Machine Learning for Predictive Analysis of CPU Power and Job States on KABRÉ supercomputer

Dylan Benavides[1], Fabricio Quirós[1] & Esteban Meneses[1]

National Center for High Technology, San José 08544, Costa Rica

Abstract. This work presents the application of machine learning models to predict the CPU power consumption of jobs executed on each of the partitions of the KABRÉ computing cluster. Additionally, as an auxiliary task, a binary classification is performed to anticipate whether a job will fail or be successfully completed. To this end, data extracted from the Simple Linux Utility for Resource Management (SLURM) system is used. These data went through a process of filtering, exploratory analysis, data mining, and model training.
After performing 10 iterations using 10-fold cross-validation on the different incorporated models and calibrating the hyperparameters of the two best models per partition, it was determined that the Random Forest algorithm was the most optimal for four of the five partitions, while the remaining partition was better modeled using the K-Nearest Neighbors algorithm. These adjusted models achieved the best results based on error metrics such as the root mean square error (RMSE) and the coefficient of determination (R^2).
For the auxiliary binary classification task, Random Forest was again employed, yielding an overall accuracy of 88%, with a per-class accuracy of 88% for failed jobs and 90% for completed jobs.

Keywords: HPC · SLURM · Machine Learning · Regression

1 Introduction

High-Performance Computing (HPC) systems have become an essential tool in various fields, including industry and scientific research, enabling users to perform complex computational tasks that require significant computational resources such as memory, processing power, and storage. These tasks include simulations, large-scale data analysis, and the training of machine learning and artificial intelligence models. It should be noted that these systems have become more accessible to users, particularly those affiliated with universities and research centers [9].

Given the growing demand in recent years, optimizing the performance of HPC systems has become a key priority, as it directly affects both operational costs and infrastructure sustainability. Achieving such an optimization requires users to manage available resources efficiently; however, it is common for users to overestimate the required resources, including memory, the number of processing

units, and execution time, in order to avoid adverse outcomes such as premature termination of jobs due to insufficient resource allocation [10].

To achieve effective resource optimization, the analysis of historical data is presented as a key strategy. By accessing historical records of the jobs executed in the cluster, it becomes possible to study in detail the distribution of key variables, such as the energy consumption of each task or the execution time utilized. For example, [6] developed a methodology for anomaly detection using metrics obtained from the logs of the Simple Linux Utility for Resource Management (SLURM) scheduler associated with the Prometheus supercomputer. Similarly, [5] created a queue time prediction tool by analyzing historical SLURM logs from the HPC Anvil supercomputer.

Currently, predicting energy consumption in HPC systems is a challenge due to the variability of workload types. This variability complicates both the measurement and the regulation of energy usage, which in turn affects the efficiency and sustainability of computing centers. Furthermore, the growing demand for artificial intelligence is driving significant investments in supercomputers, leading to an exponential increase in energy consumption. [4].

This study focuses on analyzing data obtained from the SLURM logs of the KABRÉ supercomputer, located at the National Center for High Technology (CeNAT) in San José, Costa Rica. The main objective is to train machine learning models to estimate the CPU power consumption of each task executed on the cluster, as well as, complementarily, a classification model to identify tasks with a higher probability of failure or timing out. Through these models, the goal is to optimize the use of system resources and improve the operational efficiency of the cluster.

The remainder of this paper is organized as follows. Section 2 reviews related work and key concepts relevant to this study. Section 3 introduces the proposed methodology, its rationale, and the models developed. Section 4 discusses the evaluation and performance of the models, and Section 5 presents the conclusions.

2 Background and Related work

2.1 Machine Learning for Resource Estimation in HPC

The central focus of this work is the application of supervised machine learning techniques to address a predictive task in HPC systems, specifically aimed at estimating the CPU power consumption of each job executed on the KABRÉ supercomputer. This prediction allows anticipating the CPU power consumption that a job will require before execution, which is essential to improve scheduling, reduce operational costs, and promote efficient and sustainable resource allocation. Since each HPC system has its own particular characteristics, our objective is to develop a model tailored to the KABRÉ environment that contributes to optimizing the system's energy usage, minimizing resource waste, and supporting both administrators and users in strategic decision-making. [1]

The use of machine learning techniques to predict resource requirements in HPC tasks has been a central topic in numerous previous studies, as discussed in [1], [2], and [9]. These works agree that accurate resource estimation is critical for improving system efficiency, reducing operational costs, and optimizing the overall performance of computing clusters. In particular, the analysis of energy consumption, specifically CPU power consumption (`CPUPower`) has gained relevance due to increasing concerns regarding the sustainability of supercomputer usage. In this context,CPU power consumption forecasting not only improves job scheduling, but also supports the design of energy-aware management strategies tailored to the constraints and dynamics of each HPC system.

