wjert, 2024, Vol. 10, Issue 5, 301-314. Original Article ISSN 2454-695X

World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 7.029

PREDICTIVE MODEL OF CPU-BURST TIME IN COMPUTATIONAL GRIDS: A COMPARATIVE STUDY OF MACHINE LEARNING APPROACHES

Sudarshan M. G.[1], Vinutha M. R.[2], P. S. Amrutha Lakshmi[3] and Nischith Gowda D. Y.[4]*

[1,3,4]Dept. of ISE, MCE, Hassan.

[2]Assistant Professor Dept. of ISE, MCE, Hassan.

Article Received on 21/03/2024 **Article Revised on 11/04/2024** **Article Accepted on 01/05/2024**

***Corresponding Author**
Nischith Gowda D. Y.
Dept. of ISE, MCE, Hassan.

ABSTRACT

Analysing different Machine Learning approaches to predict burst times in CPU. This study explores the proposal of machine learning algorithms, entailing Random Forest, Gradient Boosting, Decision Tree, and K-Nearest Neighbors (KNN), to predict CPU burst times in compute grids. The Auver Grid dataset, encompassing diverse job attributes such as Job ID, Submit Time, Wait Time, Run Time, N Procs, and Req Memory, was utilized for model training and evaluation.

After preprocessing and model training, the outcome depicts the potency of machine learning models in predicting CPU burst times. The Random Forest model exhibited superior performance, showing promising accuracy in predicting CPU burst times. Additionally, the Decision Tree model, particularly with a maximum depth of 5, also yielded competitive results, showcasing its capability for reliable predictions in compute grid environments. The analysis leads to the advancement of resource allocation and job scheduling strategies in compute grids, showcasing the benefits of machine learning techniques for analysing CPU burst times.

***Index Terms*:-** Computational Grids, Auver Grid, Random Forest, Machine Learning.

I. INTRODUCTION

In the realm of compute grids and high-performance computing, the efficient deployment of resources and job scheduling are paramount for optimizing system performance and throughput. Compute grids, composed of interconnected computing resources, handle a multitude of tasks ranging from scientific simulations to data processing. Efficient allocation of resources ensures that jobs are executed promptly while maximizing resource utilization.

One pivotal aspect of resource distribution is predicting the runtime of jobs or tasks. Accurate predictions enable scheduler algorithms to make knowledgeable decisions, leading to optimized resource utilization and reduced job wait times. Traditionally, predicting job runtime has been a challenging task due to the dynamic nature of compute grid environments and the diverse characteristics of jobs.

Machine learning (ML) has emerged as a promising approach for predicting job runtimes in compute grids. ML models can learn from bygone job attributes and performance metrics enabling them to make factual predictions for forecoming tasks. This study focuses on exploring the application of various ML algorithms to predict CPU burst times in compute grids.

The dataset used in this study, AuverGrid, comprises a rich set of job attributes such as JobID, SubmitTime, WaitTime, RunTime, NProcs, and ReqMemory. By leveraging these attributes, along with the power of ML algorithms, we aim to forge models that can precisely forecast CPU burst times. The models investigated include Random Forest, Gradient Boosting, Decision Tree, and K-Nearest Neighbors (KNN).

Through this exploration, we seek to enhance the efficiency of compute grid resource management by providing accurate and timely predictions of CPU burst times. Such predictions can significantly improve job scheduling strategies, reduce resource wastage, and ultimately enhance the overall performance of compute grid environments.

For the purpose of scrutinizing the proposed model for predicting CPU burst lengths, we utilized the "GWA-T-4 AuverGrid" grid workload dataset. AuverGrid is a production grid platform consisting of five clusters, each equipped with Dual 3GHz Pentium-IV Xeons running Scientific Linux. These clusters, totaling 475 processors, are located in the Auvergne region of France. The grid workload dataset comprises 404,176 jobs, each characterized by

29 attributes as outlined in Table I. From this extensive dataset, the study focuses on the first 40,000 jobs for the analysis. During the preprocessing stage, it is identified that attributes numbered 19 to 29 in Table I contained no information or values, denoted by the value -1. Consequently, these 11 attributes were excluded from the analysis due to the missing data. The remaining 18 attributes, from 1 to 18, were considered for this specific study, with the attribute "RunTime" representing the actual CPU burst time.

