Accredited SINTA 2 Ranking
Decree of the Director General of Higher Education, Research, and Technology, No. 158/E/KPT/2021
Validity period from Volume 5 Number 2 of 2021 to Volume 10 Number 1 of 2026

Published online at: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 8 No. 5 (2024) 597 - 606 e-ISSN: 2580-0760

Cloud Node Auto-Scaling System Automation Based on Computing Workload Prediction

Tri Fidrian Arya[1*], Reza Fuad Rachmadi[2], Achmad Affandi[3]
[1,3]Department of Electrical Engineering, Faculty of Intelligent Electrical and Information Technology, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[2]Department of Computer Engineering, Faculty of Intelligent Electrical and Information Technology, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[1]trifidrianarya@gmail.com, [2]fuad@its.ac.id, [3]affandi@its.ac.id

Abstract

Auto-scaling systems in cloud computing are important for handling application workload fluctuations. This research uses machine learning to predict resource requirements based on workload work patterns and design an automatic scaling system. The dataset used includes features of node name, time, CPU usage percentage, and RAM usage. The ML model is applied for prediction regression of CPU usage percentage, CPU load, and RAM usage, and then the server workload is classified into four categories: Very High, High, Low, and Very Low. The autoscaling system used is horizontal scaling. From the results of this research, it was found that the stacking algorithm with the base learner Random Forest and XGBoost had better performance in producing predictive regression. Then, after performing stability testing using K-Fold cross-validation by classifying based on workload status, it was found that the Gradient Boosting algorithm had better results compared to other algorithms, namely for the percentage of CPU usage with an accuracy of 0.998, precision 0.9, recall 0.878, f1score 0.888; CPU load average 15 minutes with accuracy 0.997, precision 0.854, recall 0.863, f1score 0.863; Meanwhile, the percentage of RAM usage is accuracy 0.992, precision 0.986, recall 0.986, and f1score 0.986. However, the XGBoost algorithm also has test results that are almost the same as Gradient Boosting.

Keywords: Auto-scaling; Cloud Computing; Forecasting; Workload

How to Cite: Tri Fidrian Arya, Reza Fuad Rachmad, and Achmad Affandi, "Cloud Node Auto-Scaling System Automation Based on Computing Workload Prediction", *J. RESTI (Rekayasa Sist. Teknol. Inf.)* , vol. 8, no. 5, pp. 597 - 606, Oct. 2024.
DOI: https://doi.org/10.29207/resti.v8i5.5928

1. Introduction

The increasingly rapid development of the digital industry and the need for flexible computing infrastructure means that Cloud Computing services have become the main choice. Service Cloud, otherwise known as Infrastructure as a Service (IaaS). To optimize system performance and avoid wasting resources, practice auto-scaling has become popular. Auto-scaling enables infrastructure Cloud to automatically adjust computing capacity according to workload fluctuations. In this case adding or reducing resources such as CPU, RAM, or storage dynamically either by mechanism vertical scaling or horizontal scaling automatically.

Despite the concept that auto-scaling is well known, implementation often requires a deep understanding of workload patterns, as well as careful monitoring. Therefore, implementation machine learning can be used to manage auto-scaling for the Infrastructure Cloud.

Research conducted [1] by Balantimuhe et al conducted research on cloud node resource management based on server workload parameters (CPU processor usage). The server workload is classified into 3 status groups, namely low, medium and high. In developing the prediction model, use the Backpropagation Neural Network (BNN) model, with several data features on CPU usage, RAM, network traffic and accuracy performance of more than 90%.

In 2018 [2] there was research to automate predictions based on actual workload patterns and resources obtained from historical data on existing service data. The prediction model used is Naive Bayes. Adane and Kakde [3] conducted research related to auto scalability using machine learning methods to predict resource

Received: 17-07-2024 | Accepted: 01-10-2024 | Published Online: 06-10-2024

usage such as CPU and memory in a cloud system environment. The algorithm of machine learning which is used is Random Forest. Algorithm Random Forest has good performance compared to other algorithms, such as Linear Regression, k-NN, Neural Networks, and SVM. The experimental results produce a learning model Random Forest that has lower prediction regression error tolerance and lower resource usage time compared to other models.

Research by Khaleq, et al in 2021 [4] developed a smart autonomous automatic scaling system for automatic scaling microservice of the cloud with QoS constraints, using the method of Reinforcement Learning (RL). Provides customized auto-scaling results for microservice in cloud applications by paying attention to the minimum QoS constraints perceived by the user.

Furthermore, research conducted in 2022 [5] is a constructive machine learning (ML) model to make predictions for computing workload patterns on the server farm based on data from time to time from previously appropriate computing activities. Building models using models SVM. Testing is carried out to obtain accurate regression predictions, and comparisons are made with the model and other machine learning. Other machine learning models used Gaussian Naive Bayes and obtained better accuracy prediction results using SVM.

Iqbal W, et al [6] Conducted research on the use of varying-performance Virtual Machines (VM) to perform automatic scaling in dealing with dynamic and fluctuating workloads. Its predictive modelling uses a Random Decision Forest (RDF) based on resource configuration, number of requests, and response time. Comparing algorithms Random Decision Forest (RDF) with AdaBoost, SVM, Naive Bayes, and K-NN algorithms. And getting results from RDF has better performance.

Manam S, et al [7] Conducted research on the automatic scaling method used, namely the Random Forest Classifier method, which is a supervised classifier model that uses a decision tree creation algorithm. Each tree in the Random Forest Classifier will produce a prediction, and the best prediction will be selected based on the number of votes from each tree. This method can produce better CPU and memory predictions compared to other algorithms.

Within the framework of this research, the aim is to develop and test an auto-scaling system prediction automation algorithm that can intelligently adjust Cloud resources based on dynamic analysis of computing workloads by utilizing several algorithms. Machine Learning (ML) to look for good performance and processes auto-scaling use horizontal scaling.

2. Research Methods

The stages that will be carried out in the research are as follows in Figure 1.

Figure 1. Research Block Diagram

2.1 System Analysis

System analysis is carried out to find out problems with the architecture and implementation of the previous system as in Figure 2, and to find out what is needed to overcome these problems.

