RUNTIME AND DURATION CLASS PREDICTION IN HPC SYSTEMS

Davide Leone

1. INTRODUCTION

This report presents the results of a series of experiments on runtime and duration class prediction in High-Performance Computing systems. The study constitutes the first phase of a broader project aimed at integrating machine learning (ML) techniques into HPC workload dispatching.

In this initial phase, we focus on replacing the traditional static, user-provided runtime estimates, which are often inaccurate and unreliable, with machine learning–based predictions (regression and classification). The objective is to improve job prioritization and scheduling efficiency by providing more accurate and data-driven runtime estimates.

1.1. DATASET

The dataset used in our experiments is derived from the PM100 dataset, relative to the Marconi100 supercomputer at CINECA, which contains 222,520 jobs.

However, the original PM100 dataset did not include all the features required, nor did it provide duration class labels to support classification. To address this, a custom `DataLoader` class was implemented, which was responsible for:

- Enriching the dataset with the additional features required (e.g., job-specific, historical, and system-state information).
- Defining and assigning duration classes based on expert knowledge and job runtimes, with two levels of granularity:
  - Four classes: *Very-Short, Short, Medium, Long*
  - Seven classes: *Very-Short, Short, Medium-Short, Medium, Medium-Long, Long, Very-Long*
- Dividing chronologically the dataset into training (earliest jobs – 70%) and testing (more recent jobs – 30%) subsets, to better reflect the real-world scheduling scenarios.
- Saving the resulting data into new, ready-to-use `Parquet` files for subsequent experiments.

1.2. SETS OF FEATURES

We considered two different set of features:

- SET 1: minimal set of features directly available when a user submit his job.
- SET 2: larger and task-specific set of features, which is a combination of features taken from different sources such as the description of the job, historical information, the current state of the system, and the environment.

1.3. PREDICTION METHODS

The main machine learning employed for the prediction task are:

- Decision Tree Regression (scikit-learn implementation)
- Normalized Polynomial Regression (custom implementation)
- k-Nearest Neighbors (k-NN) Classification (scikit-learn implementation)

To benchmark the effectiveness of machine learning methods, we also considered two baselines:

- The user estimation.
- A simple heuristic, based on the historical average of a given user's previous jobs.

1.4. METRICS

To compare the various runtime prediction methods we considered as main metric the average prediction error, but also the number of unique predictions, since higher heterogeneity in predictions allows better priority assignment.

To compare the various duration class prediction methods we considered standard metrics for classification tasks such as accuracy, macro F1-score and micro F1-score.

2. RESULTS

In this section we summarize the results of our experiments.

2.1. REGRESSION

Method	Feature Set	Average Prediction Error	Unique Predictions
User Estimation	–	15:40:16	207
Heuristic	–	01:04:04	11841
Decision Tree	SET 1	05:27:08	948
	SET 2	01:17:05	5695
Normalized Polynomial	SET 1	02:33:05	1273
	SET 2	00:47:57	13066

The table above shows that user-provided runtime estimates are highly inaccurate, with an average error of nearly 16 hours. This poor performance is largely due to the overuse of the maximum value (24 hours), chosen in more than half of the cases.

The simple heuristic based on users' historical runtimes provides a clear improvement, reducing both the prediction error and increasing the diversity of predicted values.

SET 1 is unable to provide sufficient information to guide the regression and produces worse results than the heuristic in terms of both average prediction error and number of unique predictions. For example, using Decision Tree the error is very high and the number of unique predictions very low. This indicates that the regressor fails to extract meaningful patterns from data, resulting in predictions that fall within a limited subset of all the possibilities. As a consequence, the regressor predicts the same value an extremely high percentages of the time

SET 2, which integrates richer job, historical, and system information, greatly enhances performance. In particular, in combination with Normalized Polynomial it produced the best results overall, further reducing the average prediction error and increasing the diversity in prediction values compared to the heuristic. SET 2 also helped Decision Tree, reducing the average prediction error by more than four times and increasing the number of unique predictions by almost

six times. However, despite these improvements, the numerical results obtained using Decision Tree are still worse than those obtained with the simple heuristic.