II. Literature survey

The findings of the analysis in[1] divulge that the traditional Exponential Averaging (EA) method may not consistently provide reliable burst time predictions. It is depicted that the non-linear ensemble ML technique, Random Forest outperforms other models, indicating its productiveness in estimating CPU burst times.[1] Suggests that ML-based approaches offer a positive path for boosting the efficiency of CPU scheduling algorithms by providing accurate burst time predictions, henceforth optimizing resource allocation.

The approach putforth in[2] indicates that the dual simplex method, offers a feasible solution to the snag of estimating the length of the upcoming CPU burst in SJF scheduling. The outcomes suggest that the predicted values using the dual simplex method highly correspond with the actual length of previous CPU burst requests. This implies that the dual simplex method is suitable in enhancing the correctness of predicting CPU burst times in SJF scheduling, enabling improved decision-making in resource allocation.

The outcomes of[3] suggest that the mentioned fuzzy-based algorithm, harnessing intelligent fuzzy systems, is found out to be a structured solution for estimating the next CPU-burst time of a process based on past behavior. By employing fuzzy logic, the algorithm improves the Shortest Path Next (SPN) scheduling algorithm's efficiency by providing dependable predictions of run time before processes are executed. The intelligent fuzzy systems helps in overcoming the ambiguity linked with estimating CPU burst times in SPN scheduling.

The result of[4] is that when machine learning is employed on real-time scheduling algorithms in cluster environments, it improves the accuracy of burst time predictions. Based on the predictions made the study compares First Come First Serve and Shortest Job First scheduling algorithms disclosing that SJF outruns FCFS, especially for shorter processes. Based on the reliable estimates of burst times and providing some fundamental revelations for making smart choices, the particular literature survey suggests that machine learning has

the possibilities to transform the real-time scheduling in cluster environments.

The point of view of[5] is that the ML-based approach, including Super Vector Machines (SVM), K-NN, Artificial Neural Networks (ANN), and Decision Tree, provides a powerful solution for predicting CPU burst times. Feature selection techniques contribute in refining the performance, and K-NN rises out to be an exceptional performer with respect to correlation coefficient (CC) and relative absolute error (RAE). According to this literature survey there exists a strong linear relationship between process attributes and burst CPU time, and the putforth ML-based approach depicts supremacy in relation to accuracy, estimation quality, and time efficiency.

Table 1: Dataset Attributes. This comprehensive dataset provides a detailed overview of job Attributes and Characteristics within the Auver Grid environment, enabling our study to delve into job scheduling and resource utilization patterns for CPU burst time prediction.

Attribute Name	Description
JobID	The unique identifier for space in megabytes.
Req Resources	List of comma-separated generic requested resources.
VOID	Identifier for Virtual Org.
ProjectID	Identifier for project. each job.
SubmitTime	The time of submission in seconds
WaitTime	The wait time in seconds before execution.
RunTime	The actual runtime measured in seconds.
Nprocs	The number of processors allocated for the job.
Average CPU Time	The average CPU time across all allocated processors.
Used Memory	Average used memory per processor in kilobytes.
ReqNProcs	Average used memory per processor in kilobytes.
Req Time	The requested time in seconds.
Req Memory	The requested memory per processor in kilobytes.
Status	Indicates job completion (1), failure (0), or cancellation (5).
User ID	String identifier for the user
Group ID	String identifier for the user's group.
Executable ID	Name of the executable (application).
Queue ID	String identifier for the queue.
Partition ID	String identifier for the partition
OrigSite ID	String identifier for the submission site.
Last Run Site ID	String identifier for the execution site.
JobStructure	Specifies if the job is single (UNITARY) or composite (BoT).
Job Structure Params	If composite, contains batch identifier
Used Network	Used network resources in kilobytes per second
Used Local Disk Space	The used local disk space in megabytes
Used Resources	List of comma-separated generic resources
Req Platform	The requested platform (CPU architecture, OS, OS Version).