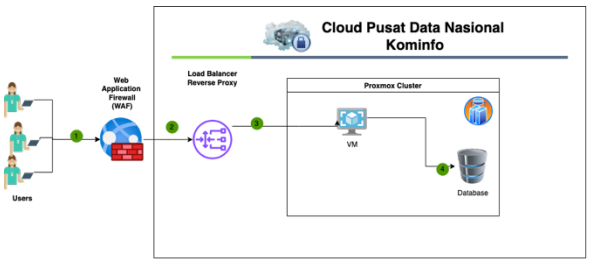

Figure 2. Existing Architecture

Users access the application there is a firewall (WAF) before accessing the server. After passing through the WAF, it is forwarded to the Reverse Proxy (RP). From the Reverse Proxy (RP) the request is forwarded to the destination application server VM according to its domain. The application VM is connected to its database (DB)

There was a problem with the previous system architecture, namely that there was no mechanism that automatically adjusted the resources in the application VM according to needs. When workload fluctuations occur in applications on a VM that are accessed with a high workload, it causes the application process on the VM to become slow or even inaccessible.

To minimize the impact of workload fluctuations on a VM, leveraging machine learning in the system autoscaling is one solution. Machine learning will analyze historical data to predict server workload. System auto-scaling The mechanism used is horizontal scaling, capacity adjustment is carried out by increasing or decreasing the number of VMs horizontally, without changing the capacity of each VM itself. [8][9][10]. Election horizontal scaling from vertical scaling is to minimize downtime during the process auto-scaling is running, because on vertical scaling Capacity adjustments are carried out by increasing the resource capacity of the running VM and requiring a restart of the VM itself.

2.2 Data Analysis

The detailed dataset used consists of 15 features according to Table 1 To characterize workloads to form a system prediction model auto-scaling. The selected feature is a performance indicator of a node in the data centre cloud.

Table 1. Dataset Features Used

No	Feature Name	Description
1	Node_name	Cloud node name
2	Day_of_week	Day of the week (1-7)
3	Time_only	Time
4	Day_number	Days of the month (1-31)
5	Cpu_load_avg_1_min	Average CPU Load per 1 minute
6	Cpu_load_avg_5_mins	Average CPU Load per 5 minutes
7	Cpu_load_avg_15_mins	Average CPU Load per 15 minutes
8	Cpu_usage_percentage	CPU Usage Percentage
9	Ram_used	RAM/memory used (MB)
10	Ram_total	RAM/memory total (MB)
11	Ram_usage_percentage	RAM usage percentage
12	Disk_used	Storage used (GB)
13	Disk_total	Total storage (GB)
14	Net_packet_send	Download bandwidth (MiB/s)
15	Net_packet_received	Upload bandwidth (MiB/s)
16	Label	Server workload status

Dataset retrieval mechanism for the system auto-scaling carried out on 10 virtual machines (VM) according to Figure 3 with data features according to Table 1 Is carried out every 15 minutes. The dataset was labelled using the criteria according to Table 2 [2]. After the data has been collected and labelled, the workload status is then sent to the machine learning Forecasting server to be formed into a dataset for the training process later.

Table 2. Performance Criteria and Percentage of Capacity Utilization

No	Performance Criteria	Percentage of Usage Capacity
1.	Very Low	0%-25%
2.	Low	25%-50%
3.	High	50%-75%
4.	Very High	75%-100%

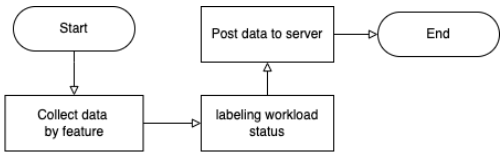

Figure 3. The process of collecting data into datasets

2.3 System Architecture Design

Following in Figure 4 Is the system architecture design autoscaling generally uses a horizontal scaling mechanism by monitoring the workload on the application server [11].

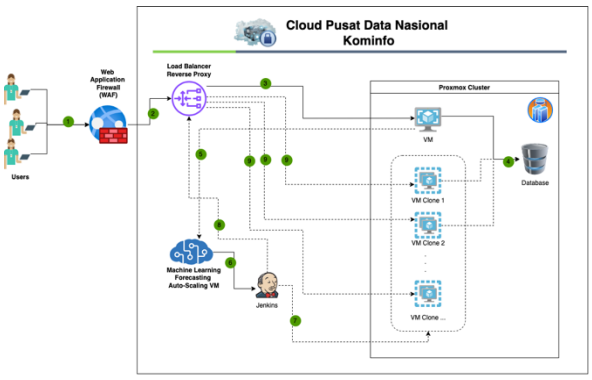

Figure 4. Auto-Scaling System Architecture

The difference from the previous architecture is that there is Machine Learning Forecasting. [12], which is used to monitor workload Application server VMs, and perform predictions and triggers to perform resource scalability, either adding resources (scale-up) horizontally, namely doing clone VM, or reducing resources (scale-down) according to the results of the monitored server workload prediction classification.

2.4 Prediction Model System Design

Designing a prediction model is part of system design by the previous system architecture design. The prediction model is formed based on previously collected datasets, which include features such as timestamp, CPU processor percentage, RAM usage, disk usage, and traffic bandwidth. Here are the steps to take:

Feature Extraction from Timestamp: Extracted features traction as in Table 3 Includes days of 1 week and the date is within 1 month, which allows the model to understand time-based work patterns.

Table 3. Prediction Model Building Feature Data

No	Feature Name	Description
1	Node_name	Cloud node name
2	Day_of_week	Day of the week (1-7)
3	Time_only	Time (hh:mm: ss)
4	Day_number	Days of the month (1-31)
5	Ram_total	RAM/memory total (MB)
6	Disk_total	Total storage (GB)

Unchanging Features: Some features such as total RAM, total storage, and node names do not change over time and are considered fixed features as in Table 3.

Predicted Features: The features whose values will be predicted are CPU workload estimates, CPU usage, memory usage, and bandwidth usage. According to Table 4.

Table 4. Prediction Model Target Class Data

No	Target	Description
1	Cpu_load_avg_15_mins	Average CPU Load per 15 minutes
2	Cpu_usage_percentage	CPU Usage Percentage
3	Ram_usage_percentage	RAM usage percentage

Prediction Model Building: The prediction model used by several algorithms [12] based on the results of previous research produces good performance among the algorithms ensemble learning [13][14][15] among others Random Forest [16], Gradient Boosting [17], XGBoost [18], as well as the algorithm Neural Network[19][20] i.e Multi-Layer Perceptron (MLP) [20] and Long Short-Term Memory (LSTM) [21]. The model is trained to learn patterns from the extracted features and predict unknown values based on these features. Figure 5 shows the general stages of machine learning, in which datasets are grouped into 2 subsets of data, namely the training subset and the testing subset, from the training data is carried out to form an ML model using the algorithm mentioned previously. Then

make predictions using the testing data that has been prepared.