In this line, [4] emphasize that traditional methods for estimating energy consumption in HPC environments are based mainly on reports generated by power supply units. However, these approaches typically assume exclusive node usage, an assumption that becomes increasingly inadequate in modern scenarios where computational resources are shared among multiple users and tasks. To address this limitation, the authors collected detailed metrics on both CPU usage and server energy consumption, in order to identify consumption patterns under varying workload conditions. By employing a non-negative linear regression model, they estimated parameters that describe the relationship between CPU utilization and energy consumption, thereby achieving a more precise understanding of how different workload types affect energy demand. Such analyses are fundamental to the development of predictive models that are both accurate and adaptable to the dynamic and shared nature of modern HPC systems.

In [4], it is noted that few studies estimate CPU power consumption based on instructions executed by programs or workloads. In contrast, predominant approaches estimate energy consumption by considering the entire system, whether it is an individual computer or a server. In response to these recent approaches, the present work proposes a more detailed methodology, developing different models that are adjusted according to the partition of the system where each job is executed, which has specific characteristics and usage patterns, and to the variables requested by the user, such as the number of CPUs required, the memory allocated and the time limit defined, with the purpose of identifying energy consumption patterns based on the data history generated by the KABRÉ supercomputer user community.

Tanash [9] developed machine learning models to predict and determine the resources required to execute jobs on a supercomputer. In addition, he compared the effects of overestimating memory versus execution time in resource allocation, aiming to identify which type of overestimation is more detrimental to system performance and efficiency. This approach provides valuable insight into how resource allocation can be improved in HPC environments.

On the other hand, Dash, Paul, and Oral [2] analyzed data from the Titan computing cluster, including job scheduler logs, GPU failure data, and project-specific information. Their study focused on how scheduler performance varies over time and how users adjust their behavior after system failures, and the primary goal of their research was to generate valuable information that would

help better understand system behavior and optimize resource allocation, which is crucial for efficiently managing system failures and improving stability and performance.

Lastly, Lovell et al. [5] developed a model called TROUT, based on neural networks, to predict job wait times in the HPC cluster Anvil. They used Slurm data, which were transformed during the feature engineering phase, which provides information about work priorities, partitions, and states. Their study explores various prediction methods for wait times in HPC environments and concludes that deep learning models are viable solutions for such predictions.

2.2 Defining machine learning task

After applying transformations to the SLURM logs, a regression task is defined to predict the CPU power consumption of each job, along with a complementary task; determining whether a job will fail or complete, using as predictor variables those features that are obtained once a job is submitted to the system.

Predicting CPU power consumption is a problem in which the response variable is numerical and is related to estimating the amount and types of resources used in the jobs executed on the system. On the other hand, classifying the state of a job is a classification problem, where the variable of interest is categorical, with two possible values: Failed(F) and Not Failed (NF). While the task of estimating CPU power consumption is considered a regression task aimed at obtaining a numerical value, classifying the state of the job is classified as a binary classification task, where the goal is to predict whether a job has failed or not. It should be noted that although the estimation of CPU power consumption could initially be considered a discrete task, the precision required to make estimates under various scenarios turns this problem into a continuous regression task.

2.3 Machine Learning Models

The purpose of the research was to predict computing power (`CPUPower`) and classify job status according to the following parameters: `TimelimitRAW`, `ReqNodes`, `ReqMem`, `ReqCPUS`, `Priority`, `ResvCPURAW`, `Submit`, and `QOS`. Therefore, we considered the following popular models for regression and classification tasks.

For the regression task, we used the following well-known models: Linear Model (LM), Lasso Regression (LASSO), Ridge Regression (RIDGE), Decision Trees (DT), Random Forest (RF), K-Nearest Neighbors (KNN), Neural Networks (NNET), Extreme Gradient Boosting (XGBoost), Naive Bayes (NB) and Support Vector Machine (SVM). To evaluate the models, we mainly used the coefficient of determination (R^2) and the root mean squared error (RMSE).

For the classification task, the following models were used: Generalized Linear Model (GLM), Generalized Linear Model with Regularization Networks (GLMNET), Decision Trees (DT), Random Forest (RF), K-Nearest Neighbors (KNN),

Neural Networks (NNET), Boosting (Boost), Extreme Gradient Boosting (XGBoost), Naive Bayes (NB), and Support Vector Machine (SVM). The model evaluation was conducted using the confusion matrix and ROC curves.

3 Methodology

This section describes the acquisition and preprocessing of the data set obtained from the SLURM logs, as well as the selection, calibration, and parameter tuning of the machine learning models used.