Req Network	The requested network in kilobytes per second
Req Local Disk Space	The requested local disk

III. Proposed approach overview

To predict the CPU burst length in grid workloads, the study proposes a machine learning-based approach that utilizes various regression models. The main aim is to estimate the runtime of jobs based on their requested memory and other relevant attributes. The particular approach consists of the following steps.

A. Dataset and Preprocessing

The selected dataset is "GWA-T-4 AuverGrid" grid workload for testing the model. It contains a broad spectrum of job attributes such as SubmitTime, WaitTime, RunTime, ReqMemory, and more (refer to Table I in the dataset description). Initial preprocessing involved handling missing values, encoding categorical variables, and removing outliers using z-score thresholding. Feature engineering was performed to extract date-time features and normalize numerical attributes.

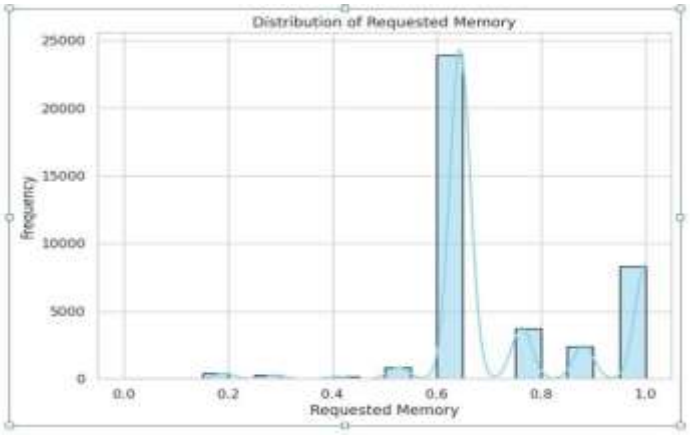

Fig. 1: Requested memory distribution.

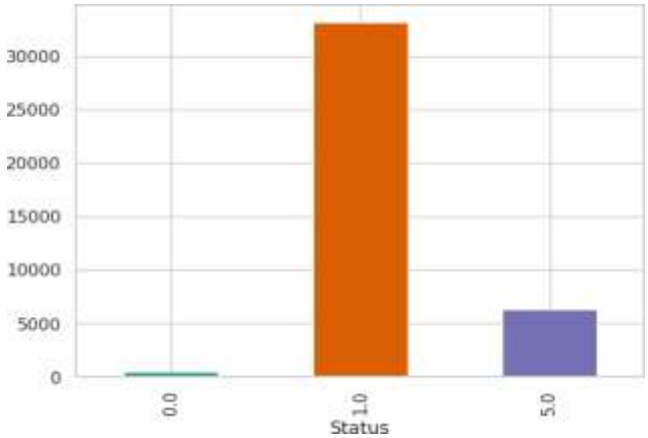

Fig. 2: Job status distribution.

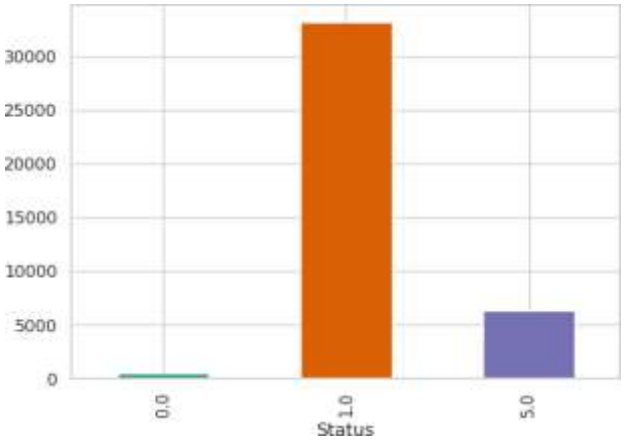

Fig. 3: Origin site distribution.