However, the two metrics considered until now are not able to show the full picture. To improve our understanding of the effectiveness of the prediction methods, we also produced six histograms which show the comparison between the Actual Runtime and the Predicted Runtime using the various prediction methods: user estimation, heuristic, and machine learning based.

Fig. 1 shows how the estimation fails completely to follow the distribution of the actual runtime of jobs, with several peaks in correspondence of the most common values of Requested Runtime, such as, notably, at 24 hours.

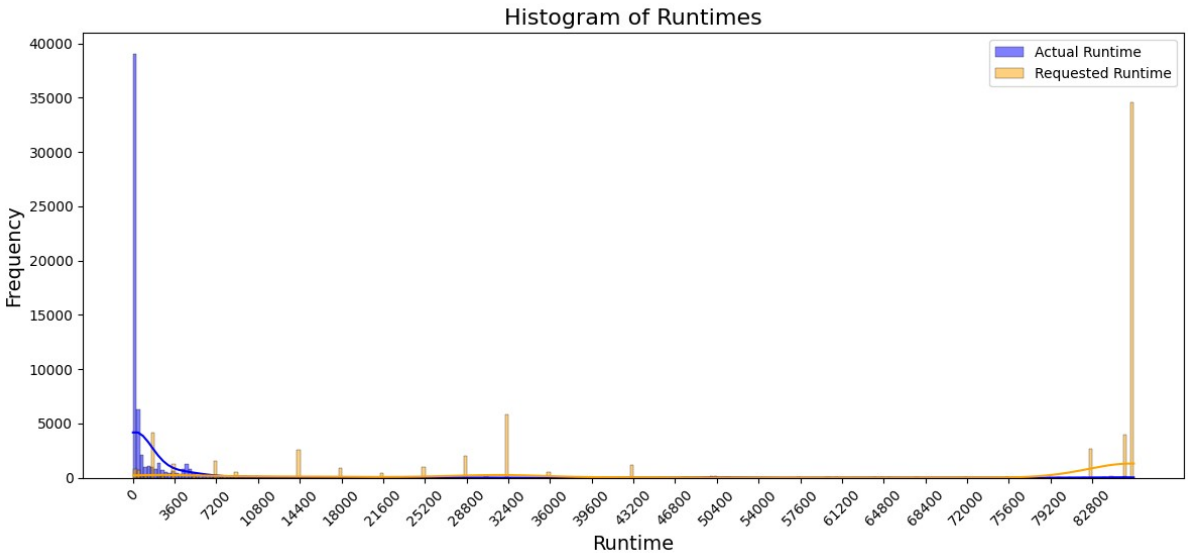


Fig. 1: User Estimation

Fig. 2 shows how, , despite the clear improvement over the user estimation, the heuristic prediction does not follow very closely the distribution of the actual runtime. In particular, we can notice several discrepancies in the frequency bars relative to the lower part of the plot.

