

UNIVERSITI TEKNOLOGI MARA

**EVALUATION OF CLOUD FAILURE
PREDICTION MODELS USING
TRADITIONAL MACHINE
LEARNING AND DEEP LEARNING
WITH GOOGLE CLUSTER TRACES**

TENGKU NAZMI BIN TENGKU ASMAWI

BACHELOR OF COMPUTER SCIENCE (HONS.)

FEBRUARY 2022

Universiti Teknologi MARA

**Evaluation of Cloud Failure Prediction
Models using Traditional Machine
Learning and Deep Learning with
Google Cluster Traces**

Tengku Nazmi Bin Tengku Asmawi

**Thesis submitted in fulfilment of the requirement
for Bachelor of Computer Science (Hons.)
Faculty of Computer and Mathematical Science**

February 2022

SUPERVISOR APPROVAL

EVALUATION OF CLOUD FAILURE PREDICTION MODELS USING TRADITIONAL MACHINE LEARNING AND DEEP LEARNING WITH GOOGLE CLUSTER TRACES

By

**TENGKU NAZMI BIN TENGKU ASMAWI
2019495972**

This thesis was prepared under the supervision of the project supervisor, Dr. Azlan Bin Ismail. It was submitted to the Faculty of Computer and Mathematical Science and was accepted in partial fulfilment of the requirement for the degree of Bachelor of Computer Science (Hons.)

Approved by



.....
DR. AZLAN BIN ISMAIL
PROJECT SUPERVISOR

FEBRUARY 21, 2022

STUDENT DECLARATION

I certify that this thesis and the project to which it refers is the product of my own work and that any idea or quotation from the work of other people, published or otherwise are fully acknowledge in accordance with the standard referring practice of the discipline.



.....

TENGKU NAZMI BIN TENGKU ASMAWI
2019495972

FEBRUARY 21, 2022

ACKNOWLEDGEMENT

In successfully completing this project, many people have helped me. I would like to thank all those who are related to this project.

Primarily, I would thank my supervisor, Dr, Azlan Bin Ismail, under whose guidance I learned a lot about this project. His suggestions and directions have helped in the completion of this project. Then, I would like to thank my lecturer for this proposal, Dr. Nasiroh Binti Omar and I really appreciate her valuable comments and advise on my thesis paper.

Finally, I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance and have been very helpful in various stages of project completion.

ABSTRACT

Cloud computing is the buzzword in the computing world nowadays. Due to COVID-19 pandemic, the demand for cloud services went to the roof as people seek the need to work from remote location in accordance with government's restriction. Cloud failure is one of critical issue that need to be addressed in the IT industries. A cloud failure may cost thousands, even millions for cloud service providers in addition to lost productivity as a result of inability to use computing power provided by these companies. As the system growing larger and larger every day, the process of failure mitigation and prediction are still a challenge for computer scientist everywhere. In this study, we try to embark the challenge of predicting cloud job and task failure using data from 2011 Google Trace Dataset. Eight machine learning model which are logistic regression, decision tree, random forest, gradient boosting, extreme gradient boosting, single layer LSTM, bi-layer LSTM and tri-layer LSTM will be used to generate the predictive model. On job-level failure prediction, we find that extreme gradient boosting has managed to achieve 94.35% accuracy with the F-Measure score of 0.9310. The model attained 91.92% sensitivity score and 96.07% specificity score. Logistic regression, decision tree, random forest, gradient boosting, single layer LSTM, bi-layer LSTM, and tri-layer LSTM obtained the accuracy score of 63.13%, 93.23%, 86.65%, 90.65%, 88.78%, 89.75%, and 85.15%. It is different for task-level failure prediction where both decision tree and random forest managed to achieve 89.75% accuracy and obtained the F-Measure score of 0.9145. Both models managed to attain 98% sensitivity score and 78% specificity score. Logistic regression, gradient boosting, extreme gradient boosting, single layer LSTM, bi-layer LSTM, and tri-layer LSTM obtained the accuracy score of 69.69%, 87.87%, 89.35%, 87.82%, 86.95%, and 87.55%.

TABLE OF CONTENT

CONTENT	PAGE
SUPERVISOR APROVAL	i
STUDENT DECLARATION	ii
ACKNOWLEDGEMENT	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF ABBREVIATIONS	x
CHAPTER ONE : INTRODUCTION	
1.1 Background Of the Study	1
1.1.1 Factor Contributing to Cloud Failure	1
1.1.2 Necessity For Failure Prediction	2
1.2 Problem Statement	2
1.3 Research Question	3
1.4 Research Objective	3
1.5 Research Scope	4
1.6 Research Significance	4
CHAPTER TWO : LITERATURE REVIEW	
2.1 Cloud System	6
2.2 Cloud Dataset	8
2.2.1 Google Cloud Traces	8
2.2.2 Azure Cloud Traces	10
2.2.3 Alibaba Cloud Traces	11
2.3 Failure In Cloud	12
2.3.1 Categories Of Cloud Failure	12
2.3.2 Cloud Failure Management	13
2.4 Traditional Machine Learning Algorithm	14
2.4.1 Logistic Regression	14
2.4.2 Decision Tree	15
2.4.3 Random Forest	15
2.4.4 Gradient Boosting	15
2.4.5 Extreme Gradient Boosting	16
2.5 Deep Learning Algorithm	16
2.5.1 Long-Short Term Memory	16
2.6 Synthetic Minority Over-Sampling Technique	17
2.7 Related Works	18

CHAPTER THREE : RESEARCH METHODOLOGY

3.1	Data Extraction	22
3.2	Data Understanding	24
3.3	Data Preparation	24
3.3.1	Data Cleaning	24
3.3.2	Data Integration	25
3.3.3	Data Reduction	26
3.3.4	Data Transformation	26
3.3.4.1	Data Relabelling	27
3.3.4.2	Class Balancing	27
3.4	Data Storage	28
3.5	Data Modelling	28
3.5.1	Traditional Machine Learning Model	28
3.5.2	Deep Learning Model	29
3.6	Model Evaluation	29
3.7	Summary	30

CHAPTER FOUR : ANALYSIS AND RESULT DISCUSSION

4.1	Google Cloud Platform	31
4.1.1	Experiment Setting	31
4.2	Result Of Descriptive Analysis	33
4.2.1	Job Event Table	33
4.2.2	Task Event Table	34
4.3	Result Of Model Performance	36
4.3.1	Evaluation Metrics	37
4.3.1.1	Accuracy And Error Rate	37
4.3.1.2	Sensitivity And Specificity	38
4.3.1.3	Precision	38
4.3.1.4	F-Measure	38
4.3.2	Job-Level Failure Prediction	38
4.3.2.1	Feature Importance	39
4.3.2.2	Model Evaluation	40
4.3.3	Task-Level Failure Prediction	42
4.3.3.1	Feature Importance	42
4.3.3.2	Model Evaluation	43
4.4	Summary	44

CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

5.1	Project Overview	46
5.2	Project Achievement	47
5.3	Project Limitation	47
5.4	Future Recommendation	48
5.5	Summary	48

LIST OF FIGURES

FIGURE	PAGE
2.1 Conceptual Map of Literature Review	5
2.2 Illustration of Single LSTM Cell	17
3.1 Conceptual Map of Research Methodology	22
3.2 Downloading Job Event Table Code	23
3.3 Function to Extract the Zipped File	23
3.4 File Extraction Process	23
3.5 Checking Null Values and Dropping Rows Containing Null Values	25
3.6 Data Aggregation Process	26
3.7 Data Integration Process	26
3.8 Data Reduction Process	27
3.9 Data Relabelling Process	27
3.10 SMOTE Process	27
3.11 Python Library for Traditional Machine Learning Model	29
3.12 Single Layer LSTM Model	29
4.1 Virtual Machine Configuration	32
4.2 Distribution of Job Scheduling Class	33
4.3 Job Termination Status Categorized by Scheduling Class	34
4.4 Distribution of Task Priority	35
4.5 Termination Status Categorized by Task Priority	35
4.6 Boxplot of Resource Request Distribution in Task Event Table	36
4.7 Feature Importance by Machine Learning Algorithm	39
4.8 Feature Importance by Machine Learning Algorithm	42

LIST OF TABLES

TABLE	PAGE
2.1 Classification of Failure and Their Cause	12
2.2 Summary of Related Works for Failure Prediction	20
4.1 Training Performance Evaluation	40
4.2 Testing Performance Evaluation	40
4.3 Training Performance Evaluation	43
4.4 Testing Performance Evaluation	43

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AUC	Area Under Curve
CloudSim	Cloud Simulator
CPU	Central Processing Unit
CSP	Cloud Service Provider
CSV	Comma Separated Values
FN	False Negative
FP	False Positive
HPC	High-Performing Computing
GB	Gigabyte
GiB	Gibibyte
GCP	Google Cloud Platform
GPU	Graphics Processing Unit
IaaS	Infrastructure as a Service
ID	Identification
I/O	Input Output
IP	Internet Protocol
IT	Information Technology
kB	kilobyte
LSTM	Long-Short Term Memory
MB	Megabyte
NIST	National Institute of Standard and Technology
PaaS	Platform as a Service
QoS	Quality of Service
SaaS	Software as a Service
SMOTE	Synthetic Minority Over-Sampling Technique
TN	True Negative

TP	True Positive
VM	Virtual Machine
XGBoost	Extreme Gradient Boosting

CHAPTER ONE

INTRODUCTION

This chapter provides the background and rationale for the study. It also gives details of the significance of cloud computing, the issues and problems that led to this research.

1.1 Background Of the Study

A decade ago, cloud computing is not a common term even in computing world. Now it is one of the solutions for expanding storage space and hosting modern applications and websites in virtual machine, transform the IT field. Cloud computing also allow business provides extra layer of security in term of information security and allowing them to rise level of efficiency of operation to a new level. Thanks to the current epidemic that has massively impact worldwide economies, the demand for cloud infrastructure increases as most of the workforce is working from home due to lockdown restriction (Alashhab et al., 2021). According to Gartner. Inc, North America market has approximately spent a whopping \$78.28 billion for cloud services. The market for cloud computing is expected to keep expanding from \$219B in 2020 to \$791.48B in 2028 (*Cloud Computing Market Size & Share | Industry Growth [2020-2027]*, n.d.)

1.1.1 Factor Contributing to Cloud Failure

Cloud user is allowed to run intensive task on computing resources provided over the network that is provided by cloud service provide (CSP). CSP on the other hand need to allow millions of users to access the same pool of computing resources (Alam, 2020). Due to the complexity of cloud operation, cloud failure in inevitable.

Any failure can negatively affect the availability of resources and may cause a domino effect that brought down the whole cloud service (*Amazon Web Services Experiences Another Big Outage - The Washington Post*, n.d.).

1.1.2 Necessity For Failure Prediction

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 (TheGuardian, 2017). Failure prediction in cloud infrastructure can be useful on improving the reliability of cloud infrastructure since any failure can affect multiple users from any region. Failure prediction in general is a very important aspect of predictive maintenance due to its ability to prevent failure occurrence and maintenance cost.

1.2 Problem Statement

There have been multiple studies of cloud failure prediction from the concept of application failure, scheduling failure and task failure. Application failure in cloud environments is a failure that is caused by cascading failure resulting in non-responsive cloud application. Research in this area would be focusing on task failure prediction focusing on job that failed multipled task in it running duration (Islam & Manivannan, 2017). Scheduling failure is a failure occurred when the allocated resources to task exceed the currently available resources. (Soualhia et al., 2015a) has done research in predicting scheduling failure by analysing the resource requested by a task and the utilization of the requested resources alongside the service time of an individual task. Lastly, a general job and task failure research would be focusing on the property of the task of job itself; such as their priority and scheduling class (Gao et al., 2020a). These studies have utilised either dataset that available publicly or using simulated cloud environment for their dataset, and each of these studies has been focusing on specific machine learning and deep learning algorithm.