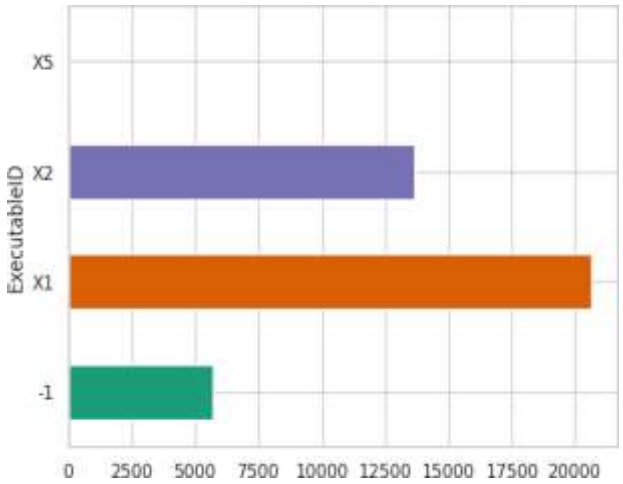

Fig. 4: Executable ID Distribution - A horizontal bar plot showing the distribution of executable IDs, representing the diversity of applications in the workload.

B. Model selection

The study choses multiple regression models to predict CPU burst lengths:

1. ***Random forest regressor*:** Capable of capturing complex relationships and handling non-linear data patterns.
2. ***Gradient boosting regressor*:** Effective for ensemble learning and improving prediction accuracy.
3. ***Decision tree regressor*:** This provides a tree-like structure for intelligible predictions.
4. ***K-Nearest Neighbors (KNN) Regressor*:** Simple and effective for predicting values based on neighboring data points.

C. Model Training and Evaluation

The model was trained on the following attributes: ['N Procs', 'Req N Procs', 'Req Time', 'Req Memory', 'Partition ID', 'Day Of Week', 'Hour Of Day', 'Month', 'Year', 'User ID', 'Group

ID', 'Executable ID', 'Queue ID', 'Orig Site ID', 'Last Run Site ID', 'Hour Bin']

The dataset was divided into training and testing sets (80:20 ratio).Each regression model was trained on the training data and assessed on the testing data. Evaluation metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) were used to assess model performance.

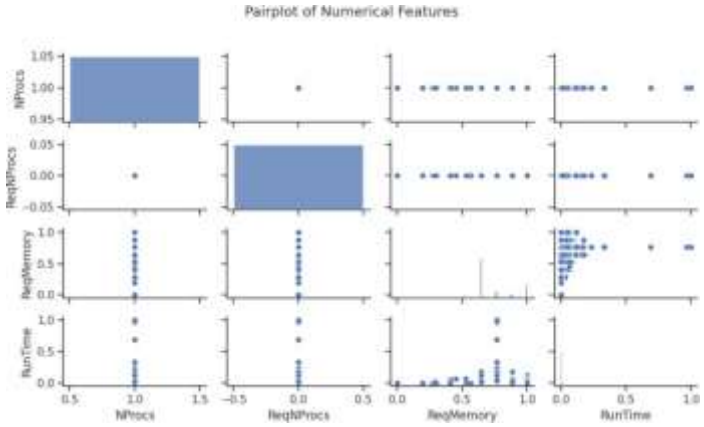

Fig. 5: Pairplot of Numerical Features-A pairplot showing the relationships between numerical features such as NProcs, ReqNProcs, ReqMemory, and RunTime. This provides understanding into potential correlations.

