Evaluation of Resource Allocation in Cloud using Machine Learning

Suhad Ibrahim Mohammed and Ziyad Tariq Mustafa Al-Ta'i
Department of Computer Science, College of Sciences, University of Diyala, 32001 Baqubah, Diyala, Iraq
{scicomphd222307, Ziyad1964tariq}@uodiyala.edu.iq

Keywords: Cloud Computing, Machine Learning, Virtual Machine, Resource Allocation.

Abstract: Cloud computing has revolutionized the way computing the resources are allocated and managed, and it offers scalability, flexibility, and the cost savings. Proper resource allocation, however, remains a difficult problem due to varying workloads, unpredictable demand, and the need for optimal performance. Machine learning (ML) techniques have been recognized as a promising solution to optimizing the resource allocation by predicting workload patterns, optimizing resource utilization, and reducing latency. In this paper, we have compared the various ML-based framework for the resource allocation in the cloud computing environments on the basis of their efficiency in order to improve the efficiency and the cost control. Through comparative evaluation, we highlight the merits and demerits of different ML models, including the contextually of their suitability in the actual implementations. The results reveal that the proposed model achieves accuracy (Decision Tree 100%, AdaBoost 72.2%, Support vector machine 98.5%, logistic regression 97.6% and Gradient boosting 100%).

1 INTRODUCTION

Industry and academia increasingly transfer applications to the cloud so developers can deploy and operate applications without server configuration and setup complexity. Cloud providers continuously innovate their services to provide greater efficiency, service quality, and resource control. Cloud computing offers scalable computer resources via the Internet, reducing job completion time, makepan, and cost of operations. Cloud computing resource allocation is the efficient distribution of compute resources such as CPU, memory, storage, and network bandwidth to tasks and applications. Correct resource allocation ensures maximum performance, cost efficiency, and user satisfaction. Virtualization plays a key role in the process since it enables many virtual machines (VMs) to execute on one physical server. VMs provide a private and dynamic operating environment for running applications, with dynamically scalable resources based on the need to enhance flexibility and efficiency within the system [1-8].

Virtualization is the foundation of cloud computing, as it creates virtual depictions of physical hardware, maximizes resource utilization, and enhances system performance, energy consumption, and data center efficiency. In addition, job scheduling and load balancing are crucial for workload distribution over cloud resources. Job scheduling schedules and assigns jobs based on resource requests and time constraints, while load balancing prevents server overload through network traffic balancing [9].

Machine learning (ML) as a subset of artificial intelligence is unfolding as a very efficient means of optimizing resource management in the cloud. By learning from experience and finding patterns, ML algorithms are able to predict workload fluctuations, enhance decision-making, and automate the process of assigning resources. The current paper explores the significance of having an even task distribution across VMs to ensure optimal resource utilization. It proposes a sequential distribution approach for optimizing resource distribution, minimizing access times, and overall cloud system performance by overcoming bottlenecks and latencies [10].

This study aims to evaluate the effectiveness of machine learning (ML) algorithms for cloud computing resource optimization based on the framework. It aims to analyse key challenges in resource allocation, such as scalability, workload prediction, and cost optimization while exploring ML-based approaches to enhance efficiency. The study focuses on maximizing the utilization of

virtual machines (VMs) through smart scheduling and load balancing to ultimately enhance system performance by reducing execution time, power consumption, and latency. In addition, it seeks to propose a sequential distribution method for the assignment of tasks in order to limit bottlenecks and enhance cloud efficiency. By comparing traditional and ML-based resource allocation approaches, this research provides perspective on performance improvement and cost saving and facilitates the design of adaptive cloud resource management solutions.