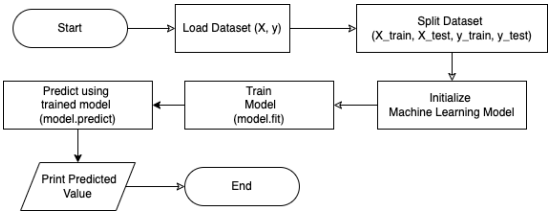

Figure 5. General Stages of Machine Learning (ML)

Classification of results from prediction regression: after getting the performance results from the best model, the results of the predictive regression are then classified according to the criteria in Table 2 and Figure 6. If the regression prediction results are less than 25% they will be categorized as very low, less than 50% be low, less than 75% be high, and above it becomes very high.

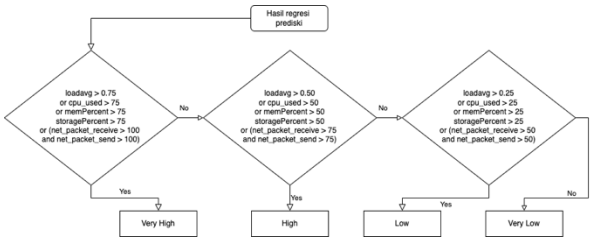

Figure 6. Prediction result classification criteria

2.5 Automation System Design

The mechanism for sending workload data is carried out according to Figure 7 Process of monitoring workload data from Virtual Machine (VM). The VM being monitored will send its workload status data according to the parameters in Table 3 and Table 4 To the ML Forecasting server. This process is carried out every 15 minutes.

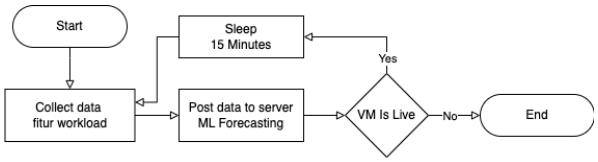

Figure 7. The process of monitoring workload data from VM

Then ML System Forecasting analyzes this data and predicts workload status for some time in the future. ML mechanism Forecasting as in Figure 8 Process forecasting data workload From the VM being monitored, the ML Forecasting server performs workload predictions, after generating the workload prediction status then triggers the Jenkins server to carry out scale-up or scale-down.

Jenkins does auto-scaling with a horizontal-scaling mechanism, which clones the monitored server up to a certain number. On Jenkins, there are 2 jobs doing scale-up and scale-down and configuring Load Balancing.[22], [23], [24] according to Figure 9 To the resulting VM auto-scaling.

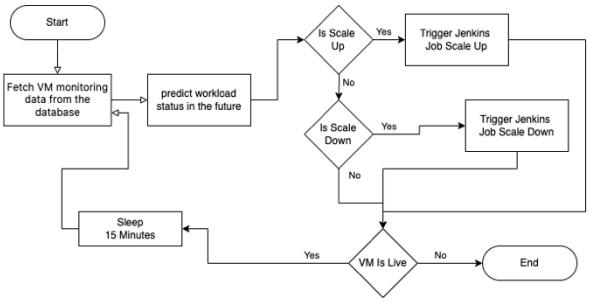

Figure 8. Workload data forecasting process of monitored VM

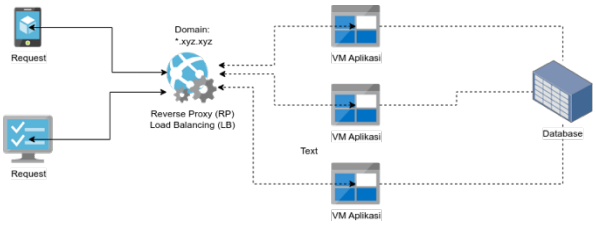

Figure 9. Reverse Proxy Server and Load Balancing

2.5 Implementation and Testing

At this stage, implementation is carried out based on architectural design and infrastructure design cloud to be built, the automation system auto-scaling, and forecasting machine learning will be combined, then system testing of the system will be carried out auto-scalinghis.

Testing is carried out to test the prediction results using MSE, MAE, and Correlation Coefficient. [25] [26]. Testing the classification results uses the method of k-fold cross-validation which uses a confusion matrix to get the value of accuracy, precision, recall, and f1 score. [27].

3. Results and Discussions

3.1 Dataset Characteristics

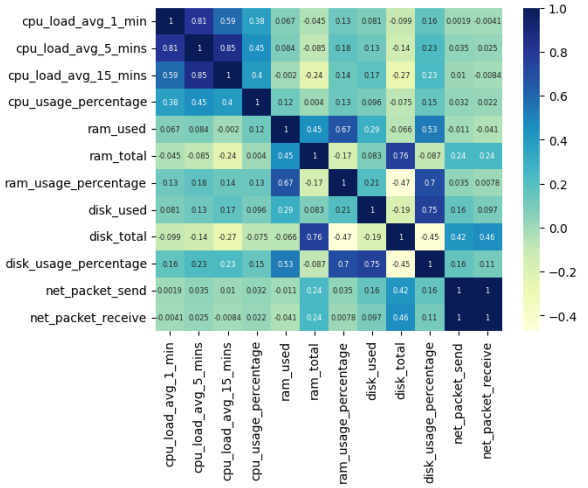

Figure 10. Correlation Heatmap Between Dataset Features

The dataset used from March to the end of May 2024 amounts to 78044 data originating from 10 cloud nodes. Based on the dataset that has been collected, the correlation between the features is obtained as follows

in Figure 10. It is found that CPU Usage and CPU Load are certainly highly correlated, and s based on correlation value.

After measuring the workload correlation between servers to determine the connectiv, isity between servers according to the dataset, the correlation matrix results were obtained for CPU usage, average CPU load in 15 minutes, and RAM usage as follows in Figure 11, Figure 12, Figure 13. From the results of Figure 11, it is obtained that the correlation between servers from the percentage of CPU usage still has a relationship that can be said to be high, especially in VM 0 and 1; VM 0 and 6; VM 1 and 6; and VM 3 and 5.

