COMPUTING RESOURCE ESTIMATION BY USING MACHINE LEARNING TECHNIQUES FOR ALICE O^2 LOGGING SYSTEM

MISS JUTHAPORN VIPATPAKPAIBOON

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING
(COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2021

Computing Resource Estimation by Using Machine Learning Techniques for ALICE O^2 Logging System

Miss Juthaporn Vipatpakpaiboon B. Eng. (Computer Engineering)

A Thesis Submitted in Partial Fulfillment
of the Requirement for
the Degree of Master of Engineering (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2021

Thesis Committee

[illegible] ………………………………………………… (Assoc. Prof. Anan Banharnsakun, Ph.D.)	Chairman of Thesis Committee
[illegible] ………………………………………………… (Asst. Prof. Khajonpong Akkarajitsakul, Ph.D.)	Member and Thesis Advisor
[illegible] ………………………………………………… (Asst. Prof. Phond Phunchongharn, Ph.D.)	Member
[illegible] ………………………………………………… (Asst. Prof. Kejkaew Thanasuan, Ph.D.	Member
[illegible] ………………………………………………… (Lect. Unchalisa Taetragool, Ph.D.)	Member

Thesis Title	Computing Resource Estimation by Using Machine Learning Techniques for ALICE O^2 Logging System
Thesis Credits	12
Candidate	Miss Juthaporn Vipatpakpaiboon
Thesis Advisor	Asst. Prof. Dr. Khajonpong Akkarajitsakul
Program	Master of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2021

Abstract

Resource estimation is a technique that has many uses. In this research, we examine how it can be used to estimate computing resources of systems based on historical data in order to make them more efficient. Many researchers have applied machine learning to estimate computing resources. The European Organization for Nuclear Research (CERN) is currently developing a new logging system for A Large Ion Collider Experiment (ALICE) detector based on the Elasticsearch, Logstash, and Kibana (ELK) software stack. Beats, a shipper installed on the First Level Processor nodes, will receive the logged data and transfer it to Logstash, a data preprocessing pipeline. Logstash ingests the data and sends the ingested data to Elasticsearch which is used for search and analytics. The system faces difficult problems when working with large clusters which are hard to estimate. Moreover, in the future, the number of nodes and the number of services in the machine can increase or decrease. Resource estimation was used to estimate and plan the number of resources by using Metricbeat to get the historical computing metrics of machines and by using Logstash to make the system more reliable and adaptable to changes. We applied different machine learning algorithms including Random Forest Regression, Multiple Linear Regression, and Multi-Layer Perceptron to create models. The efficiency of these models is measured and compared using the coefficients of determination which are Mean Absolute Error and Mean Square Error.

Keywords: ALICE/ CERN/ Regression/ Resource Estimation/ Resource Planning

หัวข้อวิทยานิพนธ์	การประมาณการทรัพยากรเพื่อเพิ่มประสิทธิภาพและความเหมาะสมในการประมวลผลด้วยการเรียนรู้ของเครื่องสำหรับระบบการจัดการล็อก
หน่วยกิต	12
ผู้เขียน	นางสาวจุฑาภรณ์ วิภัชภาคไพบูลย์
อาจารย์ที่ปรึกษา	ผศ. ดร.ขจรพงษ์ อัครจิตสกุล
หลักสูตร	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	คณะวิศวกรรมศาสตร์
ปีการศึกษา	2564

บทคัดย่อ

การประมาณทรัพยากรเป็นเทคนิคที่สามารถใช้ได้กับปัญหาลายรูปแบบ ซึ่งหนึ่งในนั้นคือสามารถใช้ในการประเมินทรัพยากรคอมพิวเตอร์จากข้อมูลประวัติการใช้งานต่าง ๆ เพื่อทำให้ระบบมีประสิทธิภาพมากขึ้น นักวิจัยหลายคนมีการประยุกต์ใช้การเรียนรู้ของเครื่องเพื่อประเมินทรัพยากรคอมพิวเตอร์และแก้ปัญหาของพวกเขา องค์กร CERN วางแผนที่จะสร้างระบบล็อกสำหรับเครื่องตรวจจับ ALICE O^2 เพื่อที่จะจัดการและวิเคราะห์ข้อมูลล็อก ระบบ ELK (Elasticsearch Logstash Kibana) ซึ่งเป็นระบบที่จัดการข้อมูลล็อกในรูปแบบของส่วนกลางที่อนุญาตให้ผู้ใช้สามารถค้นหาข้อมูลล็อกจะเป็นส่วนสำคัญในระบบนี้ การส่งข้อมูลล็อกไปที่ Logstash ซึ่งเป็นส่วนที่ใช้ในการจัดการรูปแบบข้อมูล จะถูกส่งโดย Beat ซึ่งเป็นตัวส่งล็อกที่ติดตั้งอยู่บนโหนด FLP หลังจากนั้นล็อกจะถูกส่งต่อจาก Logstash ไปยัง Elasticsearch ซึ่งเป็นส่วนที่ใช้ในการเก็บ ค้นหาและวิเคราะห์ข้อมูลล็อก ในอนาคตจำนวนโหนดของ FLP รวมถึงการใช้งานต่าง ๆ อาจเพิ่มหรืออาจลดลงทำให้การประเมินและวางแผนทรัพยากรถูกนำมาใช้ในการรองรับปัญหาเหล่านี้ การประเมินทรัพยากรจะใช้ข้อมูลประวัติการใช้งานของเครื่องที่ถูกเก็บผ่าน Metricbeat ในรูปแบบของล็อกจาก Logstash เพื่อประมาณการทรัพยากรที่เหมาะสมกับการเปลี่ยนแปลงการใช้งาน เทคนิคการเรียนรู้ของเครื่องที่ถูกนำมาประยุกต์ใช้ ได้แก่ Random Forest Regression, Multiple Linear Regression และ Neural Network ประสิทธิภาพของแบบจำลองจะถูกวัดและเปรียบเทียบโดยใช้การวิเคราะห์ความแปรปรวนร่วม (ANOVA Analysis), ค่าสัมประสิทธิ์ของการตัดสินใจ (R^2), Mean Absolute Error (MAE) และ Mean Square Error (MSE)

คำสำคัญ: ALICE/ CERN/ การถดถอย/ การประมาณทรัพยากร/ การวางแผนทรัพยากร

ACKNOWLEDGMENTS

There are many people whom I would like to express my appreciation for their contributions, both directly and indirectly during the period of my study. First, I would like to thank my undergraduate supervisor, Asst. Prof. Dr. Phond Phunchongharn of the Computer Engineering Department at King Mongkut's University of Technology Thonburi, who invited me to start doing this thesis with CERN. I would like to greatly acknowledge Asst. Prof. Dr. Khajonpong Akkarajitsakul of the Computer Engineering Department at King Mongkut's University of Technology Thonburi, who always gives me necessary suggestions, contributions, and other help to better this study. I especially thank all committees for the comments and suggestions. I would like to express sincere appreciation to Vasco Chibante Barroso for giving information about the system on ALICE O^2. Moreover, I want to thank my friends for their help. Finally, and most importantly, I am particularly grateful to my family for their full supports and for always encouraging me.

CONTENTS

CONTENTS (Cont'd)

LIST OF TABLES

LIST OF TABLES (Cont'd)

LIST OF FIGURES

LIST OF TECHNICAL VOCABULARY AND ABBREVIATIONS

ALICE	=	A large Ion Collider Experiment
ANOVA	=	Analysis of Variance
ATLAS	=	A Toroidal LHC ApparatuS
CERN	=	The European Organization for Nuclear Research
CMS	=	Compact Muon Solenoid
DT	=	Decision Tree
DTW	=	Dynamic Time Wrap
ELK	=	Elasticsearch Logstash and Kibana
EPN	=	Event Processing Node
FLP	=	First Level Processor
HMM	=	Hidden Markov Model
KNN	=	K-Nearest Neighbors
LHC	=	Large Hadron Collider
LHCb	=	Large Hadron Collider beauty
LR	=	Linear Regression
LSTM	=	Long-Short Term Memory
MAE	=	Mean Absolute Error
MAPE	=	Mean Absolute Percentage Error
MLP	=	Multi-Layer Perceptron
MSE	=	Mean Square Error
O^2	=	Online-Offline
PM	=	Physical Machine
SLA	=	Service Level Agreement
SVM	=	Support Vector Machine
SVR	=	Support Vector Regression
RF	=	Random Forest
RMSE	=	Root Mean Square Error
R^2	=	R-Squared Score

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Motivation

CERN (The European Organization for Nuclear Research) is a research organization that investigates particle physics with the largest particle physics laboratory in the world since 1954. Many experiments were conducted in the laboratory with one of them being the LHC (Large Hadron Collider) experiments. The LHC uses 4 main particle detectors: ALICE (A Large Ion Collider Experiment), ATLAS (A Toroidal LHC ApparatuS), LHCb (Large Hadron Collider beauty), and CMS (Compact Muon Solenoid). These detectors interpret the collisions from different subatomic particles in the accelerator and generates a large amount of data.

One of the four main detector experiments is ALICE (A Large Ion Collider Experiment). ALICE was created to observe heavy-ion collisions and the quark-gluon plasma interactions using proton-proton, nucleus-nucleus, and proton-nucleus collisions. ALICE then collects the experimental data in the ALICE O^2 system as stated in Wegrzynek, et al. [1]. ALICE collected data during two main experimental periods: from 2010 until 2015 (Run 1) and 2015 until 2018 (Run 2). Afterwards, ALICE was shut down for upgrades. The new system used in Run 3 (2020) produced more than 1 TBps of data and consisted of 2 different types of computing nodes, FLP (First Level Processor) and EPN (Event Processing Node) under a heterogeneous environment. A total of 250 FLP nodes and 1500 EPN nodes were used in the experiment.

FLP will collect data from the detector and then send the data to the EPN farm for data preprocessing along with further physics experiments and analyses. There are front-end servers between FLP and EPN that can access the machine and manage processes on FLPs and EPNs. The front-end server controls the configuration parameters and distributes those parameters to the processes that running on FLPs or EPNs. The FLPs and EPNs will update the configuration with the front-end server [2]. Each FLP node in the system will run different or same services, some of them can be in the idle state due to the task scheduling on each machine. The number of nodes and services may increase or decrease in the future. To make the system more reliable and appropriate to these changes, the system needs to have good computing resources. Resource estimation is thus a vital tool for this process.

Resource estimation is a method that focuses on system logs from the server. This method will learn the historical system logs and estimate the resources for the server. The way to get the log is by using the Elasticsearch, Logstash, and Kibana (ELK) stack, an open source software which provides centralized logging that allow users to search for the logs. In order to estimate computing resources, Metricbeat is used to get the computing metrics of machines and systems from Logstash. Computing metrics include system-level CPU usage and memory among other system performance metrics.

The rest of this thesis is organized as follows: Chapter 2 presents the literature review, related research, and previous works of machine learning techniques. Chapter 3 presents our proposed work and describes the design and overview of the framework.

1.2 Statement of the Problem

In this research, there are three main problem statements related to the resource estimation on system logs:

1. The number of nodes and number of services in the system may increase or decrease which affects the reliability of Logstash and also the overall logging system.
2. Resource estimation models will be helpful to find appropriate computing resources for the system and in this work, we found an effective model.
3. We applied different machine learning algorithms including random forest regression, multiple linear regression, and multi-layer perceptron. Then, we selected an appropriate regression model among these three models.

1.3 Objectives

According to the problem statements, this thesis aims to:

1. Study ALICE O^2's environment, logging system, and ELK open sources.
2. Create methodology for estimating computing resources for ALICE O^2 logging system.
3. Compare regression models based on the history of the system log data in ALICE and find an appropriate model.

4. Discuss the result and performance between different regression models, along with comparing differences with no tuning and tuning models.
5. Formulate the regression model to estimate computing resources.

This thesis is a part of the European Organization for Nuclear Research (CERN).

1.4 Scope of the Study

1. CentOS 7 was used as the server's operating system with 4 VCPUs, 7.3 GB RAM, and 40 GB memory. On the server, we installed the ELK system to monitor the node from CERN. 3 nodes were used for the FLP.
2. The datasets used in this work were collected from Metricbeat installed on Logstash, a part of the ELK system. Our dataset was focused on main system metrics such as the number of CPU cores, percentage of CPU spent, total memory, memory used, total time spent doing I/Os, etc.
3. The 3 learning algorithms that we selected for this study were Random Forest Regression, Multiple Linear Regression, and Neural Networks based on 2 main approaches, basic and modern approaches.
4. The model was developed with Python.

1.5 Expected Benefits

1. Our resource estimation would estimate the computing usages of CPU, RAM, and memory.
2. Compare and contrast results between three machine learning techniques.
3. Research methodology would be applicable to estimate future computing resources for ALICE O^2 logging system with a different scale of nodes and services.

CHAPTER 2 LITERATURE REVIEW AND THEORY

This chapter is split into four sections. Firstly, we will present the logging system and how it works. Secondly, we describe the Elasticsearch, Logstash, and Kibana stack and the Beats log shipper. We will then survey related works on resource estimation, resource types, and the overview of resource estimation. A summary of is presented in the third part. Finally, we perform literature review on the techniques and existing methods that have been used in resource estimation and evaluation metrics.

2.1 Logging System

Logging gets a detailed set of events that happened within an application. Moreover, logging can also get specific events from a system. The purpose of logging is to track and report data in a centralized way. Logging can take the form of both event logging and actual log files. The statements that logs can inform are failure, anomaly, error, and state transformation. Logs can help find the cause of application or system problems.

Logs will automatically produce entries that consist of a timestamp for each event to a particular system. It can be used for various purposes such as: production monitoring and debugging, flagging security, tracking site visitors in order to understand visitor behavior, HTTP error checking, and monitoring resource usage. Server, rather than software, errors can cause overloads on the system; therefore, we need appropriate resources to prevent this problem.

2.1.1 Centralized Logging

When the system grows to multiple hosts or nodes, accessing and managing content may be complicated. Moreover, searching for a specific error among thousands of log files on multiple hosts could be challenging. This problem is commonly solved by centralized logging. Centralized logging can aggregate thousands of logs in a central location. After aggregation, it will be easier to collect, store, and analyze all data.

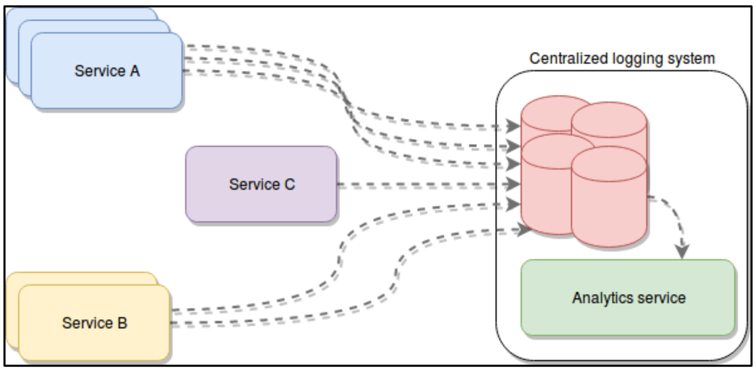


Figure 2.1 An example of centralized logging architecture [3]

Figure 2.1 is an example of centralized logging architecture. Multiple hosts of several services, Service A, Service B, and Service C, send their logging messages to a single centralized log.

2.2 Elasticsearch, Logstash, and Kibana (ELK) stack

The Elasticsearch, Logstash, and Kibana (ELK) stack consists of three open source projects which are Elasticsearch, Logstash, and Kibana. All of them are developed by the company Elastic. This system is designed to support the user to get data from any source in any format for analysis and visualization in real-time. ELK presents centralized logging that allows users to search logs from multiple hosts in a single place.

Elasticsearch [4] is a search and analytics program. It is used for storing logging data. Logstash is a data preprocessing pipeline which stays on the server-side. It is used to ingest data, maintain data formats, and transform data from multiple sources immediately. Logstash ships transformed data into Elasticsearch. Kibana is a visualization tool used to visualize data in the form of graphs and charts.

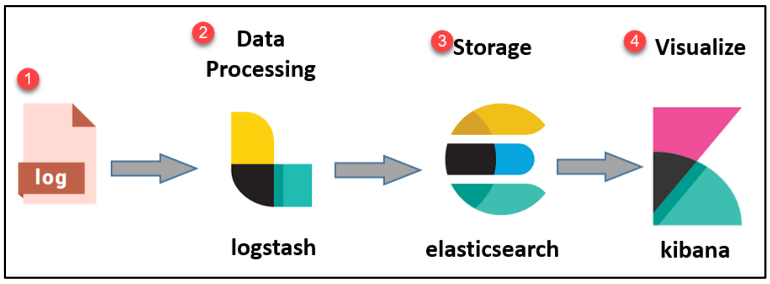


Figure 2.2 Simple architecture of ELK stack [5]

The architecture of ELK Stack is shown in Figure 2.2. This architecture starts with logs which are collected by Logstash. Logstash transforms and parses the data to Elasticsearch which stores data. Lastly, Kibana uses the Elasticsearch database to explore and visualize data. Another option to gather the logs is to include Beats in the architecture as shown in Figure 2.3.

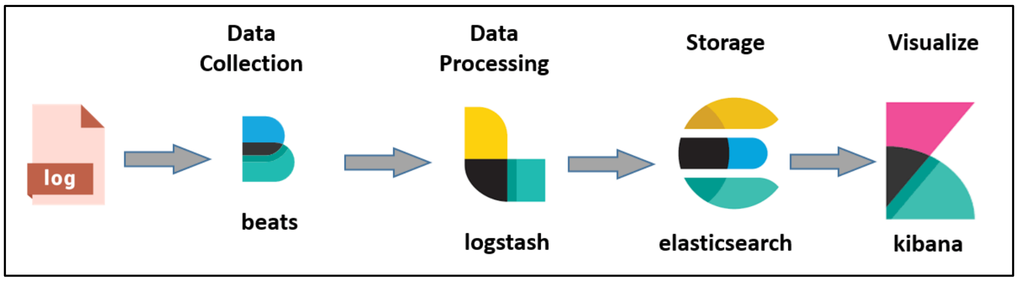


Figure 2.3 Architecture of ELK stack with Beats [5]

The normal architecture is not good enough for large amounts of data. To make the system work better we have to include buffering tools, for example, Redis and Kafka. These two tools are log aggregation tools. Figure 2.4 shows our new ELK stack architecture.

