implement a comprehensive comparison study of the prediction of job and task failure driven by the GCT dataset

fault occurring is high, cloud service providers will take a preventive measure, such as removing hardware from the service cluster or performing corrective measures on the software. Failure prediction can be built using the information collected from previous cloud failures. Machine learning is an excellent tool for predicting software and hardware failures in cloud infrastructures. Failure prediction is considered a proactive fault tolerance approach if it is implemented in the cloud infrastructure [12].

Existing work has implemented the proactive tolerance approach, such as predicting hardware failure in the cloud farm [13] and predicting memory failure in a computer system [14]. Another example is the work of [15], where a machine learning algorithm is constructed to predict 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 occurs, potentially disrupting the availability of the cloud resources.

Method

This section introduces and explains the approach proposed to carry out this study.

Overview

Figure 1 shows the approach to implement the comprehensive comparison study of the prediction of job and task failure driven by the GCT dataset. We divide the activities involved into three phases, namely *data handling*, *model generation*, and *analysis and experiments*. Phase *data handling* comprises the activity of extracting from the published tables (that is, the job event and task event tables) and preparing the two datasets, which we call data set A, which comprises the job-level termination data and dataset B, which comprises the task-level termination data. Table 2 shows the sample of dataset A, while Table 3 shows the sample of dataset B. Phase *model generation* involves the process of developing and training predictive models based on the TML and DL algorithms. The predictive model is meant to address the classification problem, that is, to classify the termination status of

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

each job or task, either success or failure. Phase *analysis and experiments* comprises four analyzes, namely exploratory data analysis (EDA), performance analysis (PA), feature importance analysis (FA) and scalability analysis (SA). The EDA is applied to uncover the behavior of the dataset. The input data for the EDA are referred to the published dataset. PA is used to determine the quality of predictive models based on certain metrics to determine the best models. FA is used to identify the feature importance of the predictive models. SA is applied to understand the scalability of predictive models in relation to different data sizes. The input data for PA, FA, and SA are referred to datasets A and B.

Furthermore, Fig. 2 shows the technical workflow and platforms for implementing the proposed approach presented in Fig. 1. There are two platforms that are used to implement the overall phases. First, we use the Google Cloud Platform (GCP) [16] to handle data and generate models. Second, we used a local machine to perform analysis and experiments. For the GCP, we configure a virtual machine with 4 CPU cores and 16 GB of memory with a NVIDIA Tesla T4 GPU. Meanwhile, the specification of local machine is based on 2 CPU cores with a clock speed of 2.30 GHz and 20 GB of memory.

Data Handling

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

Data Extraction

The sources of data are retrieved from Google Cloud Storage (GCS). Information on the dataset is available at [17] for the 2011 version. We downloaded the GCT dataset with a size of approximately 400 GB. *urllib* library was used to access the data stored in the GCS. The files were stored in a zipped file and partitioned. The number of partitions was embedded in the Uniform Resource Locator (URL) of the GCS. Then, all data were extracted using the *gzip* library to generate the intermediate dataset, namely, the *job event table* and *task event table* in csv format. To automate this task, a function was built to load, extract, and save the intermediate dataset into the respective directory.

Data Preparation

This task involves several subtasks, which are *data cleaning*, *data integration*, *data reduction*, *data transformation*, and *class balancing*. We utilized the Dask library instead of Pandas to process the data since Dask enables parallelism. Having said that, Dask provides similar capability as Pandas, since it is built on top of Pandas. The intermediate data sets, namely, the job event table and the task event table, are the main input for this task. The aim is to prepare two new datasets, namely, a job-level termination dataset (i.e., dataset A) and a task-level termination dataset (i.e., dataset B). The details of each subtask are as follows.

Data Cleaning

In the case of the GCT dataset, we perform data cleaning, in particular, we handle missing values in the task event table. There are three columns affected by this problem, namely *CPU request*, *memory request*, and *disk space request*. We solve this problem by removing related records that contain missing values using the Pandas function *dropna*. We then make use of the visualization approach to check the data. We used a boxplot to observe the data distribution for the continuous data chart and a bar and pie chart to observe the categorical data. It is

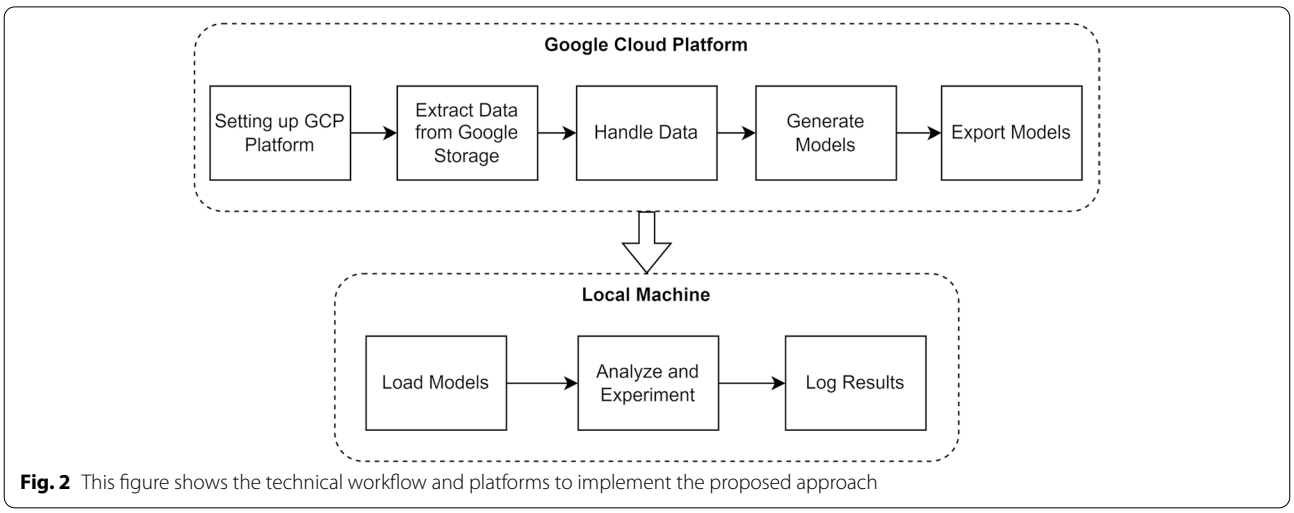

Fig. 2 This figure shows the technical workflow and platforms to implement the proposed approach

important to note that we also found some outliers in the task event table. However, we have decided to keep these records because they can potentially contribute to identifying the termination status of the job and predicting task failure.

Data Integration

In this study, we need to integrate data from the job event and task event table into a single table to produce the job-level termination data set (i.e., data set A). This task is needed to increase the relevant features for dataset A. To support data integration, we need to identify the most appropriate values from the task event table to be merged with the record in the job event table. This is important since one job event can have multiple task events. For this reason, we apply a data aggregation step in which each set of task event records associated with each job is aggregated based on their three features, namely *CPU requests*, *memory requests*, and *disk space requests*. The purpose is to find the maximum value for each of them. These maximum values are then used to merge with the job event records. The data integration task is implemented using the *merge* function of the Dask library. Meanwhile, dataset B is not involved in the integration task because the existing features are sufficient for modeling purposes.

Data Reduction

This task is carried out to obtain the subset of the entire dataset. It is needed to enable us to conduct the analysis and experiments using the available platform with a limited budget. Hence, this task involves two objectives. The first is to reduce the number of columns or features, which is supported by the correlation analysis. The second objective is to reduce the number of rows by deciding the subset of the dataset.