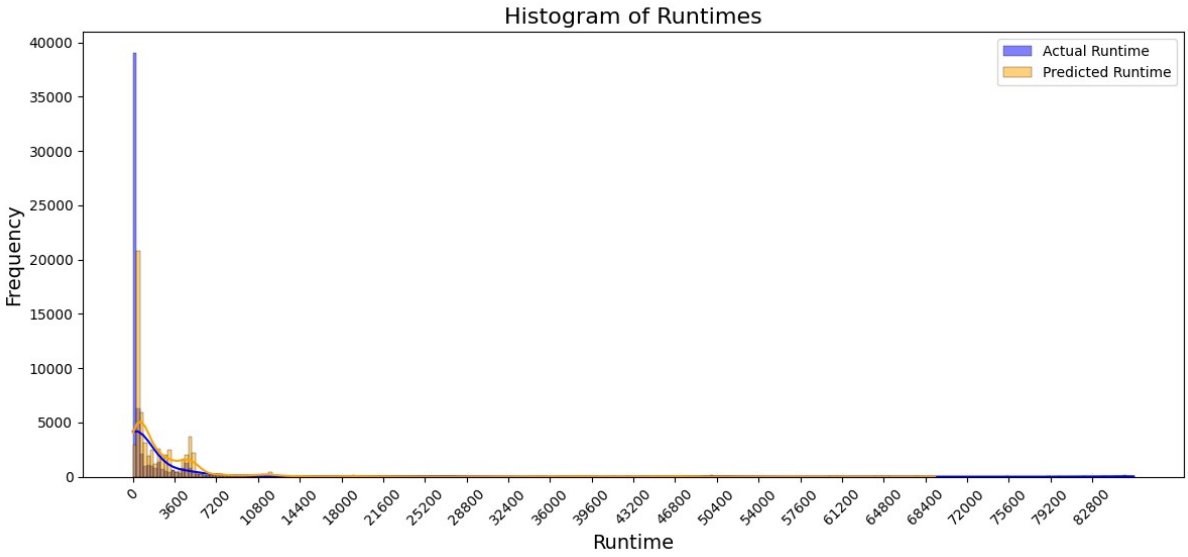
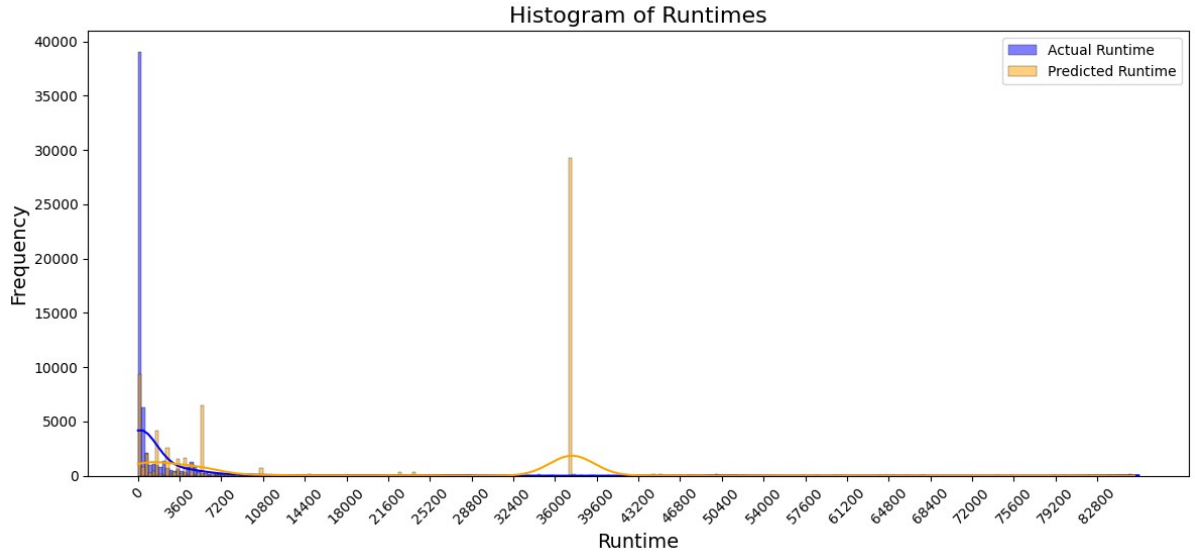
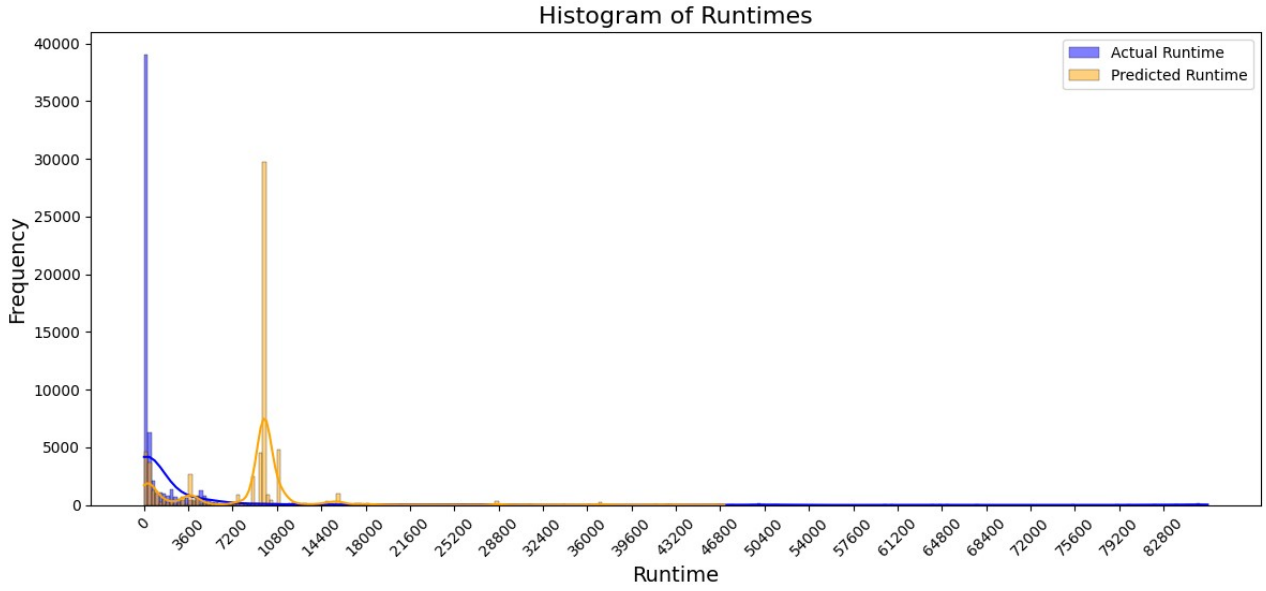