A 2019 study conducted by (Jassas & Mahmoud, 2019), has conclude that both Decision Tree Classifier and Random Forest Classifier both has managed to get the highest model accuracy for job failure description which are at 98% and 99% respectively. (Gao et al., 2020a) on the other hand has develop a prediction model using multi-layer Bidirectional LSTM (Bi-LSTM) to predict the termination status of job and task in Google Cluster Traces. Their result show that the model managed to achieve 93% accuracy for task failure prediction and 87% accuracy for job failure prediction. This research will be the first research that will be utilizing and comparing both deep learning and traditional machine learning model performance in predicting job and task termination status. Thus, we able to compare performance of difference model in predicting the failure status on cloud job and task based on the sample data of 2011 Google Cloud Traces.

1.3 Research Question

- a) What is the contributing factor for job and task failure in Google Traces Dataset?
- b) How to design and build failure prediction model using Google Cloud Traces Dataset?
- c) What is the accuracy of all models when applied on the sample dataset of Google Cloud Traces?

1.4 Research Objective

This study aims to construct an algorithm which can analyse Google Traces Dataset using machine learning approach. There are three objectives to achieve the aim.

- a. To clean and prepare Google Traces Dataset published in 2011 for failure prediction modelling
- b. To design and develop failure prediction models using traditional machine learning and deep learning algorithm

- c. To evaluate and compare the prediction models performance and their feature importance on predicting failure in cloud

1.5 Research Scope

The scope of the study will focus on predicting job and task failure in 2011 dataset by generating predictive model from machine learning algorithm and one deep learning algorithm. The sample dataset is taken from the first seven days of data from both datasets. Algorithms involve for building a failure prediction model are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting, XGBoost and LSTM

1.6 Research Significance

The significance of the study can be identified as follows:

- a) This study will provide insight on the performance of six algorithm which are Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, XGBoost and LTSM on failure prediction by comparing the result using 2011 dataset.
- b) This study can also contribute to research work in improving reliability in cloud computing. One of the Quality of Services (QoS) issues in the system reliability which are important indicator of good cloud environment. The occurrence of failure will affect the system reliability.
- c) Most business organization will be using cloud infrastructure to scale up their business. Any failure in cloud may interrupt the growth of the business and cause huge loss of revenue. Other than that, failure in cloud can cause customer loss trust toward the business which will increase revenue loss. Hence the reliability of the system must be guaranteed for the user.

CHAPTER TWO

LITERATURE REVIEW

In this chapter, the concept of cloud computing in general will be explained further in term of its characteristics, cloud type and service model offered to its users for a better understanding on how the cloud service works. Next, public cloud dataset available for analysis and academic studies will be dissected in this chapter for better understanding. Next, cloud failure concept will also be explained in this chapter alongside the ways of cloud failure management. Lastly, related works on failure prediction in cloud involving Google Traces Dataset and other dataset will also be discussed as a reference to support this study. Overall flow for this chapter is illustrated in Figure 2.1 below.

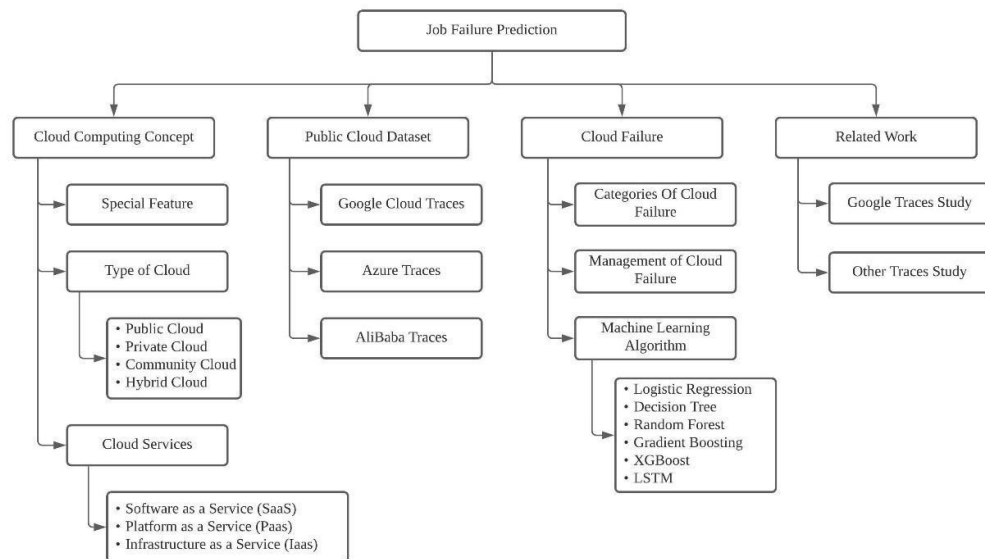


Figure 2.1 Conceptual Map of Literature Review

2.1 Cloud System

Cloud computing term is first coined by Professor Ramnath Chellappa in 1997 (Ray, 2017). Since then, the phrase has been perceived in a variety of ways by various organisations and IT firms. Despite its widespread use, there is no commonly accepted definition regarding the term. However, NIST define the concept of cloud computing is a model for an on-demand access to a shared pool of computing resources that can be rapidly acquired with minimal management or service provider interaction (Mell & Grance, 2011). In the last decade, cloud computing has evolved far from the original concept. NIST has described that cloud computing had five essential characteristics, three service models and four deployment models (Mell & Grance, 2011). The special characteristics of cloud computing are as classified below:

- a) On-demand self-service: A user can acquire and utilise computing resources such as server time or storage space without interaction to any human operator or service provider.
- b) Broad network access: The computing resources is available over the network and can be utilized by diverse set of gadgets such as laptop and smartphone.
- c) Pooling of resources: The computing resources can be accessed by multiple party by using multi-tenant model. The server that is rented by the user might be shared with other user without their knowledge. In addition, the customer has no knowledge or control on where the data is stored in the physical location but they able to specify location at higher level such as country, state, or data centre.
- d) Rapid Elasticity: The computing resources can be scale toward the user demand. The computing capabilities may appear unlimited to the user and the server must provide adequate number of resources accordingly at any time based on the user needs.
- e) Measured Serviced: Cloud system able to optimise the available computing resources by the type of service offered. Service provider may limit the utilization of resources if the service provider feels the need to do it. Resource usage also can be monitored, controlled, and reported by the service provider.

Cloud become immensely popular among organization whether it is small, medium, or large. An organization can only benefited from a specific type of cloud for their usage or need (Goyal, 2014). The deployment models are classified as below:

- a) Public Cloud: Public cloud is an infrastructure made available for public usage that is usually owned by an organization or large IT companies such as Google or Amazon. The resources may be provided for free or on a pay-per-use basis. The infrastructure is typically found at the service provider's site.
- b) Private Cloud: Private cloud is used by an organization and can only be accessed by the member of the organization. To limit the number of users who can access the cloud, the service usually operated behind a set of firewalls. The physical server could be on or off the site of the organization, and it can be managed by the organization or by a third party. Furthermore, unlike public cloud system, a private cloud could have greater cloud protection by air gapping the server from other servers.
- c) Community Cloud: Community cloud usually utilised by a group of users who may be belongs to different organization who has the same goal or needs such as data organization and data management. The resources may be managed by one organization, a group of organization or a third party.
- d) Hybrid Cloud: Hybrid cloud is an infrastructure comprise of two or more cloud models that exist as a separate entity but are linked through standardised or proprietary technology that would allow data and application portability.

In cloud computing, there are three services model can be used by cloud system: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) (Mell & Grance, 2011). The service models for cloud system are classified as below:

- a. Software as a Service (SaaS) is a model where service provider has a capability for the client to use an application running on cloud system. The application can be accessed by the client through various device or through an interface, such as web browser. The client has no control over the cloud infrastructure in this model.

- b. Unlike SaaS, Platform as a Service (PaaS) is a model where client is capable to deploy application or software to the cloud system. This software is limited to the capabilities of the machine provided by the service provider. The client may need to use programming language and tools provided by the service provider to build the software. However, the client has control over the software deployment and configuration.
- c. Infrastructure as a Service (IaaS) is a service model where the service provider has the capabilities to provide computing resources where to client able to run software can deploy and run a software. The service provider only provides and control the infrastructures, but the client has control over operating system, storage and deployed application and limited to the network properties of the infrastructures.

Each service model has its own set of benefits and drawbacks. If an organization decide to use cloud services, it may have to compromise on certain aspect of the cloud system. In essence, the IaaS service model provides the most flexibility for the customer since the provider only controls the infrastructure and networking information, while the rest is under control of the customer. SaaS service model is the most rigid structure as the organization only has the influence over the data they own. In comparison with SaaS, PaaS give user higher advantages since they oversee both data management and software deployment.

2.2 Cloud Dataset

This section will explain all the traces dataset available publicly available online for research and academic use. It will include Google Cloud Trace, Azure VM Traces and Alibaba Production Cluster Traces.

2.2.1 Google Cloud Traces

Up-to-date, Google has release three public traces dataset for researcher and academic usage. The first one is release in 2007 containing 7-hour workload details. The dataset only contains the basic information, which is the time, job ID, Task ID,

the category of job, number of cores and memory. Both the cores and the memory count has been normalised.

The second dataset is containing traces contain 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 task which has been submitted by the user (Abdul-Rahman & Aida, 2015). Like the previous dataset, this dataset contains trace of the resource usage of CPU, memory, and disk space for 5 minutes.

Machine details is described in two table which is machines events and machine attributes. Each machine is identified by a unique 64-bit identifier. Every event regarding machine such as adding and removing machine in the cluster and updating machine details is recorded in the machine event table. Each record has its own timestamp indicating when the event occurs. In the table, there is also the information on the machine platform representing the microarchitecture and chipset version of the machine, and the normalised value of the machine CPU and memory capacity. For machine attributes table, it contains the details about machine property such as kernel version, machine operation clock speed, and the external IP address that tied to the machine.

For jobs and task, there are four table describing them, which are job event table, task event table, task constraint table and resource usage table. Job event and task event table used to describe the lifecycle of the job of 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 this table. The type of event is identified by their event type where value zero indicating the job or task submitted by the user, value one indicating the task has been scheduled to run on a machine, value two to six indicating the task has been terminated while value seven and eight indicating the task details or requirement has been updated. Alongside the event details, there is also a column indicating the task or jobs has event has been synthesised as a replacement for a missing record.

Alongside all the columns mentioned above, there is a column for scheduling class in both tables. Scheduling class is indicated by a single integer valued from zero (0) to three (3) where zero indicating a non-production task while three indicating latency-sensitive task. Lastly, there is one column for username and two

more columns for job name. These columns data have been anonymised by combining data from several internal name field.

For task event table, in addition of all the mentioned columns, there is an addition of six different columns. One of the columns is machine ID which remarking which machine the task is ran. The second columns are the task priority which valued from zero (0) as the least priority to eleven (11) as the most important task. In general, there is three level for task priority; “free” for lowest priority, “production” for highest priority, and “monitoring” for task that is constructed to monitor other task. These three priority ranges are normalised to the value mentioned above. Next, there is three columns detailing the normalised amount of resources requested by each task. Lastly, there is a column attributing the constraint of running in a different machine where if there is value in the column, remarking the task must be executed on a different machine than the machine that currently running different task from the same job.

Next table is task constraint table. This table disclose the task constraint. A task may contain zero or many task constraint. Task constrain inhibit the task from running in a certain machine. Each rows representing exactly one task event record. Lastly, there is resource usage table. This table disclose the amount of computing resources that has been utilized by every task. This usage is recorded in a five minute or 300 seconds measurement period. In this table, cache amount, disk I/O time is recorded alongside the computing resources such as CPU, memory, and disk space.