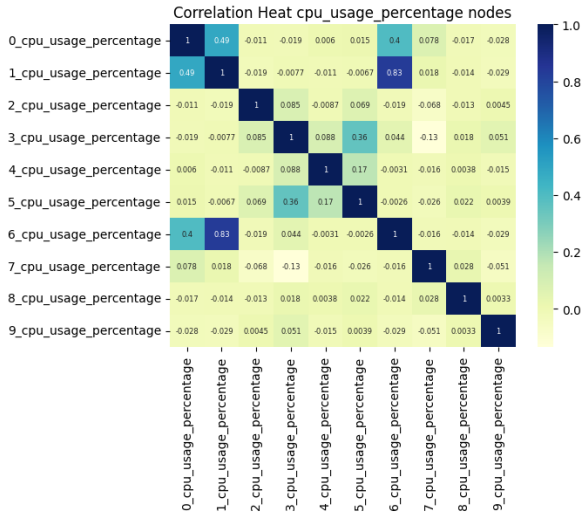

Figure 11. Correlation Heat CPU Usage Between Nodes

From Figure 12, the CPU load load shows that the server has a weak correlation. Meanwhile, in Figure 13, the percentage of RAM usage shows various strong and weak correlations, which correlate or can be said to have a connection, namely VM 0 and 6; VM 0 and 8; VM 3 and 8; VM 6 and 8; as well as VM 8 and 9. From the results of determining the correlation, it can be concluded that in general the workload between one server and another server is connected or influences each other, so that in making a prediction model you can utilise the dataset from each cloud node, to predict the workload of another server.

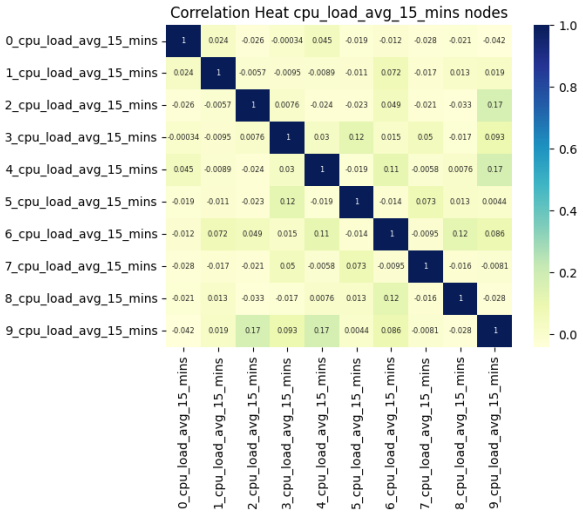

Figure 12. Correlation Heat CPU Load 15 Minutes Between Nodes

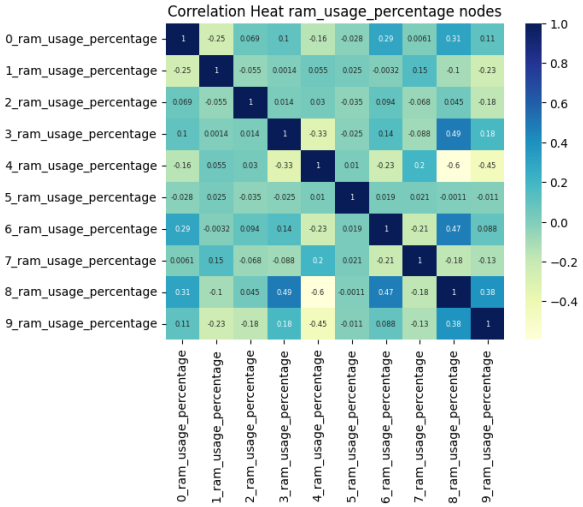

Figure 13. Correlation Heat RAM Usage Between Nodes

Table 5. Specifications of The Tools Used

Server Tools	Proxmox Cluster	Jenkins	NGINX	Forecasting Server	App Server
OS	Debian 11	Debian 11	CentOS 7	Ubuntu 22	CentOS 7
CPU	68 * 2.10 GHz	4 * 2.10 GHz	4 * 2.10 GHz	6 * 2.10 GHz	4 * 2.10 GHz
RAM	85 GB	4 GB	8 GB	4 GB	4 GB
Disk	8 TB	500GB	500 GB	100 GB	100 GB

The implementation of the system design required for hardware and software is as follows in Table 5. The following are the results of the implementation of Jenkins to carry out VM scale-up and scale-down jobs, wherein each job there is a stage or stages carried out by Jenkins in handling VM auto-scaling according to Table 6. From the results of implementing Jenkins, it takes approximately 6-8 minutes to carry out the process scale-up and takes around 1-2 minutes to carry out the process scale-down.

Table 6. State Scaling Processes And Time

No	State Scaling	Takes Time	Stages
1	Scale-Up	6min 15s	Clone VM; Configure Reverse Proxy; Reload Reverse Proxy
2	Scale-Down	1min 53s	Configure Reverse Proxy; Reload Reverse Proxy; Remove Clone VM

3.2 Implementation and Testing of Forecasting Model

Before performing the forecasting process: based on the status workload from the VM being monitored according to the process in Figure 8. Then the dataset that was previously obtained was divided into 2 data subsets, namely 70% training subset, and 30% testing subset. The following are the results of the distribution plot of the percentage of CPU usage in Figure 14, CPU Load in Figure 15 RAM usage is shown in Figure 16 for each node cloud.