For the first objective, we first change the string datatype of the respective features into a numerical datatype. This is done using Pandas' *astype* function. After that, we construct a Heatmap to identify the correlation between all the respective features and to determine which features to be removed from the dataset. In general, those features that have weak to no correlation with the outcome of a job or task are removed. Once removed, the only features that remain in dataset A are the *scheduling class* and resource-related requests (i.e. *CPU requests*, *memory requests* and *disk space requests*). Meanwhile, the remaining features of dataset B are *scheduling class*, *priority* and resource-related requests.

For the second objective, we reduce the rows or records on the basis of two filters. The first filter uses the *timestamp* feature, where we have decided to select data from the first fourteen (14). As the value is in a timestamp format, we have to convert the day into microseconds to enable the filtering task. For the second filter, we have decided to remove some records using the *event type* feature. There are 9 distinct values in *event type*. Here, we have chosen to remove those with their event types considered incomplete lifecycle (that is, job / task submission (0), job / task scheduled (1), update while still in queue (7), and update during running (8)).

Data Transformation

In this task, the focus is on re-categorizing the termination status in both datasets, A and B. Hence, the *even type* feature is used for this reason. Due to the previous data reduction, the remaining event type values are evict (2), fail (3), finished (4), kill (5) and lost (6) which indicate the termination status of each job or task. Based on these values, we categorize the type of event into two, *success* (that is, where the type of event equals 4) and *failure* (i.e., other than 4).

Class Balancing

This task is needed due to the number of records of the *failure* and *success* classes being imbalanced. There are several techniques that can be used to balance them. Here, we apply the synthetic minority sampling technique (SMOTE) [18]. In general, SMOTE facilitates the selection of 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 the minority class and will also aid in model construction and training, especially for the DL models in the later stage.

Model Generation

During this phase, two types of machine learning algorithms are implemented, classified as TML and DL algorithms. Each dataset, A and B, is divided into two parts, specifically 70% for training and 30% for testing. The algorithms involved are as follows.

Traditional Machine Learning Algorithms

We have implemented three types of algorithms to address the classification problem, which are regression, tree, and ensemble, respectively. In the case of regression, LR [19] is chosen since it is the most investigated regression algorithm in machine learning. For the tree, we have selected DT [20] since it is the primary tree machine learning algorithm for the classification problem. Lastly, for the ensemble category, we have chosen RF [21], GB [22], and XGBoost [23]. These machine learning algorithms were implemented using the scikit-learn library [24], except XGBoost, which was implemented using the XGBoost [23] library. All models were built as a default model with a small change in the default arguments.

However, for the LR model, the number of maximum iterations has been changed because the default value is insufficient for converging all solvers. Solver is an optimization algorithm for calculating the loss of the LR model.

Deep Learning Algorithms

The last three machine learning algorithms will be three different variants of the LSTM [25] based algorithms. They are differentiated by the number of layers. The three variants of the DL model are Single Layer LSTM algorithm, Bi-Layer LSTM algorithm (two hidden layers), and Tri-Layer LSTM (three hidden layers), respectively. Finally, we add a dense layer to ensure that our algorithm produces only a single value for a prediction. The epoch is set to 100 to ensure we gain the best model possible. To reduce training times and prevent overfitting of the model, training is automatically stopped if there is no improvement of the validation loss value after 10 epochs.

Analysis & Experiments

This phase comprises exploration activities, identification of the feature importance and determination of the best models in terms of performance and scalability. Each of them is explained below.

Exploratory Data Analysis

This analysis focuses on exploring the data in the job event table and the task event table before preparing the data for the machine learning task. The job event table comprises of eight columns namely Timestamp, Missing Info, Job ID, Event Type, User Name, Scheduling Class, Job Name, and Logical Job Name, Meanwhile, the task event table comprises of thirteen columns namely Timestamp, Missing Info, Job ID, Task Index, Machine ID, Event Type, User Name, Scheduling Class, Priority, CPU Request, Memory Request, Disk Space Request, and Different Machine Constraint. Several types of analysis are performed, namely data distribution analysis for exploring the continuous type features and data classification analysis for exploring the discrete type features. We then produce a series of visualizations, specifically a bar chart and a pie chart to visualize the results of data categorization and a box plot to visualize the results of data distribution.

Performance Analysis

This analysis focuses on measuring the performance of predictive models towards building highly accurate classifiers. The analysis is divided into job-level and task-level predictive models. A large amount is used to train and test the models, namely 1 million records for the job-level prediction and 14 million records for the task-level prediction. A set of evaluation metrics is applied to measure each model, namely error rate, precision, sensitivity, specificity, and F-score. These measures can be obtained from the confusion matrix generated, which consists of four parameters: True positive (TP), true negative (TN), false positive (FP) and false negative (FN).

Feature Importance Analysis

The feature importance refers to the score that measures the importance of each feature in a predictive model. Feature importance helps to understand the relative importance of each feature in estimating the models. Feature importance does not relate to the accuracy of the model. The importance of each feature is determined by a score calculated by predicting the training data. A high score means that the feature will have high priority in determining the outcome of the prediction. In this study, the scores of feature importance are calculated using the library *Dalex* [26].

Scalability Analysis

This analysis focuses on measuring the scalability of predictive models, which means the ability of the models to scale in response to the amount of data. It is determined by measuring the time taken to make the prediction based on the given input data. For this analysis, we have prepared a few sets of data for analyzing the job-level and task-level predictive models. In the case of the job level, there are three sets of input data of different sizes, namely (1) 10,000 rows, (2) 100,000 rows, and (3) 1,000,000 rows. For the task level, we prepared four sets of input data, specifically, (1) 10,000 rows, (2) 100,000 rows, (3) 1,000,000 rows, and (4) 10,000,000 rows. These input data are taken from the job event table and the task event table. For this analysis, the event type (i.e. the dependent variable) is ignored, and only the independent variables are considered. The process of preparing the data is similar to the process of data handling discussed in Section 3.2 that excludes data reduction, data transformation, and class balancing.

Results and Findings

This section explains the results of analysis and experiments, namely, related to exploratory data analysis, performance, feature importance, and scalability aspects.

Results of Exploratory Data Analysis

We explain the results based on two tables, which are the job event and the task event table.

Job Event Table

This table contains eight columns (or features) and 2,012,242 rows, where three of the columns take a hash string type and the rest are defined as integer type.

Meanwhile, there are only 672,074 unique job identifiers. Each job has at least three occurrences, namely, job submission, job scheduling, and job termination. Figure 3 shows the distribution of scheduling classes in the job event table, while Fig. 4 shows the job termination status classified by their scheduling class.

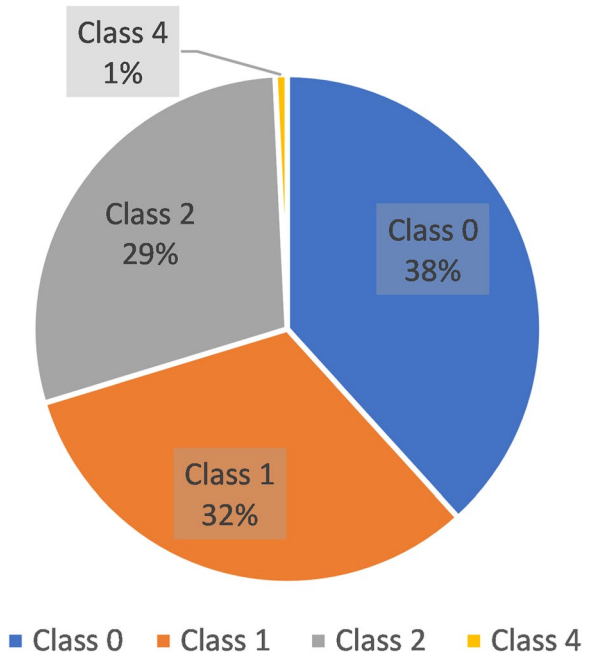

Fig. 3 The figure shows the distribution of scheduling class in job event table. Most of the job events are classified as 0 which indicates a non-production job

There are four scheduling classes where scheduling class 0 indicates the non-production jobs (e.g. non-business critical analysis), scheduling class 3 represents latency-sensitive jobs (e.g. serving revenue-generating user requests), while scheduling classes 1 and 2 represent the jobs that are somewhere between the least and the most latency one. Based on Fig. 3, almost half of the jobs are identified as class 0. There are around 5,000 jobs designated as class 3, 215,000 jobs as class 1 and 194,000 jobs as class 2. From Fig. 4, about 385,582 jobs finished normally, and 274,341 jobs were killed due to user interruption, or their dependent jobs have died. About 10,000 jobs were failed due to task failures. Lastly, about 22 jobs were evicted because of executing jobs with higher priority, the scheduler overcommitted, the actual demands exceeded the machine capacity, the machines became unusable, or the disks were failed.