2 RELATED WORKS

Resource management has been transformed by cloud computing, which has resulted in the development of a variety of methods for efficient allocation and work planning. Traditional methods, including opportunistic load balance (OLB) algorithms and min-min-min, distribute duties according to predetermined criteria; however, they encounter difficulties when dealing with dynamic cloud work. Artificial intelligence (AI) and machine learning (ML) techniques have been implemented in recent years to mitigate these constraints and enhance resource management. For instance, Mamalis et al. [8] proposed a Gravitational Search algorithm (GSA)- based work scheduling algorithm for cloud computing virtual machines. Based on Newtonian physics and agent-based optimization, GSA significantly improved scheduling efficiency. GSA-based makespan reduction had an average performance improvement of 3.5% over ESTA, 7.7% over PSO, 5.1% over Min-Min, and 27.3% over OLB. GSA outperformed these strategies by 6.7%, 9.2%, 14.5%, and 68.7% in average resource consumption, making it a promising cloud computing strategy. Anbarkhan and Samar Hussni [9] proposed a machine learning-based cloud resource management method that improves resource usage by 30% and reduces operational costs by 25%. Their method improves system performance, latency, and prediction accuracy. Simulations show it can transform cloud service management. According to Zhang et al. [10], the GAACO algorithm for cloud computing outperforms standard scheduling algorithms in time cost minimization, quality of service enhancement, and system load balancing. By balancing time, money, reliability, and system load, GAACO shows the revolutionary appeal of AI-enabled cloud computing solutions. Shetty et al. [11] optimized virtual machine job scheduling settings with feed-forward neural networks. They used Principal Component Analysis (PCA) to choose relevant features for dynamic scheduling to maximize resource consumption and cloud system performance. In Infrastructure-as-a-Service (IaaS) contexts, their approach maximizes throughput and minimizes makespan using Min-Min and MET algorithms. Khan et al. [12] explored cloud computing security approaches and distributed architectural concerns. They stressed that while cloud growth might improve security, innovative models should work with existing systems without compromising critical functions. Their goal was to improve security without compromising other system characteristics. Swarna et al. [13] examined green cloud computing load balancing in the IoE paradigm. They improved energy consumption, cloud storage, data processing, and end-user services using Wind-Driven Optimization and Firefly algorithms. They also used IoT network clustering to develop intelligent information processing solutions for energy-efficient cloud systems. Gomathi and Karthikeyan [14] introduced hybrid swarm optimization for task scheduling to reduce project completion time. Their method optimizes cloud computing workload and resource allocation. Kumar, Singh, and Buyya [15] proposed a neural network trained by black hole events for predictive workload control. They use deep learning, an evolutionary algorithm, and backpropagation to anticipate 99.9% more accurately than previous approaches. The Coronavirus Herd Immunity Optimizer-derived scheduling algorithm minimizes makespan and improves speed, efficiency, and throughput. Renyu Yang et al. [16] proposed data-driven profiling, issue formulation, and supervised modelling for end-to-end cloud computing optimization. They stressed that ML-based solutions improve system efficiency and architecture. ML algorithms must be integrated with edge and cloud devices to maximize performance and resource management.

To the best of our knowledge, substantial research has been conducted in the past few years to understand, manage, and mitigate resource allocation in the cloud. Limited work has been conducted on the subject defined by these challenging circumstances. Therefore, the purpose of this research is to establish a framework that improves the quality of resource allocation by utilizing machine learning to improve the cloud's performance.

3 MACHINE LEARNING

Classification extracts latent knowledge from mass data after pre-processing. Its essential operation determines data element internal relativity. As mentioned, machine and deep learning methods are likely utilized for classification.

3.1 Support Vector Machine Algorithm

SVM, a classification and regression technique, finds a multi-dimensional hyperplane to best segregate data classes [20]. SVM maximizes the distance between the hyperplane and nearest points from both classes to determine the best data separation hyperplane. The margin is the distance between the hyperplane and nearby points, and support vectors change its position to improve classification accuracy and performance even with high-dimensional data [21].

3.2 Decision Tree Algorithm

Decision trees are hierarchical data structures containing leaf and non-leaf nodes used for classification and regression trees, with decision criteria differing by structure [22]. A rooted decision tree with leaf and non-leaf nodes represents classification and selection choices based on input attribute values [23]. This work uses a static decision tree-based approach to pick a priority rule combination for processing jobs, allowing planning without rule modifications over the scheduling horizon [24].

3.3 Gradient Boosting

For categorization and versioning, machine learning uses Gradient Boosting. It corrects faults consecutively when building models. Cloud computing allows effective training of complex models on massive datasets due to scalable architecture and dispersed computing capacity. Cloud-based clusters can parallelize training and handle enormous datasets. [25] Gradient-boosting ensemble learning creates models successively, correcting errors by gradient descent. Fitting weak learners (usually decision trees) to residual mistakes minimizes a loss function [26].