3.1 Data Preparation and Feature Analysis

The set of observations and variables was obtained from the sacct file, which was extracted from the SLURM logs of the KABRÉ supercomputer at the National Center for High Technology (CeNAT). It includes logs of jobs executed between July 2024 and January 2025, comprising approximately 30376 observations.

An initial selection of thirteen variables deemed relevant to the research was performed. These features are described in Table 1. The variables Partition and State were used as filters, while the remaining ones were used for the mathematical modeling process.

Feature	**Type**	**Description**
ConsumedEnergyRaw	int	Total energy consumed by all tasks in job, in Joules.
CPUTimeRAW	int	Time used (Elapsed time * CPU count) by a job or step in cpu-seconds.
ResvCPURAW	int	How many CPU seconds were used as reserved time for this job.
TimelimitRaw	chr	What the timelimit was/is for the job (in minutes).
ReqMem	chr	Minimum required memory for the job (in MB per CPU or MB per node).
ReqNodes	int	Requested minimum number of nodes.
ReqCPUS	int	Number of requested CPUs.
QOS	chr	Name of Quality of Service.
Partition	chr	Identifies the partition on which the job ran.
Priority	int	Slurm priority.
State	chr	The job status.
Submit	chr	The time the job was submitted.

Table 1. Feature Selection

The variables selected for the predictive model were processed using the software **R** in three stages: data filtering, exploratory data analysis, and modeling.

Data Filtering: The KABRÉ supercomputer includes five partitions, each with different characteristics. Initially, a single dataset was used that combined

job records from all partitions, but then it was decided to analyze each partition separately, so the original dataset was divided into five subsets associated with each partition: Andalan, Dribe, Nu, Nukwa, and Kura, containing 13243, 6641, 5178, 2524 and 2790 jobs, respectively. This decision was made because each partition provides different resources distributed in specific subpartitions, so the distribution of variables may vary significantly from one partition to another.

Exploratory Data Analysis and Data Mining: As a general process for all partitions, the null values were removed. The highest number of null entries was in the Kura partition, with 857 entries. Note that all null records corresponded mostly to canceled or timeout jobs. Records with zero elapsed time were also removed, as they represent observations that do not contribute to the study of CPU power consumption; therefore, 26, 309, 197, 276, and 150 jobs were removed from the respective partitions.

Subsequently, a new variable, `CPUPower` with the unit of measurement in Watts (W), was created from the variables `ConsumedEnergyRaw`, which is measured in Joules (J) and `CPUTimeRAW` which is measured in seconds (s). The `TimelimitRAW` variable was converted to seconds, the `Submit` variable was converted to a date format and then to a numeric value, and `ReqMem` values were converted to bytes, as they were originally recorded in megabytes, gigabytes per node or per CPU (Mc, Mn, Gc, Gn).

The main transformations applied to each dataset/partition are described below.

1. **Andalan Partition.**
   In this partition, the variables `ReqMem`, `ReqNodes`, and `QOS` were discarded, as all jobs were executed on a single node. 99% of the jobs used 2000 Mn of memory, and 99% had a normal quality of service (`QOS`). Additionally, the jobs were distributed as follows: 13 canceled, 1070 completed, and 12033 failed, with the latter accounting for 91% of the total.
   Outliers were identified in the `CPUPower` response variable (about 12%) based on Tukey's rule, especially associated with completed and failed jobs. To mitigate the influence of these extreme values, a 10% winsorization was applied, replacing values below the 10th percentile with the respective percentile value, and similarly for the upper tail, replacing values above the 90th percentile with the corresponding cut-off value. Similarly, a 5% winsorization was applied to the `ResvCPURAW`, `TimelimitRAW` and `Priority` variables.
   Additionally, 3% of the outlier data in the Submit variable were removed using Tukey's rule. Following this, a Z-score normalization was applied to the numerical variables, ensuring that all variables had a mean of zero and a standard deviation of one. Finally, the Pearson correlation matrix revealed strong correlations between the `ReqCPUS`, `TimelimitRAW` and `Priority` variables with the `CPUPower` response variable, while the `ResvCPURAW` and `Submit` variables showed weak correlations with `CPUPower`, as shown in Figure 1.

2. **Dribe Partition.**

   In this partition, the `ReqNodes` variable was removed, as all jobs except one were executed on a single node. The jobs were distributed as follows: 14 canceled, 3656 completed, 2179 failed, and 15 timed out, with 62% of the jobs being completed.
   Outliers were identified in the `CPUPower` response variable, as well as in the `ResvCPURAW`, `ReqMem`, `ReqCPUS`, `TimelimitRAW`, `Priority`, and `Submit` variables. To mitigate the influence of these extreme values, winsorization at the 5% level was applied.
   Additionally, 13% of the outlier data in the `TimelimitRAW` variable were removed using Tukey's rule. Following this, a Z-score normalization was applied to the numerical variables and the factor variable `QOS` was included as a predictor. Finally, the Pearson correlation matrix showed a strong correlation between `ReqCPUS`, `ResvCPURAW`, and `CPUPower`, and weak correlations between `ReqMem`, `Submit`, `TimelimitRAW`, and `Priority` with `CPUPower`, as shown in Figure 1. A square root transformation was also applied to the response variable.