Figure 5 shows the correlation between the features in the job event table using heatmap. As shown, we can observe that *event type* is highly correlated with *missing info*. Also, note that other features have a weak or no correlation with *event type*. Two features, namely *user name* and *job name*, are not considered in this correlation, as we focus only on anonymous values.

Task Event Table

This table contains 13 columns (or features) and 144,648,288 rows, where eight of the columns are integer types, three are floating types, and one column is string type. There are 25,424,731 unique tasks in the table. A unique task is identified by the combination of the job id and the task index. Similarly to the job event table, a single task may contain at least three occurrences, namely,

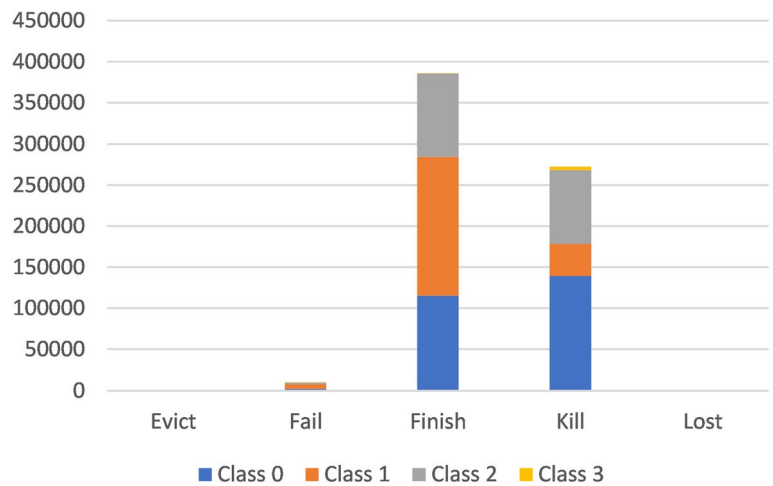

Fig. 4 The figure shows the distribution of termination status categorized based on the scheduling class. Most of the job events terminates with the finish status

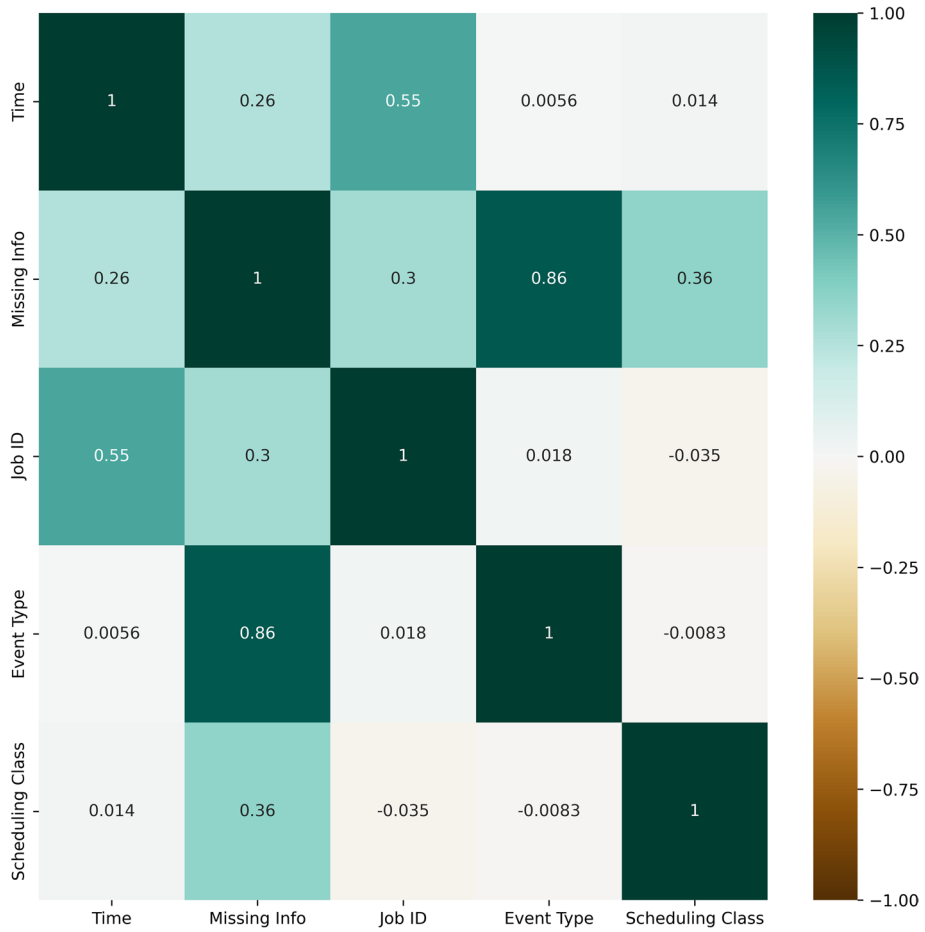

Fig. 5 Job Event Table Correlation. The figures shows the the heat map of the correlation analysis of the job event table

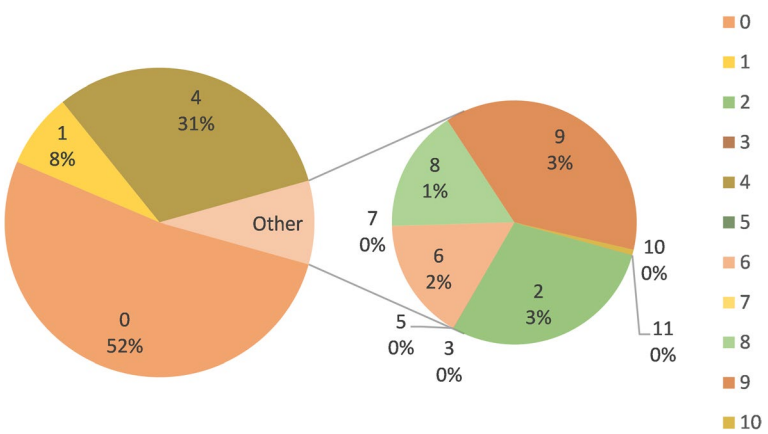

Fig. 6 The figure shows the distribution of task priority in the task event table. The left chart depicts the overall distribution of task priority, whereas the right chart depicts a detailed distribution of job priority, that is smaller than 5%

task submission, task scheduling, and task termination. Of 25,424,731 rows, 18,375 rows were synthesized, which is approximately 7% of the total records. Figure 6 shows the distribution of priority tasks, while Fig. 7 shows the termination status of tasks categorized by priority.

From Fig. 6, we can see that more than half of the task priority in the dataset is identified as level 0 which represents the least priority task. Unlike the job, task priority plays a significant role in determining the resource access for each task. From Fig. 7, 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, 3 of 4 tasks did not finish correctly. This ratio is abnormal, as a cloud service provider usually aims for 99.9% service availability. Therefore, we can assume that there were a series of outages at the site while monitoring the traces.