3.4 AdaBoost (Adaptive Boosting)

AdaBoost builds a powerful classifier from numerous weak learners. Giving misclassified data items more weight enhances categorization. It uses distributed computing resources in cloud computing for efficient large dataset handling, faster training, and cost-effective resource management [27]. AdaBoost, an ensemble learning algorithm, combines weak classifiers (usually decision stumps) to create a powerful classifier. Misclassified samples are weighted to focus on hard-to-classify cases in the following iterations.

3.5 Logistic Regression

Logistic regression is a statistical method in binary classification to estimate the chance of an instance belonging to either of two categories. It invokes a logistic (sigmoid) transformation on a linear combination of features, resulting in a value from 0 to 1. It is utilized in fields of healthcare, finance, and marketing for tasks of disease prediction, credit scoring, and customer churn prediction.

4 METHODOLOGY

Cloud computing requires an ideal dataset for efficient resource allocation based on real-world activity and system behaviour. To this end, in Figure 1, the phase 1 machine has focused on data collection, simulation, and preaching to generate high-quality datasets that are suitable for learning models. It begins with the GOCJ Google Cloud Jobs data set, a well-known data set that provides valuable information about cloud job planning and execution behaviour. Raw data sets, however, are rarely highly variable, and raw data is incapable of approximating the dynamic quality of the environment of blame. Monte Carlo simulation can bypass this deficiency by producing a representative set of sales data and modelling varied charging distribution, resource requirements, and execution time. When the dataset is generated, the following process traverses a collection dataset that captures important virtual machine (World Cup) view measurements required in examining resource allocation methods. Such matrices include:

1) Makespan: Total execution time required to complete all the scheduled functions.
2) Memory usage: The space occupied by running the charge.
3) Idle time: When the World Cup is short, there is a breakage that affects the overall efficiency.
4) Throughput: The speed at which the tasks are handled within a given time frame.

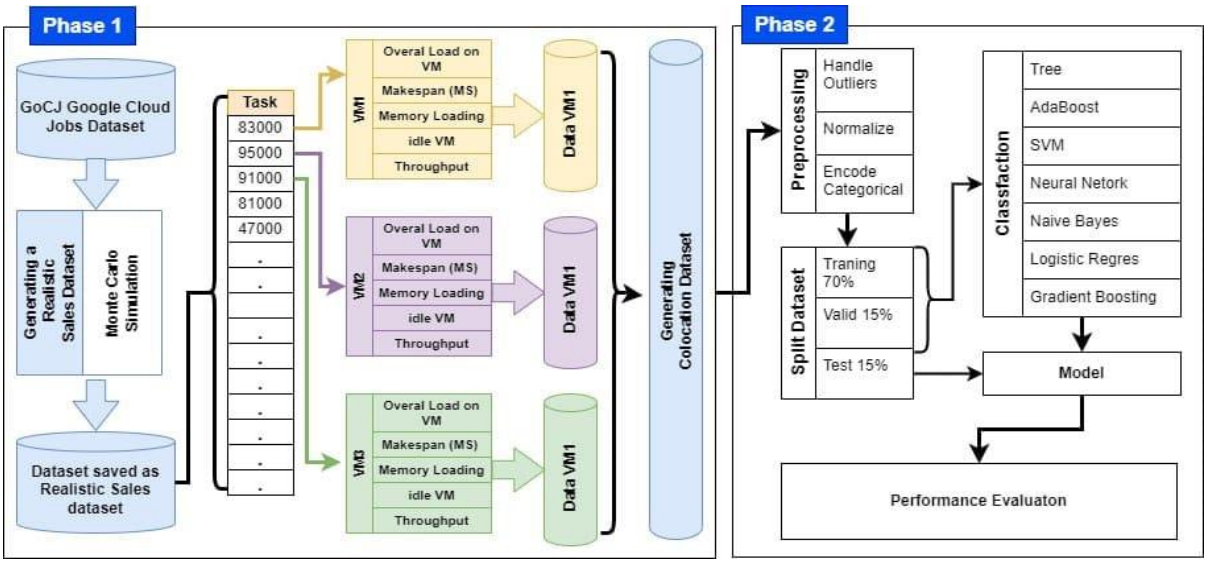


Figure 1: The proposed Framework recall and F1 score.