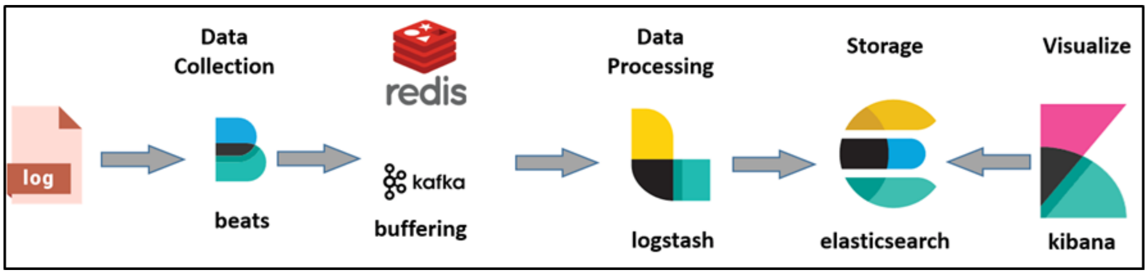


Figure 2.4 Architecture of ELK stack with buffering tools [5]

2.2.1 Beats

Beats is a platform that contains lightweight data shippers. [6] It sends data from thousands of systems to Logstash and Elasticsearch. There are 7 types of Beats which is shown in Figure 2.5. The first one is Filebeat, a shipper for log data. Metricbeat is used to collect metrics from systems and services from CPU, memory, etc. Packetbeat monitors the network traffic over an environment. Winlogbeat ships Windows event logs which are then tracked across Windows-based infrastructures. The fifth one is Auditbeat which collects Linux audit framework data. Heartbeat monitors uptime. Finally, Functionbeat is deployed as a function on a cloud provider's platform to collect and ship data from cloud services.

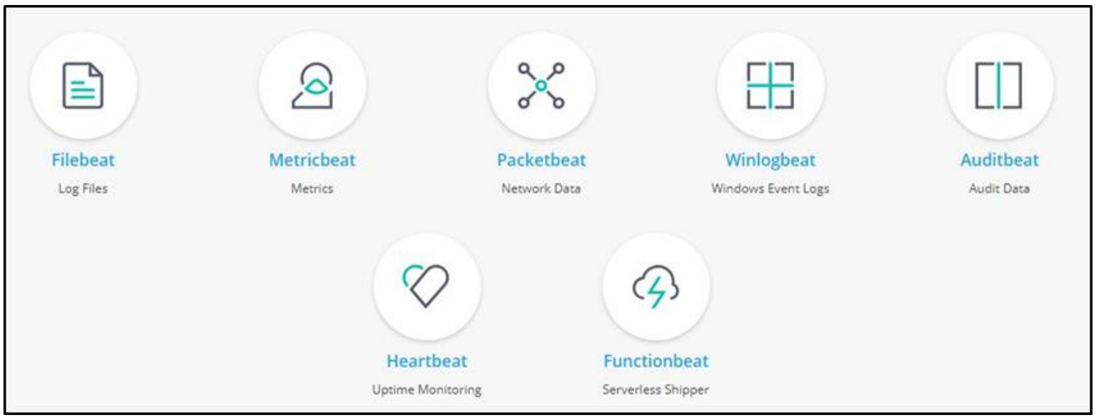


Figure 2.5 Types of Beats [7]

2.3 Related Work

2.3.1 Resource Types

In a resource management system, processes are associated with different types of resources. According to a survey by Jennings, et al. [8], resources are separated into four main types which are computing resources, networking resources, storage resources, and power resources. Computing resources are the properties of the physical machines (PMs), each of them consisting of one or more processors, network, memory, and I/O.

Networking resources of PMs are related to the bandwidth and latency, overhead, network protocols, and are also associated to network topology. Storage resources consist of the number of users, load or data volume, virtual disks, and database services. The last type of resource is power resources. If we consider the data center, power resources will be another important type of data which comprise of the amount of power consumed by a server, energy usage, and energy cost.

2.3.2 Related Application

Resource estimation can be used in many ways. As we focus on computing resource estimation, many works use this technique to fulfill their problem. In 2008 a regression-based method was implemented by Zhang, et al. [9]. The research investigated factors that impact the efficiency and accuracy of the proposed models and applied regression models, non-negative least square regression to approximate CPU demand from client transactions, and also use approximate information to create efficient capacity planning..

Another form of application that uses linear regression-based methods is to represent CPU usage for the live migration of virtual machines in the data center which as done by Fahimeh, et al. [10]. The research approximated CPU utilization based on historical usage together with live migration to minimize power consumption, energy cost, and decrease average service-level agreement (SLA) violations.

Khan, et al. [11] developed a model to characterize and predict the workload of virtual machines (VMs) in a cloud system. First of all, they used the co-clustering technique to identify groups of VMs in the form of workload patterns. The Hidden Markov Model (HMM) was used to predict changing workload patterns of input VMs. Similarly, Patel, et al. [12] used another clustering technique that grouped tasks that have the same characteristics into the same cluster and developed a workload model to estimate the workload pattern of random tasks from the log. The clustering technique used was Dynamic Time Wrap (DTW). Linear algebra was used to find the server resource utilization namely CPU, memory, disk, and network usages from several patterns of log data as mentioned in Vora [13]. Here resource utilization logs from all nodes in the system was used.

Several works used many techniques, examples include Bankole and Ajila [14]; Rajaram and Malarvizhi [15]; Adane and Kakd [16]. The first and second work [14, 15] compares 3 techniques which are Support Vector Regression (SVR), Multilayer Perceptron (MLP), and Linear Regression (LR). The last work [16] compares results between 5 models including K-Nearest Neighbors (KNN), Linear Regression, Multilayer Perceptron, Support Vector Machine (SVM), and Random Forest.

To improve the accuracy of the result, the ensemble-based method could be used as shown by Mehmood, et al. [17]. The ensemble-based method is performed on stacking mechanisms that stacks K-Nearest Neighbor and Decision Tree (DT). An example of a module based on stacking is shown in Figure 2.6. The stacking scheme is divided into two steps which are level 0 and level 1. Level 0 is assigned as a base learner, used to train the dataset. The base learner consists of KNN and DT. The next step is level 1 or meta learner, they train the DT model on level 1 by using the output of the base level to improve classification results.

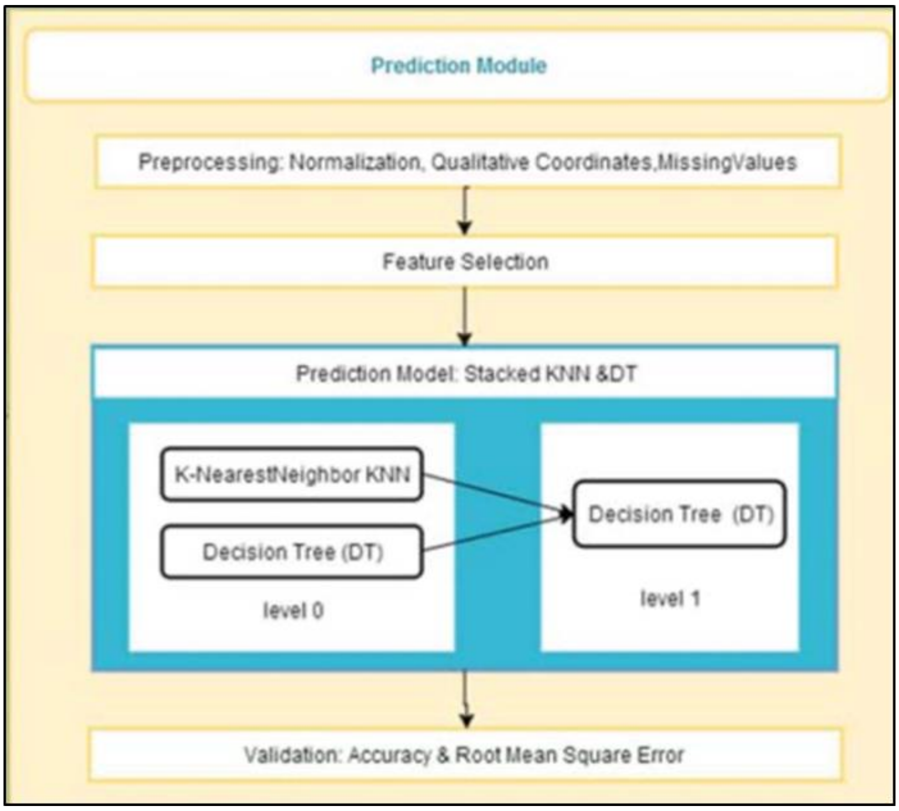

Prediction Module
Preprocessing: Normalization, Qualitative Coordinates,MissingValues
Feature Selection
Prediction Model: Stacked KNN &DT
K-NearestNeighbor KNN
Decision Tree (DT)
level 0
Decision Tree (DT)
level 1
Validation: Accuracy & Root Mean Square Error

Figure 2.6 Example of module based on stacking [17]

In recent years, the neural network is a popular approach to solve this type of problem. Thonglek, et al. [18] used Long Short-Term Memory (LSTM) which is a recurrent neural network as a model. This model was designed to learn historical trends in time series since they have memory cells. Memory cell size is an important parameter in LSTM. The memory cell was used to define how much previous data will be held in the model. Another advantage says that the model aims to solve the vanishing gradient problem by using gates.

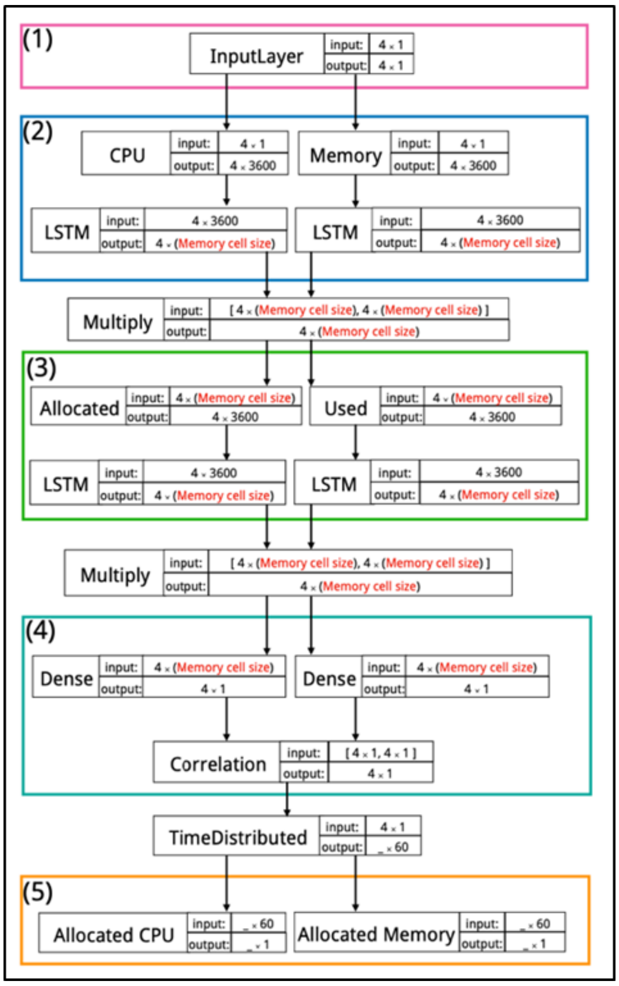


Figure 2.7 Proposed prediction model based on LSTM [18]

Figure 2.7 shows an example of an architecture based on LSTM. The design is composed of five layers: the input layer, first LSTM layer, second LSTM layer, fully connected layer, and output layer. Input data is 4 x 1 dimensional matrix of significant features consisting of requested CPU resource, requested memory resource, used CPU resource, and used memory resource. After that layers 2 and 3 are used to find the correlation between CPU resources, memory resources, and separate cells into allocated and used resources. The last layer outputs a suitable CPU and memory allocation.

Previous research can be categorized into two main types, primitive approaches and modern approaches. Primitive approaches were popular in the past, for example, many types of regression models like Linear Regression and Support Vector Regression, clustering techniques like co-clustering, Support Vector Machine, Decision Tree and Random Forest, K-nearest Neighbor and also other approaches like Linear Algebra.

When one model is not effective enough to show the result, the ensemble technique combines weaker supervised learning to reduce bias between each model, reducing both variance and overfitting.

Conversely, there are the modern approaches implemented. These approaches use new algorithms within machine learning, with the main difference between primitive and modern approaches being that modern approaches consist of various layers for analyzing and learning incoming data. Examples of modern approaches include Multilayer Perceptron and Long Short-Term Memory. An important distinction between primitive and modern approaches is the computing time and result from those models. There will be some trade-offs to each side, better results and accuracy costs computing time.

The summary of our survey on related works that we presented previously is shown in Table 2.1. The summary table below shows the different methods with different evaluations, pros, and cons.

Table 2.1 The summary scope of survey related work

Year	Paper	Method	Evaluation	Pros	Cons
2012	Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach [19]	- Co-clustering technique - HMM	- Plot percentages of correct, under- and over-prediction - Compare with the SPAR model, single time-series based prediction method	Group common VMs together with common workload pattern so it will be easier to estimate the future workload	Using HMM may take time to train a model
2013	Predicting Cloud Resource Provisioning using Machine Learning Techniques [10]	- Support Vector Regression - Neural Networks (Multilayer Perceptron) - Linear Regression	- Mean Absolute Percentage Error (MAPE) - Root Mean Square Error (RMSE) - PRED 25	- Use feature selection technique to select important features - SVM has the advantage of reducing the problems of over-fitting or local minima.	- MLP may take time to train a model - The prediction interval is 9-12 minutes which is not generalize
2014	Predicting Utilization of Server Resources from Log Data [4]	Linear Algebra	Average Absolute Error	This framework does not need to use any instrumentation, benchmarking, or load testing which causes unfeasible, time-consuming, or very intrusive.	It helps in the scenario where there are few data points available for a system running with a light workload
2015	Workload Estimation for Improving Resource Management Decisions in the Cloud [6]	- Clustering (Dynamic time wrap) - Estimate workload	- Overestimation ratio (OER) summarizes the overestimation ratio (OER) of our scheme (= (estimated – actual) / actual) - Percentage of savings of resources	In DTW distance, the data point of one time series is compared to the corresponding data point as well as the neighboring data points (increase a chance to find the similarity)	Not good for the new information, it tends to lose information from heavy outliers since those have to be put in a cluster as well.

Table 2.1 The summary scope of survey related work (Cont'd)

Year	Paper	Method	Evaluation	Pros	Cons
2017	Utilization based prediction model for resource provisioning [13]	- Linear Regression - Support Vector Regression - Multilayer Perceptron	- Mean Absolute Percentage Error (MAPE) - Root Mean Square Error (RMSE) - Mean Absolute Error (MAE), PRED 25	MLP uses the techniques of feed forwarding and backpropagation and it can deal with linear and non-linear separable data which can give better accuracy	MLP has too many parameters, inefficient and high computational times
2018	Prediction Of Cloud Computing Resource Utilization [20]	Ensemble Based Workload Predictor (KNN and DT)	- Accuracy, F-measures, Recall, Root Mean Square Error (RMSE) - Compare with other baseline models (K Nearest Neighbor, Neural Network, Decision Tree, Support Vector Machine, and Naïve Bayes)	- Ensemble help improve accuracy rather than single learner - Stacking model reduce variance, bias, and overfitting	Difficult to understand
2019	Improving Resource Utilization in Data Centers using an LSTM-based Prediction Model [8]	Long Short-Term Memory (LSTM)	Measure how much resource utilization is improved by comparing predicted (CPU/ memory resources) allocation with the requested allocation	- Overcome the vanishing gradient problem by using gates -- Memory cell size is an important parameter in LSTM that specifies how much previous data to hold in the model	Longer training time

In this research, we are interested in estimating the computing resource for CERN to make the system more reliable. Since there are many approaches to estimation problems, we want to compare and contrast between primitive and modern approaches.

2.4 Theory

2.4.1 Regression Techniques

1. Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning with the regression technique. Regression is a model related to the relationship between variables and forecasting. The relationship is modeled using a linear predictor function. Linear regression predicts a dependent variable value (Y) based on a given independent variable (X). This regression technique finds out a linear relationship between X (input) and Y (output). The linear regression function is defined as follows:

$$Y = a + bX \tag{2.1}$$

From Equation 2.1, Y is the dependent variable value and X is the independent or input variable. The slope of the line is b and a is the intercept point. Consider the resource estimation problem where X can be the number of requests, the number of nodes, or ram usage; furthermore, Y would be defined as CPU usage.

2. Support Vector Regression

Support Vector Regression is an approach based on Support Vector Machine. Support Vector Machine is a model used to classify 2 groups of data by creating a hyperplane that best separates two sets of data as shown in Figure 2.8. The best hyperplane is the one that maximizes the margins of both sets of data. Support Vector Regression uses the same principle as Support Vector Machine but the difference is in the hyperplane used. Support Vector Regression uses hyperplane as a line for predicting continuous or target values.

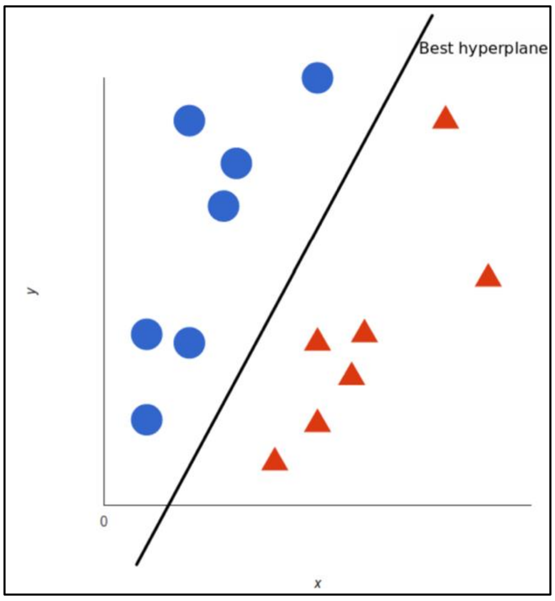
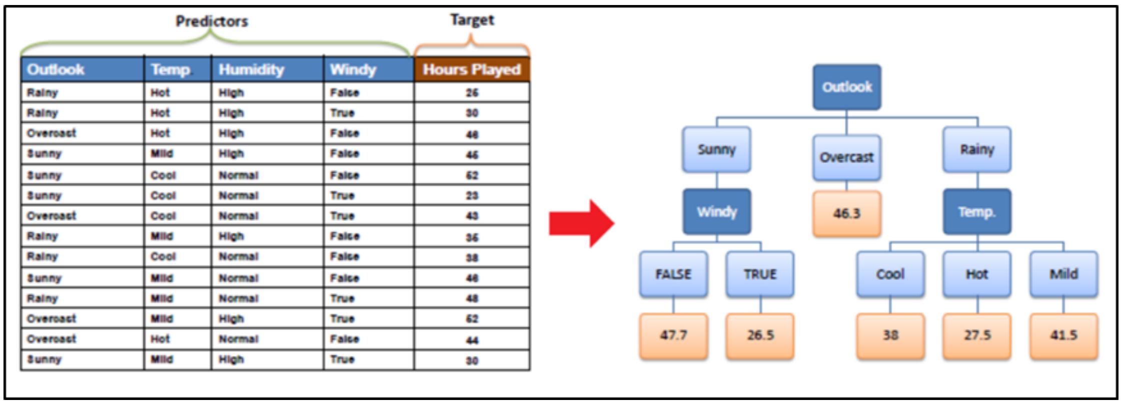


Figure 2.8 Hyperplane that was created in Support Vector Machine model [21]

2.4.2 Decision Trees

Decision Trees, used in machine learning, data mining, and statistics, can be used in both classification and regression problems in the form of a tree structure. It includes root nodes, internal nodes, and leaf nodes. An example of a decision tree is shown in Figure 2.9. The main point of the decision tree is to build a model that can predict the target value.



Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	62
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	36
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	62
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

Figure 2.9 Example of decision tree [22]

ID3 is an example of a decision tree learning algorithm. ID3 is used to build a decision tree for regression by replacing Information Gain with Standard Deviation Reduction. Standard Deviation Reduction is to decrease Standard Deviation after splitting the dataset. Therefore, building a decision tree is to find an attribute that gives the highest Standard Deviation Reduction or the most homogeneous branches. The other learning algorithm is CART or Classification and Regression Tree. This algorithm uses the Gini Index to split the data.

2.4.3 K-Nearest Neighbors

From [16], K-Nearest Neighbors is one of the machine learning algorithms that can be used in both classification and regression. This algorithm uses feature similarity to predict the values of new data points by assigning a value based on how close the new data points are to the points of the training dataset. The methods to find the distance between points are Euclidean Distance, Manhattan Distance, and Hamming Distance. When we use KNN for regression, the training data will be in the form of $\{x_i,y_i\}$, x_i is an attribute value and y_i is a real value target. We then compute distance $D\{x, x_i\}$ to every training data. Afterwards, we select k closest instances $x_{i1}\ldots x_{ik}$ and label $y_{i1}\ldots y_{ik}$. The predicted value or $\hat{y}$ is based on the mean of k-most similar instances based on Equation 2.2

$$\hat{y} = \frac{1}{k}\sum_{j=1}^{k} y_i \tag{2.2}$$

2.4.4 Multi-layer Perceptron

From [15], Multilayer Perceptron is a feedforward artificial neural network model that consists of at least three layers: nodes, input layer, hidden layer, and the output layer. The layers are fully-connected. Each node in one layer connects with a certainly weigh wij to every node in the following layer and each of them is a neuron that uses a nonlinear activation function to map sets of input data onto a set of outputs. The common activation functions are both sigmoid functions which are described in Equations 2.3 and 2.4, where v_i is the weighted sum of the input connections. Multilayer Perceptron utilizes a supervised learning technique called backpropagation for training the network.

$$y(v_i) = \tanh(v_i) \tag{2.3}$$

$$y(v_i) = (1+e^{-v_i})^{-1} \tag{2.4}$$

2.4.5 Long Short-Term Memory

Long Short-Term Memory (LSTM) is one of the artificial recurrent neural network (RNN) architectures that is popular in the deep learning field. LSTM was created for solving the vanishing gradients problem. It can work with single data points and sequential data. This model is also suitable for time series data.

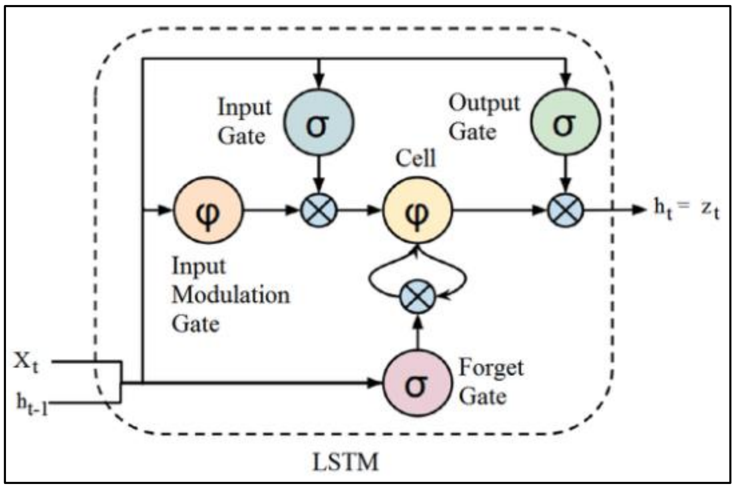


Figure 2.10 Long short-term memory architecture [23]

From Figure 2.10, the architecture of LSTM consists of a cell state, forget gate, input gate, output gate, and hidden state. The cell state is controlled by the forget gate and the input gate. The output from the forgetting gate will control the cell state in order to record or reject information. The input gate will manage which information should enter the cell state. There are two activation functions in this algorithm which are the sigmoid and Tanh functions.

2.4.6 Feature Selection Techniques

Feature selection, also known as variable selection, is a process that select important features from all features. Normally when we build a model we have to use many features to train the model but having irrelevant features for the training set will influence the model to learn irrelevant features which could decrease accuracy. Having correct features can reduce the overfitting and training time. Feature selection can be divided into 3 main methods: Filter-based, Wrapper-based, and Embedded. If we focus on regression techniques, feature selection methods that are frequently used to find the smallest set of features are Stepwise Selection, Forward Selection, and Backward Selection.

1. Filter-based Methods

Filter-based methods will create a metric to rank the features and then select the highest-ranking features. Metrics that are used for ranking consist of variance, chi-square, correlation coefficients, and information gain or mutual information. Details of this method is shown in Figure 2.11

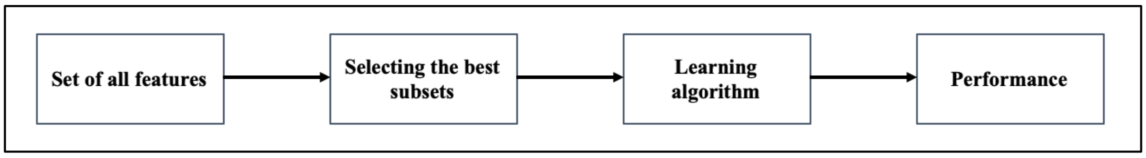


Figure 2.11 The process of filter-based method [24]

2. Wrapper-based Methods

Wrapper-based methods select features like a search problem. To solve the search problem, we have to use search strategies such as greedy search, forward selection, backward selection, heuristic feature subset selection, and stepwise selection. It is more generalized than the filter-based method and also gives higher performance accuracy. On the other hand, it has a high computation cost with a longer running time while also giving a higher chance of overfitting. The process of this method is shown in Figure 2.12

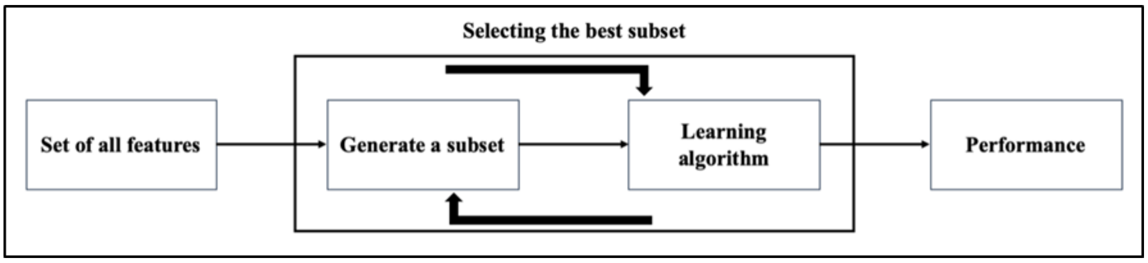


Figure 2.12 The Process of wrapper-based methods [24]

3. Embedded Method

Embedded methods use algorithms that have built-in feature selection methods. It performs feature selection during model training. Examples of built-in methods are regularization and tree-based methods. Regularization will add a penalty to the parameters of a model. Regularization is done to make the model more generalized, robust to noise, and avoid overfitting. There are three main types of regularization: lasso regularization, ridge regularization, and elastic nets. Tree-based methods can give better prediction and also provide other information that is called feature importance to inform which features are more significant on target variables. Figure 2.13 shows the steps in an embedded method.

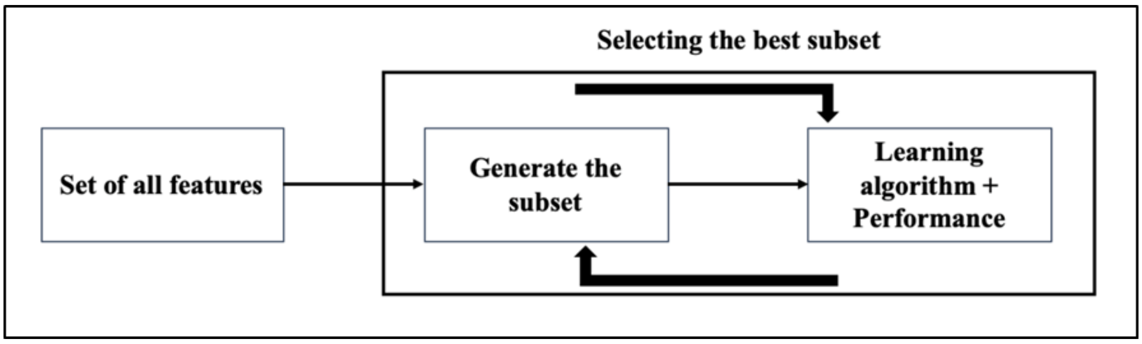


Figure 2.13 The process of embedded method [24]

4. Stepwise Selection

Stepwise selection will start with fitting the model with each predictor and return the model with the lowest p-value. Then, it increases the input predictor one by one from the remaining predictors. We keep iterating until we get a combination whose p-value is less than the threshold of .05.

5. Forward Selection

Forward selection is similar to the stepwise selection, it will begin with no variables in the first step. After that, it adds the most significant variable until there are no variables that meet the criterion or until none improves the model to a statistically significant extent.

6. Backward Selection

Backward selection is different from the forward selection in that this method is going to start with all variables. It then removes the one which has the highest p-value, for example the variables that exceed the threshold limit of .05. This method will keep repeating the process until no further variables can be deleted without a statistically significant loss of fit.

2.4.7 Evaluation Metrics

Evaluating model accuracy is an important part of creating machine learning models. A basic concept is to compare actual and predicted values. There are five main error metrics used to evaluate models which are: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Square Error (MSE), Root Mean Square Error (RMSE), and R-Squared (R^2) Score.

1. Mean Absolute Error (MAE)

Mean absolute error is an average of absolute error between 2 continuous variables, x_i, and y_i, where xi is the actual value and yi is the predicted value. Absolute error is the absolute value of the difference between the predicted value and actual value. In other meaning, it informs how big an error we can expect from the forecast on average. A formula to calculate mean absolute error is shown in equation 2.5 where n is the number of data.

$$\text{MAE} \quad = \frac{1}{n}\sum_{i=1}^{n} |x_i - y_i| \tag{2.5}$$

2. Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error is similar to the mean absolute error. It is used to measure how accurate the forecast system is. This method measures accuracy as a percentage. A formula is shown in Equation 2.6 where n is the number of data points, x_i is the actual value, and y_i is the predicted value.

$$\text{MAPE} \quad = \frac{1}{n}\sum_{i=1}^{n} \frac{|x_i - y_i|}{x_i} \tag{2.6}$$

3. Mean Square Error (MSE)

Mean square error measures the average of the squares of the difference between actual and predicted values. It is the measure of how close a fitted line is to data points. The smaller the mean square error value, the closer to the fitted line. A formula for mean square error follows Equation 2.7 where n is the number of data points, x_i is the actual value, and y_i is the predicted value.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{2.7}$$

4. Root-Mean-Square Error (RMSE)

Root mean square error is a square root of mean square error. It measures the quality of fit between actual and predicted values. Root mean square error is a method that is usually used for measuring the goodness of fit of generalized regression models. Equation 2.8 shows a formula of root-mean-square error.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.8}$$

5. R-Squared (R^2) Score

R-squared or the coefficient of a determinant is a method to measure how close the data is to the fitted regression line or it indicates how much variation of a dependent variable is explained by the independent variables in a regression model. The value of R-squared is between 0 to 1. The higher the R-squared value, the better the model fits data. The equation for the R-squared is shown in Equation 2.9 where yi is the actual value, $\hat{y}$ is the predicted value, and $\bar{y}$ is the mean value of y.

$$R^2 = 1 - \frac{\Sigma(y_i - \hat{y})^2}{\Sigma(y_i - \bar{y})^2} \tag{2.9}$$

6. Adjusted R-Squared (R^2) Score

Adjusted R-squared is a modified version of normal R-squared score. It has been adjusted for the number of predictors in the model. It identifies the percentage of variance in the target field that is explained by the input or inputs. Adjusted R-squared is calculated by dividing the residual mean square error by the total mean square error (which is the sample variance of the target field). The result is then subtracted from 1. Adjusted R-squared is always less than or equal to R-squared. A value of 1 indicates a model that perfectly predicts values in the target field. A value that is less than or equal to 0 indicates a model that has no predictive value. The equation for the adjusted R-squared is shown in Equation 2.10 where n is the number of points in your data set and k is the number of independent variables in the model, excluding the constant.

$$\text{Adjusted } R^2 = 1 - \left[\frac{\left(1\text{-}R^2\right)(n\text{-}1)}{(n\text{-}k\text{-}1)}\right] \tag{2.10}$$

CHAPTER 3 METHODOLOGY

In this chapter, we describe our methodology by mentioning the overall framework followed by the detailing each proposed step and design of the experiment. Lastly, the evaluation metric for comparing various models is presented.

3.1 Overall Framework

The objective of this research is to study which system metrics are factors affecting our resource estimation for the ALICE O^2 logging system. Moreover, we would like to develop machine learning models for estimating the computing resources for the Logstash server which is a part of the logging system. Based on the history of the system log data in ALICE, we propose different machine learning techniques including Random Forest Regression, Multiple Linear Regression, and Multiple-Layer Perceptron. After that, the results and performance of the estimation models will be collected and compared. Figure 3.1 shows the overall framework of our research. The framework consists of two key parts which are data preprocessing and modeling.

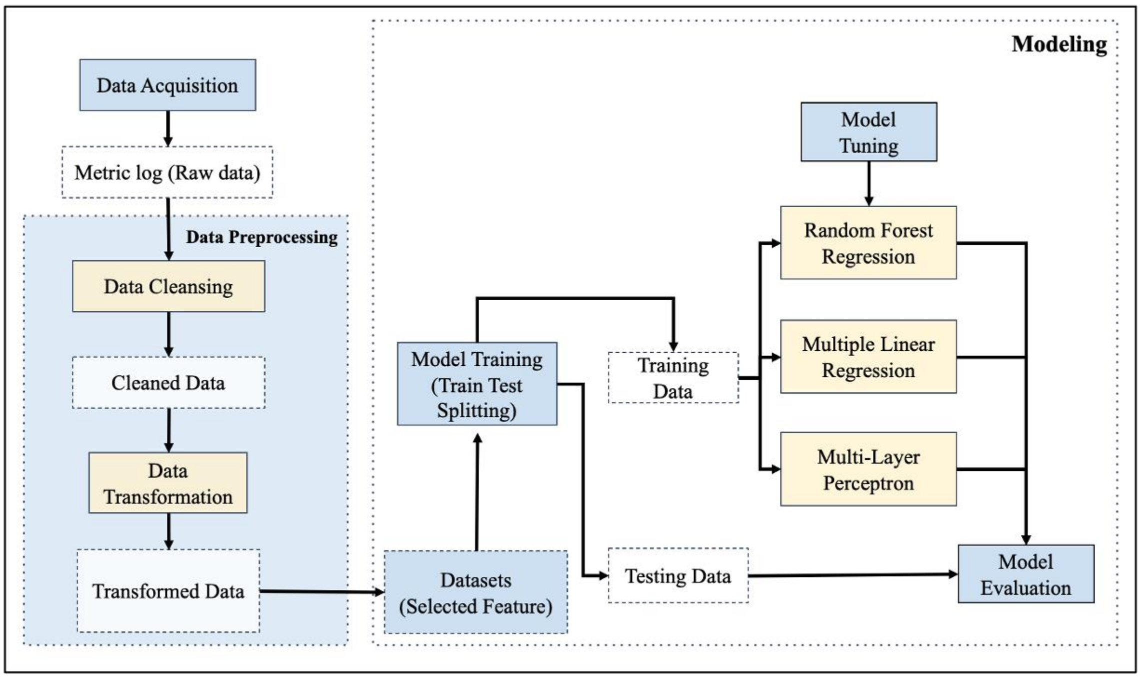


Figure 3.1 Overall framework consisting of data preprocessing and modeling parts

We divide the process into two main parts: data preprocessing and modeling. In the data preprocessing, we collect the metric log from the logging system. After that, we start data exploration to understand the data characteristics. Then, we need to satisfy whether the data is appropriate to the model or not. If not satisfied, we have to do the data transformation to transform the data before the feature selection step. For the feature selection step, first, we select the features that are usually used for resource estimation. After that, we use Pearson's correlation to select those features again to train the estimation models. Before building the estimation model, the dataset will be split into 2 parts, a training set of 80% and a testing set of 20%. In the modeling part, we use a grid search cross-validation method to find a good hyperparameter for model tuning. After that, we retrain the models again with training data and new hyperparameters from the model tuning step. Finally, the performance of models will be evaluated by using the Mean Square Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2) methods.

3.2 Data Acquisition

In our testbed, we created all instances (i.e. FLP and ELK servers) by using Linux operating system with 4 VCPUs, 7.3 GB memory, and 40 GB disk space. Figure 3.2 shows the architecture of the logging system that we have designed according to the ALICE O_2 architecture. We install Filebeat on every node for shipping log data to the ELK server. The main part that we focus on is at the Logstash node. We also install Metricbeat on this node to collect the metric log data coming out from the FLPs. When Filebeat in each agent node (i.e. FLP) ships the log data to the Logstash node, the Logstash node will start processing those log data and send them to the Elasticsearch node. The resources in the Logstash node will be collected by Metricbeat.