The distribution of resource requests is shown in Fig. 8. As shown, the average value for the CPU request per task is around 0.0125 to 0.0625 of the total CPU resources in a machine. The amount of memory request

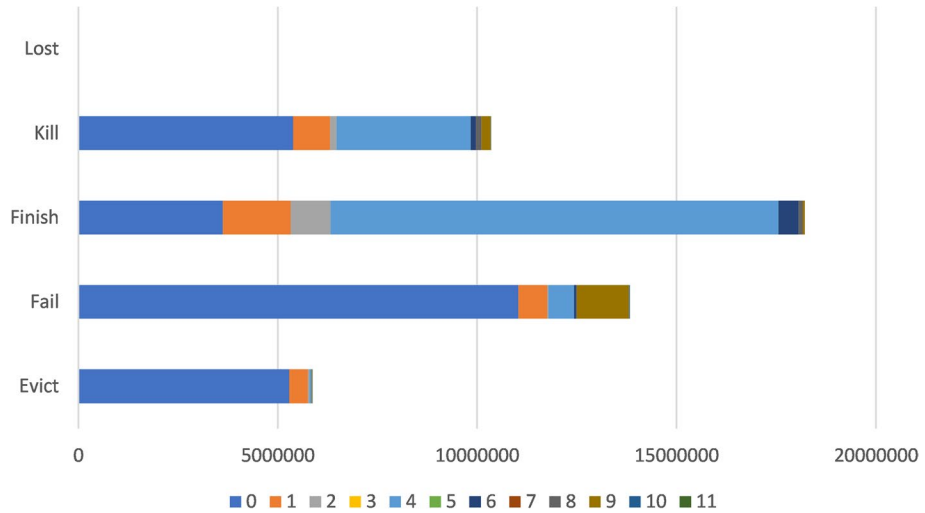

Fig. 7 The figure shows the distribution of the tasks' termination status in the task event table. Level 0 is the least priority task, while level 11 is the highest priority task

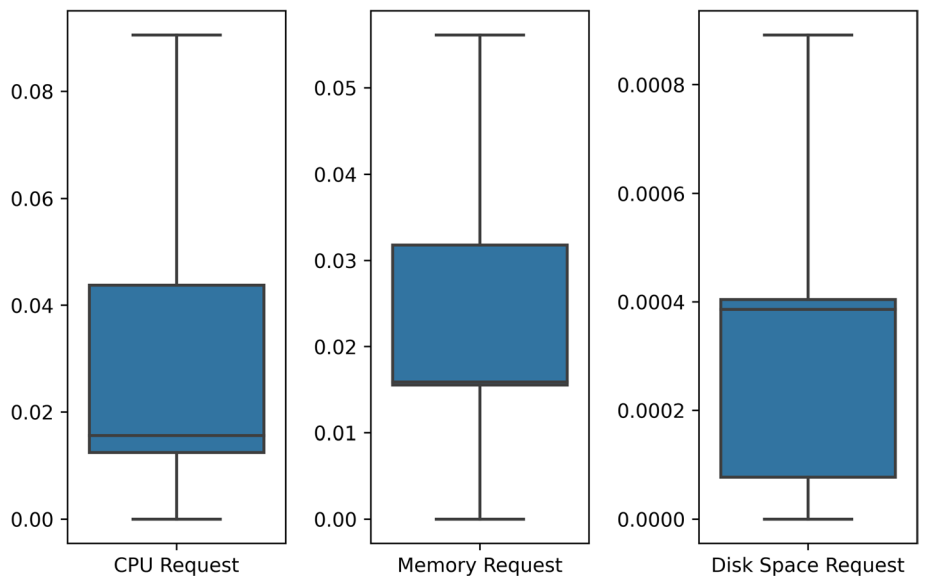

Fig. 8 The figures show the distribution of the resource requests (CPU request, memory request, disk space request)

required is usually in the range of 0.01553 to 0.03180, while the disk space request is in the range of 0.000058 to 0.000404. The values of all resources have been normalized from 0 to 1. As a result, we may conclude that the amount of disk space needed is insignificant compared to the available resource. We can also see that each requested task takes less than 10% or 0.1 of the total CPU and memory requests.

Figure 9 shows the correlation of features in the task event table using heatmap. From the figure, we can see that *event type* has a weak correlation with other features. In addition to that, we notice that there is a positive correlation between *resource usage* and *priority*. *Scheduling class* does not appear to be affected by the amount of resources requested and the status of the task. Although it is a weak correlation, *missing info* is adversely correlated with *resource requests*.

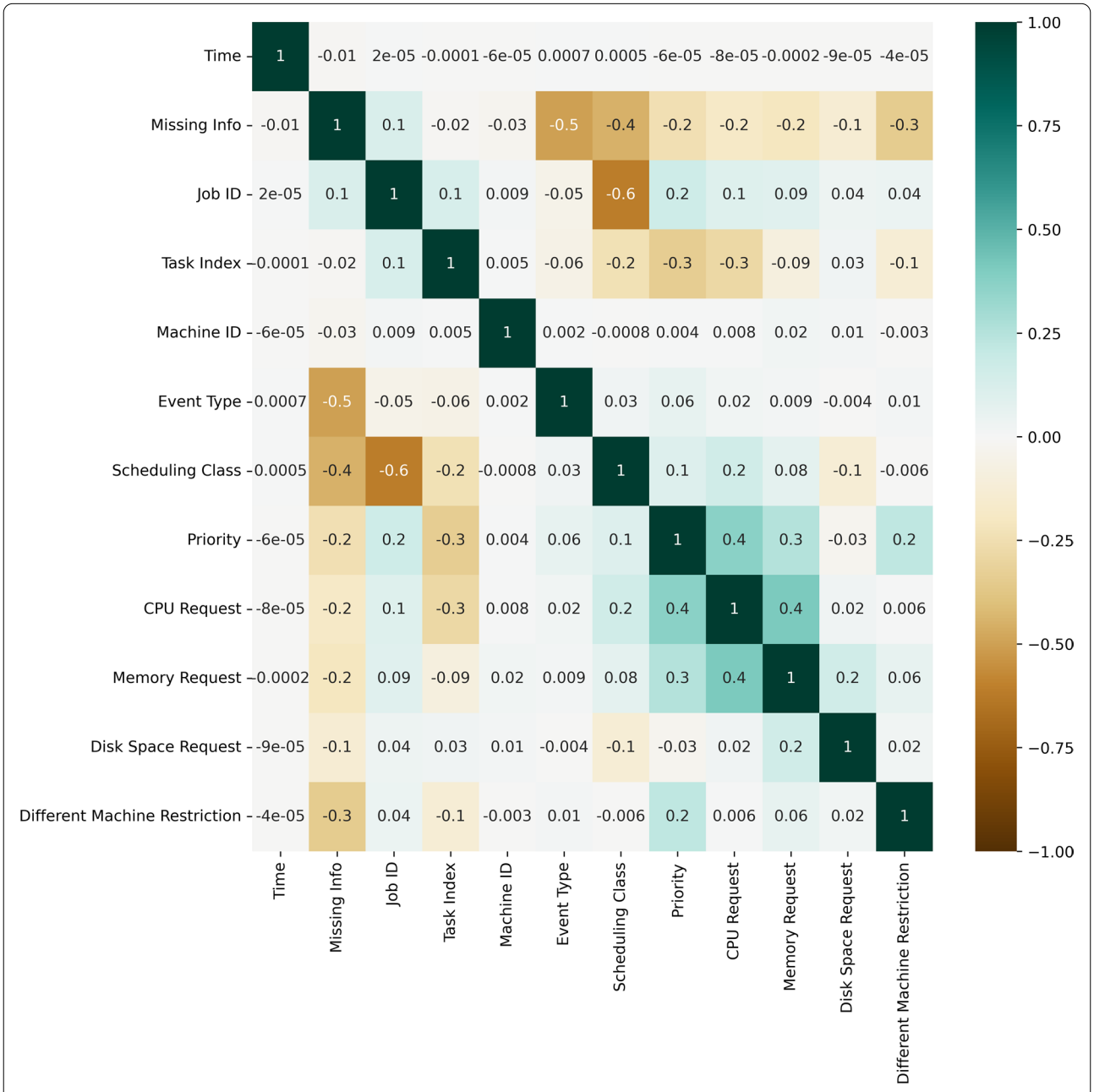

Fig. 9 This figures show the heatmap of the correlation between features of the task event table

Feature Importance Results

Herein, we present the results of feature importance based on job-level and task-level failure prediction.