Figure 14. Time-series graph of CPU usage percentage

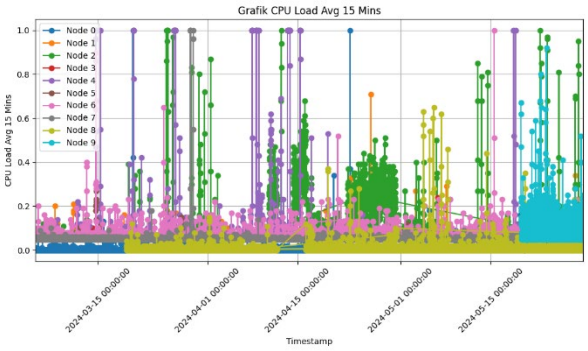

Figure 15. Time-series graph of CPU Load percentage Average 15 minutes

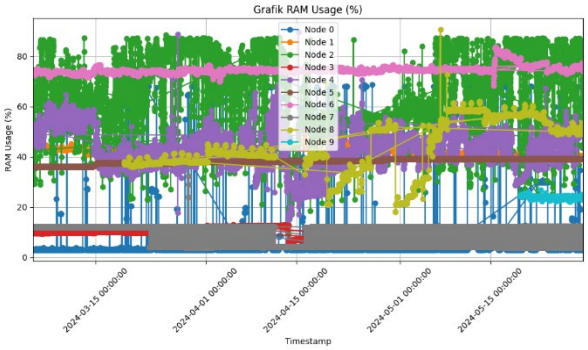

Figure 16. Time-series graph of RAM usage percentage

The results of the distribution of CPU usage from Figure 14 show an unbalanced distribution of workload status data, where the distribution data is in the same data status workload Very Low and Low (usage less than 50%) is very large compared to High or Very High status (more than 50% usage) of each node. From the results of the data distribution in Figure 15, it is found that the CPU load of 15 minutes is very low (less than 0.25 or 25%) and greater than the other workload statuses, resulting in an unbalanced data distribution from each status for each node.

From the results of the distribution of time-series data for RAM percentage usage in Figure 16, we get a distribution that can be said to be even between each workload status, namely Very Low, Low, High, and Very High.

Testing is needed in the selection of machine learning algorithms: which have good performance. Algorithm comparison machine learning used is the model ensemble learning that is Random Forest Regressor (Bagging), Gradient Boosting (Boosting), Extreme Gradient Boosting (XGBoost), and Stacking which combines between Random Forest and XGBoost (Bagging then Boosting). Apart from algorithms ensemble learning is Also used as the algorithm of the algorithm Neural Networks, MLP and LSTM. Before testing the results of the algorithm, first, create a model that uses 70% of the data from the dataset resource cloud node which is used as training data. Then, after the model is formed, the prediction regression results are tested using 30% of the data from the remaining dataset as testing data.

The following is a graphic comparison of the plot between the predicted values produced by each algorithm and the actual values. Test data samples are taken based on testing data (30% of the previous dataset as testing data) and then some data is taken from each data quartile (Q1, Q2, Q3, Q4).

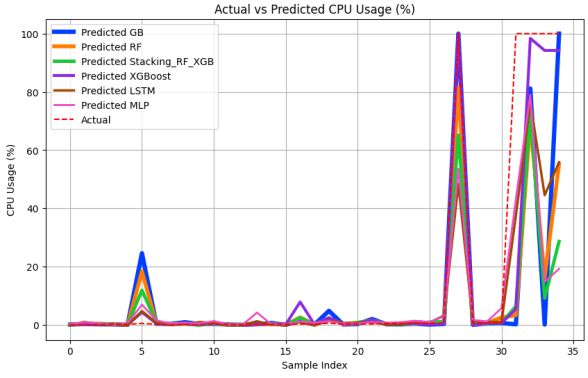

Figure 17. Comparison chart of predicted and actual CPU usage

Figure 17 shows the regression results of CPU usage predictions compared to the actual values for each algorithm. From Figure 17, which is the result of the CPU usage test data samples used, totalling 34 test data samples, you can see at a glance that the results have good predictions, namely XGBoost (purple) because the prediction results are almost close to the actual value. At first glance, MLP seems to have poor results compared to other algorithms.

Figure 18 shows the regression results of CPU Load predictions per 15 minutes compared to the actual values for each algorithm. From Figure 18, the results of the CPU Load test data samples averaged 15 minutes, totalling 27 sample data that have good performance, namely Gradient Boosting (blue).

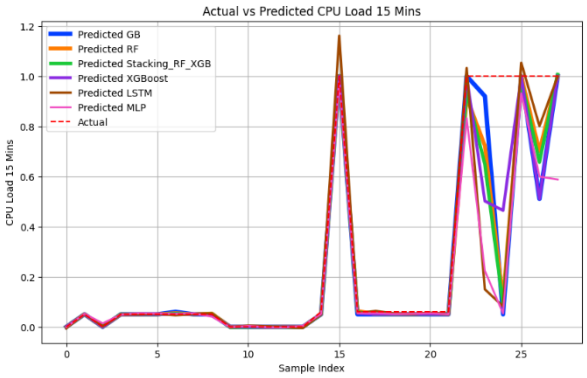

Figure 18. Comparison chart of predicted and actual CPU load average of 15 minutes

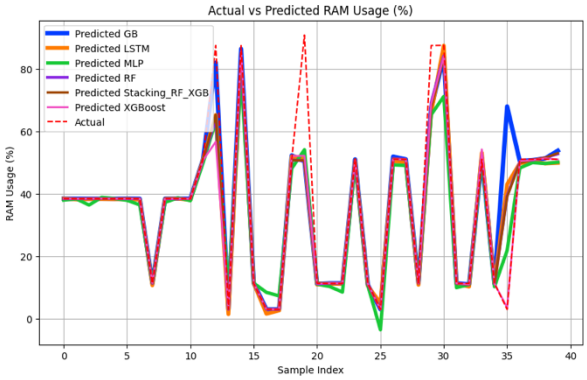

Figure 19. Comparison chart of predicted and actual RAM usage

From Figure 19, which is the result of testing samples of RAM usage totalling 39 samples, which has good performance at a glance, namely XGBoost (pink colour) because the prediction results are almost close to the actual value.

Performance testing results of prediction regression using the above algorithm: The following are the results of performance testing from predictive regression using the algorithm to produce CPU usage predictions in Table 7. From the results of testing the CPU usage regression prediction performance in it is found that the Stacking model (RF and XGBoost) has better regression prediction performance. It can be seen that the lowest MSE results are 0.00083 in Stacking (RF and XGB), the lowest RMSE is 0.02882 in Stacking (RF and XGB), the lowest RAE is 0.00002 in Stacking (RF and XGB), and the highest coefficient of determination is 0.59887.

In the performance testing results from the regression of CPU load predictions for an average of 15 minutes in Table 8 Using several algorithms and machine learning. From the results of the CPU load regression prediction performance test for an average of 15 minutes, it was found that the lowest MSE was Stacking (RF and XGB), namely 0.00064, the lowest RMSE was 0.02536 Stacking (RF and XGB), the lowest RRSE is 0.43756 Stacking (RF and XGB), and the highest coefficient of determination is Stacking (RF and XGB). So it can be concluded that the best prediction is the Stacking algorithm (Random Forest and Extreme Gradient Boosting). However, if you look at the previous comparison graph, the Random Forest algorithm has almost similar performance to Stacking (RF and XGB) and has a difference that can be said to be small.