3. **Nu Partition.**

   In this partition, the `ReqMem` variable was removed, as it was found to have no correlation with the `CPUPower` variable. The jobs were distributed as follows: 88 canceled, 1097 completed, 756 failed, and 27 timed out, with 55% of the jobs being completed.
   Outliers were identified in the CPUPower response variable, as well as in the `ResvCPURAW`, `ReqCPUS`, `TimelimitRAW`, `Priority`, and `Submit` variables. To reduce the influence of these extreme values, 5% winsorization was applied to all these variables.
   Subsequently, a Z-score normalization was applied to the numerical variables and the factor variable `QOS` was included as a predictor. Finally, the Pearson correlation matrix showed a moderate correlation between `ReqCPUS` and `CPUPower` and weak correlations between `Submit`, `TimelimitRAW`, `ResvCPURAW`, `ReqNodes`, and `Priority` with `CPUPower`, as shown in Figure 1. Additionally, a logarithmic transformation was applied to the `ResvCPURAW` variable, and a square root transformation was applied to the `TimelimitRAW` variable.

4. **Nukwa Partition.**
   In this partition, the ReqCPUS variable was removed, as all jobs used 24. The jobs were distributed as follows: 22 canceled, 215 completed, 148 failed, and 28 timed out, with 52% of the jobs being completed.
   Outliers were identified in the `CPUPower` response variable, as well as in the `ResvCPURAW`, `TimelimitRAW`, `Priority`, and `Submit` variables. To mitigate the influence of these extreme values, 5% winsorization was applied. Additionally, a square root transformation was applied to the `Priority` variable to reduce skewness.

Furthermore, 12% of the data considered outliers in the `Priority` variable were removed using the Tukey rule. Subsequently, Z-score normalization was applied to the numerical variables, and the factor variable `QOS` was included as a predictor. Finally, the Pearson correlation matrix showed low correlations between all predictor variables and the `CPUPower` variable, as shown in Figure. Additionally, a logarithmic transformation was applied to the `ResvCPURAW` variable and a square root transformation to the `TimelimitRAW` variable.

5. **Kurá Partition.**

   In this partition, no variables were removed. The jobs were distributed as follows: 87 canceled, 917 completed, 770 failed, and 9 timed out, with 51% of the jobs being completed.

   To mitigate the influence of extreme values in the `CPUPower`, `ReqCPUS`, `ResvCPURAW`, `TimelimitRAW`, `Priority` and `Submit` variables, 5% winsorization was applied.

   Subsequently, Z-score normalization was applied to the numerical variables and the factor variable `QOS` was included as a predictor. Finally, the Pearson correlation matrix showed strong correlations between `ReqCPUS`, `ReqMem`, `ResvCPURAW`, `Submit` with `CPUPower`, and weak correlations between `ReqNodes`, `TimelimitRAW`, `Priority` with `CPUPower`, as shown in Figure 1. Additionally, a square root transformation was applied to the response variable.

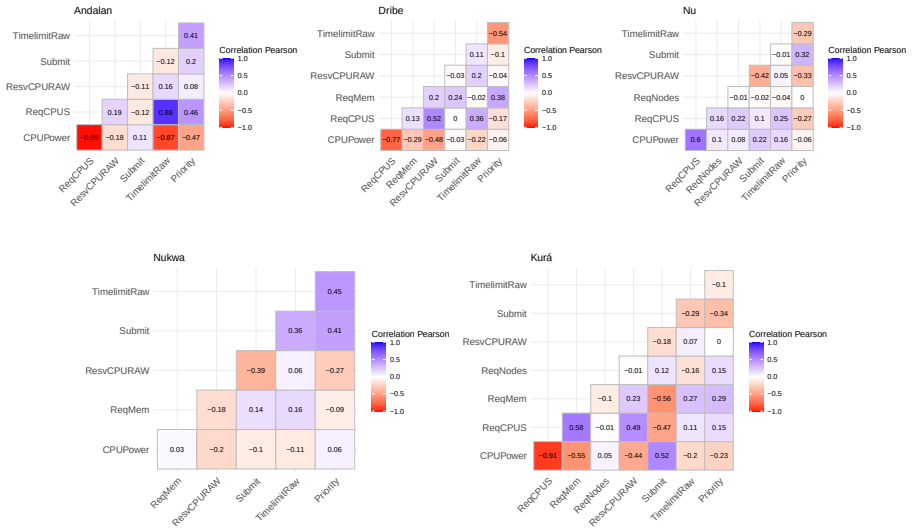

Fig. 1. Correlation matrix associated with each partition

Modeling: To evaluate the performance of various regression models, a 10 times repeated cross-validation scheme was applied. In each repetition, the data set was randomly partitioned into 10 subsets of equal size; one of these was used as the test set and the remaining 9 were used as the training set. This process was repeated for each subset, and the prediction error for each model was accumulated.