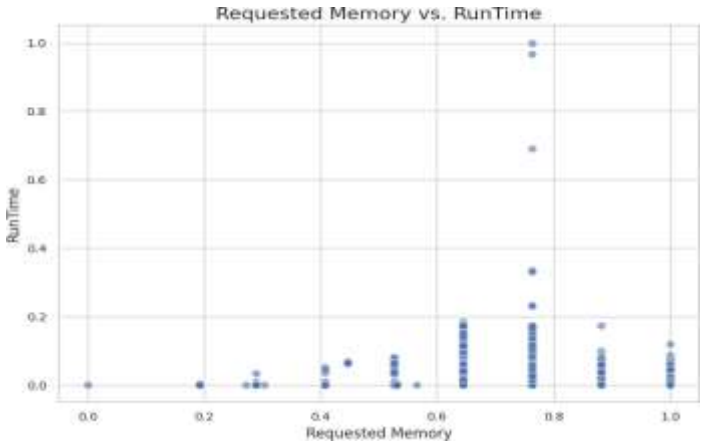

Fig. 6: Scatter Plot: Requested Memory vs. Run Time - Visualize the influence of requested memory on job runtime, aiding in understanding resource utilization patterns. It was observed that there is a strong correlation between Requested Memory and Run Time, indicating that the amount of memory requested is highly relate to the actual run time.

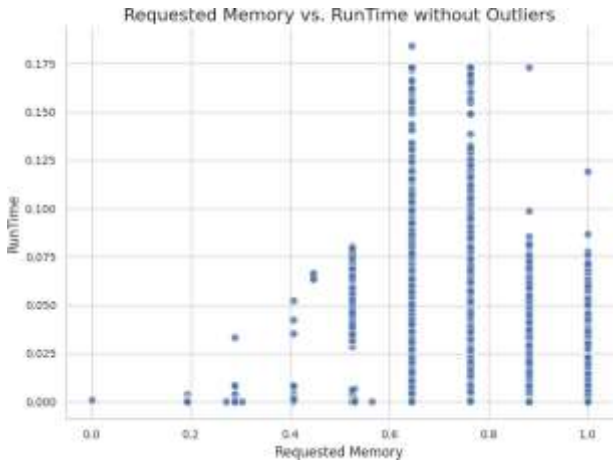

Fig. 6: Scatter Plot: Requested Memory vs. Run Time without outliers.

D. Result analysis

The performance of four regression models is evaluated: Random Forest (RF), Decision Tree (DT), K-Nearest Neighbors (KNN), and Gradient Boosting (GB).

Each model's performance was analyzed individually, followed by a relative evaluation in order to find out the most effective model for projecting CPU burst length.

Individual model performance

1. Random Forest (RF)

- Achieved an R^2 score of 0.6572, indicating a moderate fit to the data.
- Mean Squared Error (MSE): 0.00024
- Root Mean Squared Error (RMSE): 0.0155

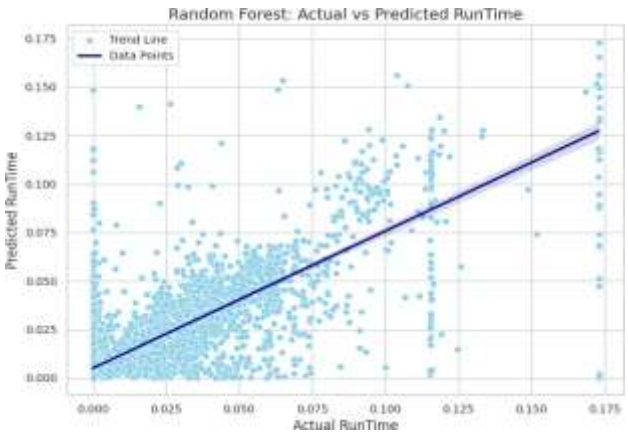

Fig. 7: Scatter plot for random forest.