Table 7. CPU Usage Prediction Performance Evaluation Results

Algorithm	MSE	RMSE	MAE	RSE	RRSE	RAE	R^2
GB	$1.35x10^{-3}$	0.037	$7.65x10^{-3}$	0.65	0.81	$2.61x10^{-5}$	0.35
LSTM	$1.07x10^{-3}$	0.033	$8.85x10^{-3}$	0.52	0.72	$3.01x10^{-5}$	0.48
MLP	$1.30x10^{-3}$	0.036	$9.89x10^{-3}$	0.63	0.79	$3.37x10^{-5}$	0.37
RF	$9.21x10^{-4}$	0.03	$6.99x10^{-3}$	0.44	0.67	$2.38x10^{-5}$	0.56
Stacking (RF_XGB)	$8.30x10^{-4}$	0.029	$7.06x10^{-3}$	0.4	0.63	$2.40x10^{-5}$	0.6
XGBoost	$1.18x10^{-3}$	0.034	$7.52x10^{-3}$	0.57	0.76	$2.56x10^{-5}$	0.43

Table 8. CPU Load Average 15 Minutes Prediction Performance Evaluation Results

Algorithm	MSE	RMSE	MAE	RSE	RRSE	RAE	R^2
GB	$9.14x10^{-4}$	0.0302	$5.79x10^{-3}$	0.272	0.522	$1.44x10^{-5}$	0.728
MLP	$1.14x10^{-3}$	0.0337	$7.67x10^{-3}$	0.338	0.581	$1.91x10^{-5}$	0.662
LSTM	$1.02x10^{-3}$	0.0319	$7.86x10^{-3}$	0.304	0.551	$1.95x10^{-5}$	0.696
RF	$6.52x10^{-4}$	0.0255	$5.67x10^{-3}$	0.194	0.441	$1.41x10^{-5}$	0.806
Stacking (RF_XGB)	$6.43x10^{-4}$	0.0254	$5.94x10^{-3}$	0.191	0.438	$1.48x10^{-5}$	0.809
XGBoost	$9.04x10^{-4}$	0.0301	$5.93x10^{-3}$	0.269	0.519	$1.48x10^{-5}$	0.731

Table 9. RAM Usage Prediction Performance Evaluation Results

Algorithm	MSE	RMSE	MAE	RSE	RRSE	RAE	R^2
GB	$2.12x10^{-3}$	0.046	0.008	$2.65x10^{-2}$	0.163	$1.49x\ 10^{-6}$	0.973
LSTM	$2.00x10^{-3}$	0.045	0.015	$2.50x10^{-2}$	0.158	$2.75x\ 10^{-6}$	0.975
RF	$1.44x10^{-3}$	0.038	0.008	$1.81x10^{-2}$	0.134	$1.45x\ 10^{-6}$	0.982
MLP	$3.28x10^{-3}$	0.057	0.026	$4.11x10^{-2}$	0.203	$4.67x\ 10^{-6}$	0.959
Stacking (RF_XGB)	$1.44x10^{-3}$	0.038	0.009	$1.81x10^{-2}$	0.134	$1.53x\ 10^{-6}$	0.982
XGBoost	$1.96x10^{-3}$	0.044	0.009	$2.46x10^{-2}$	0.157	$1.62x\ 10^{-6}$	0.975

The performance test results from the RAM usage prediction regression in Table 9 use several algorithms and machine learning. From the results of testing the regression prediction performance of RAM usage in Table IX, the lowest MSE is 0.00144, namely Stacking (RF and XGB), the lowest RMSE is 0.03799 Stacking

(RF and XGB), the lowest RRSE is 0.13436 in Stacking (RF and XGB), the highest coefficient of determination is 0.98195 in Stacking (RF and XGB).

Classification of the results of prediction regression: After regressing the predicted values from CPU usage, CPU Load, and RAM usage, the values were then classified into workload status according to Figure 6. The confusion metrics were obtained based on the classification test.

Table 10 are the results of testing the CPU usage classification of each algorithm. An algorithm that has good performance is obtained, namely Random Forest with an accuracy value of 0.9958, precision of 0.7276, recall of 0.539 and f1 score of 0.5875.

Table 10. CPU Usage Classification Performance

Algorithm	Accuracy	Precision	Recall	F1-Score
GB	0.9949	0.6670	0.6085	0.6325
LSTM	0.9953	0.6685	0.4754	0.5195
RF	0.9958	0.7276	0.5390	0.5875
Stacking (RF_XGB)	0.9956	0.4988	0.4672	0.4679
XGBoost	0.9947	0.6536	0.5744	0.5922
MLP	0.9943	0.4782	0.3013	0.3302

Table 11 are the results of the average CPU Load classification test per 15 minutes for each algorithm. It was found that Random Forest had better performance, with an accuracy value of (0.9920), precision (0.6586), recall (0.5820), and f1-score (0.6098).

Table 11. CPU Load Average 15 Minutes Classification Performance

Algorithm	Accuracy	Precision	Recall	F1-Score
GB	0.9915	0.5810	0.6096	0.5946
RF	0.9920	0.6586	0.5820	0.6098
Stacking (RF_XGB)	0.9919	0.6555	0.5878	0.6104
XGBoost	0.9906	0.5806	0.5760	0.5776
LSTM	0.9906	0.6308	0.5475	0.5821
MLP	0.9907	0.6132	0.4352	0.4946

Table 12 Are the results of RAM usage classification testing for each algorithm. It was found that Random Forest had high precision (0.9607) and f1-score (0.9550) and Gradient Boosting had high accuracy (0.9732) and recall (0.9533).

Table 12. RAM Usage Percentage Classification Performance

Algorithm	Accuracy	Precision	Recall	F1-Score
GB	0.9732	0.9549	0.9533	0.9541
RF	0.9728	0.9607	0.9497	0.9550
Stacking (RF_XGB)	0.9715	0.9554	0.9481	0.9516
XGBoost	0.9707	0.9548	0.9474	0.9510
LSTM	0.9449	0.9072	0.8884	0.8970
MLP	0.9152	0.8641	0.8096	0.8272

To test the stability of the classification performance results, the k-fold validation test is used by dividing the dataset into several subsets called folds. This test is carried out to test the consistency of machine learning algorithm model results which is formed. The dataset is separated into 5 fold subsets, then classification testing is carried out.