2.2.2 Azure Cloud Traces

Microsoft also release their own cloud traces collected tin 2017 and 2019 (*AzurePublicDataset/Azure 2019 Public Dataset V2 - Trace Analysis.Ipynb at Master · Azure/AzurePublicDataset · GitHub*, n.d.; *AzurePublicDataset/Azure Public Dataset - Trace Analysis.Ipynb at Master · Azure/AzurePublicDataset · GitHub*, n.d.). Both traces contain record from 30 consecutive days. Both traces contain only two table with the same format, which are the VM table and the deployment details table. For the VM table, it details each user subscription and deployment ID, the start and end time of Virtual Machine (VM) deployment time, and CPU usage. Each VM

is identified by its unique ID assigned during the creation of the VM. Also recorded in the traces is the number of CPU core and memory assigned to each VM. For CPU usage, the maximum, average and the 95th percentile usage is also recorded in the traces. Meanwhile for the deployment table, there is only two columns which are the deployment ID and deployment size.

2.2.3 Alibaba Cloud Traces

In 2017, Alibaba provided trace data taken from a production cluster of arounds 1300 machines within 12-hours period (Ding, 2017). The trace tracks machine and instances usage taken every 60 seconds for over 300 seconds. The dataset contains six tables which is machine events, machine utilization, task table, instance table, service instance event and service instance usage. Machine event table recorded the any event such as timestamp of the process of adding and removing machine in the cluster alongside computing resources for that particular machine. Machine utilization shows the total resource utilization for a period of 300 seconds.

Task table recorded the timestamp of the task creation time, the job and task ID, the number of instances or thread and the amount of resource requested. For instances table however shows the computing resources usage for each thread alongside with the number of times the same thread has been ran for task completion. Last two table is service instance event and service instance usage. Service instance event recorded online task which task that is created by outside user that is ran on the machine available in the Alibaba cloud cluster. Like task table, it also recorded the creation time, job and task ID, number of threads, and the amount of resource requested. Service instance usage table on the other hand mirror the instance usage table showing the resources utilization. In 2018, Alibaba once again provided trace data from 4000 machines spanning 8 days running time. Like 2017 datasets, 2018 dataset contain six table which is machine events, machine utilization, task table, instance table, service instance event and service instance usage.

2.3 Failure In Cloud

In this section, we will explain the categories of cloud failure, and cloud failure management method that is currently being utilised.

2.3.1 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 failure. More complex the computing system, the higher the chance of the system to fail. There are four type of cloud computing failure which is service failure, resource failure, correlated failure, and independent failure (Gill & Buyya, 2020). These failure categories are summarised in Table 2.1. Service failure usually happen due to software failure such as unplanned reboot and cyber-attacks and scheduling failure such as service time timeout. For our research we will be focusing on scheduling failure. The second failure type is resource failure. Resource failure cause by hardware failure such us power outage, system breakdown, memory problem and complex circuit design. Both service and resource failure can be classified as architecture-based failure.

Table 2.1 Classification of Failure and Their Cause

Type of Failure	Classification	Cause of Failure
Service Failure	Architecture Based	<ul style="list-style-type: none">• Software Failure<ul style="list-style-type: none">➤ Complex Design➤ Software update➤ Planned reboot➤ Unplanned reboot➤ Cyber attacks• Scheduling<ul style="list-style-type: none">➤ Timeout➤ Overflow
Resource Failure		<ul style="list-style-type: none">• Hardware Failure<ul style="list-style-type: none">➤ Complex circuit design➤ Memory

Table 2.1 (Continued)

		<ul style="list-style-type: none"> ➤ RAID Controller ➤ Dis Drive ➤ Network Device ➤ System Breakdown
Correlated Failure	Occurrence based	<ul style="list-style-type: none"> • Based on Spatial Correlation Between Two Failure • Based on Temporal Correlation Between Two Failure
Independent Failure		<ul style="list-style-type: none"> • Denser System Packing • Human Error • Heat Issue

Sources: Gill & Buyya, 2020

Correlated failure is the next form of failure. It is known as a correlated failure when two or more failures occurs at the same time and the failures can be linked to one another. Correlated failure can be classified into two categories: temporal correlation and spatial correlation. Temporal correlation failure occurs when the time gap between consecutive failures is within a certain threshold. Spatial Correlation failure happen when failure happen when the failed job/task share the same physical resources (Ghiasvand et al., 2015). Finally, the last failure type is independent failure. Independent failure usually a failure related to the physical component of the machine/server. An example of independent failure are human error and heat issues affecting the performance of the computing power. Correlated failure and independent failure are classified as occurrence-based failure.

2.3.2 Cloud Failure Management

Despite the advancement in technology, the performance of cloud still hampered by their vulnerabilities to failure. Therefore, failure management or fault tolerance is one of the fundamentals requirements of cloud computing. There are two major tolerance approaches that is currently being implemented in cloud computing (Bala & Chana, 2012). The first approaches are reactive fault tolerance (Abdelfattah et al., 2018; Asghar & Nazir, 2021). Reactive fault tolerance approach reduces the

effect of failure on the execution of an application. One of the techniques of reactive fault tolerance is task resubmission where a failed job is detected in the system, the same job will be rerun either on the same or different resources.

The second approach is proactive fault tolerance. Proactive fault tolerance is implemented with the principle of prevent failure from occurring altogether. For proactive fault tolerance approach, the condition of the physical system constantly monitored and there is a need to predict the occurrence of the failure to the system. If the probabilities of fault occurring is high, CSP will takes pre-emptive measure such as removing hardware from the service cluster or performing corrective measure on the software. One example of proactive fault tolerance is a research done by (Li et al., 2020) where a machine learning model is being implemented in a cloud farm in order to detect a failing hardware. Another example of proactive fault tolerance is studied by (Gao et al., 2020) where a machine learning algorithm is constructed to predict the workload of cloud system at one time. with the prediction model, the system able to efficiently schedule task in accordance to resource availability and the resource requested.

Failure prediction is the process of using predictive modelling to predict cloud services downtime. These models are built from 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 as a proactive fault tolerance approach if it is implemented in cloud infrastructure.

2.4 Traditional Machine Learning Algorithm

This section will explain some of the machine learning algorithm that is suitable for usage on Google Cloud traces and will be implemented in this study.

2.4.1 Logistic Regression

Logistic regression is a machine learning model used to estimate the likelihood of an event occurring. It is a technique that is borrowed from the field of statistic. Logistic regression works with binary data, where either the event happens

or not happen (Tolles & Meurer, 2016). Logistic regression is often use in the healthcare sector in performing diagnosis on a patient based on their risk factor. So far, only (Lin et al., 2018) that has use logistic regression as one of the machine learning model for cloud failure prediction.

2.4.2 Decision Tree

Decision tree is one of the predictive modelling approaches used in data mining and machine learning. Decision tree often represented as a tree where the leaves is representing the outcome of the prediction and the branches is representing the condition of a feature to determines the tree flow from the root to the leaves (Myles et al., 2004). Decision tree calculates the outcome of a prediction by observing the flow of the hypothetical tree from its root to the one of the leaves. Decision trees model can perform both classification and regression analysis. For cloud failure prediction, Decision tree is one of the popular algorithms as the training time for decision tree is shorter compared to the other prediction model.

2.4.3 Random Forest

Random Forest is an ensemble learning method that is commonly used for classification and regression analysis. Random forest model built multiple decision tree and calculated the mean score of the decision tree outcome to produce it outcome (Breiman, 2001). Depending on the complexity of the data, the difference of performance between decision tree and random forest is either non-existence or random forest performs slightly better on a more complex data. Research using random forest usually compared the score with decision tree as they built on the same base (Jassas & Mahmoud, 2019; Soualhia et al., 2015b).

2.4.4 Gradient Boosting

Gradient boosting is a machine learning approach that creates a prediction model in the form of an ensemble of weak prediction models, generally decision

trees, for regression and classification tasks (Natekin & Knoll, 2013). Gradient boosting is one built based on AdaBoost or Adaptive Boosting. Unlike random forest, gradient boosting will build one tree at a time to prevent overfitting and shorter tree will be prioritize compared to longer tree.

2.4.5 Extreme Gradient Boosting

Extreme Gradient Boosting or XGBoost is an improvised version of gradient boosting (T. Chen et al., 2018). The difference between gradient boosting and XGBoost is the method of regularization technique implemented to prevent overfitting. XGBoost is usually fast compared to gradient boosting as it can perform parallelization in calculating the result of a single tree. Compared to other traditional machine learning method, XGBoost is the only algorithm than can harness the all the computing power in a modern multicore computer.

2.5 Deep Learning Algorithm

This section will explain deep learning algorithm that is suitable for usage on Google Cloud traces and will be implemented in this study.

2.5.1 Long-Short Term Memory

Long-short term memory (LSTM) is a deep learning algorithm that is modelled based on recurrent neural network. LSTM is a recurrent neural network model that not only study one point of the data, but also an entire sequence of data. Due to this LSTM is extremely popular in predicting a timeseries data such as speech recognition and handwriting recognition. LSTM also popular model used to detect an anomaly in a continuously running system. Compared to other recurrent neural network algorithm, LSTM can work with most modern system as it still can make prediction despite the inconsistent gap between the records. A common LSTM cell composed of three elements; input gate, output gate and forget gate. Figure 2.2 shows an example of LSTM cell.

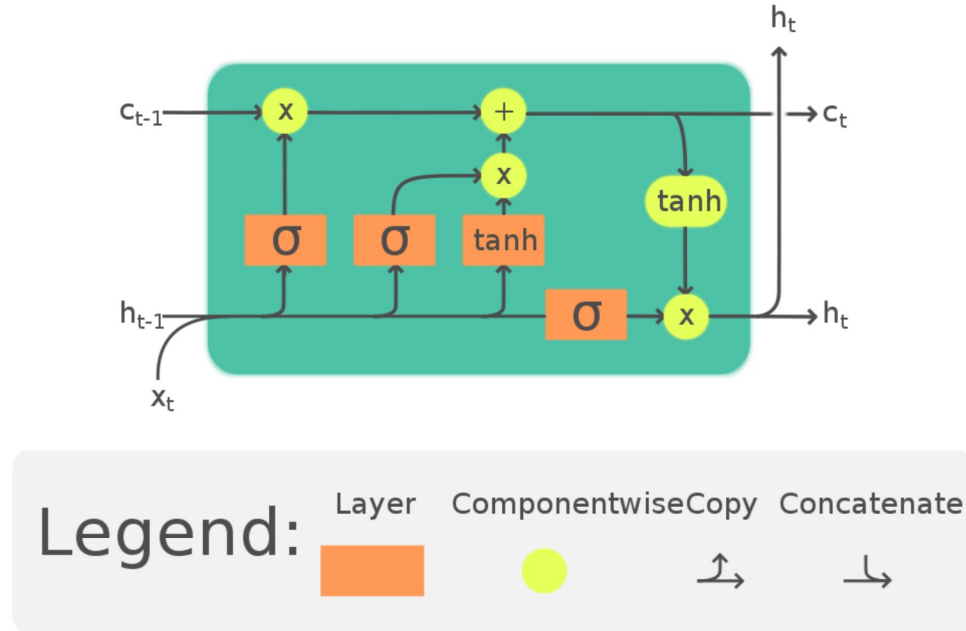


Figure 2.2 Illustration of Single LSTM Cell

The key to the LSTM is the cell state which is represented by the horizontal line that is run through the cell at the top of the diagram. The information on the cell state can be change though the change is extremely regulated by the gates. Gates usually composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer usually output a float valued between zero and one describing how much component will be let through where zero indicates do not let anything added to cell state while one implies letting every through the gate.

The first gate can be seen in the diagram is the forget gate which is represented by leftmost vertical lines on the diagram. Forget gate is a gate that will determining how much information will be remove from the cell state. The next gate we encounter in the cell is input gate. Input gates will be determining what information will be added or updated in the cell state. Last gate is the output gate. The output of the cell will be a filtered version of our cell state.