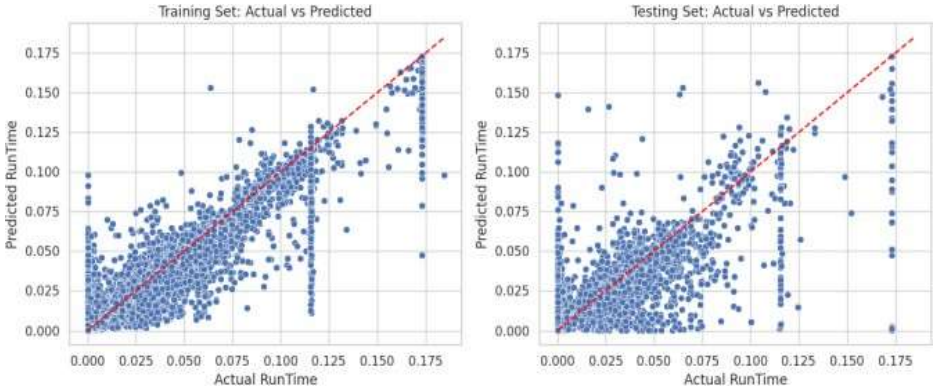

Fig. 8: This study includes testing to check if the Random Forest model was overfitting the training set, based on the results provided below, it was not the case.

Training R-squared: 0.8987458798153637

Test R-squared: 0.6572450719553862

2. Decision Tree (DT) (Max Depth 5)

- R^2 score: 0.5559, suggesting a reasonable fit to the data.
- MSE: 0.00031
- RMSE: 0.0176

Fig. 9: Scatter plot for decision tree.

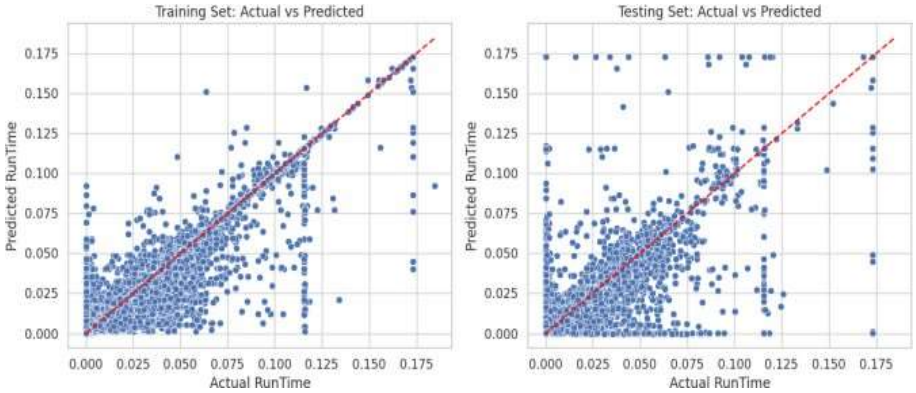

Fig. 10: The study tested if the Decision Tree model was overfitting the training set, on the basis of the results provided below, it was the case.

Training R-squared: 0.85488921290731

Test R-squared: 0.5559323141626523

3. K-Nearest Neighbors (KNN)

- R^2 score: 0.4151, indicating a weaker fit compared to RF and DT.
- MSE: 0.00041RMSE: 0.0202

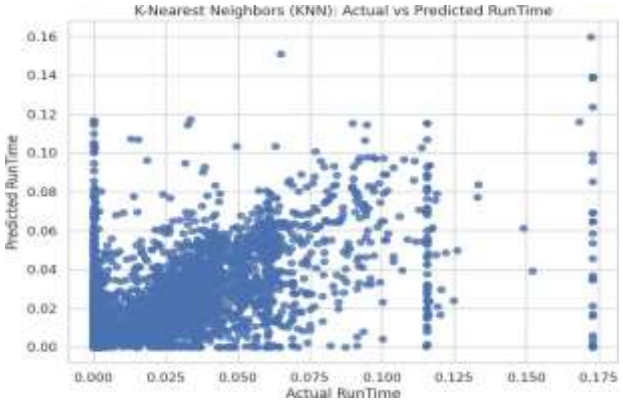

Fig. 11: Scatter plot for decision tree.