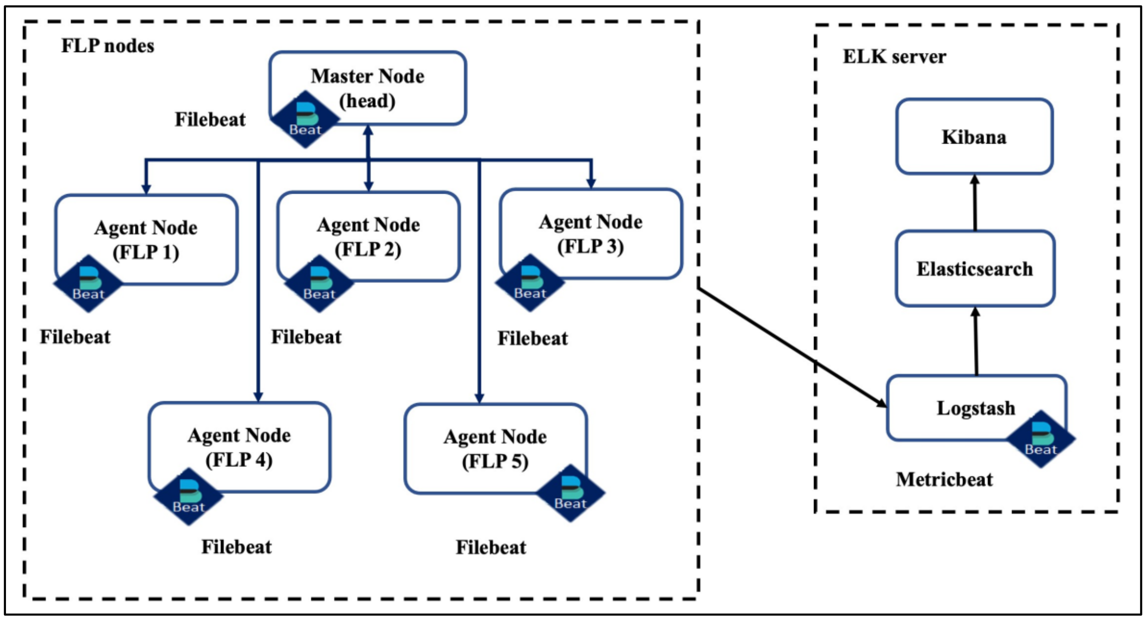


Figure 3.2 Logging system architecture

3.3 Dataset

In this work, the status of computing metrics was logged and collected by Metricbeat on the Logstash node during different time intervals. Particularly, this dataset was collected when resources such as CPU size, memory size, and disk space size are limited (i.e. no resource scaling during the collection process). Then, CPU usage, memory usage, disk space usage were measured. The number of the FLP nodes is varied. The metrics measured are as follows:

1. CPU Usage (Percentage of CPU time spent in user space, in percentage)
2. Memory Usage (Memory allocated by applications and the operating system, in percentage)
3. Memory Free (System memory free which available for applications and the operating system, in gigabytes)
4. Disk Space Usage (Disk space usage, in percentage)
5. Disk Space Free (Disk space free, in percentage)

Some examples of the features that we collected from Metricbeat are shown in Table 3.1

Table 3.1 Examples of features collected from ELK node

Column Name	Description	Type	Example
system_cpu_cores	Number of CPU cores	integer	32
system_cpu_user_pct	Percentage of CPU time spent in user space	float	16.48
system_cpu_irq_pct	Percentage of CPU time spent servicing and handling software interrupts.	float	0
system_memory_total	Total memory	int	126744059904
system_memory_actual_free	Actual free memory in bytes	int	112922484736
system_memory_actual_used_bytes	Actual use of memory in bytes	int	13821575168
system_filesystem_free	The disk space available in bytes	long	558624768
system_filesystem_used_bytes	The used disk space in bytes	long	11718656
system_diskio_io_time	Total number of milliseconds spent doing I/Os	int	51
system_diskio_read_bytes	Total number of bytes read successfully	int	1536
system_diskio_read_time	Total number of milliseconds spent by all reads	int	53
system_load_5	CPU load averages last 5 min	float	26.07
system_process_summary_running	Number of running processes on this host	int	2

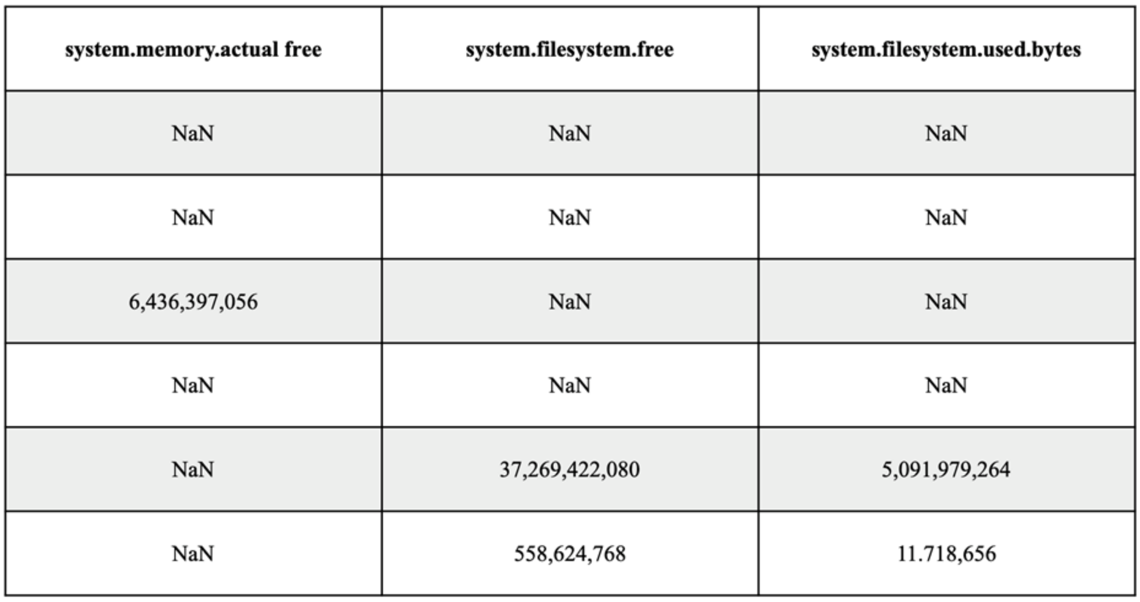
The reason that we did not use other features was because we could not control the value of other features. Thus, we decided to use only features that we could set the values of in order to answer the research question proposed on resource estimation.

3.4 Data Preprocessing

In the data preprocessing step, we perform data cleansing, data transformation, and feature extraction to delete unnecessary data which can improve the quality of data and also improve the generalizability of the models.

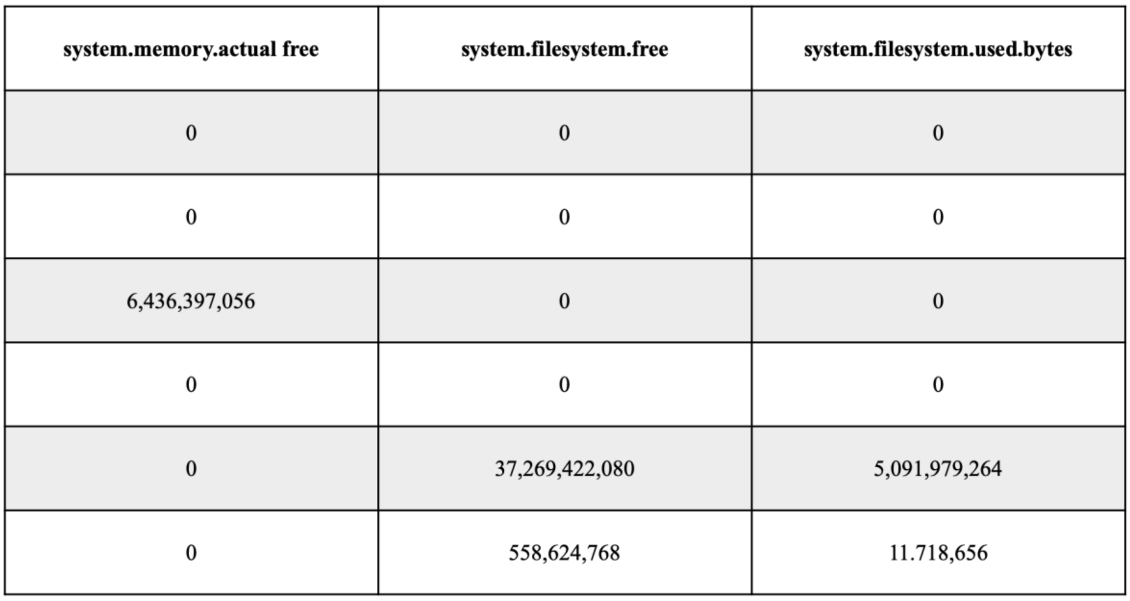
3.4.1 Data Cleansing

After the dataset is collected, we start with changing types of input data into their appropriate data types such as integer and float. We also remove unnecessary special characters or symbols before using them in the model and handle any missing values, null, or NaN values as shown in Figures 3.3 and 3.4.



system.memory.actual free	system.filesystem.free	system.filesystem.used.bytes
NaN	NaN	NaN
NaN	NaN	NaN
6,436,397,056	NaN	NaN
NaN	NaN	NaN
NaN	37,269,422,080	5,091,979,264
NaN	558,624,768	11.718,656

Figure 3.3 Example of data before filling missing value

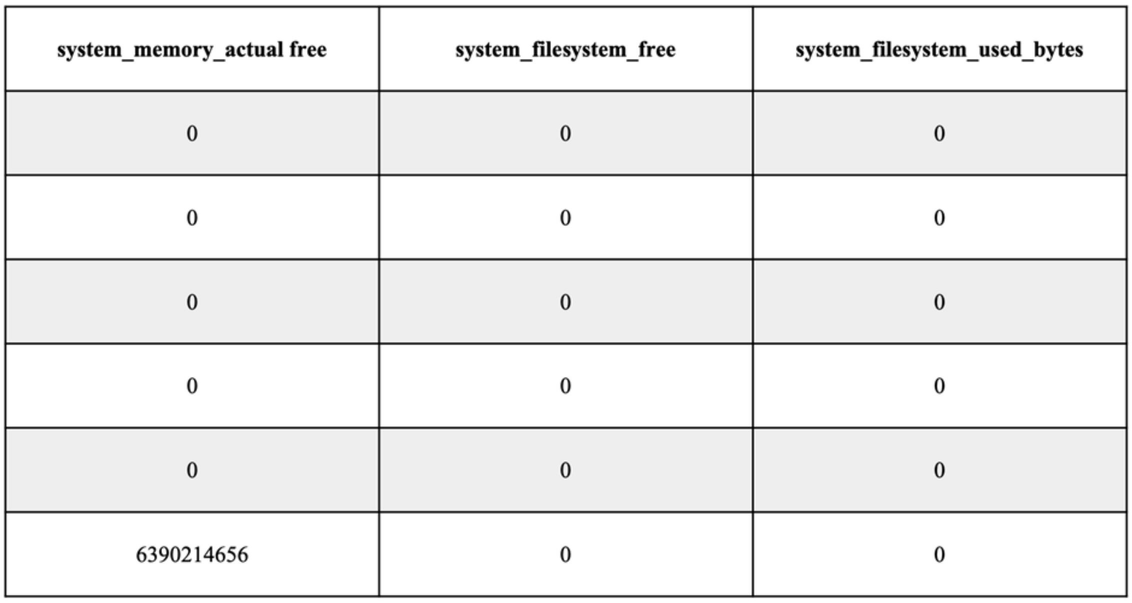


system.memory.actual free	system.filesystem.free	system.filesystem.used.bytes
0	0	0
0	0	0
6,436,397,056	0	0
0	0	0
0	37,269,422,080	5,091,979,264
0	558,624,768	11.718,656

Figure 3.4 Example of data after filling missing value

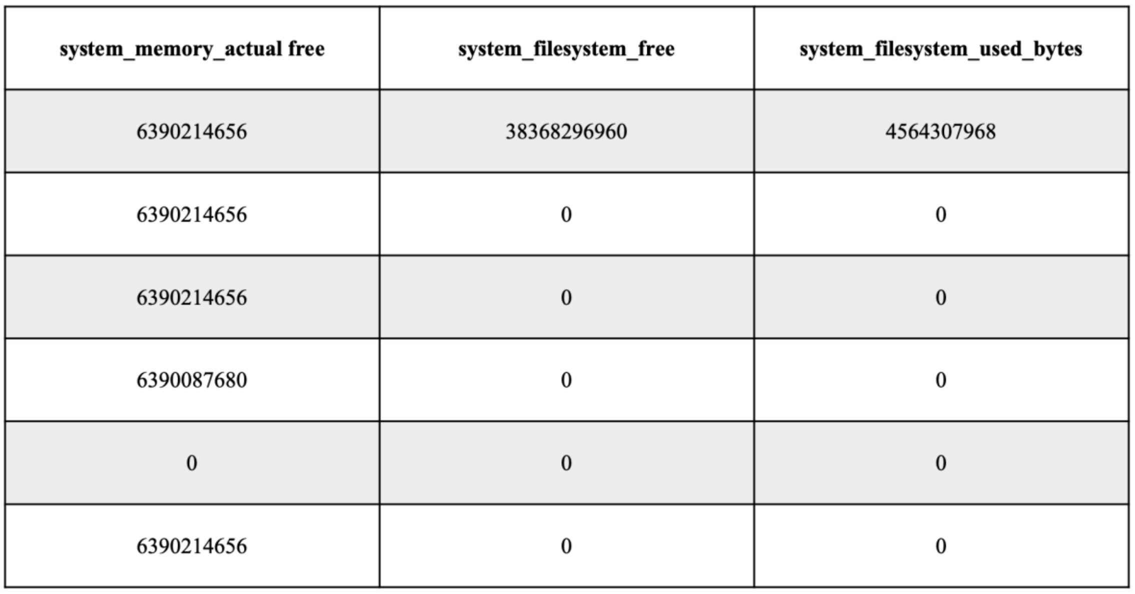
3.4.2 Data Transformation and Feature Extraction

We use average, sum, and maximum operations to aggregate our data. The time column is dropped because in our experiment we did not focus on time. After that, we create new columns (i.e. features) which are about the number of agent nodes and the number of workflows. The example of data aggregation is shown in Figures 3.5 and 3.6.



system_memory_actual free	system_filesystem_free	system_filesystem_used_bytes
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
6390214656	0	0

Figure 3.5 Example of data before aggregation



system_memory_actual free	system_filesystem_free	system_filesystem_used_bytes
6390214656	38368296960	4564307968
6390214656	0	0
6390214656	0	0
6390087680	0	0
0	0	0
6390214656	0	0

Figure 3.6 Example of data after aggregation

3.4.3 Feature Selection

To train and improve the model accuracy, we do some feature selection steps to eliminate some features that can reduce the model performance. In this study, we use Pearson's correlation coefficient to measure the relationship between 2 continuous variables. Since we want to estimate the quantitative dependent variable by using 2 or more quantitative independent variables, Pearson's correlation is suitable. We use this statistical method as critereon for the feature selection process by selecting only the features which provide a coefficient value of more than 0.25 in both positive and negative relationships. We also do the feature selection by using t-tests and find p-value for Pearson's correlation in order to provide conclusions for the hypotheses.

3.5 Model Training

In this work, we use 3 estimation models which are Random Forest Regression, Multiple Linear Regression, and Multi-Layer Perceptron. Each of them shows their ability differently, multiple linear regression aims to discover the relationship between target and predictors. Random forest regression, one of the decision tree methods, also presents the relationship between target and predictors in the form of a tree structure that is easy to read. It also can handle a large number of variables with a small number of observations. The last one is the neural network, it is a complex model that consists of 3 layers. Those complex parameters will help to improve the accuracy of estimating the results. Additional information has been explained in the section below.

3.5.1 Random Forest Regression

Random forest is a supervised ensemble learning algorithm that is not only used for classification problems but is also used for regression problems. It is made up of multiple trees to make a forest. The algorithm creates many decision trees and where each tree will be trained with different proportions of training data until all of them get the prediction result. Finally, it calculates the mean prediction of the individual tree and finds the final best solution.

In the training process, random forests use bagging or bootstrap aggregation for random sampling a small subset of data from the whole dataset to use in each decision tree in the forest. Since the data in this research consists of more than 25 features, including percentage of CPU time spent in user space, actual use of memory, the total number of milliseconds spent doing I/Os, the total number of milliseconds spent by all reads, CPU load averages last 1, 5 or 15 minutes, etc., sampling allows the random forest to handle these features without deletion. Random forests require only a few data preprocessing steps to be done so the data does not need to be rescaled or transformed. Random forest performs well with high-dimensional data since we are working with subsets of data. When we have many features in the dataset, this model can give the result whose features are important to the target. The advantage of using this method is to reduce overfitting and, due to the averaging outcome process, improve stability and accuracy.

After the training process, the random forest method uses mean predictions to find the best result of the random forest method. Mean prediction finds the average of all results which was predicted from each tree. Figure 3.7 shows an example of Random Forest Regression.

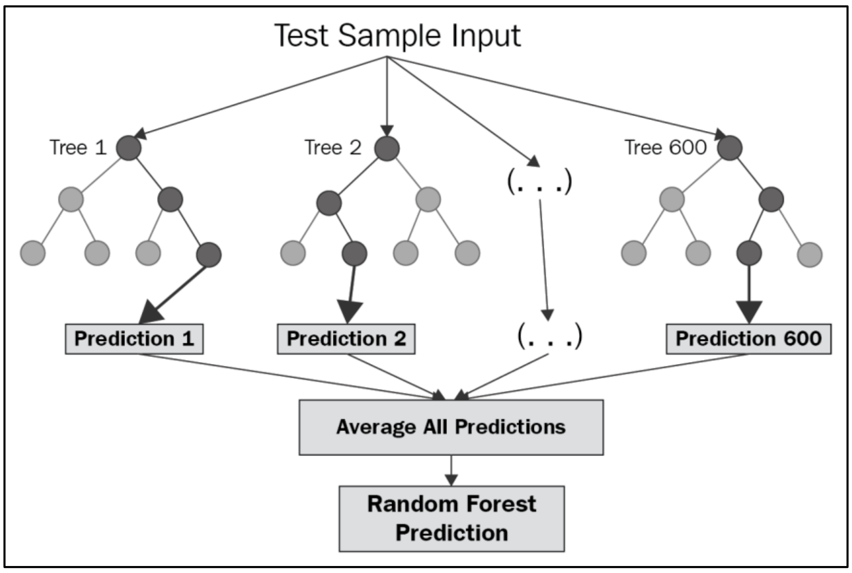


Figure 3.7 Example of random forest regression [25]

3.5.2 Multiple Linear Regression

Multiple linear regression is a statistical technique that uses several independent variables to predict the outcome of the dependent variable. The data that is good for using a multiple linear regression model is the data that have a linear relationship between the independent variables and dependent variables and also has a normal distribution pattern. The characteristic of each variable in this research has a linear relationship with each other. This model creates a relationship of variables in the form of a straight line or a regression equation that best approximates all the data points. Multiple linear regression can be shown in Equation 3.4 where y is a dependent variable, x_i is an independent variable, β_0 is the y-intercept, β_p is the slope coefficient and ϵ is the model error.

$$y = \beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots+\beta_p x_{ip}+\epsilon \tag{3.4}$$

An advantage of multiple linear regression is mostly about defining the relationship between 2 or more variables. Since this research consists of many independent variables, it is essential to identify the strength of the effect that independent features impact to the dependent variable and also help us to understand how much the dependent variable (memory needed) will change when we change independent variables, for example, actually used memory, actual free memory, number of FLP nodes, and number of log coming into Logstash. Moreover, multiple linear regression can predict the trend or future values and also get an estimated point.