2.6 Synthetic Minority Over-Sampling Technique

Synthetic Minority Over-sampling Technique or SMOTE is an oversampling technique used generates synthetic data for the minority class in order to balance the

class in an imbalance dataset (Chawla et al., 2002). SMOTE works by selecting a random record from the minority class and finding its nearest neighbour. Then the synthetic data is generated based on the lines between the random record that has been selected and its nearest neighbour. This procedure is then repeated until the amount of minority class data matched the amount of majority class data.

2.7 Related Works

Over the past year, there are several failure predictions studies that have been conducted. Table 2.2 shows the summary of selected studies that utilized the Google Cluster Traces dataset alongside several research using other datasets. For Google Cloud Traces dataset, the focused table is jobs event, task event and task usage. Other than that, the termination status of a job or task will be sorted into two groups which are jobs or tasks that finished normally and jobs or tasks that failed to finish their intended duty. The variable that is highly correlated with job or task termination status is resource usage and the amount of resource requested. Due to time and computing power constraints, some researchers opt to use a small subset of data sampled from the overall dataset. Table 2.2 shows the summary of past studies for cloud failure prediction.

There is various prediction level that is being used by these studies. Four general prediction levels for cloud failure prediction are Message Log Pattern, Job Failure, Task Failure and Scheduling Failure. (X. Chen et al., 2014), and (El-Sayed et al., 2017) build on a model focused on jobs failure prediction, while (Rosa et al., 2015), (Shetty et al., 2019) and (Gao et al., 2020a) is more focused on task failure prediction. For scheduling failure prediction, (Soualhia et al., 2015b) model was built to predict on task scheduling failure, meanwhile (Jassas & Mahmoud, 2019) model was built based on the analysis of jobs scheduling failure. (Islam & Manivannan, 2017) is the only study in table 2.2 that built a model for application failure. There are no studies for Message Log Failure cause Google Traces does not include any message in the dataset.

For predictive analytic, there are two algorithms which are classification model and regression model. These models are used depending on the feature selected for

training and type of predictive task. For failure prediction based on time-series as demonstrated by (Gao et al., 2020a) and (Rasheduzzaman et al., 2014), LSTM was used as it is special kind of recurrent neural network that is capable of learning long term dependencies in data. The most common model which is used by these studies are classification model as it is a model that useful for predicting discrete classes which are failed job and successful job for these studies.

(Jassas & Mahmoud, 2019) has initially develop a DT classification algorithm which can predict the job status with the accuracy of 98%. However, after refining the feature selection approaches, they manage to achieve a higher accuracy of 99% using Random Forest classification algorithm and with the F1-Score of 99% which is the best job failure prediction. For task failure prediction, (Tang et al., 2016) has develop K-HUNTER, a KNN based prediction model for detecting task failure. The study used three static feature which is mean CPU, memory, and disk usage. They also found out that the value of K which is the number of neighbours does not play a significant role for the prediction. The accuracy of K-HUNTER is on average of 99% which is the best for task failure prediction.

For time series prediction, LSTM model which was used by multiple studies has proven it capabilities on predicting whether task or job failure with a high accuracy level. (Rasheduzzaman et al., 2014) is the first studies to our knowledge that used LSTM as deep learning and managed to predict job and task failure with the accuracy level of 87% and 81% respectively. (Gao et al., 2020a) further improve the performance of LSTM by building a model based on multi-layer Bidirectional LSTM for job and task failure and managed to achieve the accuracy level of 87% and 93% respectively.

Among the feature used to construct a predictive model are resource usage, resource requested, scheduling class and priority, and other miscellaneous feature such as job or task duration, scheduling delay and task resubmissions. (X. Chen et al., 2014), (Rosa et al., 2015), (Jassas & Mahmoud, 2019) and (Gao et al., 2020a) has select amount of computing resources requested alongside job or task priority and scheduling class as the input vector for their prediction model. Other has included the resource usage and other miscellaneous as an addition to the input vector that being fed to their model for training.

Table 2.2 Summary of Related Works for Failure Prediction

Author & Year	Datasets			Method Used in The Study	Result		
	Google Trace Dataset	Other Dataset	Feature Studied		Job Failure Prediction Accuracy (%)	Task Failure Prediction Accuracy (%)	Node Failure Prediction Accuracy (%)
(X. Chen et al., 2014)	/		Job Priority, Resource Requested	Recurrent Neural Network, Ensemble Method	80	84	N/A
(Soualhia et al., 2015b)			Waiting Time, Service Time, Scheduling Class, Priority, Resources Requested, Resources Usage	Tree	N/A	70.8	N/A
				Boost	N/A	95.8	N/A
				General Linear Model	N/A	50	N/A
		/		Conditional Tree	N/A	88.6	N/A
				Random Forest	N/A	74	N/A
(Rosa et al., 2015)			Task Priority, Requested Resources, Scheduling Class, Job Size	Neural Network	N/A	87.4	N/A
				Linear Discriminant Analysis	63	N/A	N/A
	/			Linear Discriminant Analysis on an expanded basis	72	N/A	N/A
				Quadratic Discriminant Analysis	68	N/A	N/A
				Logistic Regression	72	N/A	N/A
(Tang et al., 2016)	/		Scheduling class, Priority, Task Duration, Hourly Failure Frequency, Resource Usage	K-Nearest Neighbour	N/A	99	N/A
(El-Sayed et al., 2017)	/		Job Priority, Scheduling class, Number of Task per Job, Resource Requested, Resource Usage	Clustering Analysis	N/A	97	N/A
(Rasheduzzaman et al., 2014)	/		Resource Usage, Job/Task Priority, Scheduling Class, Job Duration, Task Resubmission, Scheduling Delay	Random Forest	95	N/A	N/A
(Lin et al., 2018)			Resource Usage, Node's Status, Group Policy, Domain Group, Rack Location	Long Short-Term Memory (LSTM)	81	87	N/A
				MING	N/A	N/A	*92.4
				Logistic Regression	N/A	N/A	*69.4
		/		Support Vector Machine	N/A	N/A	*66.0
				Random Forest	N/A	N/A	*76.2
(Jassas & Mahmoud, 2019)	/		Resource Requested, Priority, Scheduling Class	LSTM	N/A	N/A	*72.2
				Decision Tree	98	N/A	N/A
				Random Forest	99	N/A	N/A
(Shetty et al., 2019)	/		Resource Usage, Job Duration	XGBoost	N/A	*92	N/A
(Gao et al., 2020a)	/		Task Priority, Task Resubmissions, Scheduling Delay, Resource Usage	Bi-LTSM	87	93	N/A
(Li et al., 2020)		/	Resource Usage, Node's Status, Group Policy, Domain Group, Rack Location	MING	N/A	N/A	**90
				LSTM	N/A	N/A	**90
				Random Forest	N/A	N/A	**92

* Precision Score ** AUC Score

The research on prediction of failure on cloud computing has not only been conducted on Google Cloud Trace dataset, but there were a lot of studies conducted on failure prediction on other cloud services such as Amazon Cloud, HPC, and many more. (Soualhia et al., 2015b) build a prediction model of task failure using multiple machine learning algorithm which are Tree, Boost, General Linear Model, Conditional Tree, Random Forest and Neural Network. This study is different from previously mentioned study as this study used different dataset for training phase and test phase, where Google CloudSim Data were used as training data and the model is implemented in Hadoop in Amazon EMR. From all the model, Neural Network is the best model with the highest accuracy of 72.8%, precision rate of 97.2% and recall rate of 72.7%. Lastly, (Lin et al., 2018) has develop MING, a combination of LSTM and Random Forest. MING has successfully identified 92.4% of failed node or hardware. (Li et al., 2020) further continue the research on MING by implementing in real-world system over a period of six month. Due to non-disclosure agreement, only the AUC score is published. In this research, MING managed to obtain the AUC score of 90%. Random Forest however managed to outperform MING by obtaining the AUC score of 92%.

CHAPTER THREE

RESEARCH METHODOLOGY

This section provides a brief introduction on the theoretical framework of this study and presents the machine learning techniques which will be used to design the failure prediction models.

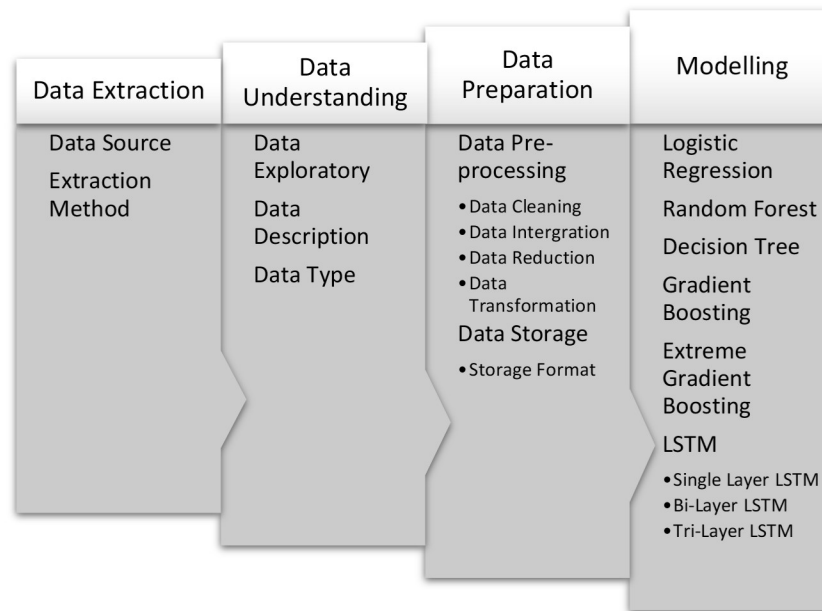


Figure 3.1 Conceptual Map of Research Methodology

3.1 Data Extraction

For this study, the sources of data are retrieved from Google Cloud Storage (GCS). The information about the dataset is available at the website <https://github.com/google/cluster-data> for all version of google cluster dataset. We downloaded the 2011 version of the dataset with the size of the dataset approximately around 400 GiB. Only two table needed to be downloaded for this experiment which are job events and task event. Urllib.request package is used to access the data stored in Google Cloud Storage.

```

print('Download Started')
#Job Events Table
jEvent1 = 'https://commondatastorage.googleapis.com/clusterdata-2011-2/job_events/part-'
jEvent2 = '-of-00500.csv.gz'
for x in range(500):
    while True:
        try:
            url = jEvent1+("%05d" % x)+jEvent2
            fname = 'data/' + 'job_events/part-' + ("%05d" % x) + '-of-00500.csv.gz'
            data = urlopen(url).read()
            with open(fname, 'wb') as f:
                f.write(data)
        except:
            continue
        break
print('Download Completed')

```

Figure 3.2 Downloading Job Event Table Code

After all the data needed has been downloaded, we need to extract the data using gzip package as all the file is stored in a zipped manner. A function is built to read save the newly extracted data into our director. After extracting all the data, all the original zipped file will be deleted to save storage space.

```

def gunzip(source_filepath, dest_filepath, block_size=65536):
    with gzip.open(source_filepath, 'rb') as s_file, \
        open(dest_filepath, 'wb') as d_file:
        while True:
            block = s_file.read(block_size)
            if not block:
                break
            else:
                d_file.write(block)

```

Figure 3.3 Function to Extract the Zipped File

```

path1 = 'data/job_events/part-'
path2 = '-of-00500.csv.gz'
path3 = '-of-00500.csv'
print('Extraction Started')
for x in range(500):
    spath = path1 + ("%05d" % x) + path2
    dpath = path1 + ("%05d" % x) + path3
    gunzip(spath, dpath)
    os.remove(spath)
print('Extraction Finished')

```

Figure 3.4 File Extraction Process

3.2 Data Understanding

The second stage of this research is data understanding. Based on the schema provided by Google, the dataset contains six (6) separate table which are job event, task event, machine event, machine attribute, task constraint and task usage. Every job and machine are given a 64-bit unique identifier. For task however, it is identified by their job ID and index. Every task belongs to one job, while one job may have one or many tasks. Machine description are place in two (2) tables, which are machine attribute and machine events. Machine events contain the information about the status of the machine, along with the capacity of its CPU and memory. Machine attributes contain data representing machine properties, such as kernel version and clock speed. We will do our own data exploratory to analyse the dataset. the finding of this process will be explained in Chapter 4.