Organizing the dataset in this way enables a more accurate and in-depth analysis of resource usage, job planning efficiency, and workload balancing. Colocation is a core input in dataset phase 2, where machine learning techniques make resource allocation dynamic.

In phase 2, the dataset undergoes a series of reservative steps to increase quality and ensure optimal performance in the machine learning model. Raw data often consists of functions with deviations, lack of values, outliers and separate scales, which can significantly affect the model's accuracy. To solve these challenges, the preparation stage includes dealing with generalization and classified coding and ensuring that the dataset is well structured and the training machine is suitable for training the learning algorithms. After preprosautting, the dataset is divided into three Multier to facilitate model training, verification and evaluation:

- Training (70%): A machine is used to train learning models so that they can learn from previous computer patterns and develop predicting abilities.
- Verification (15%): The fine-tuning model helps with parameters and ensures that the algorithm normalizes unseen data without overfitting.
- Test set (15%) – This is Reserved for final evaluation and provides a fair assessment of the model's actual world performance.

When the dataset is prepared, various machine learning classifiers are used to predict optimal resource allocation in the cloud environment. We evaluated several models, including decision trees, AdaBoost, SVM, logistic regression, and gradient boosting, chosen for their potential benefits in workload planning, resource optimization, and system efficiency. To determine their effectiveness for cloud resource management, the final models were evaluated based on accuracy, precision, recall, and resource utilization metrics.

Selecting and integrating these ML techniques for cloud resource allocation leverages their efficiency, accuracy, scalability, and ability to handle complex workloads. Rigorous evaluation ensures intelligent and adaptable resource allocation in the cloud environment. This ML-driven approach optimizes operational planning, reduces execution time, enhances VM utilization, and lowers operational costs – leading to more efficient and scalable cloud systems. The evaluation results identify the most suitable algorithm(s) for real-world cloud resource management, ensuring reliability and performance improvements in modern cloud infrastructures. Algorithm 1 details task processing in virtual machine memory.

Algorithm 1: Memory Task Processing.

Input:

```
tasks_list → List of tasks to process
virtual_memory_count → Number of virtual memory (VM) units
```

```
   time_execution → Execution time
parameter
   buffer_size = 1024 → Default buffer
3jk3
   size.
```

Output:generation collocation dataset

Step 1: //Initialize VM resources

```
   Create
VM_array[virtual_memory_count]
   Set VM.flag = 0 (0 = available, 1 =
busy)
   Set   VM.wait_time,   VM.throughput,
VM.makespan = 0
```

Step 2: // Initialize counters

```
   task_id = 0
   iteration = 0
```

Step 3: While task_id < tasks_list. length do: //Task Processing Loop

Step 3.1://Get Current Task

```
   If task_id > max_tasks then
   set current_task = 0, resource_need
= 0,
   time_id = 0
   Else,
   assign        current_task        =
tasks_list[task_id],
   calculate resource_need and time_id
```

Step 3.2:// Find and Assign Task to a VM

```
   Iterate through VM_array
   If VM is available and within task
bounds Then
   Assign current_task to VM
   Update VM metrics:
   load = resource need + (VMP / VM)
   load percent = (load / VMP) * 100
   wait_time += resource need
   makespan += time_id
   throughput = ((end - current_task) /
buffer_size) /  (makespan + time_id)
   Compute overall load using random
scaling factors
   Update historical counters for task
tracking
```

Step 3.3: //Update Wait Times & Performance Metrics:

```
   For each VM, adjust:
   wait_time -= (core * time_execution)
   makespan = max(0, makespan - (core *
(time_execution / 10000)))
   Ensure non-negative values for
wait_time, makespan
```

Step 3.4: //Compute Total Performance Metrics:

```
   total_makespan = sum(VM.makespan)
   total_throughput                    =
sum(VM.throughput)
```