3.5.3 Multi-layer Perceptron

Multi-Layer Perceptron is a feed-forward artificial neural network model that consists of at least 3 layers of nodes. First, the input layer is made of the number of perceptions equal to the number of input features, followed by at least one hidden layer and an output layer. In the hidden or output layer, there is one perceptron in the case of regression. The layers are fully connected. Each node in one layer connects with a certain weight to every node in the following layer. Each layer has an activation function such as Softmax (Equation 3.5 where K is probabilities proportional to the exponentials of the input numbers), Sigmoid (Equation 3.6), and ReLU (Equation 3.7). Each activation is better performed with a different kind of dataset. Moreover, many functions can be used between each layer such as dropout which can reduce overfitting.

$$\sigma(x)_i = \frac{e^{x_i}}{\Sigma_{j=1}^{K} e^{x_j}} \tag{3.5}$$

$$S(x) = \frac{1}{1+e^{-x}} \tag{3.6}$$

$$f(x) = \max(0,x) \tag{3.7}$$

3.6 Experimental Design

In this section, we present the experimental design in this research. We collect the data when the number of agent nodes (FLP nodes) in the system is varied. The number of agents is varied as 1, 3, 4, and 5 agent nodes with different 1, 2, and 3 workflows running on the agent nodes. Since our collected data consists of continuous dependent variables which are CPU usage, memory usage, disk space usage, and multiple independent variables such as the number of agent nodes, free memory, etc., we decided to use the regression model to estimate these dependent variables. After the data preprocessing step, three estimation models are used to find the estimated results together with a comparison among these models whether the log data is appropriate to each model or not. In the experiment, we also focus on the outliers in the dataset. We decided to clean up data from outliers by using interquartile range (IQR) and select data in the range of first quartile (Q1) and third quartile (Q3).

In addition, each learning algorithm has different hyperparameters that we have to adjust to improve the accuracy. These hyperparameters are defined below.

1. Random forest regression hyperparameters:
    - max depth is maximum depth of the tree.
    - n-estimators is the number of trees that we want to build before taking the maximum voting or averages of predictions.
    - min samples split is the minimum number of samples required to split an internal node.
    - Min samples leaf is the minimum number of samples required to be at a leaf node.

2. Multi-layer perceptron parameters:
    - Learning rate init is the initial learning rate used.
    - Hidden layer sizes is The ith element represents the number of neurons in the ith hidden layer.
    - Activation is activation function for the hidden layer which are, 'logistic' or logistic sigmoid function, 'tanh' or hyperbolic tan function and 'relu' or rectified linear unit function.
    - Max iteration is maximum number of iterations.
    - Solver is the solver for weight optimization which are, 'lbfgs' is an optimizer in the family of quasi-Newton methods, 'sgd' refers to stochastic gradient descent and 'adam' refers to a stochastic gradient-based optimizer.

We will apply the grid search method to find the best hyperparameter of each model. The results of our experiments are presented in Chapter 4.

3.7 Model Evaluation

To evaluate the performance of models, we use statistical metrics including R-squared (R^2), Adjusted R-squared (R^2), Mean Absolute Error (MAE), and Mean Square Error (MSE). R-squared and Adjusted R-squared are measures that presents the proportion of variance for the dependent variable and independent variables in the regression model. It explains the correlation between dependent and independent variables in the range of 0 to 1 for example if the R-squared of the model is 0.8 means that approximately 80 percent of the output can be explained by the input. To calculate R-squared we need to follow Equation (2.9) from section 2.4.7.

Mean Absolute Error or MAE is used for measuring the accuracy of continuous variables. This metric is used to find the absolute difference between forecast and observation values. The calculation was presented in section 2.4.7 and Equation (2.5). Another metric is Mean Square Error (MSE), it uses to measure the average squared error of the prediction result. For both MAE and MSE, the values are inversely proportional to model performance. Therefore, we have to minimize these metrics.

CHAPTER 4 EXPERIMENTAL RESULTS

In this chapter, we present the results from the experiments according to the proposed framework. Firstly, we show the information about data collection, data preprocessing and feature selection experiments. We explain the training and test datasets and follow the relation between CPU, memory, and disk space usage. Secondly, the detail of the model performance comparison is shown in Section 4.3. We then discuss the results between five different preprocessed datasets and three different regression models in Section 4.4. Lastly, from the results, we can formulate regression equations for estimating the computing resources which are CPU usage, memory usage, and disk space usage as shown in Section 4.5.

4.1 Data Collection, Data Preprocessing, and Feature Selection

50,352 records are collected by Metricbeat on the Logstash node from our logging system architecture. For the data cleansing process, since the dataset contains extreme values that are outside the range of what is expected and unlike the other data, we decided to perform an outlier cleansing step by using the interquartile range (IQR) together with dropping duplicated data and other cleansing processes as mentioned in Chapter 3. Moreover, the IQR method is appropriate with a dataset that is not normally distributed like our dataset.

4.1.1 Feature Selection with Pearson’s Correlation

From the Pearson’s correlation, the correlation coefficients between each feature are presented in Table 4.1. The result shows that ‘number of node (num_node)’ feature gets the lowest correlation coefficient which was -0.00604, -0.00403 and 0.00737 with system_cpu_user_pct, system_memory_actual_used_pct and system_filesystem_used_pct respectively. The cut-off value will be at +0.25 or -0.25 Therefore, the ‘number of node features’ will not be included for training our models.

Table 4.1 Pearson's correlation coefficient between each feature

	system_cpu_ user_pct	**system_memory_ actual_used_pct**	**system_filesystem _used_pct**
system_cpu_user_pct	-	0.393570719	0.698185749
system_memory_actual_ used_pct	0.393570719	-	0.57612405
system_memory_actual_ free	0.569776973	-	0.666020046
system_filesystem_free	0.593083769	0.742371526	-
system_filesystem_used_pct	0.698185749	0.57612405	-
num_node	**-0.006041439**	**-0.004030482**	**0.007369189**
num_workflow	-0.728292637	-0.291515985	-0.645816407

Moreover, we also perform the t-statistical testing and find p-value for Pearson's correlation and conclude with the hypothesis. The hypotheses were:

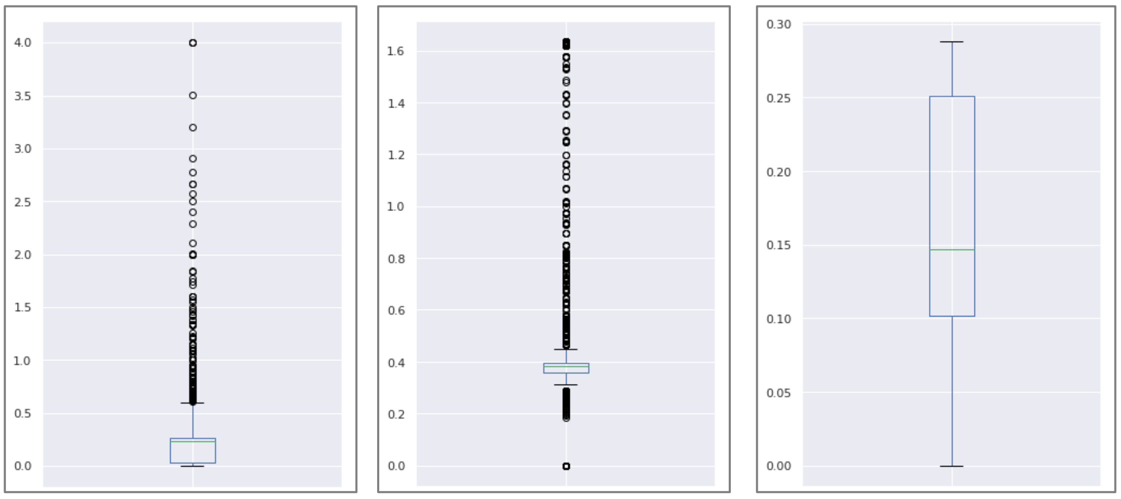
Hypothesis (H_0):	There is a significant correlation between 2 variables ($p\text{-value} < 0.05$)
Null Hypothesis (H_1):	There is no significant correlation between 2 variables ($p\text{-value} > 0.05$)

The results in Table 4.2 shows that p-values between number of nodes and each dependent variable, system_cpu_user_pct, system_memory_actual_used_pct and system_filesystem _used_pct respectively are 0.1752, 0.3658 and 0.0982 respectively. We can conclude from the hypothesis that number of nodes has no significant correlation with system_ cpu_user_pct, system_memory_actual_used_pct and system_filesystem_used_pct. Although, the feature that we deleted may not affect to the performance of multiple linear regression but it may affect to the performance of random forest regression and multi-layer perceptron. Since we have to use the same dataset in every model, we decided to delete feature 'number of node' out.

Table 4.2 The p-values for Pearson's correlation test by using t-statistical test between each features

	system_cpu_user_pct	**system_memory_actual_used_pct**	**system_filesystem_used_pct**
system_cpu_user_pct	-	0	0
system_memory_actual_used_pct	0	-	0
system_memory_actual_free	0	-	0
system_filesystem_free	0	0	-
system_filesystem_used_pct	0	0	-
num_node	**0.1752**	**0.3658**	**0.0982**
num_workflow	0	0	0

Since we designed the experiment to cut the outliers of dependent variables out and do the train test split. But before we cut outliers and do the train test split, we start with plotting the boxplots of dependent features which are system_cpu_user_pct, system_memory_actual_used_pct and system_filesystem_used_pct.



system_cpu_user_pct system_memory_actual_used_pct System_filesystem_used_pct

Figure 4.1 Boxplots of dependent variables

From Figure 4.1, we decide to cut the outliers by using the rule saying that a data point is an outlier if it is more than 1.5IQR above the third quartile or below the first quartile or it means the data point is between Q1–1.5IQR and Q3+ 1.5IQR.

After finishing the data cleansing, data transforming, and feature selection process, we split each preprocessed datasets after cutting the outliers for the experiments by using k-fold cross validation split and set number of folds equal to five. The number of records in each dataset and each fold are mentioned in Table 4.3. We then we split data for each fold as the training and test datasets. The training and test split percentage was 80 percent and 20 percent, respectively.

Table 4.3 The number of records in each preprocessed dataset

Experiment	Dataset	Processing Step	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5
CPU Usage	CPU Usage Dataset 1	Not remove outliers + not drop duplicated data	8236	8236	8236	8236	8236
	CPU Usage Dataset 2	Not remove outliers + drop duplicated data	7681	7681	7681	7681	7681
	CPU Usage Dataset 3	Remove outliers from feature ‘CPU usage’ + drop duplicated data	7656	7656	7656	7656	7656
	CPU Usage Dataset 4	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	6855	6855	6855	6854	6854
	CPU Usage Dataset 5	Remove outliers from all features + drop duplicate data	6847	6847	6847	6847	6847
Memory Usage	Memory Usage Dataset 1	Not remove outliers + not drop duplicated data	8287	8286	8286	8286	8286
	Memory Usage Dataset 2	Not remove outliers + drop duplicated data	7861	7860	7860	7860	7860
	Memory Usage Dataset 3	Remove outliers from feature ‘memory usage’ + drop duplicated data	6860	6860	6860	6860	6859
	Memory Usage Dataset 4	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	6859	6859	6859	6858	6858
	Memory Usage Dataset 5	Remove outliers from all features + drop duplicated data	6849	6849	6849	6848	6848

Table 4.3 The number of records in each preprocessed dataset (Cont'd)

Experiment	Dataset	Processing Step	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5
Disk Space Usage	Disk Space Usage Dataset 1	Not remove outliers + not drop duplicated data	8039	8039	8039	8039	8038
	Disk Space Usage Dataset 2	Not remove outliers + drop duplicated data	7647	7646	7646	7646	7646
	Disk Space Usage Dataset 3	Remove outliers from feature 'disk space usage' + drop duplicated data	7647	7646	7646	7646	7646
	Disk Space Usage Dataset 4	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	6855	6855	6855	6855	6855
	Disk Space Usage Dataset 5	Remove outliers from all features + drop duplicated data	6849	6849	6849	6848	6848

The reason that we split dataset into five different folds was that we wanted to compare and conclude which one will be an appropriate dataset for building the models. We try to delete outliers from only one feature, three features, and from all features and see that which one is the best dataset.

4.1.2 Relationship between dependent variables

In Figure 4.2, the relation between CPU, memory, and disk space usage is shown. When the range of memory and disk space usage changes, we can see that the range of CPU usage falls between 0% and 20%. This means that neither the memory nor disk space usage significantly affects the CPU usage. If we focus on the relation between memory and disk space usage, we will see that when the memory usage increases, the disk space usage also increases. This means that they are proportional to each other.

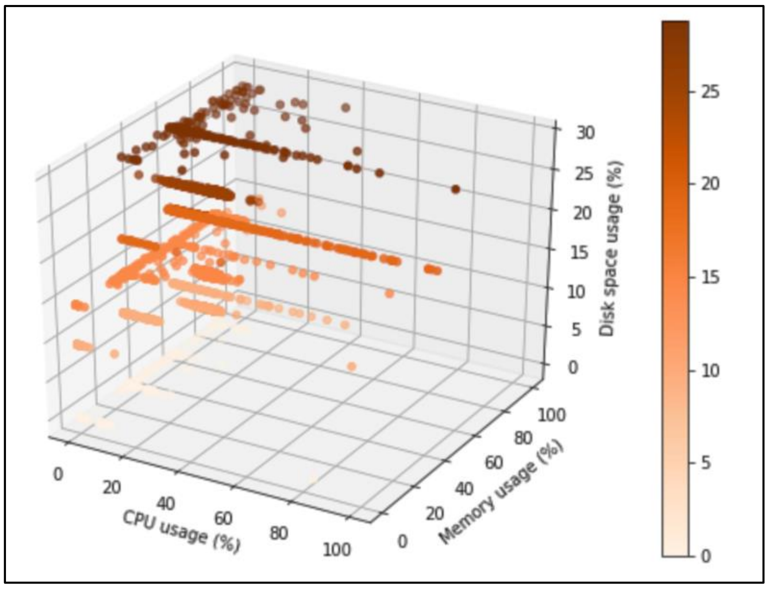


Figure 4.2 Relationship between CPU, Memory and disk space usage

4.2 Experiments for Model Performance Comparison

To estimate the three computing resources (CPU usage, memory usage, and disk space usage) three models were created with different input features as shown in Table 4.4.

Table 4.4 Different input features in each model

Model	Features
CPU usage model	Memory free
	Memory usage
	Disk space free
	Disk space usage
	Number of workflows
Memory usage model	CPU usage
	Disk space free
	Disk space usage
	Number of workflows
Disk space usage model	Memory free
	Memory usage
	CPU usage
	Number of workflows

4.3 Model Performance Results

We separate the experiments into two main experiments, the first experiment is to compare between five preprocessed datasets with no removed outliers, no removed outliers with duplicated data dropped, removed outliers of one feature which is the dependent variable of each dataset with dropped duplicated data, removed outliers of three features which are dependent variables of every dataset with dropped duplicated data, removing outliers of all features and dropping duplicated data to conclude that which one will be an appropriate one for building the models. For the first experiment, we trained five different preprocessed datasets with three regression models without any model tuning process. For the second experiment, we trained five different preprocessed datasets with three regression models with the model tuning process. All experiments in this section used only the test dataset for performance evaluation. We then compare the first and second experiments and discuss which preprocessed datasets will be appropriate. Lastly, we will compare models and find the best model for these datasets.

4.3.1 Model Performance Results for not tuning models

For the first experiment, we used multiple linear regression, random forest regression and multi-layer perceptron in both tuned and not-tuned models to find the differences between the five preprocessed datasets. We trained the model for each fold of each dataset and averaged them. Experiments 1.1-1.3 show the results from the untuned models.

Experiment 1.1: Used five different preprocessed CPU datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process. The performance measures are presented in Table 4.5.

Table 4.5 Performance comparison between different preprocessed CPU datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process

Models	Type of dataset	Training Time (Seconds)	Mean Square Error (MSE)	Mean Absolute Error (MAE)	R-squared (R^2)	Adjusted R-square
Multiple Linear Regression	Not remove outliers + not drop duplicated data	0.0060	0.0060	0.0242	0.37272	0.37222
	Not remove outliers + drop duplicated data	0.0026	0.0071	0.0243	0.3089	0.3083
	Remove outliers from 1 feature (CPU usage) + drop duplicate data	0.0027	0.0013	0.0192	**0.4554**	**0.4549**
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.0026	0.0012	0.0187	0.4163	0.4157
	Remove outliers from all features + drop duplicated data	0.0026	**0.0004**	**0.0120**	0.3892	0.3886
Random Forest Regression	Not remove outliers + not drop duplicated data	1.0645	0.0073	0.0220	0.2749	0.2743
	Not remove outliers + drop duplicated data	1.0244	0.0085	0.0148	0.3169	0.3167
	Remove outliers from 1 feature (CPU usage) + drop duplicate data	1.0036	0.0011	0.0177	**0.4059**	**0.4054**

Table 4.5 Performance comparison between different preprocessed CPU datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process (Cont'd)

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Random Forest Regression (Cont'd)	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.9481	0.0005	0.0142	0.3419	0.3416
	Remove outliers from all features + drop duplicated data	0.9460	**0.0004**	**0.0140**	0.3263	0.3261
Multi-layer Perceptron	Not remove outliers + not drop duplicated data	0.4041	0.0059	0.0247	0.3041	0.3036
	Not remove outliers + drop duplicated data	0.3336	0.0071	0.0241	0.2695	0.2689
	Remove outliers from 1 feature (CPU usage) + drop duplicate data	0.3328	0.0011	0.0196	**0.4347**	**0.4342**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.2815	0.0005	0.0137	0.3283	0.3284
	Remove outliers from all features + drop duplicated data	0.2751	**0.0004**	**0.0130**	0.3287	0.3287

The results in Table 4.5 show the model performance from different preprocessed CPU datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models. The result shows that in all models, both MSE and MAE values of “Remove outliers from all features + drop duplicated data” dataset was better than ones of the others but in terms of R-square and Adjusted R-square value, “Remove outliers from 1 feature (CPU usage) + drop duplicate data” was better than ones of the others.

Experiment 1.2: Used five different preprocessed memory datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process. The performance measures are presented in Table 4.6.

Table 4.6 Performance comparison between different preprocessed memory datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process

Models	Type of dataset	Training Time (Seconds)	Mean Square Error (MSE)	Mean Absolute Error (MAE)	R-squared (R^2)	Adjusted R-square
Multiple Linear Regression	Not remove outliers + not drop duplicated data	0.0025	0.0141	0.0362	0.8754	0.8753
	Not remove outliers + drop duplicated data	0.0027	0.0151	0.0376	0.8799	0.8798
	Remove outliers from 1 feature (memory usage) + drop duplicate data	0.0027	0.000004	0.0012	0.7844	0.7842
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.0024	0.00001	0.0013	0.7811	0.7812
	Remove outliers from all features + drop duplicated data	0.0023	**0.000002**	**0.0011**	**0.9158**	**0.9157**
Random Forest Regression	Not remove outliers + not drop duplicated data	0.3897	0.0051	0.0148	**0.9547**	**0.9546**
	Not remove outliers + drop duplicated data	0.3733	0.0067	0.0161	0.9472	0.94722
	Remove outliers from 1 feature (memory usage) + drop duplicate data	0.3378	0.000002	0.0007	0.8533	0.85318
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.3337	0.000002	0.0007	0.8900	0.88988
	Remove outliers from all features + drop duplicated data	0.3292	**0.000001**	**0.0006**	0.9280	0.92798

Table 4.6 Performance comparison between different preprocessed memory datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process (Cont'd)

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Multi-layer Perceptron	Not remove outliers + not drop duplicated data	0.6943	0.0117	0.0375	0.3072	0.3072
	Not remove outliers + drop duplicated data	0.6537	0.0128	0.0449	0.3671	0.3677
	Remove outliers from 1 feature (memory usage) + drop duplicate data	0.2868	**0.0001**	**0.0048**	**0.7154**	**0.7164**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.2646	0.0003	0.0078	0.6226	0.6245
	Remove outliers from all features + drop duplicated data	0.2635	0.0002	0.0094	0.3550	0.3582

The result in Table 4.6 shows the model performance from different preprocessed memory datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models. The results show that in multiple linear regression, all evaluation metrics of "Remove outliers from all features + drop duplicated data" were the best. For random forest regression, both MSE and MAE values of "Remove outliers from all features + drop duplicated data" were best. For R-square and Adjusted R-square value, "Not remove outliers + not drop duplicated data" performed best. For multi-layer perceptron, all evaluation metrices of "Remove outliers from one feature (memory usage) + drop duplicate data" were best.