3.3 Data Preparation

The next step in this experiment is data preparation. This stage involves several simple processes which are data cleaning, data integration, data reduction and data transformation. All of these steps are commonly known as data pre-processing. Data pre-processing is done to make raw data understandable and useful for our usage. In this step, we will be using Dask library instead of Pandas for data processing as Dask able to parallelize all of it task. All the details about data preparation will be explained in the following subsection.

3.3.1 Data Cleaning

The first step for data pre-processing is data cleaning. Data cleaning is a process of removing or replacing outlier, missing value, and correcting inconsistent data. There is two type of data we will be focusing on this step which are missing data and noisy data.

There are several known methods on how to deal with missing data in a dataset. we can either remove the row containing missing data from the dataset or we can replace the missing data value to a specified value. For this purpose, we need to identify whether our data contains any missing value. This can be done easily in python by using the is null function from pandas' library. If there are missing value in the dataset, we can remove it as there is a large amount of data available on the dataset.

```
[9] ✓ 11m 29.3s
... 291
    291
    291

task = task.drop(columns=['CPU request', 'memory request', 'disk space request']).compute()
[12] ✓ 7m 49.1s
```

Figure 3.5 Checking Null Values and Dropping Rows Containing Null Values

Another process in data cleaning is dealing with noisy data. In order to identify noisy data, we can use box plot and scatter plot as all of our data is in numeric form. Unlike missing data, we may not remove the noisy data as they may be contributing the outcome of the job or task completion status.

3.3.2 Data Integration

Because the data is from multiple tables and multiple files, data integration is done to merge all the dataset into a single file for modelling later on. For our model, we will merge two table which are job event table and task event table. Before the integration is done, we will aggregate the data in task event table. This is because there are multiple rows representing a single job. Data aggregation is done by choosing the maximum resources requested for each job. After data aggregation is done, we perform data integration using merge function. Finally, we will be removing the data with empty columns indicating the rows does not have matching entry in another table.

```
task_event = task_event.groupby('Job ID').agg({'CPU Request':'max', 'Memory Request':'max', 'Disk Space Request':'max'})
```

Figure 3.6 Data Aggregation Process

```
job_event = job_event.merge(task_event, how='outer', on="Job ID", suffixes=({' Job', ' Task'})  
job_event = job_event.dropna(subset=['Job Name', 'CPU Request'])
```

Figure 3.7 Data Integration Process

3.3.3 Data Reduction

Data reduction is a process of reducing the amount of data from a large dataset. Due to the amount of data available data, we will not use the whole dataset for model construction. Instead, we took data from the first fourteen (14) days as a sample dataset. We also remove the rows where the any job or task has not yet completed its job/task lifecycle. Both of this done by filtering the value inside two column which are timestamp and event type

```
task_event = task_event[task_event.Timestamp != 0]  
task_event = task_event[task_event.Timestamp < 604800600000]  
task_event = task_event[task_event.Event_Type > 1]  
task_event = task_event[task_event.Event_Type < 7]
```

Figure 3.8 Data Reduction Process

Next, we did the correlation analysis to determine the most relevant feature to be use in the model construction. We decide to predict the outcome of cloud computing job or task outcome based on its resource request, scheduling class and priority. Therefore, there will be only five columns for final dataset for job level failure prediction. For task level failure prediction, there is only one different compared to dataset for job level prediction, which is there is a task priority for task prediction.

3.3.4 Data Transformation

Next, we will proceed to the next step which is data transformation. This step will be including several processes which are data relabelling and class balancing. The details about these steps will be explained in the following subsection.

3.3.4.1 Data Relabelling

Next, we will proceed to the next step which is data transformation. This step will be including several processes which are data relabelling. The dataset event contains four event type which is FAIL, FINISH, KILL and LOST. FAIL is an event where a task or job was forcefully removed from its process due to user program failure. FINISH is an event where a task or jobs is finish normally. KILL is an event where the user cancelled the task. LOST is an event where the task presumably finishes but not captured by the system. We will group FAIL, KILL and LOST as a failed task group and FINISH will be labelled as successful group. The relabelled event type will be the class variable for this research.

```
job = job.replace({'Event_Type':{2:0, 3:0, 4:1, 5:0, 6:0}})
```

Figure 3.9 Data Relabelling Process

3.3.4.2 Class Balancing

Lastly, we will do an over sampling process to further solve the problem of imbalance class problem. There are many algorithms can be used for over sampling a class in the dataset. We decided to use SMOTE function. SMOTE is short for Synthetic Minority Oversampling Technique. SMOTE works by selecting examples that are close 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. SMOTE process will increase the amount of data for minority class, and it will help our deep learning algorithm later on.

```
x = job.drop('Event Type')
y = job[['Event_Type']]
sampling_method = SMOTE()
x,y = sampling_method.fit_resample(x,y)
```

Figure 3.10 SMOTE Process

3.4 Data Storage

The last step before modelling is data storage. We will store all the pre-processed dataset into two new files. One file for job failure prediction and another one is for task level failure prediction. We decided to store the new dataset into CSV file format. CSV file format is chosen because it is the most compact file for large data type, and it is also the most common data exchange format that is supported by many applications. In addition, CSV file format can be read at a satisfactory speed compared to other file format.

3.5 Data Modelling

Data modelling will take place after data pre-processing, and during this stage, two type of machine learning algorithm implemented in this experiment which are traditional machine learning and deep learning algorithm. The details about this algorithm will be explained in the following subsection.

3.5.1 Traditional Machine Learning Model

For traditional machine learning algorithm, we implemented three different type of algorithm bases, which are regression, tree, and ensemble. For regression base, we use logistic regression as our machine learning algorithm as it is the most use regression algorithm in machine learning. For tree base, Decision Tree Classifier is used it is only tree machine learning algorithm for classification of a class. Lastly, the third method is ensemble method. Ensemble method is a machine learning technique that combine several base models to produce one optimal model. For machine learning with ensemble method, we use different type of machine learning algorithm which are random forest, gradient boosting, and extreme gradient boosting. All of these machine learning algorithms can be fetched from scikit-learn packages in python.

```

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier

```

Figure 3.11 Python Library for Traditional Machine Learning Model

3.5.2 Deep Learning Model

Next the last three machine learning algorithms will be three different variants of Long-Short Term Memory based algorithm. Deep learning is designed to replicates human brain neural network. For our Long-Short Term Memory algorithm, we will build the deep learning model using the Sequential model available in TensorFlow. Sequential models allow the user to manually arrange the hidden layer to our experiment requirement. For our deep learning model, they will be only to type of hidden layer which are Long-Short Term Memory layer and Dense Layer. For the different type of deep learning variant, the number of Long-Short Term Memory layer will be manipulated. The three variant of deep learning model are single layer LSTM algorithm, bi-layer LSTM algorithm and Tri-Layer LSTM algorithm. Lastly, we add the dense layer to ensure the output of our algorithm is in the correct shape. The epoch is set to 100 to ensure we gain the best model possible. To reduce the training times and prevent overfitting of the model, the training will be automatically stop if there is no improvement in validation loss after 10 epochs.

```

LSTM1 = Sequential()

LSTM1.add(LSTM(units = 50, input_shape = (X_train.shape[1], 1)))
LSTM1.add(Dropout(0.2))

LSTM1.add(Dense(units = 2))

LSTM1.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics=['accuracy'])

LSTM1.fit(X_train, Y_train, validation_data=(X_test,Y_test), epochs=100)

LSTM1.save('LSTM1_new.keras')

```

Figure 3.12 Single Layer LSTM Model

3.6 Model Evaluation

Model evaluation is one of the most important parts in validating the quality of the model by comparing its performance with other algorithm used to create the best prediction model to predict failure prediction in cloud services. The performance of each model will be compared based on the model accuracy, sensitivity, precision, and specificity which will be further explained in Chapter 4.

3.7 Summary

This chapter explain on the theoretical framework of this experiment as a guideline for the experiment. This chapter has also explained on the methodology applied to create the prediction model.

CHAPTER FOUR

ANALYSIS AND RESULT DISCUSSION

This chapter will explain the analysis of the dataset, the result and analysis of failure prediction using Google Cloud Traces 2011 Dataset. Sample data preparation has been explained in Chapter 3.

4.1 Google Cloud Platform

This experiment was run by using Google Cloud Platform (GCP). GCP is a suite of public cloud computing service offered by Google. The platform includes a range of host service for computer storage and application development which run on hardware owned by Google.

4.1.1 Experiment Setting

In this experiment, we execute our Jupyter Notebook on GCP. To set up the Jupyter notebook on GCP, an account needs to be created in Google Cloud Platform. For every new account created, Google provided free \$300 or RM1200 free credit as a welcome gift. To use Jupyter notebook in GCP, we must first create a virtual machine. This can be done on 'Vertex AI' workbench. To create a virtual machine for our notebook, we click the new notebook button on the top of the page and select Python 3 with CUDA 11 option. The virtual machine configuration is created with the specific shown in Figure 4.1




New notebook


Notebook name *

63-char limit with lowercase letters, digits, or '-' only. Must start with a letter. Cannot end with a '-'.

Region * **Zone ***
(i) Restricted by GPU selection.

Notebook properties

Environment 	Python 3 (with Intel® MKL and CUDA 11.0)
Machine type	4 vCPUs, 15 GB RAM
GPUs 	1 NVIDIA Tesla T4
Boot disk	100 GB Standard persistent disk
Data disk	100 GB Standard persistent disk
Subnetwork	<input type="text" value="default(10.138.0.0/20)"/>
External IP	Ephemeral(Automatic)
Permission	Compute Engine default service account
Estimated cost 	\$281.58 monthly, \$0.386 hourly

☒ Install NVIDIA GPU driver automatically for me 

ADVANCED OPTIONS CANCEL CREATE

Figure 4.1 Virtual Machine Configuration

It is recommended to run the experiment using resource in us-west1 region as it is the resource on this region is the cheapest compared to another region. The VM instance set up was constructed with 4 vCPU cores and 16 GB memory with an addition of NVIDIA Tesla T4 GPU. This configuration is chosen as it is the economical option to run a deep learning analysis with the amount of data we use. After successfully requisitioned our resources, we can install the needed library for our experiment through Jupyter.

4.2 Result Of Descriptive Analysis

In this section we will explaining our finding four our data understanding process focusing on two table which are job event table and task event table. The datasets have been partially cleaned before it is made public. All the details about this table are explained in the following subsection.

4.2.1 Job Event Table

Job event table is partitioned to 500-part file, with each file size approximately 500 kB. The table containing 8 columns and 2,012,242 rows. From the 8 columns, there is three hash string column and five integer columns. From all of the rows, there is only 672,074 unique Jobs ID. This is because a single job may contain at least three occurrences in the dataset because every event such as job submission, job scheduling and job termination is recorded. From all these 19 rows of traces were synthesised. The data is synthesised because the transition data is not recorded or missing from the traces record. Figure 4.2 shows the distribution of scheduling class in the jobs event table while figure 4.3 shows the job termination status categorized by their scheduling class.