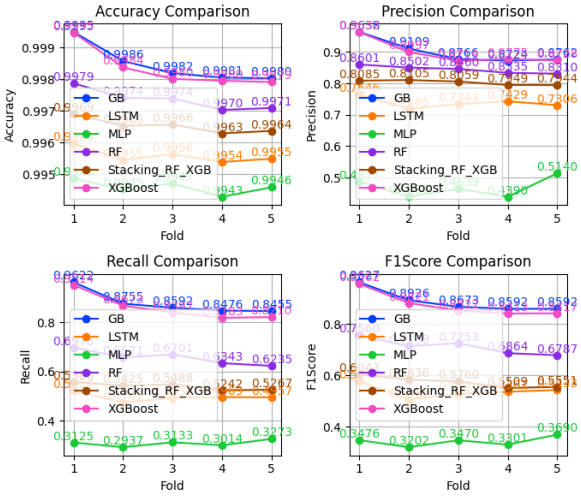

Figure 20. K-Fold results of the classification test of CPU usage

Table 13. Average Performance Results of CPU Usage Classification Using K-Fold Validation

Algorithm	Accuracy	Precision	Recall	F1-Score
GB	0.9985	0.9000	0.8780	0.8882
LSTM	0.9957	0.6869	0.5077	0.5481
MLP	0.9943	0.4989	0.3041	0.3359
RF	0.9973	0.8285	0.6550	0.7098
Stacking (RF_XGB)	0.9965	0.8029	0.5402	0.5740
XGBoost	0.9984	0.8970	0.8595	0.8747

Figure 20 is a graphic plot of accuracy, precision, recall and f1score from the k-fold cross-validation test for the percentage of CPU usage carried out in 5-fold validation. From the results in the graphic image, it was found that those with better test results were the Gradient Boosting and XGBoost algorithms.

From the graph in Figure 20, the average accuracy, precision, recall and f1score of each validation fold were then calculated. The results obtained as follows: in Table 13 are the average test results. k-fold cross-validation for CPU usage. An algorithm with better and more stable performance was obtained, namely Gradient Boosting (GB), with an accuracy value of (0.998), precision (0.9), recall (0.878), and f1-score (0.888). However, the Extreme Gradient Boosting (XGBoost) algorithm has almost similar performance to Gradient Boosting (GB).

Figure 21 shows a plot graph of accuracy, precision, recall and f1score from the k-fold cross-validation test for CPU Load on average for 15 minutes carried out in 5-fold validation. From the results in the graphic image, it was found that the test results were better, namely the Gradient Boosting algorithm, but XGBoost also had almost similar test results.

From the graph plot in Figure 21, the results of k-fold validation are then displayed perform an average calculation for accuracy of several of the models used, precision, recall, and f1score are contained in Table 14 are the results of k-fold validation testing for a CPU Load of 15 minutes. An algorithm with better and more stable performance was obtained, namely Gradient Boosting (GB), with an accuracy value of (0.997),

precision (0.854), recall (0.873), and f1-score (0.863). However, the Extreme Gradient Boosting (XGBoost) algorithm has almost similar performance to Gradient Boosting (GB).

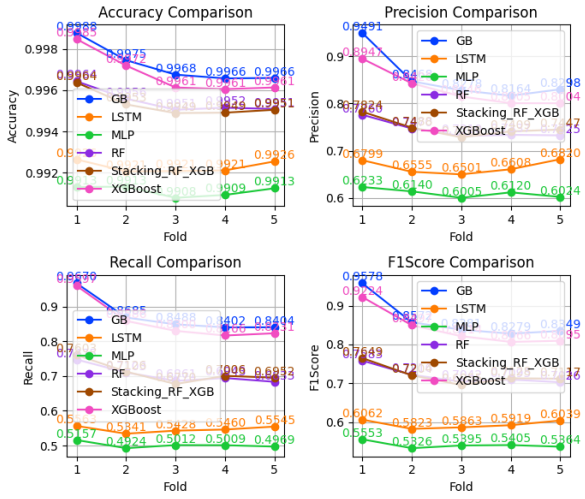

Figure 21. K-Fold results of the classification test of CPU load average 15 minutes

Table 14. Average Performance Results of Cpu Load Average 15 minutes Classification Using K-Fold Validation

Algorithm	Accuracy	Precision	Recall	F1-Score
GB	0.9985	0.9000	0.8780	0.8882
LSTM	0.9957	0.6869	0.5077	0.5481
MLP	0.9943	0.4989	0.3041	0.3359
RF	0.9973	0.8285	0.6550	0.7098
Stacking (RF_XGB)	0.9965	0.8029	0.5402	0.5740
XGBoost	0.9984	0.8970	0.8595	0.8747

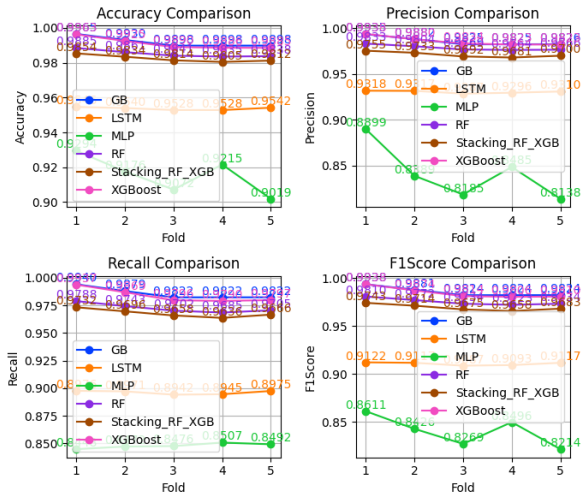

Figure 22. K-Fold results of the classification test of RAM usage percentage

Figure 22 shows a plot graph of accuracy, precision, recall and f1score from the k-fold cross-validation test for the percentage of RAM usage carried out in 5-fold validation. From the results in the graphic image, it was found that those with better test results were the Gradient Boosting and XGBoost algorithms.