Step 3-5:// Store Data in DataFrames

```
   data_wait[iteration] = VM.wait_time
   data_flag[iteration] = VM.flag
   data_load[iteration] = VM.load
   data_overall[iteration]             =
VM.overall_load
   data_makespan[iteration]            =
VM.makespan
   data_throughput[iteration]          =
VM.throughput
   data_tasks[iteration]               =
VM.current_task
```

Step 3-6: Reset Completed VMs

```
   If VM.wait_time ≤ 0, reset then
   VM.wait_time = 0
   VM.flag = 0
   VM.makespan = 0
   VM.throughput = 0
```

Step 3-7: Increment Counters:

```
   task_id += 1,      teration += 1
   End
```

5 RESULT AND EXPERIMENT

5.1 Datasets

Figure 2 presents a dataset that includes several tasks using Monte Carlo expansion. The dataset assists in deciding the suitability of a virtual machine for tasks in accordance with the characteristics of RAM, CPU, bandwidth, and processor. These characteristics serve as input to the virtual machine. The GoCJ dataset incorporates several job sizes, which can be created by duplicating formulas within an Excel file ("GoCJ_Dataset_Monte_Carlo.xlsx"). The dataset facilitates the generation of new datasets and the examination of distributed tasks, including virtual machine selection, environmental parameters, and performance metrics such as overload, makespan, throughput, wait time, and load balance.

	df_OL	Lm_p	makespan	Throughput	waite	df_task_save	Class
0	67.38162686	69.3995	0.126553672	432.129	0	127000	V_Memory0
1	67.38162686	69.3995	0.126553672	432.129	0	127000	V_Memory0
2	19.93270117	21.8585	6.971751412	4943.555	0	40000	V_Memory0
3	32.49255976	28.9623	6.957062147	2956.671	0	53000	V_Memory0
4	64.18514062	62.8421	6.88700565	587.695	45000	115000	V_Memory0
5	64.18514062	62.8421	0.11299435	587.695	0	115000	V_Memory0
6	93.79233023	63.935	6.884745763	559.226	47000	117000	V_Memory0
7	93.79233023	63.935	0.115254237	559.226	0	117000	V_Memory0
8	58.15995418	55.1918	6.902824859	824.06	31000	101000	V_Memory0
9	58.15995418	55.1918	0.097175141	824.06	0	101000	V_Memory0
10	85.32750249	57.3776	6.898305085	749.023	35000	105000	V_Memory0
11	85.32750249	57.3776	0.101694915	749.023	0	105000	V_Memory0
12	23.65245068	36.6126	6.941242938	1927.96	0	67000	V_Memory0
13	68.79390898	55.1918	6.902824859	824.06	31000	101000	V_Memory0
14	68.79390898	55.1918	0.097175141	824.06	0	101000	V_Memory0
15	58.03542264	49.7273	6.914124294	1046.207	21000	91000	V_Memory0
16	58.03542264	49.7273	0.085875706	1046.207	0	91000	V_Memory0
17	18.52379671	25.6836	6.963841808	3673.096	0	47000	V_Memory0
18	31.35114395	33.3339	6.948022599	2292.162	0	61000	V_Memory0
19	65.47290857	62.8421	6.88700565	587.695	45000	115000	V_Memory0
20	24.79879606	45.3557	0.076836158	1270.967	0	83000	V_Memory0
21	24.79879606	45.3557	0.076836158	1270.967	0	83000	V_Memory0
22	13.69257678	30.0552	6.95480226	2765.625	0	55000	V_Memory0

Figure 2: Section from our datasets in GoCJ Excel worksheet generator.

Figure 3 displays a 3D scatter plot that demonstrates the distribution of tasks in the dataset, showing the relationship between Job Size, Arrival Time, and Service Time for 100 tasks. The plot helps identify patterns, such as larger tasks requiring more time to complete. This evaluation can be used to optimize task scheduling, adjust resource allocation, and improve overall efficiency by analysing arrival times and task performance.