Job Level Failure Prediction

Figure 10 shows the features that matter the most for all models driven by machine learning. With the exception of the GB and XGBoost models, in general, *scheduling class* and *CPU request* are the most significant features in determining the result of the job termination status. These two features score around 30 to 35% for their importance in cloud job termination status. For the three DL algorithms, it has been shown that *CPU request* is the most important feature in determining the termination status of the cloud job, whereas for the LR model, *memory request* is the most important feature in predicting the termination status of the job.

Task Level Failure Prediction

As shown in Fig. 11, the *priority* is the most important feature in predicting the termination status of the task. For the TML algorithms, with the exception of GB, the priority importance score is 0.2, which means that it has been the main factor in determining 20% of the result of the given dataset. In the case of resource-related requests (i.e., *CPU request*, *memory request*, *disk space request*), it has been determined that *memory request* is the most important feature for TML models, excluding the DT model where *disk space request* is the most important. For DL models, it has shown that *memory request* is the most important feature to predict the outcome in Single-Layer LSTM and Bi-Layer LSTM, while for Tri-Layer LSTM, *disk space request* in the most important resource request.

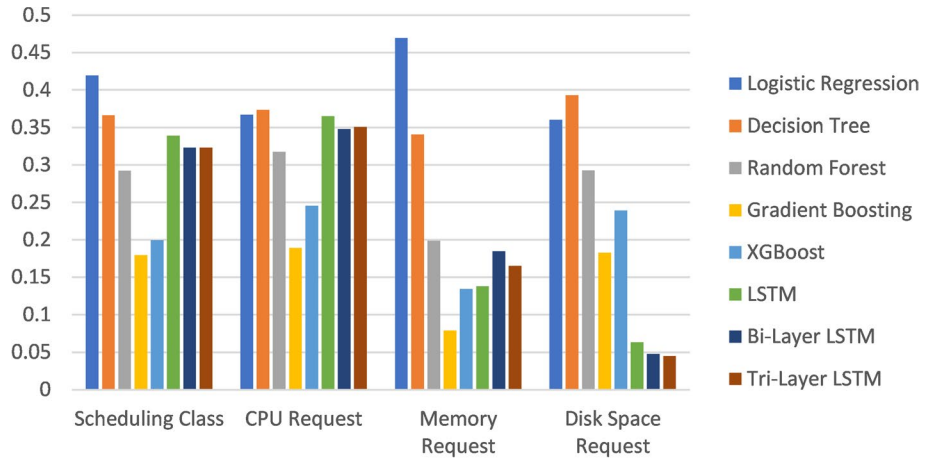

Fig. 10 This figures show the feature importance for the job level prediction models

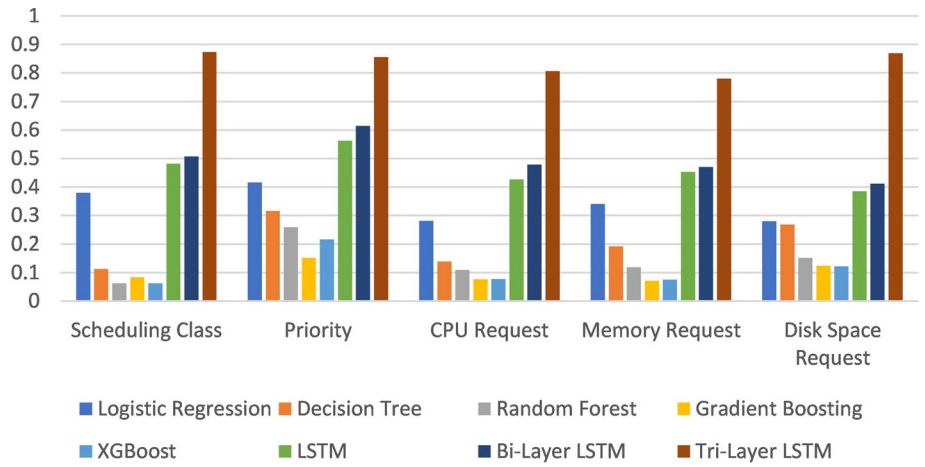

Fig. 11 This figure shows the feature importance for the task level prediction models

Performance Results

We present the performance results based on the prediction of failures for the job level and the task level.

Job Level Failure Prediction

Table 4 shows the performance of each model using the training dataset, while Table 5 shows the performance of all models trained to classify the termination status of the test dataset. We can observe that the performance of all LSTM models is lower than the performance of the TML models, excluding the LR model. Specifically, the XGBoost model has the best accuracy at 93.25% end at 93.10%. The F-score scores of 0.9325 and 0.9310 further demonstrated that XGBoost is the best model of all the models generated in this experiment. XGBoost also demonstrated the highest precision, correctly identifying 94.31% of the termination statuses of the job. Furthermore, the XGBoost sensitivity and specificity scores are 91.92% and 96.07%, respectively, indicating that it can successfully predict the termination status of the job.

The second highest accuracy is recorded by the DT model. The results for the XGBoost and DT models show only a slight difference in terms of accuracy, precision, and specificity performance during the testing and training phase. The DT model has 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. Similarly to XGBoost, the DT model also obtained a high score for sensitivity and specificity during the training and testing phase, demonstrating that the model can correctly classify the job termination status.

The GB model has the third highest accuracy at 90.72% during the testing phase compared to the RF at 89.65%. However, during the training phase, the GB model shows a lower accuracy at 90.72% compared to the RF at 93.27%. It has also 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. Therefore, the RF model is the fourth model used to predict job failure prediction, although the accuracy is slightly lower during testing, which is 89.65%. The LR model has the least accuracy compared to another model. This model has the lowest F-score and accuracy among the other TML models in this experiment. The LR model managed to obtain the accuracy score of 65% and incorrectly classify the result of 35% of the jobs in the dataset.

Finally, with the exception of the LR model, the three LSTM models developed for this experiment performed

Table 4 This table presents the performance results of each predictive model for the job level based on the training dataset, where the model in bold represents the best model

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

Table 5 This table presents the performance results of each prediction model for the job level based on the test dataset, where the model in bold represents the best model

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

somewhat worse than other TML algorithms. This may be caused by underfitting the models when there is not enough data for the model to interpret the complexity of the data to make the correct classification. We also observed that there were not many differences in performance despite their differences in the number of hidden layers. We also can conclude that 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.

Task Level Failure Prediction

Table 6 shows the performance of each model using the training dataset, while Table 7 shows the performance of each model trained based on the test dataset. As shown, the best models to predict task-level failure prediction are the RF and DT models. Both achieved 89.75% accuracy during the testing phase. However, the RF model gains a slight edge during the training phase with 92.47% accuracy, while the obtained DT model achieved 89.75%. Both the DT and RF models managed to correctly classify 78.4% of the task that ended normally. This trend can be seen in most machine learning models.