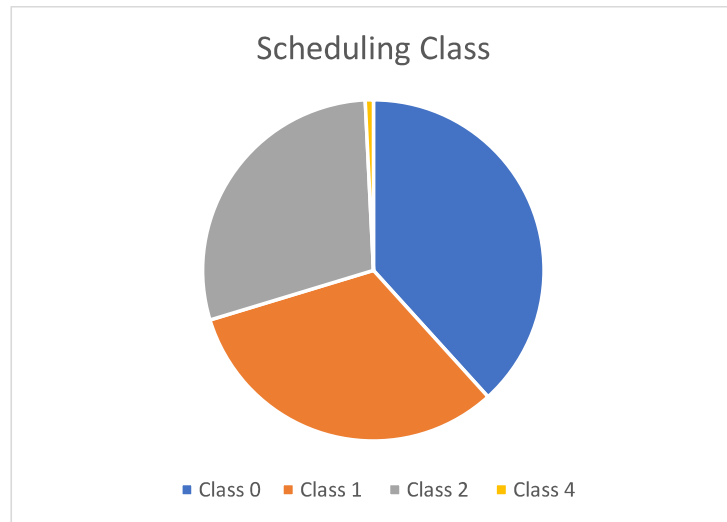


Figure 4.2 Distribution of Job Scheduling Class

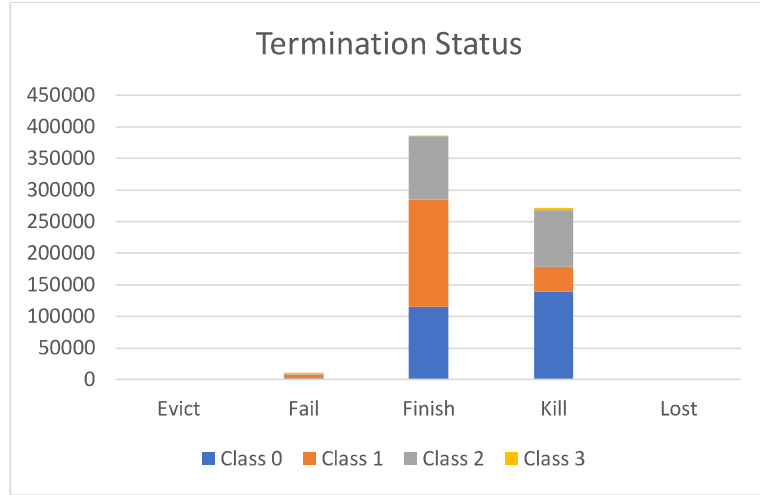


Figure 4.3 Job Termination Status Categorized by Scheduling Class

There are three four scheduling class whereas class 0 job indicates non-production job such as non-business critical analysis and class 3 representing latency sensitive job such as serving revenue-generating user requests. Based on figure 4.2, almost half of the job in the dataset were classified as class 0 jobs, while only around 5,000 jobs were classified as class 3 job. From figure 4.3, about 385,582 jobs finishes normally, 274,341 jobs were killed due to interruption from the user, or their dependant job died. About 10,000 jobs was failed where the job was de-scheduled due to task failure. Lastly, about 22 jobs was evicted. Several reason for job to be evicted are because of a higher priority task or job, the scheduler overcommitted, and the actual demand exceeded the machine capacity, the machine on which it was running became unusable, or because a disk holding the task's data was lost.

4.2.2 Task Event Table

Task event table is partitioned to 500-part file, we each file size approximately 15 MB. The table containing 13 columns and 144,648,288 rows. In the table, there is 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 is 25,424,731 unique tasks in the table. Same as job, a single task may contain at least three occurrences in the dataset because every event such as task submission, task scheduling and task termination is recorded. Out of the 25,424,731 rows, 18,375 rows of data were synthesised, which is around 7 percent of the total records. Figure

4.4 shows the distribution of task priority in the dataset while figure 4.8 shows the termination status of task categorized by their priority.

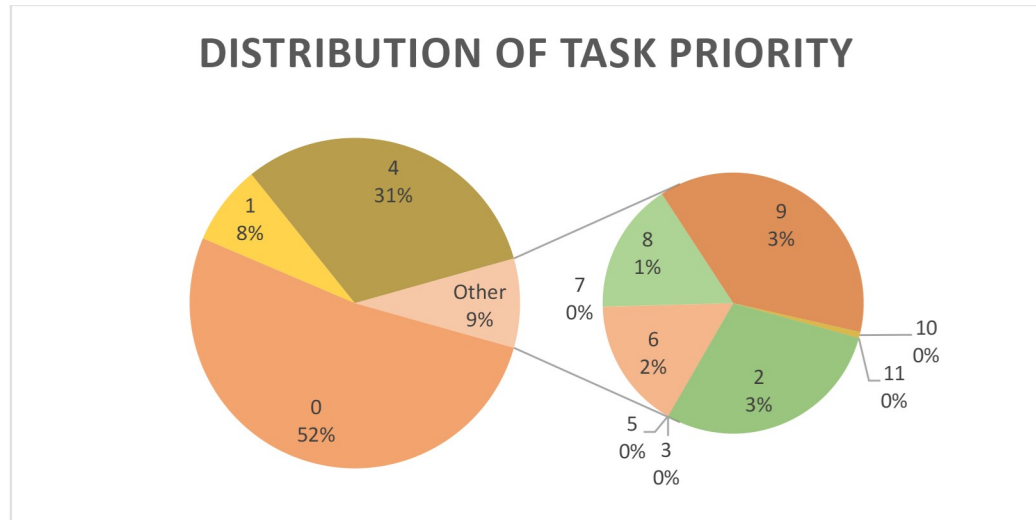


Figure 4.4 Distribution of Task Priority

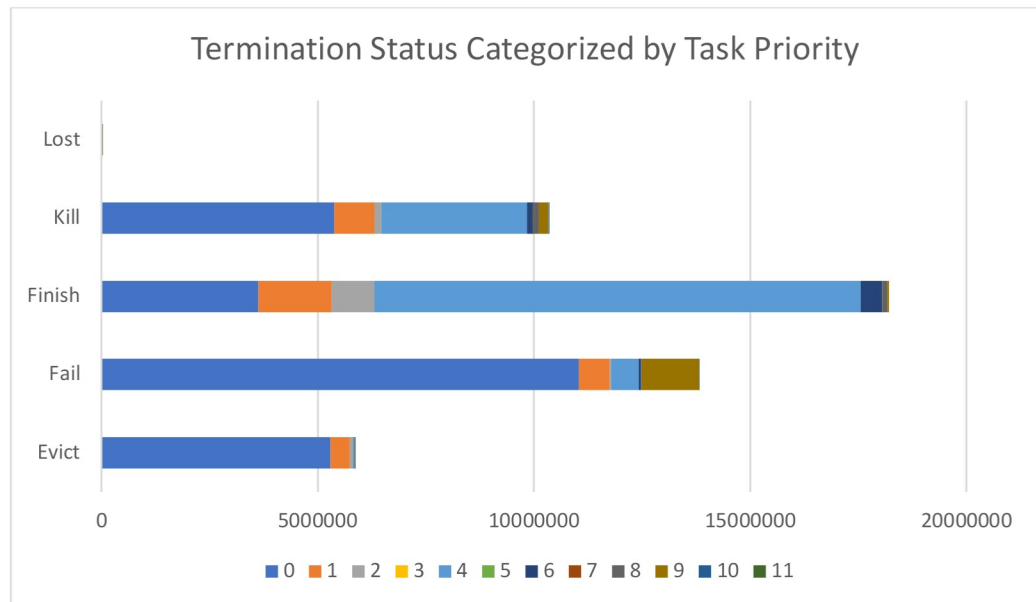


Figure 4.5 Termination Status Categorized by Task Priority

From figure 4.4, 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 ranged from zero (0) to eleven (11) whereas zero is the least priority task while level eleven priority means the task at its highest priority level. Unlike job, task priority plays more significant roles is determining the resources access for each task. From figure 4.5, we can see that the ratio of failed task to finish task is 3 to 4. This means that from the total of failed task and finished task, three out of four tasks did not finish

properly. This amount is abnormal as cloud service provider usually goals of service availability of 99.9%. We theorized that there is an outage occurring on the site during the traces' duration. Figure 4.6 shows the distribution of resource request.

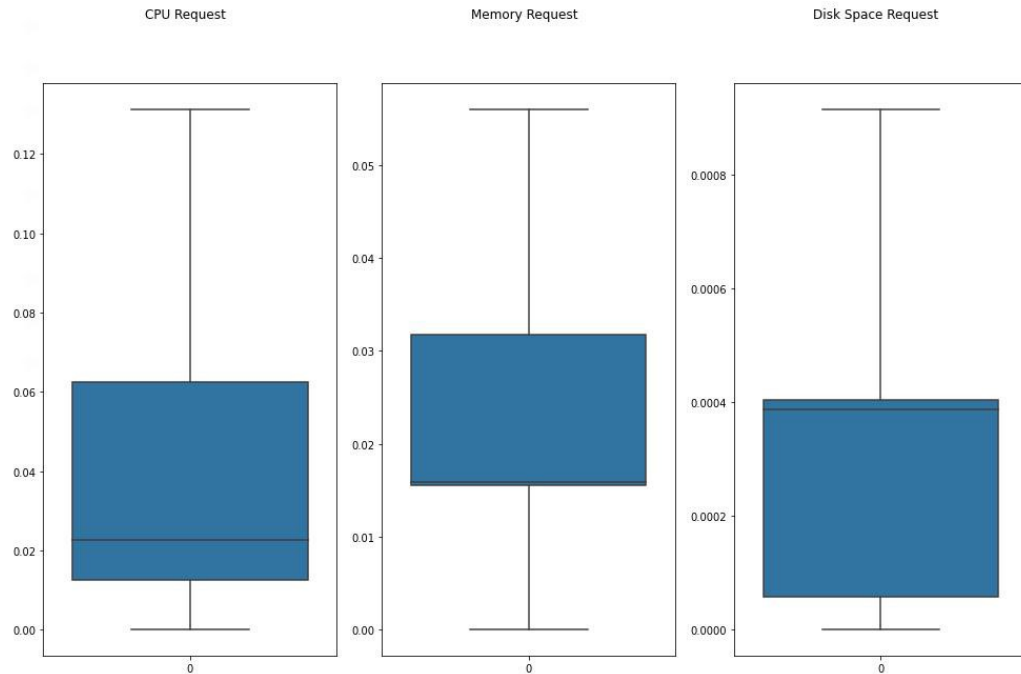


Figure 4.6 Boxplot of Resource Request Distribution in Task Event Table

From figure 4.6, we can see that the mean amount for CPU requested for task is around 0.15% to 6% out of the total CPU resources in a machine. For memory, the range of amount memory requested is usually around 1.5% to 3%, while for memory is between 0.01% to 0.04%. Do note that all of this value has been normalised to the amount of each machine.

4.3 Result Of Model Performance

Job or task that does not finished successfully will be identified as failure while only job or task that successfully perform it duties will be classified as a success. The dataset will be divided to two group with the ratio of 70 to 30 where 70 percent of the dataset will be used for training the model while the remaining 30 percent if the dataset will be used for model evaluation after training process has completed. the performance of the model is explained in the following subsection.

4.3.1 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 predictor to predict failure correctly and cover as many failure events as possible while generating very few false positives. The performance of training and testing has been compared by evaluate the accuracy, error rate, precision, sensitivity, specificity, and F-measure of the model. The measures can be obtained from the confusion matrix True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).

- a) True positives (TP): Cloud job or task that labelled as failure and evaluated as failure
- b) True negatives (TN): Cloud job or task that labelled as success and evaluated as success
- c) False positives (FP): cloud job or task that labelled as success but is evaluated as failure
- d) False negatives (FN): cloud job or task that labelled as failure but is evaluated as a success

4.3.1.1 Accuracy And Error Rate

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

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

$$Error Rate = 1 - Accuracy$$

4.3.1.2 Sensitivity And Specificity

Sensitivity measures the percentage of failed task correctly identified while specificity measure the percentage of non-faulty task correctly identified. The equation for sensitivity and specificity is as shown below

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

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

4.3.1.3 Precision

Precision is defined as the ratio of correctly predicted failure compared to the number of all recognized failure. The equation is as shown below.