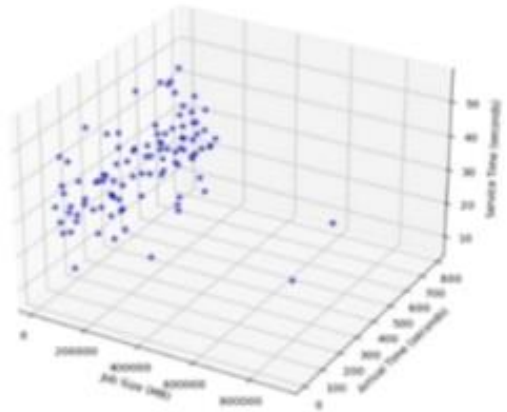
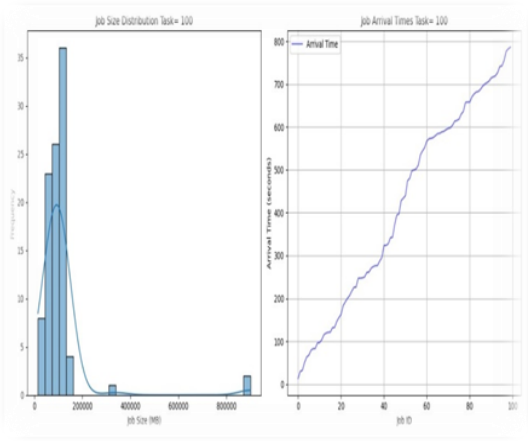


Figure 3: 3D scatter plot size, arrival times, and service times.

Figure 4 shows two plots: the left plot represents job size distribution in megabytes (MB) for 100 jobs, in which most of the jobs are distributed between 0 MB and 200,000 MB. The right plot shows the arrival times with uniform job arrival pattern and maximum arrival time of about 800 seconds. All these provide data for improving resource utilization and job scheduling in cloud computing.



(a) (b)

Figure 4: a) Left Plot: Job Size Distribution (Task = 100), b) Right Plot: Job Arrival Times (Task = 100).

5.2 Evaluation Performance

Cloud Resource Allocation is the accuracy, accurate, recall and F1 score important matrix for machine learning models. The accurate is the percentage of accurately estimated allocation between all estimated resources, while accuracy models assess purity. The Model Recall evaluates its ability to identify all necessary activities and resources. F1 score balances accuracy and the model remembers to assess performance. These provide a complete evaluation of the performance of cloud resource allocation of the Matrix model as in (1) to (4).

$$Accuracy = \frac{TP+NT}{TP+TN+FP+FN} \quad (1)$$

$$Percision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-score = \frac{2*Precision*Recall}{Precision+Recall} \quad (4)$$

5.3 Results

The Testing Phase compares the performance of various machine learning classifiers – Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), and Gradient Boosting (GB) – for cloud resource optimization based on four performance measures: precision, recall, F1-score, and accuracy, as shown in the results.

Precision computes the proportion of correctly assigned resources to the overall predicted assignments. GB and DT achieved the highest precision, indicating their ability to reduce misallocations. On the other hand, SVM achieved lower precision, indicating a higher likelihood of incorrect predictions. Recall measures the ability of the model to correctly identify all necessary resource allocations. GB and DT both performed consistently well, assigning appropriate resources to high-priority tasks, while SVM performed worst in recall, i.e., it missed some significant resource allocations.

The F1-score, being the trade-off between recall and precision, confirms that DT and GB had excellent overall performance, making them reliable for cloud resource management. SVM had a poorer F1-score, signifying its failure to balance these metrics effectively. Finally, accuracy, being the model's general rightness in its predictions, shows that GB and DT performed better than other models (see Fig. 5), making them the best classifiers for cloud environments. SVM had the worst accuracy, implying inefficiency in handling dynamic workloads.

In the summary, although there is no discussion of model verification procedures in article, for example in the setting of KEFOLD cross-valuation, overfitting, the dataset is divided into numerous parts and the model is trained and valid on several datasets. This prevents overheating and helps the model to remain in capacity. Thus, on all matrix utilized to evaluate them, GB and DT were shown to be the best artists among models to allocate sheltering resources (see Fig. 6). For responsible and competent resource management in Cloud Computing, his capacity to balance precision, recall, F1 score and accuracy makes them the best choice see Table 1. Weak SVM indicates that it lacks experience in managing the load and arranging the sliding operation effectively.

Table 1: Performance metrics for the classification algorithms.