Experiment 1.3: Used five different preprocessed disk space datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models without any model tuning process. The performance measures are presented in Table 4.7.

Table 4.7 Performance comparison between different preprocessed disk space datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Multiple Linear Regression	Not remove outliers + not drop duplicated data	0.0027	0.0008	0.0174	0.7609	0.7608
	Not remove outliers + drop duplicated data	0.0027	0.0007	0.0163	0.7848	0.7847
	Remove outliers from 1 feature (disk space usage) + drop duplicate data	0.0025	0.0007	0.0163	0.7848	0.7847
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.0025	0.0003	0.0075	0.8256	0.8255
	Remove outliers from all features + drop duplicated data	0.0025	**0.0002**	**0.0074**	**0.8879**	**0.8878**
Random Forest Regression	Not remove outliers + not drop duplicated data	0.3797	0.0001	0.0015	0.9597	0.9596
	Not remove outliers + drop duplicated data	0.3739	0.0001	0.0014	0.9665	0.9665
	Remove outliers from 1 feature (disk space usage) + drop duplicate data	0.3765	0.0001	0.0014	0.9665	0.9665
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.2577	0.000002	0.0002	0.9895	0.9895
	Remove outliers from all features + drop duplicated data	0.2623	**0.0000001**	**0.0001**	**0.9998**	**0.9998**

Table 4.7 Performance comparison between different preprocessed disk space datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models without any model tuning process (Cont'd)

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Multi-layer Perceptron	Not remove outliers + not drop duplicated data	0.2987	0.0010	0.0256	0.6456	0.6453
	Not remove outliers + drop duplicated data	0.2570	0.0010	0.0256	0.6610	0.6608
	Remove outliers from 1 feature (disk space usage) + drop duplicate data	0.2569	0.0010	0.0256	**0.6612**	**0.6610**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.2401	0.0006	0.0172	0.5306	0.5302
	Remove outliers from all features + drop duplicated data	0.2452	**0.0005**	**0.0138**	0.5552	0.5549

The results in Table 4.7 shows the model performance from different preprocessed disk space datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models. The result shows that in multiple linear regression and random forest regression, all evaluation metrices of “Remove outliers from all features + drop duplicated data” were best. For multi-layer perceptron, both MSE and MAE values of “Remove outliers from all features + drop duplicated data” were best. For R-square and Adjusted R-square value, “Remove outliers from 1 feature (disk space usage) + drop duplicate data” performed best.

From the previous experiments, we can see that outliers were significant factors in model performance for some datasets (CPU, memory, disk space dataset) trained with different regression models. However, outliers may not be important for the model performance in some cases too. For the CPU dataset, the results show that it will be better if we delete some outliers out and the one that shows the better R-squared and Adjusted R-square is “Remove outliers from one feature (CPU usage) + drop duplicate data”. For the memory dataset, the results show that “Remove outliers from all features + drop duplicated data” provided the best result for multiple linear regression. “Not remove outliers + not drop duplicated data” gave the best result for random forest regression and “Remove outliers from 1 feature (memory usage) + drop duplicate data” provided the best result for multi-layer perceptron. For the disk space dataset, the results show that “Remove outliers from all features + drop duplicated data” provided the best result for multiple linear regression and random forest regression but the one that gave the best result for multi-layer perceptron was “Remove outliers from 1 feature (disk space usage) + drop duplicated data”

4.3.2 Model Performance Results for Tuned Models

Before we describe the next series of experiments, we will introduce the model tuning process. For the hyperparameter tuning process, grid search method was applied to all of the models to tune for the best hyperparameters. The parameter that we use in the experiments are shown in Tables 4.8, 4.9, and 4.10.

Table 4.8 Parameter used for multiple linear regression model tuning

Parameters	Values
Fit intercept	True, False
Normalize	True, False

Table 4.9 Parameter used for random forest regression model tuning

Parameters	Values
Max depth	5, 10, 15, 25, 40, None
Min samples leaf	1, 2, 4, 8
Min samples split	2, 5, 10, 40
n estimators	50, 100, 250, 300, 500

Table 4.10 Parameter used for multi-layer perceptron model tuning

Parameters	Values
Hidden layer sizes	(5,), (10,), (20,), (40,), (60,), (80,), (100,)
Activation	'logistic', 'tanh', 'relu'
Solver	'lbfgs', 'sgd', 'adam'
Learning rate init	0.01, 0.001
Max iteration	100, 150, 200, 250

We tuned the model for each fold of each dataset and the sets of parameters that gave the best results in each fold are given in Table 4.11-4.16.

Note that for multiple linear regression, since this model does not have any important parameters, we decided not to tune this model. The other reason is that after we tuned this model, the performance of the model did not improve. The default parameters are:

- Fit intercept: True
- Normalize: False
- n jobs: None

The parameters of random forest regression are Max depth, n estimators, Min samples split, and Min samples leaf. The set of parameters for each of dataset was given in Table 4.11 – 4.13.

Table 4.11 The best sets of parameters for each fold of CPU datasets trained by random forest regression model

Dataset	**Fold**	**Max Depth**	**n estimators**	**Min samples split**	**Min samples leaf**	**Bootstrap**
Not remove outliers + not drop duplicated data	1	5	50	2	1	False
	2	10	50	2	2	True
	3	5	500	10	4	True
	4	5	50	5	1	True
	5	5	100	40	8	True
Not remove outliers + drop duplicated data	1	5	500	2	4	True
	2	10	100	2	2	True
	3	5	100	10	4	True
	4	5	500	2	1	True
	5	5	500	40	8	True
Remove outliers from 1 feature (CPU usage) + drop duplicate data	1	5	500	2	8	True
	2	10	300	2	1	True
	3	5	100	2	1	True
	4	5	500	2	1	True
	5	5	100	40	1	True
Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	1	5	250	2	1	True
	2	15	500	2	1	True
	3	5	500	2	4	True
	4	10	500	2	4	True
	5	5	50	40	8	True
Remove outliers from all features + drop duplicated data	1	5	500	40	8	True
	2	15	500	2	1	True
	3	5	300	2	8	True
	4	10	100	10	1	True
	5					

Table 4.12 The best sets of parameters for each fold of memory datasets trained by random forest regression model

	Fold	**Max Depth**	**n estimators**	**Min samples split**	**Min samples leaf**	**Bootstrap**
Not remove outliers + not drop duplicated data	1	5	50	2	1	True
	2	5	50	2	4	True
	3	5	250	40	1	True
	4	5	500	40	1	True
	5	25	300	2	2	True
Not remove outliers + drop duplicated data	1	10	250	40	1	True
	2	25	50	10	2	True
	3	5	500	40	1	True
	4	5	100	2	1	True
	5	15	500	5	1	True
Remove outliers from 1 feature (memory usage) + drop duplicate data	1	5	500	40	1	True
	2	15	50	40	1	True
	3	5	500	2	4	True
	4	5	50	10	1	True
	5	10	50	5	2	True
Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	1	5	500	40	4	True
	2	25	100	5	1	True
	3	5	250	40	4	True
	4	5	500	10	2	True
	5	15	100	2	2	True
Remove outliers from all features + drop duplicated data	1	5	500	40	1	True
	2	25	300	10	2	True
	3	5	100	40	4	True
	4	5	250	2	1	True
	5	5	300	2	8	True

Table 4.13 The best sets of parameters for each fold of disk space datasets trained by random forest regression model

	Fold	**Max Depth**	**n estimators**	**Min samples split**	**Min samples leaf**	**Bootstrap**
Not remove outliers + not drop duplicated data	1	5	50	5	1	True
	2	10	500	40	8	True
	3	5	50	2	1	True
	4	5	50	40	1	True
	5	5	500	40	4	True
Not remove outliers + drop duplicated data	1	5	100	40	8	True
	2	5	500	2	1	True
	3	5	100	5	2	True
	4	5	500	40	8	True
	5	5	500	5	2	True
Remove outliers from 1 feature (disk space usage) + drop duplicate data	1	5	500	40	4	True
	2	5	500	2	1	True
	3	5	100	5	2	True
	4	5	500	40	8	True
	5	5	500	5	2	True
Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	1	5	500	2	8	True
	2	5	50	2	1	True
	3	10	500	40	2	True
	4	5	50	2	1	True
	5	10	100	40	2	False
Remove outliers from all features + drop duplicated data	1	5	100	2	8	True
	2	5	250	2	1	True
	3	5	500	2	1	True
	4	5	50	2	1	True
	5	10	500	40	1	True

The parameters of multi-layer perceptron are learning rate init, Hidden layer sizes, Activation, Max iteration, and Solver. The sets of parameters for each dataset are given in Table 4.14 – 4.16.

Table 4.14 The best sets of parameters for each fold of CPU datasets trained by multi-layer perceptron model

	Fold	**Learning rate int**	**Hidden layers size**	**Activation**	**Max iteration**	**Solver**
Not remove outliers + not drop duplicated data	1	0.001	(20,)	relu	100	adam
	2	0.001	(20,)	relu	100	adam
	3	0.01	(10,)	relu	100	lbfgs
	4	0.01	(20,)	relu	100	lbfgs
	5	0.01	(5,)	tanh	150	lbfgs
Not remove outliers + drop duplicated data	1	0.001	(20,)	relu	100	adam
	2	0.01	(60,)	tanh	100	lbfgs
	3	0.01	(10,)	relu	100	lbfgs
	4	0.01	(10,)	relu	100	lbfgs
	5	0.01	(60,)	tanh	250	lbfgs
Remove outliers from 1 feature (CPU usage) + drop duplicate data	1	0.01	(80,)	tanh	100	lbfgs
	2	0.01	(80,)	tanh	100	lbfgs
	3	0.01	(10,)	relu	100	lbfgs
	4	0.01	(20,)	relu	100	lbfgs
	5	0.01	(40,)	tanh	250	lbfgs
Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	1	0.01	(10,)	relu	100	lbfgs
	2	0.01	(5,)	relu	100	lbfgs
	3	0.01	(20,)	relu	100	lbfgs
	4	0.01	(10,)	relu	100	lbfgs
	5	0.01	(40,)	tanh	150	lbfgs
Remove outliers from all features + drop duplicated data	1	0.01	(10,)	relu	100	lbfgs
	2	0.01	(5,)	relu	100	lbfgs
	3	0.01	(20,)	relu	100	lbfgs
	4	0.01	(10,)	relu	100	lbfgs
	5	0.01	(60,)	tanh	100	lbfgs

Table 4.15 The best sets of parameters for each fold of memory datasets trained by multi-layer perceptron model

	Fold	**Learning rate int**	**Hidden layers size**	**Activation**	**Max iteration**	**Solver**
Not remove outliers + not drop duplicated data	1	0.01	(5,)	tanh	100	adam
	2	0.01	(5,)	tanh	100	adam
	3	0.01	(40,)	tanh	100	lbfgs
	4	0.01	(20,)	tanh	100	lbfgs
	5	0.001	(10,)	tanh	200	adam
Not remove outliers + drop duplicated data	1	0.01	(5,)	relu	100	lbfgs
	2	0.01	(80,)	tanh	200	lbfgs
	3	0.01	(40,)	tanh	100	lbfgs
	4	0.01	(20,)	tanh	100	lbfgs
	5	0.001	(5,)	tanh	200	adam
Remove outliers from 1 feature (memory usage) + drop duplicate data	1	0.01	(20,)	tanh	100	lbfgs
	2	0.01	(10,)	tanh	100	lbfgs
	3	0.01	(100,)	tanh	100	adam
	4	0.01	(5,)	relu	100	lbfgs
	5	0.001	(60,)	tanh	100	adam
Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	1	0.01	(20,)	tanh	100	lbfgs
	2	0.01	(10,)	logistic	100	lbfgs
	3	0.01	(10,)	tanh	100	adam
	4	0.01	(5,)	relu	100	relu
	5	0.001	(60,)	tanh	100	adam
Remove outliers from all features + drop duplicated data	1	0.01	(10,)	tanh	100	lbfgs
	2	0.01	(20,)	tanh	100	lbfgs
	3	0.01	(100,)	tanh	100	adam
	4	0.01	(5,)	relu	100	lbfgs
	5	0.001	(100,)	logistic	100	sgd

Table 4.16 The best sets of parameters for each fold of disk space datasets trained by multi-layer perceptron model

	Fold	Learning rate int	Hidden layers size	Activation	Max iteration	Solver
Not remove outliers + not drop duplicated data	1	0.01	(20,)	relu	50	lbfgs
	2	0.01	(20,)	relu	250	lbfgs
	3	0.01	(10,)	relu	100	lbfgs
	4	0.01	(5,)	tanh	100	lbfgs
	5	0.01	(20,)	relu	150	lbfgs
Not remove outliers + drop duplicated data	1	0.01	(80,)	relu	250	lbfgs
	2	0.01	(20,)	relu	250	lbfgs
	3	0.01	(10,)	relu	100	lbfgs
	4	0.01	(5,)	tanh	150	lbfgs
	5	0.01	(40,)	tanh	100	lbfgs
Remove outliers from 1 feature (memory usage) + drop duplicate data	1	0.01	(80,)	relu	200	lbfgs
	2	0.01	(80,)	relu	250	lbfgs
	3	0.01	(10,)	relu	100	lbfgs
	4	0.01	(5,)	tanh	100	lbfgs
	5	0.01	(100,)	relu	200	lbfgs
Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	1	0.01	(10,)	relu	100	lbfgs
	2	0.01	(80,)	relu	100	lbfgs
	3	0.01	(20,)	relu	100	lbfgs
	4	0.01	(10,)	tanh	150	lbfgs
	5	0.01	(60,)	tanh	100	lbfgs
Remove outliers from all features + drop duplicated data	1	0.01	(10,)	relu	100	lbfgs
	2	0.01	(100,)	tanh	100	lbfgs
	3	0.01	(20,)	relu	100	lbfgs
	4	0.01	(40,)	relu	100	lbfgs
	5	0.01	(10,)	relu	100	lbfgs

We trained the models for each fold of each dataset and averaged them. Experiments 1.1, 1.2 and 1.3 showed the results from the untuned models.

After we did the model tuning process, we used the best set of parameters of each fold and trained the model for each fold of each dataset and averaged them. Experiments 2.1, 2.2 and 2.3 show results from tuned models.

Experiment 2.1: Used five different preprocessed CPU datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models with model tuning process. The performance measures are presented in Table 4.17.

Table 4.17 Performance comparison between different preprocessed CPU datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models with model tuning process

Models	Type of dataset	Training Time (Seconds)	Mean Square Error (MSE)	Mean Absolute Error (MAE)	R-squared (R^2)	Adjusted R-square
Multiple Linear Regression	Not remove outliers + not drop duplicated data	0.0060	0.0060	0.0242	0.3727	0.3722
	Not remove outliers + drop duplicated data	0.0026	0.0071	0.0243	0.3089	0.3083
	Remove outliers from 1 feature (CPU usage) + drop duplicate data	0.0027	0.0013	0.0192	**0.4554**	**0.4549**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.0026	0.0012	0.0187	0.4163	0.4157
	Remove outliers from all features + drop duplicated data	0.0026	**0.0004**	**0.0120**	0.3892	0.3886
Random Forest Regression	Not remove outliers + not drop duplicated data	0.6746	0.0055	0.0196	0.4334	0.4330
	Not remove outliers + drop duplicated data	1.3521	0.0069	0.0216	0.3951	0.3946
	Remove outliers from 1 feature (CPU usage) + drop duplicate data	1.5804	0.0009	0.0156	**0.5116**	**0.5112**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	1.9698	**0.0003**	**0.0123**	0.4455	0.4450
	Remove outliers from all features + drop duplicated data	1.5421	**0.0003**	**0.0123**	0.4110	0.4105

Table 4.17 Performance comparison between different preprocessed CPU datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models with model tuning process (Cont'd)

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Multi-layer Perceptron	Not remove outliers + not drop duplicated data	0.2878	0.0056	0.0209	0.4030	0.4026
	Not remove outliers + drop duplicated data	1.1267	0.0069	0.0222	0.3112	0.3107
	Remove outliers from 1 feature (CPU usage) + drop duplicate data	1.1984	0.0009	0.0157	**0.4813**	**0.4808**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.2620	0.0003	0.0124	0.2291	0.3303
	Remove outliers from all features + drop duplicated data	0.3728	**0.0002**	**0.0118**	0.3464	0.3603

The results in Table 4.17 shows the model performance from different preprocessed CPU datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models. The results show that in all models both MSE and MAE values of "Remove outliers from 1 feature (CPU usage) + drop duplicate data" and "Remove outliers from all features + drop duplicated data" dataset were better than others but in terms of R-square and Adjusted R-square value, "Remove outliers from 1 feature (CPU usage) + drop duplicate data" was better than one of the others.

Experiment 2.2: Used five different preprocessed memory datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models with the model tuning process. The performance measures are presented in Table 4.18.

Table 4.18 Performance comparison between different preprocessed memory datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models with model tuning process

Models	Type of dataset	Training Time (Seconds)	Mean Square Error (MSE)	Mean Absolute Error (MAE)	R-squared (R^2)	Adjusted R-square
Multiple Linear Regression	Not remove outliers + not drop duplicated data	0.0025	0.0141	0.0362	0.8754	0.8753
	Not remove outliers + drop duplicated data	0.0027	0.0151	0.0376	0.8799	0.8798
	Remove outliers from 1 feature (memory usage) + drop duplicate data	0.0027	0.000004	0.0012	0.7844	0.7842
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.0024	0.00001	0.0013	0.7811	0.7812
	Remove outliers from all features + drop duplicated data	0.0023	**0.000002**	**0.0011**	**0.9158**	**0.9157**
Random Forest Regression	Not remove outliers + not drop duplicated data	1.1367	0.0050	0.0172	0.8549	0.8548
	Not remove outliers + drop duplicated data	1.0196	0.0066	0.0163	**0.9484**	**0.9483**
	Remove outliers from 1 feature (memory usage) + drop duplicate data	0.6218	0.000003	0.0006	0.8348	0.8347
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.7321	0.000003	0.0007	0.8822	0.8821
	Remove outliers from all features + drop duplicated data	1.1426	**0.000001**	**0.0005**	0.9393	0.9392

Table 4.18 Performance comparison between different preprocessed memory datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models with model tuning process (Cont'd)

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Multi-layer Perceptron	Not remove outliers + not drop duplicated data	0.7868	0.0112	0.0315	0.7476	0.7475
	Not remove outliers + drop duplicated data	1.8164	0.0123	0.0332	0.7413	0.7412
	Remove outliers from 1 feature (memory usage) + drop duplicate data	0.3637	0.00001	0.0015	0.7819	0.7818
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.3320	0.00001	0.0017	0.5794	0.5796
	Remove outliers from all features + drop duplicated data	0.4515	**0.000005**	**0.0013**	**0.8616**	**0.8615**

The results in Table 4.18 show the model performance from different preprocessed memory datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models. The results show that all evaluation metrics of multiple linear regression and multi-layer perceptron, “Remove outliers from all features + drop duplicated data” dataset performed best. For random forest regression, MSE and MAE values performed best when trained with “Remove outliers from all features + drop duplicated data” but R-square and Adjusted R-square values of “Not remove outliers + drop duplicated data” were better than others.