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

4.3.1.4 F-Measure

F-Measure value are computed as a from the precision score and recall score. The higher the value of F-Measure, the better the overall model performance. The equation for F-Measure is as shown below

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

4.3.2 Job-Level Failure Prediction

In this section we will be explaining our findings of the model for job-level failure prediction.

4.3.2.1 Feature Importance

Predictor importance helps in understand the relative importance of each predictor in estimating the model. This chart will help to focus on the predictor fields that matter the most and consider dropping or ignoring those that matter least. Predictor importance does not relate to model accuracy, it just relates to the importance of each predictor in making prediction. The predictor importance is calculated from the training partition, otherwise the test data is used.

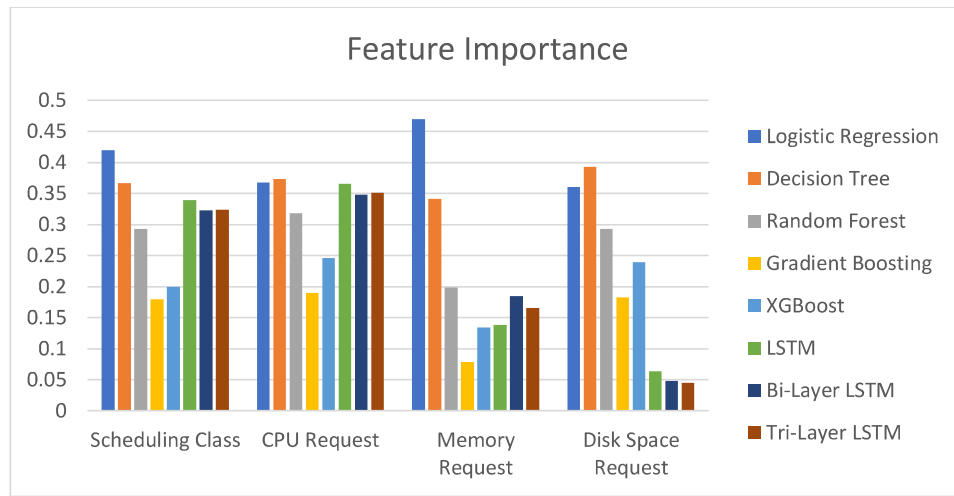


Figure 4.7 Feature Importance by Machine Learning Algorithm

Figure 4.7 shows that for all machine learning model with the exception of Gradient Boosting and Extreme Gradient Boosting, Scheduling Class and CPU request is the most important feature in determining the outcome of the termination status of the job. These two features score around 30 to 35 percent for their importance in the cloud job termination status. The score is varied depending on how the machine learning algorithm work. For LSTM, it seems that disk space request is insignificant in determining the termination status of cloud job. Other than that, logistic regression algorithm use memory request as the determining feature in order to evaluate a job or task outcome.

4.3.2.2 Model Evaluation

For job level failure prediction, the amount of data selected is around one million rows of data. Due to the amount of data, the model deep learning process experience underfitting as there is not enough data to for deep learning model to understand the complexity of the dataset. Therefore, we can see the performance of LSTM model as job level failure predictor worse compared to the traditional machine learning model. Table 4.1 shows the performance of every model using the training dataset while table 4.2 show the performance of all models trained in predicting the result of the test dataset.

Table 4.1 Training Performance Evaluation

Model	Accuracy (%)	Error Rate (%)	Precision (%)	Sensitivity (%)	Specificity (%)	F-Measure
Logistic Regression	63.11	34.66	41.44	55.65	66.30	0.4751
Decision Tree Classifier	93.43	8.08	92.19	91.56	94.70	0.9187
Random Forest Classifier	93.27	7.63	91.47	91.80	94.26	0.9163
Gradient Boosting Classifier	90.72	9.43	91.40	86.35	93.96	0.8098
XGBoost Classifier	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
Tri-Layer LSTM	85.10	14.77	81.75	81.34	87.65	0.8155

Table 4.2 Testing Performance Evaluation

Model	Accuracy (%)	Error Rate (%)	Precision (%)	Sensitivity (%)	Specificity (%)	F-Measure
Logistic Regression	63.13	36.87	41.36	55.90	66.21	0.5755
Decision Tree Classifier	93.23	6.77	91.95	91.34	94.52	0.9165
Random Forest Classifier	89.65	10.35	50.97	52.21	94.07	0.5158
Gradient Boosting Classifier	90.65	9.35	91.25	86.37	93.83	0.8874
XGBoost Classifier	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	86.26	90.82	88.19	0.8722
Tri-Layer LSTM	85.14	14.86	81.57	81.64	87.52	0.8161

From table 4.1 and 4.2, we can see that Extreme Gradient Boosting or XGBoost has the best accuracy at 93.25% and 93.10%. The F-Measure score further proven that XGBoost is the best model out of the rest with the score of 0.9325 and 0.9310. XGBoost also shown has the highest precision where the managed to correctly identify 94.31% of the job outcome. Furthermore, their sensitivity and specificity are also high meaning they able to predict the job failure and non-failure successfully.

The second highest accuracy was recorded by Decision Tree in overall algorithm. The result for XGBoost and Decision Tree only show slightly difference in term of performance of accuracy, precision and specificity during testing and training phase. Decision tree recorded 93.27% accuracy during training and 93.23% during testing with 92.19% and 91.95% precision. Like XGBoost, Decision Tree show high sensitivity and specificity score during testing and training respectively shows that can correctly identified both job failure and job success correctly.

Gradient boosting has the third highest accuracy at 90.72% during testing phase compared to random forest at 89.65%. But during training phase gradient boosting shows lesser accuracy at 90.72% compared to random forest at 93.27%. Gradient Boosting show better performance during the testing phase where the F-Measure at testing phase is 0.8874 while the F-Measure at 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 are 89.65% . Logistic regression has the least accuracy compared to another model. This model has lowest F-measures 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.

Lastly, three LSTM model that has been built for this experiment performance slightly behind other machine learning algorithm with the exception of logistic regression. This may be caused by underfitting the model where there is no large enough data for the model to understand to complexity of the data in order to make the correct classification. We also observed that there is 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.

4.3.3 Task-Level Failure Prediction

In this section we will be explaining our findings of the model for job-level failure prediction.

4.3.3.1 Feature Importance

For task level prediction, we once again considering feature importance in order to learn how the machine learning use the available feature to produce the prediction for the outcome of a task. Like job failure prediction, all the predictor importance is calculated using the training dataset.

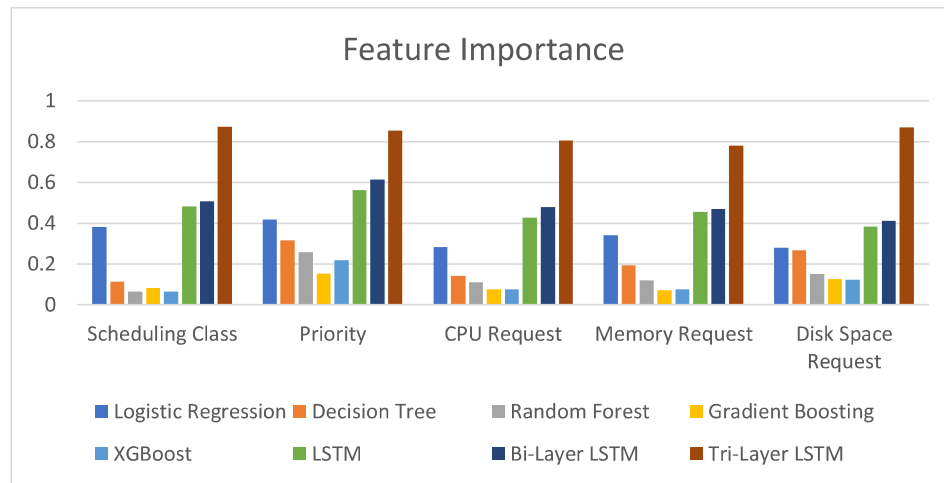


Figure 4.8 Feature Importance by Machine Learning Algorithm

As seen in figure 4.8, we can see that the priority is considered the most importance feature in evaluating the outcome of the task termination status. For traditional machine learning algorithm with the exception of gradient boosting, the priority importance is levelled at 0.2 meaning it became the main factor in determining 20% of the dataset outcome is determined by the priority. For resource request, only logistic regression has a higher score in feature importance compared to the rest of the traditional machine learning algorithm. For deep learning algorithm, all three models considered all the feature as important as we can see in figure 4.8

where the feature importance score for deep learning model is higher compared to traditional machine learning model we have trained.

4.3.3.2 Model Evaluation

For task level failure prediction, the amount of data selected is around fourteen million rows of data. Table 4.3 shows the performance of every model using the training dataset while table 4.4 show the performance of all models trained in predicting the outcome of the test dataset.

Table 4.3 Training Performance Evaluation

Model	Accuracy (%)	Error Rate (%)	Precision (%)	Sensitivity (%)	Specificity (%)	F-Measure
Logistic Regression	69.68	30.32	71.09	79.95	55.67	0.7526
Decision Tree Classifier	89.75	10.25	85.44	98.56	78.42	0.9153
Random Forest Classifier	92.47	7.53	90.66	99.13	78.40	0.9471
Gradient Boosting Classifier	87.81	12.19	84.81	95.92	76.86	0.9003
XGBoost Classifier	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
Tri-Layer LSTM	87.54	12.46	82.90	79.53	75.27	0.8962

Table 4.4 Testing Performance Evaluation

Model	Accuracy (%)	Error Rate (%)	Precision (%)	Sensitivity (%)	Specificity (%)	F-Measure
Logistic Regression	69.69	30.31	71.12	79.95	55.68	0.7528
Decision Tree Classifier	89.75	10.25	85.45	98.57	78.42	0.9154
Random Forest Classifier	89.75	10.25	85.43	98.58	78.40	0.9154
Gradient Boosting Classifier	87.87	12.13	84.80	95.94	76.88	0.9003
XGBoost Classifier	89.35	10.65	85.05	98.31	77.89	0.9120
Single Layer LSTM	87.82	12.18	83.95	96.86	76.20	0.8994
Bi-Layer LSTM	86.95	13.05	81.39	98.18	73.87	0.8900
Tri-Layer LSTM	87.55	12.45	82.92	97.53	75.28	0.8963

From table 4.4 and 4.5, the best performing model in predicting task level failure prediction is tied between random forest and decision tree. Both of them managed to obtain the accuracy score of 89.75% during the testing phase. However, the random forest model gains a slight edge during the training phase. During the training phase, random forest gains the accuracy score of 92.47% while decision tree obtained the accuracy 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 model.

Extreme gradient boosting has the third highest accuracy at 89.35% during testing phase compared to gradient boosting at 87.87%. The same is true for training phase where the accuracy for gradient boosting at 87.81% and accuracy for XGBoost at 89.34%. F-Measure for XGBoost is at 0.9120 and 0.9119 while the F-Measure for gradient boosting is at 0.9003. Logistic regression has the least accuracy compared to another model. This model has lowest F-measures and accuracy among the other traditional machine learning model 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.

Lastly, same as the job level failure prediction, three LSTM model that has been built for this experiment performance slightly behind other machine learning algorithm with the exception of logistic regression. The accuracy score for all three deep learning algorithms over around 86% to 87%. We also observed that there is not much difference in performance despite their difference in the number of hidden layers.

4.4 Summary

Scheduling class is the most important feature for determining job level failure prediction while task priority is the most important feature for determining task level failure prediction. As we mention above XGBoost is the best performing algorithm out of all algorithms we built in this research. For task-level failure prediction, the best performing model is tied between random forest and decision tree. This however

may be affected by the amount of failure event in the task record. We also found out that deep learning model perform slightly behind compared to other traditional machine leaning model. The performance however is in sync with past finding where the accuracy for cloud failure prediction hover below 90%

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

This chapter will conclude the project purpose of finding the best machine learning algorithm in determining the outcome of job or task termination status in cloud services starting from the overview of the project until the accomplishment gained. Finally, it will describe the limitations of this project and provide recommendations for future work.