Subsequently, different metrics were studied to select the three most robust models for each partition. These metrics are as follows: Root Mean Square Error (RMSE), Residual Standard Error (RSE), Mean Absolute Error (MAE), Relative Error (RE) and Coefficient of Determination and the three best models were selected, mainly in terms of goodness of fit and RMSE. Then, some hyperparameters of each model were calibrated to reduce the mean squared error, using a training-testing methodology with 5 repetitions, keeping the same split of 80 training and 20 testing. Finally, the metrics of the best resulting model for each partition were analyzed and the assumptions of the model were checked to ensure its reliability.

4 Results and discussion

As mentioned previously, the cross-validation technique with 10 partitions (folds) and 5 repetitions was used to train and evaluate a total of ten regression models on each partition of the KABRÉ dataset, and later the performance of the models was compared using various evaluation metrics. Figure 2 shows the average Root Mean Square Error (RMSE) obtained by each initial model, from which it can be observed that for the Andalan, Dribe, Nu, and Kurá datasets, the best-performing models were Random Forest and K-Nearest Neighbors, while for the Nukwa dataset, Random Forest and Decision Tree were the most effective.

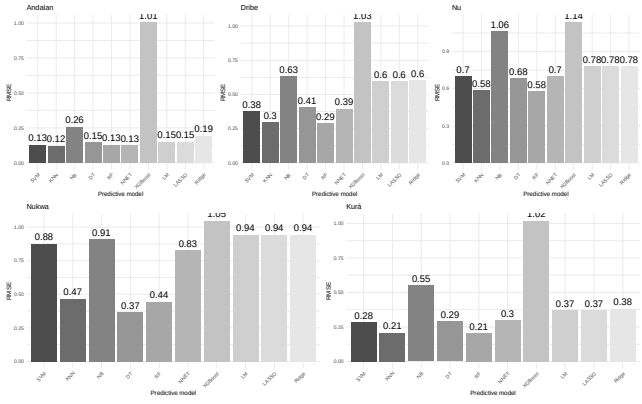

Fig. 2. Average RMSE per model across partitions

Next, a model calibration stage was performed using a training-testing approach, applied in five random data partitions. In each partition, 80% of the data was used for training and the remaining 20% for testing. During this stage, the best hyperparameters were selected for the evaluated models; for K-Nearest Neighbors (KNN), the best `kernel` was determined for all partitions except Nukwa, for Random Forest, the parameters `mtry` and `ntree` were optimized and for Decision Tree, for the Nukwa dataset, the values of `cp` and `maxdepth` were adjusted. The optimal models were selected on the basis of multiple performance metrics. Figures 3, 4, and 5 present the RMSE values obtained in each iteration, comparing the different models generated from the hyperparameter combinations evaluated for Random Forest, Decision Tree, and the various KNN kernels.