XGBoost has the third highest accuracy at 89.35% during the testing phase compared to GB at 87.87%. The same is true for the training phase where the accuracy for the GB model is 87.81% and the accuracy for XGBoost at 89.34%. F-Score for XGBoost is 0.9120 and 0.9119, while the F-Score for the GB model is 0.9003. The LR model has the lowest accuracy compared to another model. This model has the lowest F-score and accuracy among the other TML models for this experiment. The LR model has achieved 70% accuracy and wrongly classified 30% of the task in the dataset.

Finally, similar to the job-level failure prediction, the three LSTM models performed marginally worse than all TML models excluding the LR model. The accuracy score for all LSTM models is between 86% and 87%. We also observed that there are not many differences in terms of performance between the three LSTM models despite their differences in the number of hidden layers.

Scalability Results

We present the scalability results based on the prediction of the failure level of the job and the task level.

Table 6 This table presents the performance results of each predictive model for the task level based on the training dataset, where the model in bold represents the best model

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

Table 7 This table presents the performance results of each predictive model for the task level based on the test dataset, where the models in bold represent the best models

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

Job Level Failure Prediction

Figure 12 illustrates the scalability result of the job-level prediction models. For the TML models, we found that on average a single prediction will take around 2 to 3 microseconds per prediction, whereas the DL models take around 2 to 3 milliseconds for a single prediction. The best model for TML is based on the LR model, where it can predict up to 1 million inputs in under one second.

Task Level Failure Prediction

We notice that the time taken for a task-level prediction is similar to that for the job-level prediction. Figure 13 illustrates the scalability result of the task-level prediction models. For the TML models, we found that on average a single prediction will take around 2 to 3 microseconds per prediction, whereas the DL models take around 2 to 3 milliseconds for a single prediction. The best model for the TML model is based on the LR model, where it can predict up to 10 million inputs in less than two seconds.

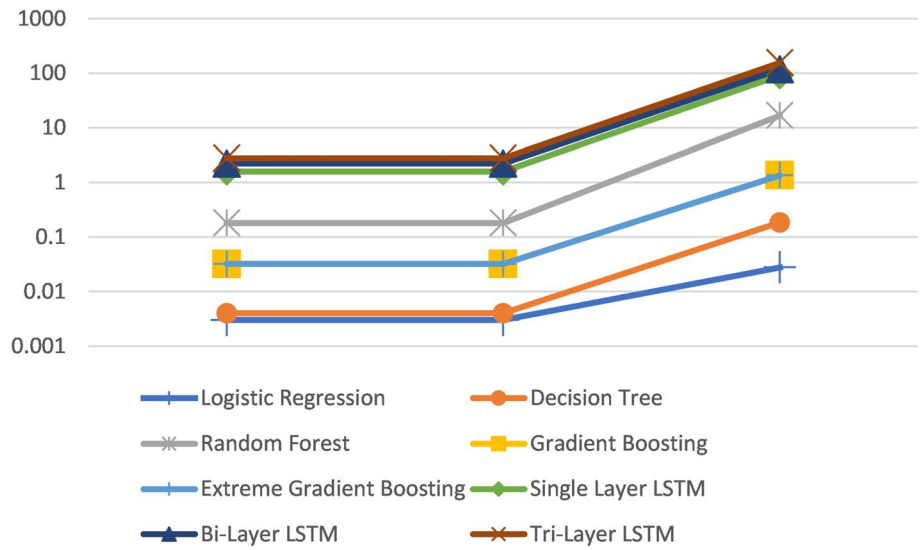

Fig. 12 This figure shows the scalability of job level failure prediction models in relation to different amount of data

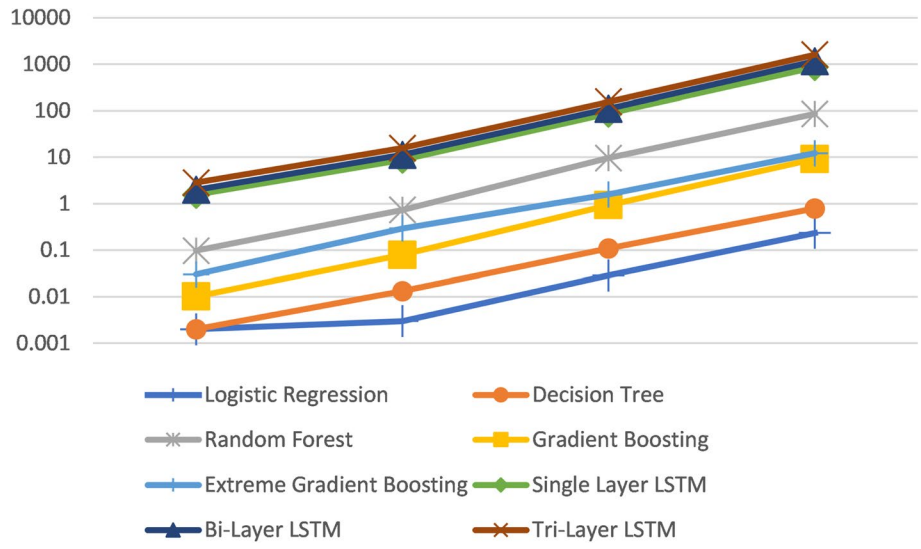

Fig. 13 This figure shows the scalability of task level failure prediction models in relation to different amount of data

Related Works

In this section, we discuss related work in three aspects. First, we review the work that has addressed the prediction of job and task failure using the GCT dataset, as shown in Table 8. Second, we review the work that addressed different types of failure prediction using different types of dataset, as shown in Table 9. Third, we review other types of prediction that have specifically used the GCT dataset as shown in Table 10.

The review resulting in Table 8 is based on three key elements. First, we identify the prediction scope to determine whether the related work addresses job failure or task failure, or both. Second, we identify the feature studied in relation to the GCT dataset. Third, we determine the applied machine learning algorithms for producing predictive models, which we categorize into SML, TML, and DL. For comparison purposes, we focus on the first and third elements, whilst the second element is meant to provide more information of the related works. From the prediction scope, we can conclude that most related work addresses either job or task failure prediction. Limited work has addressed both failures. With regard to the algorithms applied, we notice that most of the related work has applied TML algorithms. There are limited studies that applied DL and only one study applied SML. Our work concerns addressing both job and task failure prediction, where highly accurate predictive models are constructed and evaluated from two categories of algorithms, namely the TML and DL algorithms. Thus, compared to related works, our work is more comprehensive in determining the best predictive models, since we cover both types of failure and utilize more algorithms from TML and DL.