Methods	Accuracy %	F1-score	Recall	Precision
SVM	98.5	0.98	0.98	0.98
Decision Tee	100	1.00	1.00	1.00
Logistic Regression	97.6	0.98	0.98	0.98
Gradient Boosting	100	1.00	1.00	1.00
AdaBoost	72.2	0.72	0.72	0.76

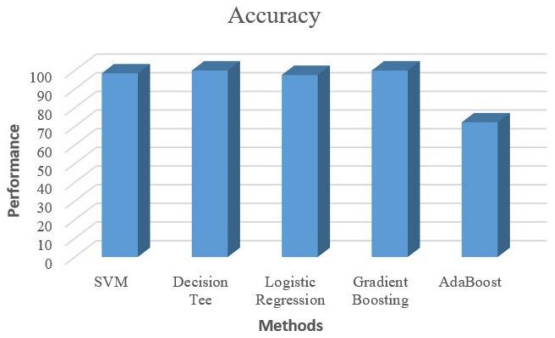


Figure 5: Classification algorithm performance accuracy metrics.

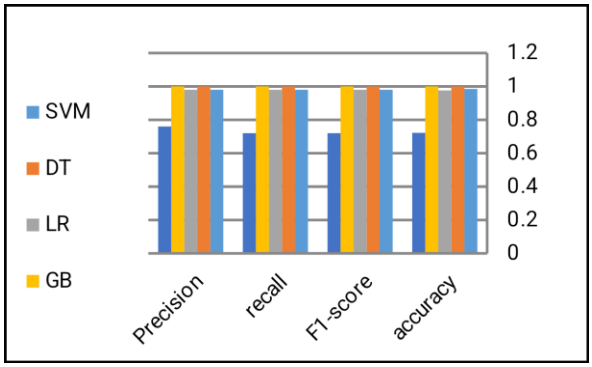


Figure 6: Classification algorithm performance metrics.

6 CONCLUSIONS

The research findings proved that machine learning models are suitable for the process of optimizing resource management in cloud computing. The results demonstrated that the Decision Tree and Gradient Boosting algorithms outperformed others, achieving 100% accuracy, precision, recall, and F1-score, indicating their exceptional reliability and suitability for real-time cloud environments. Support Vector Machine (SVM) followed closely with an accuracy of 98.5%, while Logistic Regression achieved 97.6%. However, AdaBoost lagged behind with a significantly lower accuracy of 72.2%, indicating its sensitivity to data imbalance and weaker generalization in dynamic settings.The top-performing models were decision-boosting and gradient-boosting models, while the remaining, such as Support Vector Machine and Logistic Regression, would perform well in other processes. The above findings prove that machine learning plays an important role in better resource management as it provides enhanced cost-cutting measures. Development of hybrid models that leverage the advantages offered by multiple approaches,

development of the scalability of enormous-scale cloud computing systems, and incorporation of real-time data to make dynamic resource allocation feasible are some future avenues for research that could be pursued. As long as we continue to develop energy-efficient models and embed such models into leading cloud platforms, applications of machine learning for cloud resource allocation will take us one step closer to reality.

REFERENCES

[1] J. Praveenchandar and A. Tamilarasi, "RETRACTED ARTICLE: Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 3, pp. 4147-4159, 2021.

[2] J. Bhatia et al., "HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud," in International Symposium on Cloud and Services Computing, Mangalore, KA, 2012.

[3] D. Alsadie et al., "Dynamic resource allocation for an energy efficient vm architecture for cloud computing," in Proceedings of the Australasian Computer Science Week Multiconference, 2018.

[4] M. B. Hassan, "Green machine learning approaches for cloud-based communications," in Green Machine Learning Protocols for Future Communication Networks, CRC Press, 2024, pp. 129-160.

[5] H. Shukur et al., "Cloud computing virtualization of resources allocation for distributed systems," Journal of Applied Science and Technology Trends, vol. 1, no. 2, pp. 98-105, 2020.

[6] Y.-F. Wen and C.-L. Chang, "Load balancing job assignment for cluster-based cloud computing," in 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), IEEE, 2014.