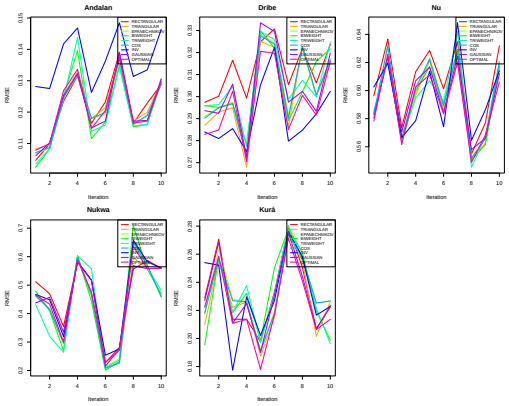

Fig. 3. Average RMSE by KNN kernel for each partition

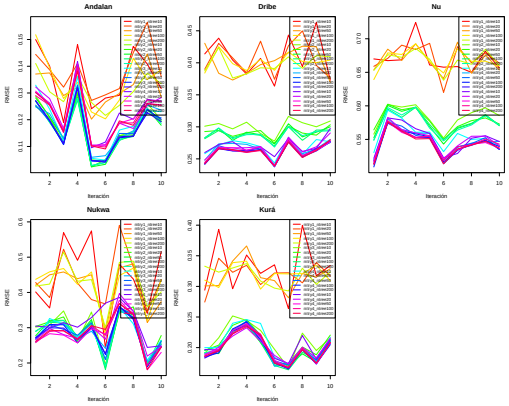

Fig. 4. Average RMSE by RF parameter combination for each partition

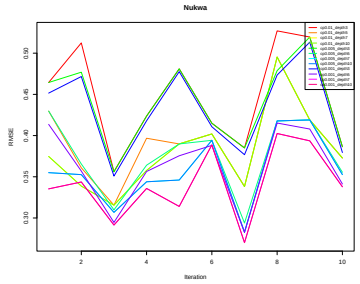

Fig. 5. Average RMSE by DT parameter combination

Finally, the calibration results indicated that for the KNN model, the best kernel was *optimal* for the Andalan and Nu datasets, *inv* for Dribe, and *epanechnikov* for Kurá. For the Random Forest model, the optimal value for the number of trees was 100 in all partitions except Andalan, where the best performance was achieved with 10 trees. Regarding the number of predictors selected in each partition, the optimal values were 2 for Andalan and Nukwa, 3 for Kurá, and 4 for Dribe and Nu. Lastly, for the Decision Tree model applied to Nukwa, the best results were achieved with a complexity parameter of 0.001 and a maximum tree depth of 10 levels.

Thereafter, the performance metrics were compared again by partition, considering the preselected models and their respective optimal configurations, to determine a final model for each dataset. As a result, for the Andalan dataset, the selected model was KNN with the *optimal* kernel. For the other datasets (Dribe, Nu, Nukwa and Kurá), the model with the best performance was RF. In particular, the following optimal configurations were identified: for Dribe and Nu, 4 predictors and 100 trees; for Nukwa, 2 predictors and 100 trees; and for Kurá, 3 predictors and 100 trees.

The metrics for each model are presented in Table 2. The highest coefficient of determination (R^2) was obtained for the Andalan partition model, with 0.98, followed by Dribe with 0.91, Kurá with 0.90, Nukwa with 0.89 and Nu with 0.69. Thus, in four partitions, the percentage of variance explained by the model exceeds 89%, except for Nu. Regarding the Root Mean Square Error (RMSE), small errors were obtained in all partitions except Nu, where Andalan presented the lowest RMSE of 0.12. The same pattern is observed for the Residual Standard Error (RSE).

The Mean Absolute Error (MAE) shows that, on average, the Andalan model errors by 0.07 units from the true value, similar to Nukwa. Kurá shows an average error of 0.08, while Dribe has an MAE of 0.17, and Nu an MAE of 0.38, which is considered moderate. In terms of Relative Error (RE), all models have values less than 0.19, except for Nu, where a value of 0.52 is obtained. Finally, all models show a good correlation (COR) between the actual and predicted values.

Model	R^2	RMSE	RSE	MAE	RE	COR
Andalan	0.98	0.123	0.123	0.079	0.128	0.992
Dribe	0.91	0.287	0.288	0.177	0.193	0.958
Nu	0.69	0.524	0.527	0.386	0.526	0.845
Nukwa	0.89	0.206	0.208	0.078	0.174	0.976
Kurá	0.90	0.183	0.186	0.089	0.097	0.982

Table 2. Performance metrics for each model

Figure 6 illustrates the comparison between the actual values and the predictions for each model, confirming that the best models correspond to Andalan, Dribe, and Nukwa. Kurá is presented as an acceptable model, while the Nu model does not adequately fit the data.

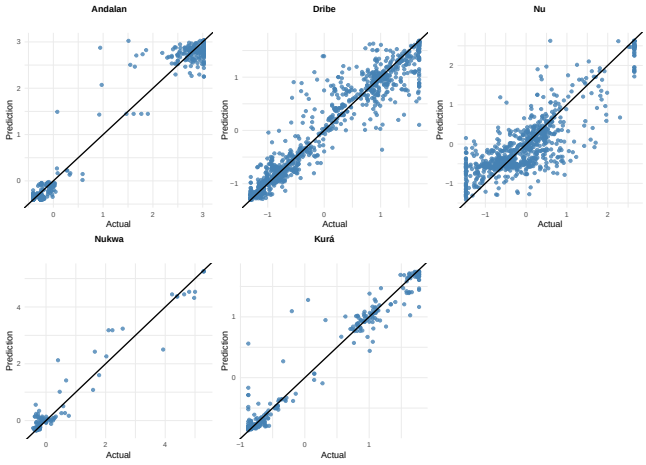

Fig. 6. Scatter Plot of Real vs Predicted Values

Classification model.

Complementarily, a binary classification model was developed with the goal of detecting whether a job will end in a failed state (considering timeout cases as failures) or will complete successfully. For this model, canceled jobs were excluded as they cannot be considered complete or failed.