We then expand our review beyond the prediction of job and task failures. We are concerned with the coverage of algorithms applied in related work. The review resulting in Table 9 is also based on three key elements, similar to Table 8. The main difference is that the table contains related works that utilized a dataset other than GCT. Therefore, its failure prediction scope is not intended to train predictive models for job and task failure. However, we take them into account, since these works are still within the cloud computing area, and the applied algorithms can also be categorized into SML, TML, and DL. This review can provide additional references to interested readers in a broader context of failure prediction and also beyond the use of the GCT dataset. Although these works are not comparable to our work in relation to the targeted failure prediction problem, we can conclude that there are still limited works that comprehensively evaluate the failure

Table 8 This table presents the related works of job and task failure prediction using GCT dataset. The algorithm in bold refers to the best model mentioned in the respective article. [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. [27]	Job and Task Failure	Job Priority, Resource Requested	-	-	RNN
Soualhia et al. [28]	Task Failure	Waiting Time, Serving Time, Scheduling Class, Priority, Resource Request, Resource Usage	-	Tree, Boost, GLM, CT, **RF**	NN
Rosa et al. [29]	Job Failure	Task Priority, Resource Request, Scheduling Class, Job Size	LDA, **ELDA**, QDA	LR	-
Tan et al. [30]	Task Failure	Scheduling Class, Priority, Task Duration, Hourly Failure Frequency, Resource Usage	-	**K-Means**, Clustering	-
Islam and Manivannan [31]	Job and Task Failure	Resource Usage, Priority, Scheduling Class, Job Duration, Number Of Task Re-Submission, Scheduling Delay	-	-	LSTM
Liu et al. [32]	Job Failure	Scheduling Time, Scheduling Class, Job Size, Task Priority, Resource Request	-	SVM, OS-SVM	**ELM**, OS-ELM
El-Sayed et al. [33]	Job Failure	Job Priority, Sheduling Class Job Size, Resource Request, Resource Usage	-	RF	-
Jassas and Mahmoud [34]	Job Failure	Resource Request, Scheduling Class, Priority	-	DT, **RF**	-
Shetty et al. [35]	Task Failure	Resource Usage, Job Duration	-	XGBoost	-
Gao et al. [15]	Task Failure	Task Priority, Task Re-Submission, Scheduling Delay, Resource Usage	-	-	Bi-LSTM
Jassas and Mahmoud [36]	Job Failure	Resource Request, Scheduling Class, Priority	-	DT, RF	ANN

Table 9 This table presents the related works of other types of failure prediction using other datasets. The algorithm in bold refers to the best model mentioned in the respective article. [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. [37]	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. [38]	NERSC	Component Failure	Failed Component, Failure Time	-	LR, SVM	-
Pitakrat et al. [39]	Private Data	System Failure	Resource Usage, Failure Information, Failure Time	-	RF	-
Zhang et al. [40]	Public Dataset	Switch Failure	Message Template Sequence, Frequency, Seasonality, Surge	HSMM	**RF**, SKSVM	-
Lin et al. [41]	Private Cloud	Node Failure	Resource Usage, Group Policy, Domain Group, Rack Location	-	LR, SVM, RF	**LSTM**
Han et al. [42]	Alibaba's Cloud, Backblaze SMART	Disk Failure	SMART Log, SysLog, Trouble Ticket	-	LGBM	-
Mohammed et al. [43]	NERSC	Component Failure and Service Failure	Multiple Sources of Failure	ARIMA	LDA, CART, RF, **SVM**, KNN	-
Chen et al. [44]	Microsoft Cloud System	Outage Prediction	Storage Location, Physical Networking, Storage Streaming Component	-	SVM, PLR, Bayesian Network, **XGBoost**	-
Li et al. [13]	System X	Node Failure	Resource Usage, Group Policy, Domain Group, Rack Location	-	**RF**	LSTM
Rawat et al. [45]	Private Cloud	VM Failure	Number of VM Used	ARIMA	-	-
Yu et al. [46]	Alibaba Cloud System	DRAM Failure	System Kernel Log, MCA Data Log	-	XGBoost	-

Table 10 This table presents the related works of other kind of prediction using GCT dataset. The algorithm in bold refers to the best model mentioned in the respective article. [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. [47]	Workload Prediction	Resource Usage, Resource Request	ANFIS, **NARX**, ARIMA	-	-
Liu et al. [48]	Workload Prediction	Job Duration, Job Waiting Time, Job Status, Machine Availability	MA, **WMA**	MR	NN
Zhang et al. [49]	Workload Prediction	Resource Request	-	-	RNN
Hemmat and Hafid [50]	SLA Violation	Resource Request, Resource Usage, Resource Availability	-	DT, **RF**	-
Zhang et al. [51]	Request Prediction	Resource Request	-	-	DBN
Chen et al. [52]	Workload Prediction	Resource Usage	-	SA	RNN, LSTM, **GRU**, ESN
Gao et al. [53]	Workload Prediction	Resource Usage, Resource Request	ARIMA	**BRR**, SVR	LSTM
Di et al. [54]	Workload Prediction	Resource Usage	SMA, LWMA, EMA, AR	**Bayesian Network**	-

predictive models (i.e., using multiple types of machine learning algorithm) as presented in the table.

Furthermore, the review that resulted in Table 10 is also based on three key elements, similar to Table 8. The main similarity is that the related work is those utilized in the GCT dataset. Meanwhile, the key difference is that we broaden the scope of the prediction beyond the failure prediction perspective. This summary may provide a wider context for the use of GCTs to interested readers. As shown, we can conclude that, in addition to failure prediction, GCT has been used mainly to predict workload. We can also conclude that more work has applied multiple categories of machine learning algorithms to find the best models. Therefore,

it supports our strategy to comprehensively address the failure prediction problem with TML and DL. Furthermore, our work also contributes to the use of GCT from a failure prediction perspective with a comprehensive evaluation.

Conclusion and Future Work

In this paper, we have proposed a comprehensive evaluation approach to predict task failure and failure. For this reason, we constructed five TML models and three DL models and compared their performance in predicting job and task failure using the GCT dataset. Our performance analysis showed that the best performing model for predicting job-level failures is the XGBoost classifier, which has achieved an accuracy score of 94.35% and an F-score of 0.9310. In the case of task-level prediction, we found two best-performing models, which are based on DT and RF. Both models have obtained an accuracy score of 89.75% and an F-score of 0.9154. Overall, the results have shown that the TML models perform slightly better than the DL models in classifying job and task termination status. Furthermore, our analysis of feature importance determined that *scheduling class* and *CPU request* are the most significant features for TML, while *disk space request* and *memory request* are the most important features for DL in the context of prediction of job failure. Meanwhile, for task-level prediction, *resource requests* is the dominant one for TML while *priority* is the most important feature for DL models. Finally, our scalability analysis found that TML models can make the prediction in a reasonably short time, even though they are executed on consumer-level hardware.

There are several recommendations for future work. First, a multi-objective prediction will be useful for supporting cloud resource management. For example, the prediction of workload and failure can be addressed simultaneously when automatically deciding on resource allocation, scheduling, or provisioning. Training for this kind of predictive model can benefit from the GCT dataset. Second, the scope of the prediction can be expanded to the energy efficiency aspect. This challenge is important for cloud service providers to minimize their costs. Additionally, it also supports the Sustainable Development Goals agenda. However, training for this kind of model should use other potential datasets. Third, the application of transfer learning techniques can be explored towards producing the best quality of predictive models for the cloud resource management.

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

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.

Funding
Similar as in the acknowledgment.

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

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

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

Received: 14 May 2022 Accepted: 9 September 2022
Published online: 19 September 2022

References

1. Stein M, Campitelli V, Mezzio S (2020) Managing the Impact of Cloud Computing. CPA J N Y 90(6):20–27
2. Fortune Business Insight (2021) 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
3. Gill SS, Buyya R (2018) Failure management for reliable cloud computing: a taxonomy, model, and future directions. Comput Sci Eng 22(3):52–63
4. Press Association (2017) British Airways it failure caused by 'uncontrolled return of power'. Guardian. https://www.theguardian.com/business/2017/may/31/ba-it-shutdown-caused-by-uncontrolled-return-of-power-after-outage. Accessed 24 Jan 2022
5. Nazari Cheraghlou M, Khadem-Zadeh A, Haghparast M (2016) A survey of fault tolerance architecture in cloud computing. J Netw Comput Appl 61:81–92
6. Abdul-Rahman OA, Aida K (2014) 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. IEEE, Los Alamitos, p 272–277
7. Verma A, Pedrosa L, Korupolu MR, Oppenheimer D, Tune E, Wilkes J (2015) Large-scale cluster management at Google with Borg. In: Proceedings of the European Conference on Computer Systems (EuroSys). Association for Computing Machinery (ACM), France, p 1–17
8. Bala A, Chana I (2012) Fault tolerance-challenges, techniques and implementation in cloud computing. Int J Comput Sci Issues (IJCSI) 9(1):288
9. Shahid MA, Islam N, Alam MM, Mazliham M, Musa S (2021) Towards Resilient Method: An exhaustive survey of fault tolerance methods in the cloud computing environment. Comput Sci Rev 40:100398
10. Setlur AR, Nirmala SJ, Singh HS, Khoriya S (2020) An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud. J Parallel Distrib Comput 136:14–28