Experiment 2.3: Used five different preprocessed disk space datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models with model tuning process. The performance measures are presented in Table 4.19.

Table 4.19 Performance comparison between different preprocessed disk space datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models with model tuning process

Models	Type of dataset	Training Time (Seconds)	Mean Square Error (MSE)	Mean Absolute Error (MAE)	R-squared (R^2)	Adjusted R-square
Multiple Linear Regression	Not remove outliers + not drop duplicated data	0.0027	0.0008	0.0174	0.7609	0.7608
	Not remove outliers + drop duplicated data	0.0027	0.0007	0.0163	0.7848	0.7847
	Remove outliers from 1 feature (disk space usage) + drop duplicate data	0.0025	0.0007	0.0163	0.7848	0.7847
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.0025	0.0003	0.0075	0.8256	0.8255
	Remove outliers from all features + drop duplicated data	0.0025	**0.0002**	**0.0074**	**0.8879**	**0.8878**
Random Forest Regression	Not remove outliers + not drop duplicated data	0.2659	0.0002	0.0057	0.9157	0.9156
	Not remove outliers + drop duplicated data	1.6049	0.0001	0.0015	0.9710	0.9710
	Remove outliers from 1 feature (disk space usage) + drop duplicate data	1.3425	0.0001	0.0015	0.9710	0.9710
	Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data	0.5265	0.00002	0.0010	0.9811	0.9763
	Remove outliers from all features + drop duplicated data	0.8451	**0.000002**	**0.0001**	**0.9941**	**0.9941**

Table 4.19 Performance comparison between different preprocessed disk space datasets trained by multiple linear regression, random forest regression and multi-layer perceptron models with model tuning process (Cont'd)

Models	**Type of dataset**	**Training Time (Seconds)**	**Mean Square Error (MSE)**	**Mean Absolute Error (MAE)**	**R-squared (R^2)**	**Adjusted R-square**
Multi-layer Perceptron	Not remove outliers + not drop duplicated data	1.1706	0.0004	0.0120	0.8197	0.8196
	Not remove outliers + drop duplicated data	1.2148	0.0004	0.0113	0.8630	0.8629
	Remove outliers from 1 feature (disk space usage) + drop duplicate data	0.8765	0.0004	0.0112	**0.8645**	**0.8644**
	Remove outliers from features 'CPU usage', 'memory usage', 'disk space usage' + drop duplicated data	0.2987	**0.0002**	**0.0074**	0.7585	0.7583
	Remove outliers from all features + drop duplicated data	0.2988	0.0003	0.0078	0.8197	0.8196

The result in Table 4.19 shows the model performance from the different preprocessed disk space datasets trained by multiple linear regression, random forest regression, and multi-layer perceptron models. The result shows that all the evaluation metrics of multiple linear regression and random forest regression, “Remove outliers from all features + drop duplicated data” dataset performed best. For multi-layer perceptron, MSE and MAE values were best when trained with “Remove outliers from features ‘CPU usage’, ‘memory usage’, ‘disk space usage’ + drop duplicated data” but R-square and Adjusted R-square value of “Remove outliers from 1 feature (disk space usage) + drop duplicate data” are better than one of the others.

From Experiments 2.1-2.3, we can see that the results of the second experiment were similar to Experiments 1.1-1.3. The results show that outliers were sometimes important for model performance for some datasets (CPU, memory, disk space dataset) trained with different tuned regression model, or it may not be important for model performance in some cases such as memory datasets trained with random forest regression. The CPU dataset results show that it will be better if we delete some outliers out and the one that shows better R-squared and Adjusted R-square is “Remove outliers from 1 feature (CPU usage) + drop duplicated data”. For the memory dataset, the results show that “Remove outliers from all features + drop duplicated data” gave the best result for multiple linear regression and multi-layer perceptron, and “Not remove outliers + drop duplicated data” gave the best result for random forest regression. For the disk space dataset, the result shows that “Remove outliers from all features + drop duplicated data” gave the best result for multiple linear regression and random forest regression but the one that give the best result for multi-layer perceptron was “Remove outliers from 1 feature (disk space usage) + drop duplicated data”

4.4 Discussion about the results between five different preprocessed datasets and three different regression models

If we compare the six experiments mentioned in Section 4.3, the results show that the deletion of outliers can give better model performance. The datasets gave good results were “Remove outliers from one feature (CPU usage or memory usage or disk space usage) + drop duplicated data” and “Remove outliers from all features + drop duplicated data” depending on different regression models in both not tuned and untuned models.

For the CPU dataset, the preprocessed dataset that gave the best results was “Remove outliers from one feature (CPU usage) + drop duplicate data”. For the memory dataset, the preprocessed datasets that gave the best performance were datasets that did not delete outliers, however, the result with deleted outliers was also good. For the disk space dataset, the preprocessed datasets that gave the best performance are “Remove outliers from one feature (disk space usage) + drop duplicate data” for multi-layer perceptron model and “Remove outliers from all features + drop duplicated data” for multiple linear regression and random forest regression models.

Since we used the same preprocessed datasets to compare three different regression models. We conclude that “Remove outliers from one feature (CPU usage or memory usage or disk space usage) + drop duplicated data” and “Remove outliers from all features + drop duplicated data” were the datasets that gave the best results in both not tuned and tuned models and we will compare the performance of each model by using these two preprocessed datasets.

We will show the performance of using three different regression models with no tuning and tuning on “Remove outliers from 1 feature (CPU usage or memory usage or disk space usage) + drop duplicated data” and “Remove outliers from all features + drop duplicated data” datasets. The results are shown in Tables 4.20-4.22.

Table 4.20 Model performance of different regression models on CPU usage datasets with "Remove outliers from 1 feature (CPU usage) + drop duplicate data" and "Remove outliers from all features + drop duplicated data"

Type of dataset	**Models**	**Training Time (Seconds)**	**Mean Absolute Error (MAE)**	**Mean Square Error (MSE)**	**R-squared (R^2)**	**Adjusted R-squared**
Performance of not tuned model						
Remove outliers from 1 feature (CPU usage) + drop duplicate data	Multiple Linear Regression	0.0027	0.0013	0.0192	0.4554	0.4549
	Random Forest Regression	1.0036	**0.0011**	**0.0177**	0.4059	0.4054
	Multi-layer Perceptron	0.3328	**0.0011**	0.0196	**0.4347**	**0.4342**
Remove outliers from all features + drop duplicated data	Multiple Linear Regression	0.0026	**0.0004**	**0.0120**	**0.3892**	**0.3886**
	Random Forest Regression	0.9460	0.0004	0.0140	0.3263	0.3261
	Multi-layer Perceptron	0.2751	0.0004	0.0130	0.3287	0.3287
Performance of tuned model						
Remove outliers from 1 feature (CPU usage) + drop duplicate data	Multiple Linear Regression	0.0027	0.0013	0.0192	0.4554	0.4549
	Random Forest Regression	1.5804	**0.0009**	**0.0156**	**0.5116**	**0.5112**
	Multi-layer Perceptron	1.1984	**0.0009**	0.0157	0.4813	0.4808
Remove outliers from all features + drop duplicated data	Multiple Linear Regression	0.0026	0.0004	0.0120	0.3892	0.3886
	Random Forest Regression	1.5421	0.0003	0.0123	**0.4110**	**0.4105**
	Multi-layer Perceptron	0.3728	**0.0002**	**0.0118**	0.3464	0.3603

The result in Table 4.20 shows that for untuned models, the model that gave the best result for "Remove outliers from one feature (CPU usage) + drop duplicate data" dataset is multi-layer perceptron and the one that gave the best performance for "Remove outliers from all features + drop duplicated data" was multiple linear regression. For the tuned model, the model that gave the best result for "Remove outliers from 1 feature (CPU usage) + drop duplicate data" and "Remove outliers from all features + drop duplicated data" dataset was random forest regression for R-squared and Adjusted R-squared. For training time, multiple linear regression used less time than the other models with 0.0027 and 0.0026 seconds for tuned and untuned models. The model that took the longest training time is random forest regression with 1.0036 and 0.9460 seconds the untuned model and 1.5804 and 1.5421 seconds for the tuned model.

Table 4.21 Model performance of different regression models on memory usage datasets with "Remove outliers from 1 feature (memory usage) + drop duplicate data" and "Remove outliers from all features + drop duplicated data"

Type of dataset	**Models**	**Training Time (Seconds)**	**Mean Absolute Error (MAE)**	**Mean Square Error (MSE)**	**R-squared (R^2)**	**Adjusted R-squared**
Performance of not tuned model						
Remove outliers from 1 feature (memory usage) + drop duplicate data	Multiple Linear Regression	0.0027	0.000004	0.0012	0.7844	0.7842
	Random Forest Regression	0.3378	**0.000002**	**0.0007**	**0.8533**	**0.85318**
	Multi-layer Perceptron	0.2868	0.0001	0.0048	0.7154	0.7164
Remove outliers from all features + drop duplicated data	Multiple Linear Regression	0.0023	0.000002	0.0011	0.9158	0.9157
	Random Forest Regression	0.3292	**0.000001**	**0.0006**	**0.928**	**0.92798**
	Multi-layer Perceptron	0.2635	0.0002	0.0094	0.3550	0.3582
Performance of tuned model						
Remove outliers from 1 feature (memory usage) + drop duplicate data	Multiple Linear Regression	0.0027	0.000004	0.0012	0.7844	0.7842
	Random Forest Regression	0.6218	**0.000003**	**0.0006**	**0.8348**	**0.8347**
	Multi-layer Perceptron	0.3637	0.00001	0.0015	0.7819	0.7818

Table 4.21 Model performance of different regression models on memory usage datasets with "Remove outliers from 1 feature (memory usage) + drop duplicate data" and "Remove outliers from all features + drop duplicated data" (Cont'd)

Type of dataset	**Models**	**Training Time (Seconds)**	**Mean Absolute Error (MAE)**	**Mean Square Error (MSE)**	**R-squared (R^2)**	**Adjusted R-squared**
Remove outliers from all features + drop duplicated data	Multiple Linear Regression	0.0023	0.000002	0.0011	0.9158	0.9157
	Random Forest Regression	1.1426	**0.000001**	**0.0005**	**0.9393**	**0.9392**
	Multi-layer Perceptron	0.4515	0.000005	0.0013	0.8616	0.8615

Table 4.21 shows that random forest regression gave the best performance in both the "Remove outliers from one feature (memory usage) + drop duplicated data" and "Remove outliers from all features + drop duplicated data" datasets in all the evaluation metrics. For training time, multiple linear regression used less time than other models with 0.0027 and 0.0023 second for untuned and tuned models. The model that took the longest training time was random forest regression with 0.3378 and 0.3292 seconds for the untuned model and 0.6218 and 1.1426 second for the tuned model.

Table 4.22 Model performance of different regression models on disk space usage datasets with "Remove outliers from 1 feature (disk space usage) + drop duplicated data" and "Remove outliers from all features + drop duplicated data"

Type of dataset	**Models**	**Training Time (Seconds)**	**Mean Absolute Error (MAE)**	**Mean Square Error (MSE)**	**R-squared (R^2)**	**Adjusted R-squared**
Performance of not tuned model						
Remove outliers from 1 feature (disk space usage) + drop duplicate data	Multiple Linear Regression	0.0025	0.0007	0.0163	0.7848	0.7847
	Random Forest Regression	0.3765	**0.0001**	**0.0014**	**0.9665**	**0.9665**
	Multi-layer Perceptron	0.2569	0.0010	0.0256	0.6612	0.6610

Table 4.22 Model performance of different regression models on disk space usage datasets with "Remove outliers from 1 feature (disk space usage) + drop duplicated data" and "Remove outliers from all features + drop duplicated data" (Cont'd)

Type of dataset	**Models**	**Training Time (Seconds)**	**Mean Absolute Error (MAE)**	**Mean Square Error (MSE)**	**R-squared (R^2)**	**Adjusted R-squared**
Remove outliers from all features + drop duplicated data	Multiple Linear Regression	0.0025	0.0002	0.0074	0.8879	0.8878
	Random Forest Regression	0.2623	**0.0000001**	**0.0001**	**0.9998**	**0.9998**
	Multi-layer Perceptron	0.2452	0.0005	0.0138	0.5552	0.5549
Performance of tuned model						
Remove outliers from 1 feature (disk space usage) + drop duplicate data	Multiple Linear Regression	0.0025	0.0007	0.0163	0.7848	0.7847
	Random Forest Regression	1.3425	**0.0001**	0.0015	**0.9710**	**0.9710**
	Multi-layer Perceptron	0.8765	0.0004	0.0112	0.8645	0.8644
Remove outliers from all features + drop duplicated data	Multiple Linear Regression	0.0025	0.0002	0.0074	0.8879	0.8878
	Random Forest Regression	0.8451	**0.000002**	**0.0001**	**0.9941**	**0.9941**
	Multi-layer Perceptron	0.2988	0.0003	0.0078	0.8197	0.8196

Table 4.22 shows that random forest regression gave the best performance in both the "Remove outliers from one feature (disk space usage) + drop duplicate data" and "Remove outliers from all features + drop duplicated data" datasets in all the evaluation metrics. For training time, multiple linear regression uses less time than other models with 0.0025 and 0.0025 seconds for the untuned and tuned models. The model that took the longest training time is random forest regression with 0.3765 and 0.2623 seconds for the untuned model and 1.3425 and 0.8451 seconds for the tuned model.

Moreover, training time shows the trade-off between model performance and training time. To have a good model performance (low MAE and MSE with high R-squared and Adjusted

R-squared), we have to take time on training models like random forest regression. On the other hand, the model that takes shorter training times may not have a good model performance.

Instead of comparing model performance by using MSE, MAE, R-squared, Adjusted R-squared and training time we also want to know about the model robustness by finding the percent difference between actual and predicted values of the model that gave the best performance for "Remove outliers from one feature (CPU usage or memory usage or disk space usage) + drop duplicated data" and "Remove outliers from all features + drop duplicated data" datasets from Tables 4.20 – 4.22. From the discussion in Section 4.4, the model that gave the best performance is random forest regression. We found percent difference by separating dataset into five folds and trained the model for each fold of each dataset and averaged them. The results are presented in Tables 4.23 – 4.25.

Table 4.23 Percent difference between actual and predicted values on the CPU usage dataset trained by not tuned and tuned random forest regression model

Type of dataset	**Percent Difference**	
	Fold	**Mean**
Performance of not tuned model		
Remove outliers from 1 feature (CPU usage) + drop duplicate data	1	6.0101
	2	5.8924
	3	6.2265
	4	6.5393
	5	107.9301
	Average	**26.5196**
Remove outliers from all features + drop duplicated data	1	5.7440
	2	5.4018
	3	6.1600
	4	7.5344
	5	4.7451
	Average	**5.9170**

Table 4.23 Percent difference between actual and predicted values on the CPU usage dataset trained by not tuned and tuned random forest regression model (Cont'd)

Type of dataset	**Percent Difference**	
	Fold	**Mean**
Performance of tuned model		
Remove outliers from 1 feature (CPU usage) + drop duplicate data	1	5.1138
	2	5.4150
	3	5.6090
	4	5.9856
	5	93.0780
	Average	**23.0402**
Remove outliers from all features + drop duplicated data	1	4.8746
	2	5.0905
	3	5.2900
	4	6.4912
	5	4.8704
	Average	**5.3233**

Table 4.24 Percent difference between actual and predicted values on memory usage dataset trained by not tuned and tuned random forest regression model

Type of dataset	**Percent Difference**	
	Fold	**Mean**
Performance of not tuned model		
Remove outliers from 1 feature (memory usage) + drop duplicate data	1	0.2482
	2	0.0312
	3	0.2928
	4	0.1534
	5	0.1427
	Average	**0.1736**

Table 4.24 Percent difference between actual and predicted values on memory usage dataset trained by not tuned and tuned random forest regression model (Cont'd)

Type of dataset	**Percent Difference**	
	Fold	**Mean**
Remove outliers from all features + drop duplicated data	1	0.2557
	2	0.0261
	3	0.2912
	4	0.1527
	5	0.1020
	Average	**0.1655**
Performance of tuned model		
Remove outliers from 1 feature (memory usage) + drop duplicate data	1	0.2406
	2	0.0306
	3	0.2897
	4	0.1500
	5	0.1460
	Average	**0.1714**
Remove outliers from all features + drop duplicated data	1	0.2435
	2	0.0263
	3	0.2936
	4	0.1480
	5	0.0979
	Average	**0.1619**

Table 4.25 Percent difference between actual and predicted values on disk space usage dataset trained by not tuned and tuned random forest regression model

Type of dataset	**Percent Difference**	
	Fold	**Mean**
Performance of not tuned model		
Remove outliers from 1 feature (disk space usage) + drop duplicate data	1	0.0725
	2	0.0254
	3	0.0899
	4	0.0763
	5	4.0987
	Average	**0.8726**
Remove outliers from all features + drop duplicated data	1	0.0593
	2	0.0670
	3	0.0751
	4	0.0228
	5	0.0665
	Average	**0.0581**
Performance of tuned model		
Remove outliers from 1 feature (disk space usage) + drop duplicate data	1	0.0742
	2	0.0256
	3	0.0886
	4	0.0760
	5	4.5489
	Average	**0.9626**
Remove outliers from all features + drop duplicated data	1	0.0594
	2	0.0668
	3	0.0766
	4	0.0228
	5	0.2763
	Average	**0.1004**

The percent differences in Tables 4.23-4.25 show the robustness of the random forest model. The results from all the tables show that for the “Remove outliers from one feature (CPU usage or memory usage or disk space usage) + drop duplicated data” dataset trained by the untuned and tuned random forest regression models, the means of percent difference of the Folds 1-4 gave similar percent difference values with each other. Fold 5 gave high percent difference value comparatively. For “Remove outliers from all features + drop duplicated data” dataset trained by the untuned and tuned random forest regression model, the means of percent difference of the Folds 1-4 gave similar percent difference values between each other. For the Fold 5, the percent difference for this dataset is quite similar to other folds than from “Remove outliers from one feature (CPU usage or memory usage or disk space usage) + drop duplicated data” dataset. We can determine that “Remove outliers from all features + drop duplicated data” dataset can make the random forest regression model more robust than “Remove outliers from 1 feature (CPU usage or memory usage or disk space usage) + drop duplicated data” dataset. Even though the percent differences are not practically identical, they still have low percent differences between actual and predicted values.