From the `Submit` variable, two new variables were generated: `SubmitHour` and `SubmitWeekday`. Then, the records with null values were removed and a 95% winsorization was applied to the variables `ResvCPURAW`, `Priority`, `ReqMem`, `ReqCPUS`, and `TimelimitRAW`, in order to reduce the effect of extreme outliers.

Subsequently, a Student's t-test was applied to each of the variables to compare their means between the two groups: Failed (Fa) and Non-Failed (NF), and to determine if there were statistically significant differences. The results of this test for each variable are presented in Table 4.

Table 3. Comparison of variables by group (Fa vs. NF)

Variable	**p-value**	**Fa Mean**	**NF Mean**
ReqCPUS	2.2e-16	18.53	7.58
ReqMem	2.2e-16	7.81e+09	1.83e+10
ReqNodes	0.002	1.02	1.01
ResvCPURAW	2.2e-16	37.69	78.86
TimelimitRAW	2.2e-16	234349.6	161490.3
Priority	7.6e-12	5605.73	5935.31
SubmitHour	2.2e-16	12.23	13.52
SubmitWeek	0.0016	3.89	3.82

The results indicate that there is statistically significant evidence of differences between failed and non failed jobs in all the analyzed variables. In Figure 7, the distribution of variables by category can be observed. In particular, jobs that end in a failed state tend to exhibit certain distinctive characteristics compared to those that are successfully completed: on average, they request a higher number of CPUs, a lower amount of memory, are assigned a lower priority, and have a longer time limit. The combination of these configurations may be associated with a higher risk of failure in the execution environment considered.

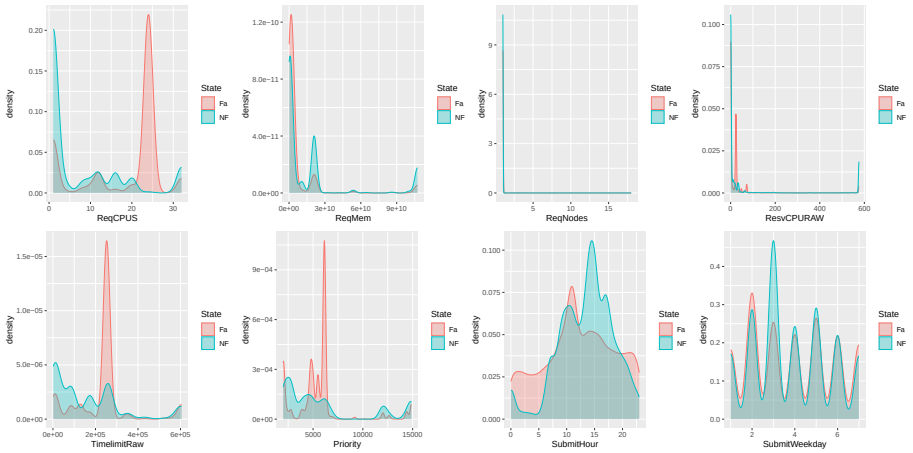

Fig. 7. Distribution of variables by State

Furthermore, the Chi-square test applied to the `QOS` variable revealed a statistically significant dependency between the status of the job and the quality of service, with a moderate strength of association $V = 0.207$.

After applying a 5-iteration training testing procedure with 85% of the training data, the best models in terms of overall precision were Random Forest, with an accuracy of 89.8%, followed by Extreme Gradient Boosting at 89%, and K-Nearest Neighbors (KNN) at 88.5%. For detecting failed jobs, Random Forest identified 91%, KNN 91.8% and Extreme Gradient Boosting 90.5%. For classifying completed jobs, the best models were Random Forest at 87%, Extreme Gradient Boosting at 86% and KNN at 81%. Considering the model with the best accuracy in each category, Random Forest was selected as the final model.

Subsequently, the cutoff probability for the Random Forest model was calibrated, selecting a probability value of 0.4. With this adjustment, the best results were achieved in both overall accuracy and per-category accuracy. Additionally, the ROC curve (false positives versus true positives) was generated for the model, resulting in an area under the curve (AUC) of 0.96. This indicates that the model has a 96% probability of assigning a higher failure probability to a job that actually failed compared to one that did not, which reflects a very high capacity to distinguish between failed and non-failed jobs.

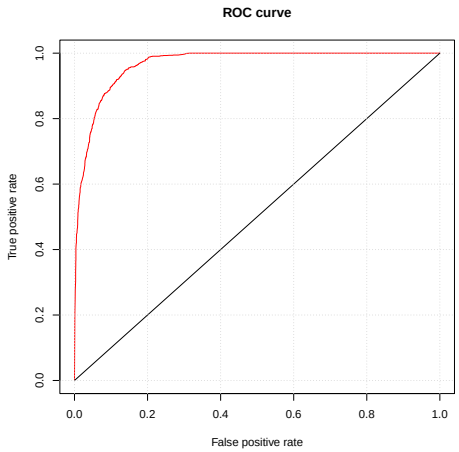

Fig. 8. ROC curve of RF model

5 Conclusions

Of the various Machine Learning models evaluated, Random Forest was the one that best adapted to the data from the computing cluster, excelling in both prediction and classification tasks. Other models, such as K-Nearest Neighbors, also produced positive results. It can be concluded that, for this research, non-parametric models were the most effective at explaining the variability of the data.