4. Gradient Boosting (GB)

- R^2 score: 0.4316, similar to KNN but slightly better.
- MSE: 0.00040
- RMSE: 0.0199

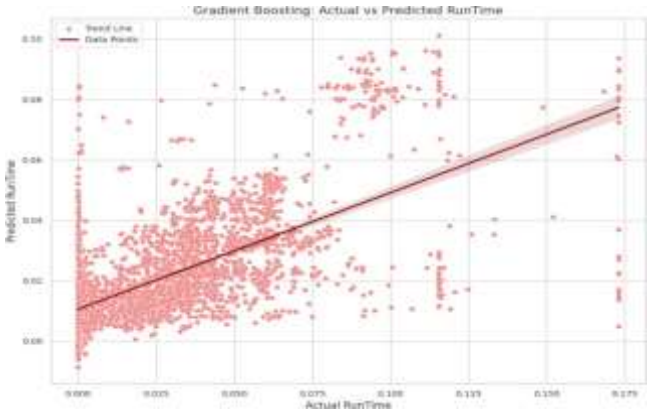

Fig. 12: Scatter plot for decision tree.

IV. RESULTS AND FINDINGS

The study aimed to predict CPU burst length using various regression models on the "GWA-T-4 AuverGrid" grid workload dataset. Below are the summarized results and findings of our analysis:

1. Key Observations

Random Forest (RF): Achieved the highest R^2 score of 0.6572, indicating a good fit to the data.

***Decision Tree (DT) (*Max Depth 5):** Followed closely with an R^2 of 0.5559, providing interpretable results.

***K-Nearest Neighbors (KNN) and Gradient Boosting (GB)*:** Showed weaker performance compared to RF and DT.

2. Comparative analysis

***Model performance*:** RF exhibited the lowest Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), indicating better predictive accuracy.

***Scatter plots*:** Visual inspection of the Actual vs.

Predicted Run Time plots revealed

- RF and DT effectively capturing underlying patterns.
- KNN and GB models showing room for improvement.

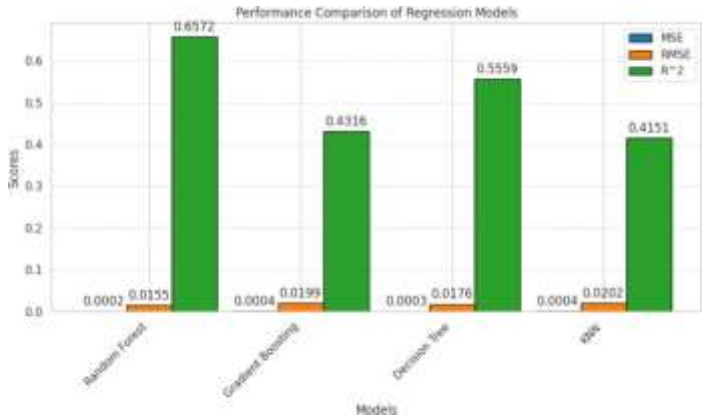

Fig. 13: Performance comparison.

V. DISCUSSION

The particular study reveals the potency of various regression models in predicting CPU burst length. Random Forest (RF) emerged as the top performer with an R^2 of 0.6572, showcasing its ability to capture complex patterns. The Decision Tree (DT) model, though slightly behind with an R^2 of 0.5559, offered valuable insights due to its interpretability.

Models like K-Nearest Neighbors (KNN) and Gradient Boosting (GB) showed less

satisfactory results, with R^2 scores of 0.4151 and 0.4316, respectively. Further fine-tuning or feature engineering may enhance their performance. The scatter plots of actual versus predicted runtimes underscored the strengths and weaknesses of each model.

This study used the "GWA-T-4 AuverGrid" grid workload, so generalization to other environments ought to be carried out cautiously. Future research could explore ensemble methods or hyperparameter tuning to improve KNN and GB models. Incorporating additional features may lead to more accurate predictions. In summary, RF and DT stand out as promising models for estimating CPU burst lengths.