Then, from the graph in Figure 22, the average calculation of the 5-fold validation results for evaluation metrics as in Table 15 is the test result. k-fold cross-validation for RAM usage. An algorithm with better and more stable performance was obtained, namely Gradient Boosting (GB), with an accuracy value of (0.992), precision (0.986), recall (0.986), and f1-score (0.986). However, the Extreme Gradient Boosting (XGBoost) algorithm has almost similar performance to Gradient Boosting (GB).

Table 15. Average Performance Results of RAM Usage Percentage Classification Using K-Fold Validation

Algorithm	Accuracy	Precision	Recall	F1-Score
GB	0.9918	0.9859	0.9857	0.9858
MLP	0.9164	0.8622	0.8153	0.8310
RF	0.9850	0.9782	0.9722	0.9751
XGBoost	0.9910	0.9860	0.9838	0.9848
LSTM	0.9482	0.9109	0.8927	0.9011
Stacking (RF_XGB)	0.9824	0.9712	0.9678	0.9695

4. Conclusions

From the results of the implementation and analysis in this research, it was found that the Stacking algorithm uses a base learner, namely Randon Forest However, when classifying cloud computing workload status, there are 4 statuses, namely: Very High, High, Low, and Very Low It was found that the Random Forest algorithm produced relatively better accuracy, precision, recall and f1-score values. Then, after carrying out stability testing using K-Fold Cross Validation for classification based on workload status, it was found that the Gradient Boosting algorithm had relatively better results among other algorithms, namely for the percentage of CPU usage with an accuracy of 0.998, precision 0.9, recall 0.878, f1score 0.888; CPU Load average 15 minutes with accuracy 0.997, precision 0.854, recall 0.863, f1score 0.863; Meanwhile, the percentage of RAM usage is accuracy 0.992, precision 0.986, recall 0.986, and f1score 0.986. However, the XGBoost algorithm also has test results that are almost close to the results of Gradient Boosting.

In the future, it is hoped that this paper can be developed by implementing a deep learning model in depth to train the model, with the hope that the deep learning model used will produce even better tests.

Acknowledgements

Support and funding for the publication of this paper were provided by the Indonesian Ministry of Communications and Information Republic Indonesia (KOMINFO RI) and for that we thank you.

References

[1] A. S. Balantimuhe, S. H. Pramono, and H. Suyono, "Konsolidasi Beban Kerja Kluster Web Server Dinamis dengan Pendekatan Backpropagation Neural Network," *Jurnal EECCIS (Electrics, Electronics, Communications, Controls, Informatics, Systems)*, vol. 12, no. 2, pp. 72–77, Sep. 2018, doi: 10.21776/jeeccis.v12i2.536.

[2] A. M. Al-Faifi, B. Song, M. M. Hassan, A. Alamri, and A. Gumaei, "Performance prediction model for cloud service selection from smart data," *Future Generation Computer*

Systems, vol. 85, pp. 97–106, Aug. 2018, doi: 10.1016/j.future.2018.03.015.

[3] P. D. Adane and O. G. Kakde, "Predicting Resource Utilization for Cloud Workloads Using Machine Learning Techniques," in *Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 1372–1376.

[4] A. A. Khaleq and I. Ra, "Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications," *IEEE Access*, vol. 9, pp. 35464–35476, 2021, doi: 10.1109/ACCESS.2021.3061890.

[5] S. T. Singh, M. Tiwari, and A. S. Dhar, "Machine Learning based Workload Prediction for Auto-scaling Cloud Applications," in *2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development, OTCON 2022*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/OTCON56053.2023.10114033.

[6] W. Iqbal, A. Erradi, M. Abdullah, and A. Mahmood, "Predictive Auto-Scaling of Multi-Tier Applications Using Performance Varying Cloud Resources," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 595–607, 2022, doi: 10.1109/TCC.2019.2944364.

[7] S. Manam, K. Moessner, and P. Asuquo, "A Machine Learning Approach to Resource Management in Cloud Computing Environments," in *IEEE AFRICON Conference*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/AFRICON55910.2023.10293275.

[8] R. A.) Eric Bauer, *Reliability and Availability of Cloud Computing*. Wiley-IEEE Press, 2012.

[9] V. Millnert and J. Eker, "HoloScale: horizontal and vertical scaling of cloud resources," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, 2020, pp. 196–205. doi: 10.1109/UCC48980.2020.00038.

[10] C.-Y. Liu, M.-R. Shie, Lee Yi-Fang, and K.-C. Lai, *ICISA 2014 : 2014 Fifth International Conference on Information Science and Applications : 6-9 May, 2014, Seoul, Korea*. 2014.

[11] J. Bi *et al.*, "Application-Aware Dynamic Fine-Grained Resource Provisioning in a Virtualized Cloud Data Center," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1172–1184, 2017, doi: 10.1109/TASE.2015.2503325.

[12] V. P. M. Arif Wani, *Deep Learning Applications, Volume 4*. in Advances in Intelligent Systems and Computing, 1434. Springer, 2023.

[13] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15.

[14] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022, doi: 10.1109/ACCESS.2022.3207287.

[15] C. Zhao, R. Peng, and D. Wu, "Bagging and Boosting Fine-Tuning for Ensemble Learning," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 4, pp. 1728–1742, 2024, doi: 10.1109/TAI.2023.3296685.

[16] L. Breiman, "Random Forests," *Mach Learn*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[17] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189 – 1232, 2001, doi: 10.1214/aos/1013203451.

[18] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[19] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992, doi: https://doi.org/10.1016/S0893-6080(05)80023-1.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Jul. 1997, doi: 10.1162/neco.1997.9.8.1735.

[22] S. Sharma, R. Garg, and D. K. Lobiyal, "Load Balancing Algorithms in Cloud Computing: A Comparative Study," *International Journal of Advanced Research in Computer Science and Software Engineering*, 2014.

[23] S. Muthukrishnan and V. Sankaranarayanan, "A Survey of Load Balancing Techniques in Cloud Computing Environments," *Journal of Network and Computer Applications*, 2016.

[24] M. G. Nair, S. Bhuvaneswari, and S. S. Baboo, "A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms," *Int J Comput Appl*, 2015.

[25] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[26] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013.

[27] H. T. Jiawei Han Jian Pei, *Data Mining: Concepts and Techniques*, 4th ed. in The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2022.