We conclude that random forest regression is a suitable model for all the datasets. Table 4.26 shows the rank of features importance of each random forest regression model. The feature having highest feature importance value in all the models is the number of workflows.

Table 4.26 Feature importance values from random forest regression model

Model	Features	Feature importance values
CPU Usage	Number of workflows	0.8790
	Disk space usage	0.0841
	Disk space free	0.0318
	Memory usage	0.0038
	Memory free	0.0013
Memory Usage	Number of workflows	0.7893
	Disk space usage	0.1947
	Disk space free	0.0103
	CPU usage	0.0057

Table 4.26 Feature importance values from random forest regression model (Cont'd)

Model	Features	Feature importance values
Disk Space Model	Number of workflows	0.6973
	CPU usage	0.3000
	Memory usage	0.0025
	Memory free	0.0002

4.5 Regression Equation

The result from other experiments suggest that random forest regression is the best model appropriate to the data. We also wanted to formulate the regression equation from multiple linear regression to show the estimated result such as CPU usage, memory usage, and disk space usage by using selected factors too. We separated the equation into three equations for each resource. We use "Remove outliers from all features + drop duplicated data" dataset for building the multiple linear regression model due to the robustness of the dataset with the multiple linear regression model. Before we formulate the models, we use the t-test to investigate the linear relationship between dependent and independent factors.

The hypothesis states that:

Hypothesis (H_0):	The slope of the regression line is equal to zero. ($B_1 = 0$)
Null Hypothesis (H_1):	The slope of the regression line is not equal to zero. ($B1 \neq 0$)

For this analysis, the significance level is 0.05, if p-value is less than 0.05 we reject the null hypothesis. In this case, we will state that there is a significant linear relationship between the independent variable and the dependent variable. But if the p-value is more than 0.05 then we do not have enough evidence to reject the null hypothesis which suggests that there was no significant linear relationship between independent variable and dependent variable.

We sample 5% of the datasets to do analysis and set the significant level to 0.05, the result is shown in Table 4.27. Note that in the table, 'Not Sig' means not statistically significant, 'Sig' means statistically significant to each dependent variable (CPU usage, memory usage, and disk space usage), and 'x' means that factor was not used in that model.

Table 4.27 T-test statistic results between 3 different estimated factors and other factors

Factors (Independent Variables)	**CPU Usage (Dependent Variable)**	**Memory Usage (Dependent Variable)**	**Disk Space Usage (Dependent Variable)**
CPU Usage	x	Sig p-value = 0.013	Sig p-value = 0
Memory Free	Not Sig p-value = 0.163	x	Sig p-value = 0
Memory Usage	Not Sig p-value = 0.174	x	Sig p-value = 0
Disk Space Free	Sig p-value = 0	Sig p-value = 0	x
Disk Space Usage	Sig p-value = 0	Sig p-value = 0	x
Number of Workflow	Sig p-value = 0	Not Sig p-value = 0.284	Sig p-value = 0

Table 4.27 shows that memory free and memory usage are not statistically significant to CPU usage. The number of workflow was not statistically significant to memory usage. The variables that are not statistically significant or had a p-value that was more than 0.05 meant that we do not reject the null hypothesis or there is no significant linear relationship between independent variable and the dependent variable.

The finalized factors were:

1. The equation for estimating CPU usage
    - Disk space free
    - Disk space usage
    - Number of workflows

2. The equation for estimating memory usage
    - CPU usage
    - Disk space free
    - Disk space usage

3. The equation for estimating Disk space usage
    - CPU usage
    - Memory usage
    - Memory free
    - Number of workflows

With these factors, we can formulate the equations by finding the coefficient and intercept of each multiple linear regression model. The finalized equation format is shown in Equation 3.4 in Section 3.6.2. The equation is shown below.

The equation for estimating CPU usage

$$\text{CPU usage} = 2.9587 + (0.0129 \times \text{Disk space free} + (0.4960 \times \text{Disk space usage}) \quad (4.1)$$
$$(0.2214 \times \text{Number of workflow})$$

With the above equation, we can estimate the CPU usage when we know the value of each factor. For example, when there are 20 GB free disk space, 10% disk usage, and 8 workflows on FLP, then the estimated CPU usage can then be calculated as follows:

$$\text{CPU usage} = -2.9587+(0.0129 \times 20) + (0.4960 \times 10) - (0.2214 \times 8)$$

The CPU usage is 0.4881 or we can say that 48.81 percent of CPU will we used in this situation.

The equation for estimating memory usage is given as

$$\text{Memory usage} = 1.1276 + (0.0150 \times \text{CPU usage}) - (0.0189 \times \text{Disk space free}) \quad (4.2)$$
$$-(0.2679 \times \text{Disk space usage})$$

From the above equation, we can estimate the memory usage when we know the value of each factor. For example, with 40% of CPU used, 15 GB free disk space, 5% disk usage, the estimated memory usage can then be calculated as follows:

$$\text{Memory usage} = 1.1276+(0.0150\times40)-(0.0189\times15)-(0.2679\times5)$$

The memory usage is 0.1050 or 10.50% of memory will be used in this situation.

The equation for estimating disk space usage is given as

$$\begin{aligned}\text{Disk space usage} = {} & 23.2600 + (0.8313 \times \text{CPU usage}) - (3.7803 \times \text{Memory free}) \\ & - (10.9545 \times \text{Memory usage}) + (0.0793 \times \text{Number of workflow})\end{aligned} \quad (4.3)$$

We can estimate the disk space usage when we know the value of each factor. For example, with 60% of CPU used, 2 GB free memory, 6% used memory, and 10 workflows on FLP, the estimated disk space usage can then be calculated as follows:

$$\begin{aligned}\text{Disk space usage} = {} & 23.2600 + (0.8313 \times 60) - (3.7803 \times 2) - (10.9545 \times 6) \\ & + (0.0793 \times 10)\end{aligned}$$

The disk space usage is 0.6434 or 64.34% of disk space will be used in this situation.

The reason that random forest regression is better than linear regression model is because the datasets that we have used in the experiments may be nonlinear as shown in Table 4.1 with Pearson's correlation coefficient. But we have formulated the multiple linear regression because in terms of usability, to formulate an equation from the multiple linear regression is easier than to use random forest regression model to estimate the computing resources.

CHAPTER 5 CONCLUSION

Resource estimation is used to estimate the computing resources of a system based on historical data. The European Organization for Nuclear Research (CERN) is currently developing a new logging system for A Large Ion Collider Experiment detector (ALICE) based on the Elasticsearch, Logstash, and Kibana (ELK) software stack. This system will use Beats to receive log data and transfer them to Logstash in order to be sent to the search and analytics parts. Additionally, the number of nodes might increase or decrease. If there is a lack of computing resources, then the whole system may be overloaded due to the bottleneck which could affect the whole logging system.

Our purposed method compared regression models and formulated regression equations to estimate CPU usage, memory usage, and disk space usage for the system. We are focusing on system logs from Logstash located at the server-side of the system. We have used Metricbeat to get the status of historical computing metrics of machines from the Logstash node.

Five preprocessed dataset were compared which were (1) not remove outliers, (2) not remove outliers with dropping duplicated data, (3) remove outlier of one feature which is the dependent variable of each dataset with dropping duplicated data, (4) remove outlier of three features which are dependent variables of every dataset with dropping duplicated data, and (5) remove outlier of all features and dropping duplicated data.

The results from our two experiments showed that outliers significantly affected model performance for most datasets (CPU, memory, disk space dataset) trained with different regression models. Dataset (3) achieved the best performance in the CPU dataset. Datasets (1), (3), and (5) gave the best results for random forest regression, multi-layer perceptron, and multiple linear regression respectively in the memory dataset. In terms of disk space, the results showed that dataset (5) gave the best results for multiple linear regression and random forest regression; dataset (3) gave the best result for multi-layer perceptron. Moreover, datasets with removed outliers gave better results when compared to datasets with outliers.

With some exceptions (MSE, MAE, R-squared and Adjusted R-squared), training time exhibited trade-offs with model performance. To have good model performance (low MAE and MSE with high R-squared and Adjusted R-squared), we had to take time on training models like random forest regression.

The most robust model was random forest regression. The results from all tables showed that, in both untuned and tuned random forest regression models, dataset (3) made random forest regression models more robust than dataset (5). Lastly, we formulated equations that can estimate CPU usage, memory usage, and disk space usage by using multiple linear regression.

Limitations and Future Work

The limitation of this research was that we used our own testbed which did not collect enough data. Furthermore, the system at that time was not stable. The collected data may not good enough. Stable systems could give better results. Additionally, we had only one Logstash in the architecture. The amount of Logstash could expand and new features such as the number of Logstash or labeled Logstash origin data should be used in estimation models.

This thesis could be used as a prototype framework or guideline for the ALICE O^2 Logging System project in real environments on production. Different environments may vary from our results.

REFERENCES

1. Wegrzynek, A., Chibante Barroso, V. and Vino, G., 2018, “Monitoring the New ALICE Online-offline Computing System”, **Proceedings of 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)**, 8-13 October 2017, Barcelona, Spain, pp. 195-200.

2. Pugdeethosapol, K., Chibante Barroso, V., Akkarajitsakul, K. and Achclakul, T., 2016, "Dynamic Configuration of the Computing Nodes of the ALICE O^2 System", **Proceedings of 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)**. 13-15 July 2016, Khon Kaen, Thailand, pp. 1-5.

3. Wilder, J., 2012, **Centralized Logging** [Online], Available: http://jasonwilder.com/blog/2012/01/03/centralized-logging/ [2020, March 18].

4. Elasticsearch B.V., 2020, **What is the ELK Stack?** [Online], Available: https://www.elastic.co/what-is/elk-stack [2020, March 18].

5. Taylor, D., 2020, **ELK Stack Tutorial: Learn Elasticsearch** [Online], Available: https://www.guru99.com/elk-stack-tutorial.html [2020, March 18].

6. Elasticsearch B.V., 2020, **Beats Lightweight Data Shippers** [Online], Available: https://www.elastic.co/beats/ [2020, March 18].

7. JavaInUse, 2020, **File Beat + ELK (Elastic, Logstash and Kibana) Stack to index logs to Elasticsearch - Hello World Example** [Online], Available: https://www.javainuse.com/elasticsearch/filebeat-elk [2020, March 18].

8. Jennings, B. and Stadler, R., 2014, “Resource Management in Clouds: Survey and Research Challenges,” **Journal of Network and Systems Management,** Vol. 23, No. 3, pp. 567–619.

9. Zhang, Q., Cherkasova, L and Smirni, E., 2008, "A Regression-based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications," **4th International Conference on Autonomic Computing (ICAC'07),** 11-15 June 2008, Jacksonville, FL, USA, pp. 27-27.

10. Farahnakian, F., Liljeberg, P. and Plosila, J., 2013, "LiRCUP: Linear Regression Based CPU Usage Prediction Algorithm for Live Migration of Virtual Machines in Data Centers," **Proceedings of 2013 39th Euromicro Conference on Software Engineering and Advanced Applications, Santander**, 4-6, September 2013, Santander, Spain, pp. 357-364.

11. Khan, A., Yan, X., Tao, S. and Anerousis, N., 2012 "Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach," **Proceedings of 2012 IEEE Network Operations and Management Symposium,** 16-20 April 2012, Maui, HI, USA, pp. 1287-1294.

12. Patel, J., Jindal, V., Yen, I., Bastani, F., Xu J. and Garraghan, P., 2015 "Workload Estimation for Improving Resource Management Decisions in the Cloud," **Proceedings of 2015 IEEE 12th International Symposium on Autonomous Decentralized Systems,** 25-27 March 2015, Taichung, Taiwan, pp. 25-32.

13. Vora, M. N., 2014, "Predicting Utilization of Server Resources from Log Data", **International Journal of Computer Theory and Engineering,** Vol. 6, No. 2, p. 118123.

14. Bankole, A. A. and Ajila, S.A., 2013, "Predicting Cloud Resource Provisioning using Machine Learning Techniques," **Proceedings of 2013 IEEE 26th Canadian Conference on Electrical and Computer Engineering (CCECE)**, 5-8 May 2013, Regina, SK, Canada, pp. 1-4.

15. Rajaram, K. and Malarvizhi, M. P., 2017, "Utilization Based Prediction Model for Resource Provisioning," **Proceedings of 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP)**, 10-11 January 2017, Chennai, India, pp. 1-6.

16. Adane, P. D. and Kakde, O. G., 2018, "Predicting Resource Utilization for Cloud Workloads Using Machine Learning Techniques," **Proceedings of 2018 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT)**, 20-21 April 2018, Coimbatore, India, pp. 1372-1376.

17. Mehmood, T., Latif, F. and Malik, S., 2018, "Prediction of Cloud Computing Resource Utilization," **Proceedings of 2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)**, 8-10 October 2018, Islamabad, Pakistan, pp. 38-42.

18. Thonglek, K., Ichikawa, K., Takahashi, K., Iida H. and Nakasan, C., 2019, "Improving Resource Utilization in Data Centers using an LSTM-based Prediction Model," **Proceedings of 2019 IEEE International Conference on Cluster Computing (CLUSTER)**, 23-26 September 2019, Albuquerque, NM, USA, pp. 1-8.

19. Charfaoui, Y., 2020, **Hands-on with Feature Selection Techniques: Embedded Methods** [Online], Available: https://heartbeat.fritz.ai/hands-on-with-feature-selection-techniques-embedded-methods-84747e814dab [2020, March 18].

20. Gupta, M., 2018, **ML | Linear Regression** [Online], Available: https://www. geeksforgeeks.org/ml-linear-regression/ [2020, 15 March].

21. Stecanella, B., 2017, **An Introduction to Support Vector Machines (SVM)** [Online], Available: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/ [2020, 15 March].

22. Sayad, S., 2020, **Decision Tree - Regression** [Online], Available: https://www. saedsayad.com/decision_tree_reg.htm/ [2020, March 15].

23. Kang, E., 2017, **Long Short-Term Memory (LSTM): Concept** [Online], Available: https://medium.com/@kangeugine/long-short-term-memory-lstm-concept-cb328 3934359 [2020, March 15].

24. Kaushik, S., 2016, **Introduction to Feature Selection Methods with an Example (or How to Select the Right Variables?)** [Online], Available: https://www.analytics vidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/ [2020, March 15].

25. Chakure, A., 2019, **Random Forest Regression** [Online] Available: https://towards datascience.com/random-forest-and-its-implementation-71824ced454f [2020, March 15].

APPENDIX A

Guidelines for estimating computing resources for the ALICE O^2 logging system

Guidelines for estimating computing resources for the ALICE O^2 logging system

1. Design the logging system together with ELK stack
2. Collect the computing metric logs with Metricbeat
3. Preprocess the collected logs to estimating computing resources for Logstash

    3.1 Data cleansing

    - Change types of input data into their appropriate data types such as integer and float
    - Remove unnecessary special characters or symbols before using them in the model and handle any missing values, null, or NaN
    - Perform an outlier cleansing step by using interquartile range (IQR) together with the rule saying that a data point is an outlier if it is more than 1.5IQR above the third quartile or below the first quartile and dropping duplicated data

    3.2 Data transformation

    - Use average, sum, and maximum operations to aggregate our data
    - Drop time column out

    3.3 Feature Extraction

    - Create 2 more features which are the number of workflows and the number of nodes

    Note: If in the future there can be 2 or more Logstashs, new feature can be created which is about number of Logstash or a feature going to label that the collected data are from which Logstash and use these features in the estimation model.

4. Feature selection by using t-statistical testing and finding p-value for Pearson's correlation to conclude that there is significant correlation between 2 variables or not

5. Estimate computing resources for Logstash by using regression model

    5.1 Regression model are multiple linear regression, random forest regression and multi-layer perceptron

    5.2 Train model with not tuned and tuned process

    Note: In the experiment, random forest regression is an appropriate model.

6. Formulate an equation from the multiple linear regression by finding the coefficient and intercept

   Note: We have formulated the multiple linear regression because in term of usability, to formulate an equation from the multiple linear regression is easier than use random forest regression model to estimate the computing resources.

7. The formulated equation will be used to estimate computing resources for the ALICE O^2 logging system by substitute value of each feature into the equation.
   7.1 Formulated the equation for CPU usage, memory usage and disk space usage
   7.2 For example, if the factors of equation for estimating CPU usage are disk space free, disk space usage and number of workflows, then substitute the value of each feature into the equation to find the CPU usage

For estimated CPU usage that come out will be in the form of percentage of CPU used in that situation. In the same way, the estimated memory usage and disk space usage will be in the form of percentage of memory used and percentage of disk space used in each situation. After finding those 3 estimated resources, we can compare and decide that if the CPU, memory and disk space usage are given like this, what will be an appropriate specification of CPU core, number of gigabytes of memory and number of gigabytes of disk space.

In our testbed, we create all instances (i.e., FLP and ELK server) by using Linux operating system with 4 VCPUs, memory 7.3 GB, and 40 GB disk space. We collect, preprocess and train those data and formulate multiple linear regression.

We found that if we know the value of each factor. For example, when there are disk space 20 GB is free, 10 percent of disk space used and the number of workflows that running on FLP node is 8 workflows. The CPU usage is 0.4881 or we can say that 48.81 percent of CPU will we used in this situation. But if the user want to increase the number of workflow from 8 workflows to 15 workflows and know that the value of disk space usage is 14 percent, the estimated CPU usage will be 92.23 percent. Therefore, we have to increase the number of CPU core from 4 to 6 or 8 cores in order to decrease the CPU usage.

From all the information that we have mentioned, this work is be a prototype framework or guideline for CERN, ALICE O^2 Logging System project to use on a real environment on the production in the future. To make the estimation model more specific, our suggestion is to collect more data from specific cases for example, collect more data from different specification of computer resources.

CURRICULUM VITAE

NAME	Miss Juthaporn Vipatpakpaiboon
DATE OF BIRTH	1 February 1996
EDUCATIONAL RECORDS	
HIGH SCHOOL	High School Graduation Samsenwittayalai School, 2013
BACHELOR'S DEGREE	Bachelor of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2017
MASTER'S DEGREE	Master of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2020
SCHOLARSHIP / RESEARCH GRANT	BX Scholarship, 2018
PUBLICATION	Vipatpakpaiboon, J., Barroso, V.C. and Akkarajitsakul, K., 2021, "Computing Resource Estimation by Using Machine Learning Techniques for ALICE O^2 Logging System", **2021 12th International Conference on Advances in Information Technology (IAIT2021),** 29 June – 1 July, Bangkok, Thailand, pp. 1-7.