11. Kochhar D, Jabanjalin H (2017) An approach for fault tolerance in cloud computing using machine learning technique. Int J Pur Appl Math 117(22):345–351
12. Mukwevho MA, Celik T (2018) Toward a smart cloud: A review of fault-tolerance methods in cloud systems. IEEE Trans Serv Comput 14(2):589–605
13. Li Y, Jiang ZM, Li H, Hassan AE, He C, Huang R et al (2020) Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. ACM Trans Softw Eng Methodol (TOSEM) 29(2):1–24
14. Costa CH, Park Y, Rosenburg BS, Cher CY, Ryu KD (2014) 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. IEEE, Los Alamitos, p 707–718
15. Gao J, Wang H, Shen H (2020) Task failure prediction in cloud data centers using deep learning. IEEE Trans Serv Comput 15(3):1411–22
16. Bisong E (2019) An overview of google cloud platform services. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, p 7–10
17. Reiss C, Wilkes J, Hellerstein JL (2011) Google cluster-usage traces: format + schema. Mountain View, Google Inc. Revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data. Accessed 24 Jan 2022
18. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
19. Tolles J, Meurer WJ (2016) Logistic regression: relating patient characteristics to outcomes. JAMA 316(5):533–534
20. Myles AJ, Feudale RN, Liu Y, Woody NA, Brown SD (2004) An introduction to decision tree modeling. J Chemom J Chemometr Soc 18(6):275–285
21. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
22. Natekin A, Knoll A (2013) Gradient boosting machines, a tutorial. Front Neurorobotics 7:21
23. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H et al (2015) Xgboost: extreme gradient boosting. R package version 04-2 1(4):1–4
24. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O et al (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res 12:2825–2830
25. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
26. Baniecki H, Kretowicz W, Piatyszek P, Wisniewski J, Biecek P (2020) dalex: Responsible Machine Learning with Interactive Explainability and Fairness in Python. arXiv preprint arXiv:2012.14406
27. Chen X, Lu CD, Pattabiraman K (2014) Failure prediction of jobs in compute clouds: A google cluster case study. In: 2014 IEEE International Symposium on Software Reliability Engineering Workshops. IEEE, Los Alamitos, p 341–346
28. Soualhia M, Khomh F, Tahar S (2015) 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. IEEE, Los Alamitos, p 58–65
29. Rosa A, Chen LY, Binder W (2015) Predicting and mitigating jobs failures in big data clusters. In: 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, Los Alamitos, p 221–230
30. Tang H, Li Y, Jia T, Wu Z (2016) 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). IEEE, Los Alamitos, p 1–12
31. Islam T, Manivannan D (2017) Predicting application failure in cloud: A machine learning approach. In: 2017 IEEE International Conference on Cognitive Computing (ICCC). IEEE, Los Alamitos, p 24–31
32. Liu C, Han J, Shang Y, Liu C, Cheng B, Chen J (2017) Predicting of job failure in compute cloud based on online extreme learning machine: a comparative study. IEEE Access 5:9359–9368
33. El-Sayed N, Zhu H, Schroeder B (2017) 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). IEEE, Los Alamitos, p 1333–1344
34. Jassas MS, Mahmoud QH (2019) Failure characterization and prediction of scheduling jobs in google cluster traces. In: 2019 IEEE 10th GCC Conference & Exhibition (GCC). IEEE, Los Alamitos, p 1–7
35. Shetty J, Sajjan R, Shobha G (2019) Task resource usage analysis and failure prediction in cloud. In: 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). IEEE, Los Alamitos, p 342–348
36. Jassas MS, Mahmoud QH (2021) A Failure Prediction Model for Large Scale Cloud Applications using Deep Learning. In: 2021 IEEE International Systems Conference (SysCon). IEEE, Los Alamitos, p 1–8
37. Guan Q, Zhang Z, Fu S (2012) Ensemble of Bayesian predictors and decision trees for proactive failure management in cloud computing systems. J Commun 7(1):52–61
38. Adamu H, Mohammed B, Maina AB, Cullen A, Ugail H, Awan I (2017) An approach to failure prediction in a cloud based environment. In: 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, Los Alamitos, p 191–197
39. Pitakrat T, Okanović D, van Hoorn A, Grunske L (2018) Hora: Architecture-aware online failure prediction. J Syst Softw 137:669–685
40. Zhang S, Liu Y, Meng W, Luo Z, Bu J, Yang S et al (2018) Prefix: Switch failure prediction in datacenter networks. Proc ACM on Measurement and Analysis of Computing Systems 2(1):1–29
41. Lin Q, Hsieh K, Dang Y, Zhang H, Sui K, Xu Y, et al (2018) 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. Association for Computing Machinery, New York, p 480–490
42. Han S, Wu J, Xu E, He C, Lee PP, Qiang Y, et al (2019) Robust data preprocessing for machine-learning-based disk failure prediction in cloud production environments. arXiv preprint arXiv:1912.09722
43. Mohammed B, Awan I, Ugail H, Younas M (2019) Failure prediction using machine learning in a virtualised HPC system and application. Clust Comput 22(2):471–485
44. Chen Y, Yang X, Lin Q, Zhang H, Gao F, Xu Z, et al (2019) Outage prediction and diagnosis for cloud service systems. In: The World Wide Web Conference. Association for Computing Machinery, New York, p 2659–2665
45. Rawat A, Sushil R, Agarwal A, Sikander A (2021) A new approach for vm failure prediction using stochastic model in cloud. IETE J Res 67(2):165–172
46. Yu F, Xu H, Jian S, Huang C, Wang Y, Wu Z (2021) DRAM Failure Prediction in Large-Scale Data Centers. In: 2021 IEEE International Conference on Joint Cloud Computing (JCC). IEEE, Los Alamitos, p 1–8
47. Rasheduzzaman M, Islam MA, Islam T, Hossain T, Rahman RM (2014) Study of different forecasting models on Google cluster trace. In: 16th Int'l Conf. Computer and Information Technology. IEEE, Los Alamitos, p 414–419
48. Liu B, Lin Y, Chen Y (2016) Quantitative workload analysis and prediction using Google cluster traces. In: 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, Los Alamitos, p 935–940
49. Zhang W, Li B, Zhao D, Gong F, Lu Q (2016) 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). IEEE, Los Alamitos, p 104–109
50. Hemmat RA, Hafid A (2016) SLA violation prediction in cloud computing: A machine learning perspective. arXiv preprint arXiv:1611.10338
51. Zhang W, Duan P, Yang LT, Xia F, Li Z, Lu Q et al (2017) Resource requests prediction in the cloud computing environment with a deep belief network. Softw Pract Experience 47(3):473–488
52. Chen Z, Hu J, Min G, Zomaya AY, El-Ghazawi T (2019) Towards accurate prediction for high-dimensional and highly-variable cloud workloads with deep learning. IEEE Trans Parallel Distrib Syst 31(4):923–934
53. Gao J, Wang H, Shen H (2020) Machine learning based workload prediction in cloud computing. In: 2020 29th international conference on computer communications and networks (ICCCN). IEEE, Los Alamitos, p 1–9
54. Di S, Kondo D, Cirne W (2012) 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. IEEE, Los Alamitos, p 1–11

Publisher's Note