[7] Q.-H. Zhu et al., "Task scheduling for multi-cloud computing subject to security and reliability constraints," IEEE/CAA Journal of Automatica Sinica, vol. 8, no. 4, pp. 848-865, 2021.

[8] B. Mamalis and M. Perlitis, "Improved Task Scheduling for Virtual Machines in the Cloud based on the Gravitational Search Algorithm," arXiv:2311.07004, 2023, [Online]. Available: https://arxiv.org/abs/2311.07004.

[9] S. H. Anbarkhan, "Optimizing Cloud Resource Allocation with Machine Learning: Strategies for Efficient Computing," Ingénierie des Systèmes d'Information, vol. 30, no. 1, 2025.

[10] Y. Zhang et al., "Application of machine learning optimization in cloud computing resource scheduling and management," in Proceedings of the 5th International Conference on Computer Information and Big Data Applications, 2024.

[11] C. Shetty, H. Sarojadevi, and S. Prabhu, "Machine learning approach to select optimal task scheduling algorithm in cloud," Turkish Journal of Computer and Mathematics Education, vol. 12, no. 6, pp. 2565-2580, 2021.

[12] Khan et al., "Learning from Privacy Preserved Encrypted Data on Cloud Through Supervised and Unsupervised Machine Learning," in Proceedings of the International Conference on Computing, Mathematics and Engineering Technologies, Sindh, Pakistan, 2019, pp. 1-5.

[13] S. P. Swarna et al., "Load balancing of energy cloud using wind driven and firefly algorithms in internet of everything," Journal of Parallel and Distributed Computing, vol. 142, pp. 16-26, 2020, [Online]. Available: https://doi.org/10.1016/j.jpdc.2020.02.010.

[14] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach," Applied Soft Computing, vol. 93, p. 106411, 2020.

[15] B. Gomathi and K. Karthikeyan, "Task scheduling algorithm based on hybrid particle swarm optimization in cloud computing," Applied Information Technology, vol. 55, pp. 33-3, 2013.

[16] J. Kumar, A. K. Singh, and R. Buyya, "Self-directed learning-based workload forecasting model for cloud resource management," Information Sciences, vol. 543, pp. 345-366, 2021.

[17] Hussain et al., "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures," Data, vol. 3, no. 4, p. 38, 2018.

[18] A. Y. Hamed et al., "An efficient firefly algorithm for optimizing task scheduling in cloud computing systems," Information Sciences Letters, vol. 12, pp. 1637-1647, 2023, [Online]. Available: https://doi.org/10.18576/isl/120348.

[19] B. Qian et al., "Orchestrating the development lifecycle of machine learning-based IoT applications: A taxonomy and survey," ACM Computing Surveys, vol. 53, no. 4, pp. 1-47, 2020.

[20] Z. Mohamad et al., "A genetic algorithm for optimal job scheduling and load balancing in cloud computing," International Journal of Engineering & Technology, vol. 7, no. 3, pp. 290-294, 2018.

[21] Sidhu et al., "Analysis of load balancing techniques in cloud computing," International Journal of Computers & Technology, vol. 4, no. 2, pp. 737-74, 2013.

[22] Jeyaraman et al., "Optimizing Resource Allocation in Cloud Computing Using Machine Learning," European Journal of Technology, vol. 8, no. 3, pp. 12-22, 2024.

[23] Mahmoud et al., "Multiobjective task scheduling in cloud environment using decision tree algorithm," IEEE Access, vol. 10, pp. 36140-36151, 2022.

[24] Chauhan et al., "Probabilistic optimized kernel naive Bayesian cloud resource allocation system," Wireless Personal Communications, vol. 124, no. 4, pp. 2853-2872, 2022.

[25] Kirasich et al., "Random forest vs logistic regression: binary classification for heterogeneous datasets," SMU Data Science Review, vol. 1, no. 3, p. 9, 2018.

[26] Khurana et al., "A fine tune hyper parameter Gradient Boosting model for CPU utilization prediction in cloud," 2023.

[27] Al-Kateeb et al., "AdaBoost-powered cloud of things framework for low-latency, energy-efficient chronic kidney disease prediction," Transactions on Emerging Telecommunications Technologies, vol. 35, no. 6, p. e5007, 2024.