Fig. 2: Heuristic Estimation

Fig. 3 shows a comparison of prediction distributions for SET 1 using the two different regression models. These histograms confirm that SET 1 is unable to provide sufficient information to guide the regression and produce predictions which follow the actual runtime distribution. Furthermore, they clearly show how both Decision Tree (Fig. 3a) and Normalized Polynomial (Fig. 3b) tend to predict a small subset of all the possibilities, with some values predicted an abnormally high percentage of times.



(a) Decision Tree + SET 1

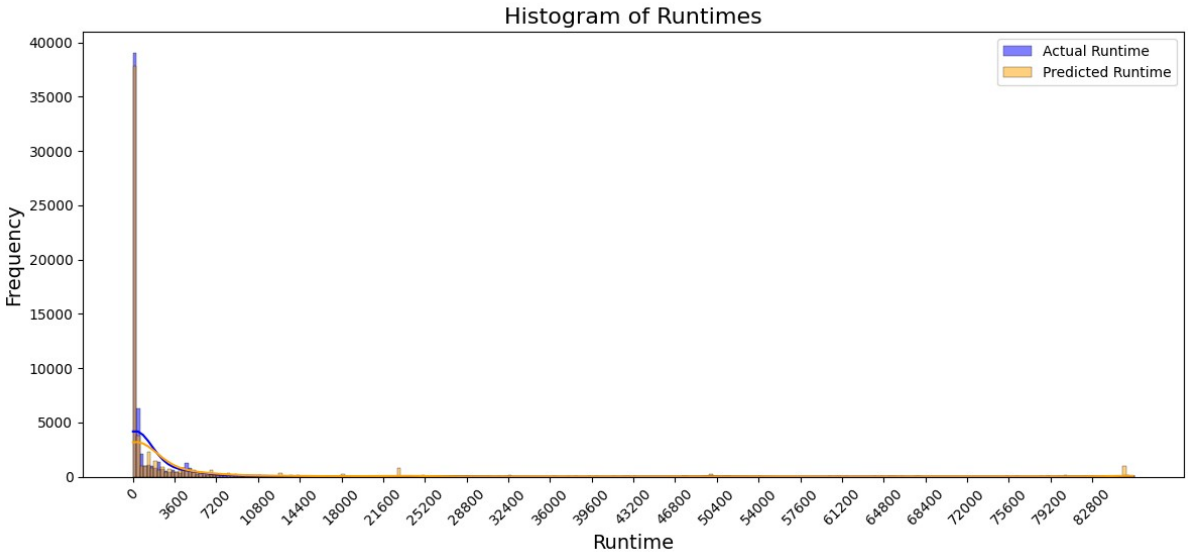


(b) Normalized Polynomial + SET 1

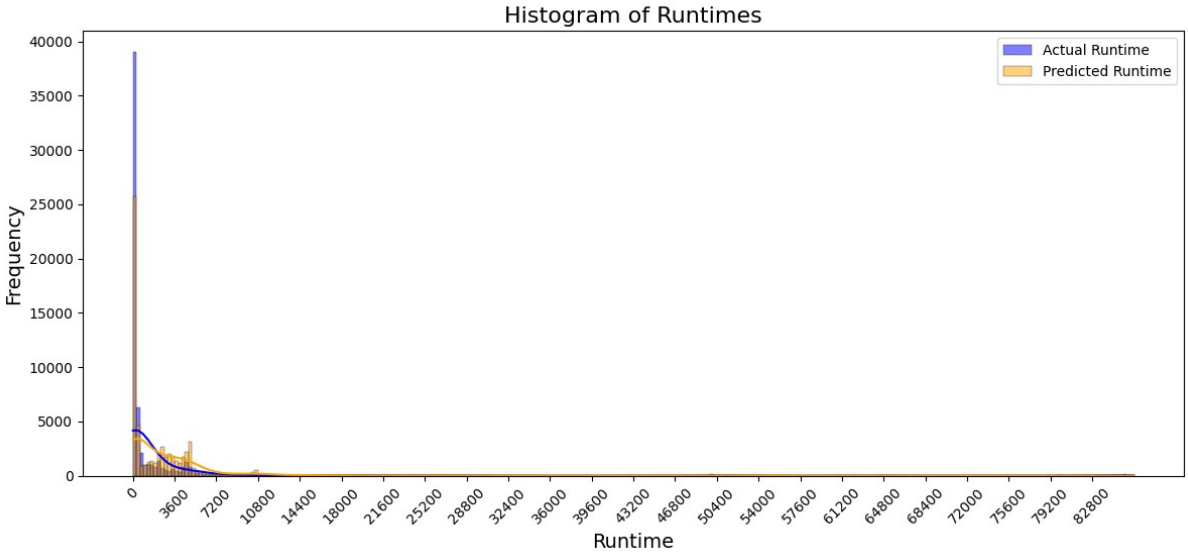
Fig. 3: Comparison of prediction distributions for SET 1 using the two different regression models.

Lastly, Fig. 4 shows a comparison of prediction distributions for SET 2 using the two different regression models. We can immediately notice that SET 2 produces predictions which follow the actual runtime distribution more closely. Decision Tree (Fig. 4a) is the method which provides the best approximation of the actual distribution for lower values of runtime, but with the presence of some odd peaks in frequency at higher values of runtime, which explain the higher average prediction error. Normalized Polynomial (Fig. 4b), instead, is less precise in predicting lower

runtime values, but does not mistakenly predict higher runtime values as often as Decision Tree, resulting in the best overall approximation of the actual runtime distribution.



(a) Decision Tree + SET 2



(b) Normalized Polynomial + SET 2