VI. FUTURE WORK

While this study provides valuable insights into CPU burst length prediction using machine learning models, there are multiple pathways for further research and improvement.

1. ***Ensemble methods*:** Exploring ensemble methods such as stacking or blending could potentially improve the predictive performance further by combining the strengths of multiple models.

2. ***Feature engineering*:** Investigating additional features or extracting new features from the prevailing ones could enhance the models' ability to capture complex patterns in CPU burst lengths.

3. ***Deep learning:*** Employing deep learning techniques, like recurrent neural networks (RNNs) or long short- term memory (LSTM) networks, may offer improved performance in capturing temporal dependencies in CPU burst lengths.

4. ***Dynamic model updating*:** Implementing a system where the models are continuously renewing with new data to adapt to changing workload patterns could lead to more robust and adaptive predictions.

5. ***Optimization techniques*:** Considering optimization techniques specific to grid computing environments, such as task scheduling algorithms or resource allocation strategies, could complement the predictive models to improve overall system efficiency.

6. ***Real-Time prediction*:** Developing real-time prediction models that can forecast CPU burst lengths as new jobs are submitted to the grid could be beneficial for efficient

resource management and workload scheduling.

7. ***Benchmarking*:** Performing parallel examinations with other grid workload datasets or benchmarking against existing scheduling algorithms to evaluate the models' performance and scalability.

By concentrating on the above areas, future research can advance the field of CPU burst length prediction in grid computing, leading to more efficient resource utilization and enhanced grid system performance.

REFERENCES

1. "CPU Burst-Time Estimation using Machine Learning." IEEE, Prathamesh Samal, Sagar Jha, Raman Kumar Goyal, 2022.
2. "Prediction of length of the next CPU burst in SJF scheduling algorithm using dual simplex method". M. R. Mahesh Kumar, B. Renuka Rajendra, C. K. Niranjan, M. Sreenatha, Second International Conference on Current Trends In Engineering and Technology – ICCTET, 2014.
3. "A Fuzzy-Based Scheduling Algorithm for Prediction of Next CPU-Burst Time to Implement Shortest Process Next", Abdolghader Pourali, Amir Masoud Rahmani, International Association of Computer Science and Information Technology - Spring Conference, 2009.
4. "A Study of Real-Time Scheduling Algorithms in Cluster Environment Based on Machine Learning", Ruiyang Zhang, Zehai Liu, Gengchen Tian, Yuhao Lu, Krishamoorthy Sujatha, International Conference on Consumer Electronics and Computer Engineering (ICCECE), 2023; 3.
5. "A Machine Learning-Based Approach to Estimate the CPU- Burst Time for Processes in the Computational Grids", Tarek Helmy, Sadam AI-Azani, Omar Bin-Obaidellah, Third International Conference on Artificial Intelligence, Modelling and Simulation, 2015.
6. "A Machine Learning Approach for Performance Prediction and Scheduling Heterogeneous CPU's", Daniel Nemirovsky, Tugberk Arkose, Nikola Markovic, Mario Nemirovsky, Osman Unsal, Adrian Cristal.
7. "Applying Machine Learning Techniques to Improve Linux Process Scheduling", Atul Negi, Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, INDIA 500046, Kishore Kumar P, Member of ITDE Shaw India Software Pvt Ltd., Hyderabad, INDIA 500016.

8. "Criticality-aware dynamic task scheduling for heterogeneous architectures,"K. Chronaki, A. Rico, R. M. Badia, E. Ayguade, J. Labarta, and M. Valero, in Proceedings of the ACM on International Conference on Supercomputing. ACM, 2015; 29: 329–338.
9. "Performance Contracts: Predicting and monitoring grid application behavior", Fredrik Vraalsens in Proceedings of the International Workshop on Grid computing, November, 2001; 2.
10. "A Dynamically Adaptive CPU Scheduler", Andrew Marks, Master Thesis, Department of Computer Science, Santa Clara University, 2003; 5-9.