5.1 Project Overview

The experiment has successfully developed to detect any job or task in cloud services that may ended in failure during the scheduling phase. There are several mitigation methods that has been implemented in real world environment in addition of much research in regard of failure prediction in cloud services platform. However, there is no deep learning algorithm being studied for failure detection during the scheduling phase. By identifying job or task that may ended abruptly will saves scarce computing resources and may also prevent cloud services outage due to cascading failure.

This project set to identify the best algorithm that able to produce high accuracy to predict the termination status of cloud job and task. This machine learning algorithm will later be implemented in a web application system to help solving the problems stated. By using machine learning, it will help filter thousands to millions of jobs and task that is being scheduled every seconds. Thus, it will save resources that may been wasted in performing task or job that ended in failure.

5.2 Project Achievement

The project has accomplished all the objective stated in chapter one. It begins with studying past works to identify the best machine learning algorithm can be implemented to solve our problems. In the end, we decide to build five machine learning model (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting and Extreme Gradient Boosting) and one deep learning model (Long-Short Term Memory). Using Jupyter notebook, the model is developed and compared using python programming language.

The model development process including data extraction, data understanding, data preparation and data modelling. Finally, all the model developed is saved to be used for future studies. Based on the result, extreme gradient boosting is the best algorithm for job level failure prediction while random forest is the model is the best model for task level failure prediction. Surprisingly, our deep learning model performance is slightly behind compared to other machine learning algorithm.

5.3 Project Limitation

One of the limitations of this project is the usage of secondary data that has been partially processed, therefore some of the data may have been lost in the process. Furthermore, all the data has been normalised and anonymised further limiting our ability to derive more insight of the data.

The second limitation of this project is the there is large amount of task failure where one over four of task ended in failure. We theorised there is outage occurring during the period of traces. This resulted in reduced specificity score where the model has a problem of successfully identifying only 70% of failed task.

Besides that, the model is not implemented in a simulation environment due to time constraint. Because of this, we do not know the performance of the model in a real-world environment. Thus, it will be left out for the future research to conduct it.

5.4 Future Recommendation

There are several recommendations of future work. Firstly, it is recommended to obtain a primary data to further understands the failure causes. With primary data, the data will be not anonymised further help us gain more insight on the context of cloud job and task failure.

Next, as a way to evaluate the model, this study should be implemented on several difference cloud computing the evaluate the viability of the model built for real world environments. If the performance of the model is similar as the training performance, there will be more confidence the system can solve the problem.

Lastly, future work might want to consider enhancing this project to work with big data and data visualisation because it would be fruitful area for further work.

5.5 Summary

To summarize this chapter, this project is important in study to build a failure prediction algorithm. This chapter also concludes the accomplishment of this project that satisfy the main objectives of this project which are finding the best algorithm for failure prediction in scheduling job and task on cloud environment. Throughout completing this project, several limitations and challenges had been identified. The limitations faced in this project helps to provide recommendations for future researcher as there is room for improvement.

REFERENCE

- Abdelfattah, E., Elkawkagy, M., & El-Sisi, A. (2018). A reactive fault tolerance approach for cloud computing. *ICENCO 2017 - 13th International Computer Engineering Conference: Boundless Smart Societies, 2018-Janua*, 190–194. <https://doi.org/10.1109/ICENCO.2017.8289786>
- Abdul-Rahman, O. A., & Aida, K. (2015). Towards understanding the usage behavior of google cloud users: The mice and elephants phenomenon. *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2015-Febru*(February), 272–277. <https://doi.org/10.1109/CloudCom.2014.75>
- Alam, T. (2020). Cloud Computing and its role in the Information Technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, 1(2), 108–115. <https://doi.org/10.34306/itsdi.v1i2.103>
- Alashhab, Z. R., Anbar, M., Singh, M. M., Leau, Y. B., Al-Sai, Z. A., & Alhayja'a, S. A. (2021). Impact of Coronavirus Pandemic Crisis on Technologies and Cloud Computing Applications. *Journal of Electronic Science and Technology*, 19(1), 25–40. <https://doi.org/10.1016/j.jnlest.2020.100059>
- Amazon Web Services experiences another big outage - The Washington Post*. (n.d.). Retrieved January 24, 2022, from <https://www.washingtonpost.com/business/2021/12/22/amazon-web-services-experiences-another-big-outage/>
- Asghar, H., & Nazir, B. (2021). Analysis and implementation of reactive fault tolerance techniques in Hadoop: a comparative study. *Journal of Supercomputing*, 77(7), 7184–7210. <https://doi.org/10.1007/s11227-020-03491-9>
- AzurePublicDataset/Azure 2019 Public Dataset V2 - Trace Analysis.ipynb at master · Azure/AzurePublicDataset · GitHub*. (n.d.). Retrieved January 25, 2022, from [https://github.com/Azure/AzurePublicDataset/blob/master/analysis/Azure 2019 Public Dataset V2 - Trace Analysis.ipynb](https://github.com/Azure/AzurePublicDataset/blob/master/analysis/Azure%2019%20Public%20Dataset%20V2%20Trace%20Analysis.ipynb)
- AzurePublicDataset/Azure Public Dataset - Trace Analysis.ipynb at master · Azure/AzurePublicDataset · GitHub*. (n.d.). Retrieved January 25, 2022, from

[https://github.com/Azure/AzurePublicDataset/blob/master/analysis/AzurePublicDataset - Trace Analysis.ipynb](https://github.com/Azure/AzurePublicDataset/blob/master/analysis/AzurePublicDataset-TraceAnalysis.ipynb)

- Bala, A., & Chana, I. (2012). Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing. *International Journal of Computer Science Issues*, 9(1), 288–293. www.IJCSI.org
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- Chen, T., He, T., & Benesty, M. (2018). XGBoost: eXtreme Gradient Boosting. *R Package Version 0.71-2*, 1–4.
- Chen, X., Lu, C. Da, & Pattabiraman, K. (2014). Failure prediction of jobs in compute clouds: A Google cluster case study. *Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014*, 341–346. <https://doi.org/10.1109/ISSREW.2014.105>
- Cloud Computing Market Size & Share | Industry Growth [2020-2027]*. (n.d.). Retrieved January 24, 2022, from <https://www.fortunebusinessinsights.com/cloud-computing-market-102697>
- Ding, H. (2017). *Cluster data collected from production clusters in Alibaba for cluster management research*. GitHub. <https://github.com/alibaba/clusterdata>
- 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. *Proceedings - International Conference on Distributed Computing Systems*, 1333–1344. <https://doi.org/10.1109/ICDCS.2017.317>
- Gao, J., Wang, H., & Shen, H. (2020a). Task Failure Prediction in Cloud Data Centers Using Deep Learning. *IEEE Transactions on Services Computing*. <https://doi.org/10.1109/TSC.2020.2993728>
- Gao, J., Wang, H., & Shen, H. (2020b). Machine Learning Based Workload Prediction in Cloud Computing. *Proceedings - International Conference on Computer Communications and Networks, ICCCN, 2020-Augus.*

<https://doi.org/10.1109/ICCCN49398.2020.9209730>

- Ghiasvand, S., Ciorba, F. M., Tschüter, R., & Nagel, W. E. (2015). Analysis of Node Failures in High Performance Computers Based on System Logs. In *Supercomputing*. <https://doi.org/10.5451/unibas-ep40834>
- Gill, S. S., & Buyya, R. (2020). Failure Management for Reliable Cloud Computing: A Taxonomy, Model, and Future Directions. *Computing in Science and Engineering*, 22(3), 52–63. <https://doi.org/10.1109/MCSE.2018.2873866>
- Goyal, S. (2014). Public vs Private vs Hybrid vs Community - Cloud Computing: A Critical Review. *International Journal of Computer Network and Information Security*, 6(3), 20–29. <https://doi.org/10.5815/ijcnis.2014.03.03>
- Islam, T., & Manivannan, D. (2017). Predicting Application Failure in Cloud: A Machine Learning Approach. *Proceedings - 2017 IEEE 1st International Conference on Cognitive Computing, ICCCC 2017*, 24–31. <https://doi.org/10.1109/IEEE.ICCC.2017.11>
- Jassas, M. S., & Mahmoud, Q. H. (2019, April 1). Failure characterization and prediction of scheduling jobs in google cluster traces. *2019 IEEE 10th GCC Conference and Exhibition, GCC 2019*. <https://doi.org/10.1109/GCC45510.2019.1570516010>
- Li, Y., Jiang, Z. M. J., Li, H., Hassan, A. E., He, C., Huang, R., Zeng, Z., Wang, M., & Chen, P. (2020). Predicting Node Failures in an Ultra-Large-Scale Cloud Computing Platform. *ACM Transactions on Software Engineering and Methodology*, 29(2). <https://doi.org/10.1145/3385187>
- Lin, Q., Hsieh, K., Dang, Y., Zhang, H., Sui, K., Xu, Y., Lou, J. G., Li, C., Wu, Y., Yao, R., Chintalapati, M., & Zhang, D. (2018). Predicting node failure in cloud service systems. *ESEC/FSE 2018 - Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 480–490. <https://doi.org/10.1145/3236024.3236060>
- Mell, P., & Grance, T. (2011). The NIST definition of cloud computing. In *Cloud Computing and Government: Background, Benefits, Risks* (pp. 171–173). <https://doi.org/10.1016/b978-0-12-804018-8.15003-x>
- Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An

- introduction to decision tree modeling. In *Journal of Chemometrics* (Vol. 18, Issue 6, pp. 275–285). John Wiley & Sons, Ltd. <https://doi.org/10.1002/cem.873>
- Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, 7(DEC), 21. <https://doi.org/10.3389/fnbot.2013.00021>
- Rasheduzzaman, M., Islam, M. A., Islam, T., Hossain, T., & Rahman, R. M. (2014). Study of different forecasting models on Google cluster trace. *2013 16th International Conference on Computer and Information Technology, ICCIT 2013*, 414–419. <https://doi.org/10.1109/ICCITechn.2014.6997346>
- Ray, P. P. (2017). An Introduction to Dew Computing: Definition, Concept and Implications. *IEEE Access*, 6, 723–737. <https://doi.org/10.1109/ACCESS.2017.2775042>
- Rosa, A., Chen, L. Y., & Binder, W. (2015). Predicting and mitigating jobs failures in big data clusters. *Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015*, 221–230. <https://doi.org/10.1109/CCGrid.2015.139>
- Shetty, J., Sajjan, R., & Shobha, G. (2019). Task resource usage analysis and failure prediction in cloud. *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, 342–348. <https://doi.org/10.1109/CONFLUENCE.2019.8776612>
- Soualhia, M., Khomh, F., & Tahar, S. (2015a). *Predicting Scheduling Failures in the Cloud*. <http://arxiv.org/abs/1507.03562>
- Soualhia, M., Khomh, F., & Tahar, S. (2015b). Predicting scheduling failures in the cloud: A case study with google clusters and hadoop on Amazon EMR. *Proceedings - 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, H*, 58–65. <https://doi.org/10.1109/HPCC-CSS-ICSS.2015.170>
- Tang, H., Li, Y., Jia, T., & Wu, Z. (2016). Hunting Killer Tasks for Cloud System through Machine Learning: A Google Cluster Case Study. *Proceedings - 2016 IEEE International Conference on Software Quality, Reliability and Security, QRS 2016*, 1–12. <https://doi.org/10.1109/QRS.2016.11>

- TheGuardian. (2017). British Airways IT failure caused by “uncontrolled return of power.” *The Guardian*. <https://www.theguardian.com/business/2017/may/31/ba-it-shutdown-caused-by-uncontrolled-return-of-power-after-outage>
- Tolles, J., & Meurer, W. J. (2016). Logistic regression: Relating patient characteristics to outcomes. In *JAMA - Journal of the American Medical Association* (Vol. 316, Issue 5, pp. 533–534). <https://doi.org/10.1001/jama.2016.7653>