Fig. 4: Comparison of prediction distributions for SET 2 using the two different regression models.

Therefore, we can conclude that SET 2 features enable both machine learning models to outperform the heuristic, since:

- Despite having an higher average prediction error than the heuristic, Decision Tree is far more precise in predicting the runtime of shorter jobs, which represent the majority of the jobs in the workload.
- Normalized Polynomial is the method which has the lowest average prediction error, improving upon the heuristic by almost 25%

2.2. CLASSIFICATION

Labeling	Method	Accuracy	F1-Score (Macro)	F1-Score (Micro)
Four Classes	User Estimation	0.16	0.16	0.13
	Heuristic	0.76	0.76	0.58
	K-NN (SET 1)	0.69	0.55	0.69
	K-NN (SET 2)	0.80	0.70	0.80
Seven Classes	User Estimation	0.10	0.10	0.10
	Heuristic	0.67	0.67	0.33
	K-NN (SET 1)	0.57	0.34	0.57
	K-NN (SET 2)	0.71	0.59	0.73

From the table above, it is clear that using four classes yields better results across all feature sets compared to seven classes. Additionally, regardless of the number of classes, SET 2 consistently outperforms SET 1, demonstrating the value of the additional features included in SET 2. All k-NN methods notably outperform user estimation, which suffers from very low accuracy and F1-scores, underscoring the limitations of relying solely on user-provided runtime estimates. Compared to the heuristic baseline, machine learning models using SET 2 yield modest improvements in accuracy and notable gains in Micro F1-score. Since the classes are imbalanced, the Micro F1-score better reflects the model’s overall predictive performance.