The recorded variables exhibit different behaviors and distributions in each partition, which justifies the decision to build an independent model for each. Although this strategy requires more computing time, it is more useful for obtaining more reliable and accurate information. Furthermore, hyperparameter calibration played a crucial role, as it significantly improved the error metrics in the various models evaluated, thus optimizing their performance.

The variables selected for the classification problem exhibit strong discriminative power, as evidenced by the results of the Student's t-test. This is also reflected in the performance of the model, which achieves an AUC value of 0.96. Therefore, it is possible to accurately predict whether a job will result in a failed or non-failed state based on the submission characteristics defined by the user.

For future work, it is recommended to develop a model that jointly analyzes the energy consumption of both CPUs and GPUs. In addition, updating the database is essential to improve the robustness of the models, allowing them to better capture data variability and detect relevant behavioral patterns.

Additionally, it would be beneficial to incorporate a time series analysis of CPUPower over time to identify key components such as seasonality and trend. This would allow the predictive models to be adjusted according to different times of the year or user participation patterns.

References

1. Andresen, D., Hsu, W., Yang, H., & Okanlawon, A. (2018). Machine Learning for Predictive Analytics of Compute Cluster Jobs. http://arxiv.org/abs/1806.01116
2. Dash, S., Paul, A. K., Oral, S., & Wang, F. (2021). SMC Data Challenge 2021: Analyzing Resource Utilization and User Behavior on Titan Supercomputer. https://doi.ccs.ornl.gov/ui/doi/334.
3. León, F., & Diaz, G. (2019). Development of a web interface for submitting jobs to SLURM. Revista UIS Ingenierías, 18(4), 95–98. https://doi.org/10.18273/revuin.v18n4-2019008
4. León-Vega, L. G., Tosato, N., & Cozzini, S. (2024). EfiMon: A Process Analyser for Granular Power Consumption Prediction. http://arxiv.org/abs/2409.17368
5. Lovell, A., Wisniewski, P., Rodenbeck, S., & Ashish. (2024). A Hierarchical Deep Learning Approach for Predicting Job Queue Times in HPC Systems. Proceedings of SC 2024-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis, 621–628. https://doi.org/10.1109/SCW63240.2024.00086
6. McSpadden, D., Alanazi, Y., Hess, B., Hild, L., Jones, M., Lub, Y., Mohammed, A., Moore, W., Ren, J., Schram, M., & Smirni, E. (2023). Dataset for Investigating Anomalies in Compute Clusters. https://doi.org/10.5281/zenodo.10058230
7. Morán, M., Balladini, J., Rexachs, D., & Rucci, E. (2020, December 1). Towards management of energy consumption in HPC systems with fault tolerance. 2020 IEEE Congreso Bienal de Argentina, ARGENCON 2020 - 2020 IEEE Biennial Congress of Argentina, ARGENCON 2020. https://doi.org/10.1109/ARGENCON49523.2020.9505498
8. Tanash, M., Andresen, D., & Hsu, W. (2021). AMPRO-HPCC: A Machine-Learning Tool for Predicting Resources on Slurm HPC Clusters HHS Public Access. In ADVCOMP Int Conf Adv Eng Comput Appl Sci.
9. Tanash, M., Dunn, B., Andresen, D., Hsu, W., Yang, H., & Okanlawon, A. (2019, July 28). Improving HPC system performance by predicting job resources via supervised machine learning. ACM International Conference Proceeding Series. https://doi.org/10.1145/3332186.3333041
10. Tanash, M., Yang, H., Andresen, D., & Hsu, W. (2021, July 17). Ensemble Prediction of Job Resources to Improve System Performance for Slurm-Based HPC Systems. ACM International Conference Proceeding Series. https://doi.org/10.1145/3437359.3465574
11. Taghavi, T., Lupetini, M., & Kretchmer, Y. (2016). Compute job memory recommender system using machine learning. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 609–616. https://doi.org/10.1145/2939672.2939717
12. Wilcox, R. R. (2020). Applying contemporary statistical techniques.
13. Xu, L., Hong, Y., Morris, M. D., & Cameron, K. W. (2022). Prediction for Distributional Outcomes in High-Performance Computing I/O Variability. http://arxiv.org/abs/2205.09879