Proactive Precision: Enhancing High-Performance Computing with Early Job Failure Detection

1st Dipanwita Mallick
Hewlett Packard Enterprise
Seattle, United States of America
dipanwita.mallick@hpe.com

2nd Siddhi Potdar
Hewlett Packard Enterprise
Fort Collins, United States of America
siddhi.potdar@hpe.com

3rd Saptashwa Mitra
Hewlett Packard Enterprise
Fort Collins, United States of America
saptashwa.mitra@hpe.com

4th Charlie Vollmer
Hewlett Packard Enterprise
Fort Collins, United States of America
charlie.vollmer@hpe.com

5th Nithin Singh Mohan
Hewlett Packard Enterprise
Fort Collins, United States of America
nithin-singh.mohan@hpe.com

***Abstract*—In the realm of high-performance computing (HPC), the efficient allocation and utilization of computational resources are paramount. A significant challenge in this domain is the early detection of job failures, which, if unaddressed, can lead to considerable inefficiencies and increased operational costs. This study introduces an innovative approach to preemptively identify potential job failures within HPC environments by leveraging an adaptive machine learning model. Utilizing the Slurm workload manager as a basis for our investigation, we analyze over 300,000 job records from a 1027-node internal HPC system. Our methodology encompasses a comprehensive preprocessing of job data, including an in-depth analysis of historical job information and resource usage, to train a predictive model capable of distinguishing between successful and failed job submissions with a high degree of accuracy. To address the inherent challenge of class imbalance within our dataset—characterized by a predominance of successful job submissions—we employ advanced strategies such as active learning and oversampling techniques. The result is a robust model that not only accurately predicts job failures within two minutes of job submission but also offers insights and suggestions for effective system adjustments and resource reallocation to enhance overall system efficiency. Additionally, we suggest ways to integrate our work within a Jupyter notebook environment, making our predictive model and post-prediction analytics capabilities accessible to a broad range of users within the HPC community. Through a comprehensive evaluation, including precision, recall, F1 score, and the receiver operating characteristic (ROC) curve, our model demonstrates good performance, with an area under the curve (AUC) of 0.99, indicating a near-perfect capability in predicting job outcomes. This research aims at improving HPC job scheduling and management by utilizing descriptive and predictive analytics to understand the data and forecast potential outcomes, subsequently leveraging prescriptive analytics to suggest optimal methods for adjusting resource allocations.**

***Index Terms*—class imbalance, adaptive modeling, active learning, Jupyter notebook integration, job scheduling, resource optimization, high-performance computing**

I. Introduction

High-Performance Computing (HPC) has become an indispensable tool across many scientific, engineering, and business domains, enabling complex and computationally intensive tasks to be performed with unprecedented speed and efficiency. At the core of HPC's success is the effective scheduling and management of computing jobs, which are often subject to stringent performance and time constraints. Despite the advanced capabilities of HPC systems, job failures remain a significant challenge, affecting system efficiency, resource utilization, and the timely completion of computational tasks. These failures can arise from many factors, including hardware malfunctions, software bugs, system overload, or misconfigurations, leading to wasted resources and delays. The ability to quickly detect and respond to these failures is critical, yet traditional methods often fall short, lacking the foresight and adaptability needed to preemptively address potential issues.

The primary challenge this research addresses is the proactive detection of job failures within HPC environments. Traditional approaches to job management and scheduling within HPC systems have been reactive, with failures often identified only after they occur, resulting in significant resource wastage and operational inefficiencies. This research aims to transform the approach from reactive to proactive by utilizing sophisticated machine learning techniques to predict job failures within two minutes of job launch. By harnessing descriptive and predictive analytics, this study aims to deeply understand job data and accurately forecast outcomes, thus enabling the preemptive adjustment of resource allocations. Furthermore, leveraging prescriptive analytics allows for the recommendation of optimal resource adjustments, enhancing overall system performance. The objectives of this research are twofold: to develop an adaptive machine learning model capable of early job failure prediction within HPC systems, and to create a user-friendly interface that facilitates the application of this model within existing workflows. In summary, we aim to achieve following objectives:

- The combination of supervised and unsupervised learning to enhance prediction accuracy and gain deeper insights into job failures.
- The development of a user-centric interface and implementation of a pipeline for viewing results in real-time,

providing actionable feedback to HPC administrators and users.
- The application of an oversampling methodology to effectively handle the significant class imbalance present in job failure data, ensuring more reliable and accurate predictions.
- The adoption of a continuous retraining mechanism for the predictive model, ensuring its adaptability and ongoing improvement in response to new data and evolving system dynamics.
- The provision of post-prediction insights that allow for a deeper understanding of the causes of failure, facilitated by analyzing historical data and the model's interpretation of features and their importance.

By addressing these aspects, our research not only advances the field of HPC job management but also sets a new benchmark for predictive analytics in high-performance computing environments, significantly improving upon existing methodologies and tools.

In the rest of the paper, starting with a literature review in Section II, it assesses existing HPC solutions, machine learning's impact on efficiency and reliability, and identifies current research gaps. Section IV and V details the data collection and preprocessing efforts, including feature analysis crucial for the predictive model discussed in Section IX, where we elaborate on model development, training, and the integration of learning techniques. Section XI introduces a user-focused interface, designed for ease of use within a Jupyter notebook environment, enhancing user interaction and decision-making. A comprehensive performance evaluation in Section X assesses the model's effectiveness, followed by Section XII, which outlines future research directions, including advanced machine learning techniques and potential for scalability. The paper concludes in Section XIII, summarizing the study's contributions to HPC job management and efficiency.

II. Literature Review

Job failure prediction in HPC systems is a critical research area that aims to improve system efficiency and utilization. Prior studies have embarked on this journey, employing various machine learning strategies to enhance the accuracy and efficiency of the systems.

In 2021, team [1] developed a framework to predict job failure in HPC systems at two different points: job submit-state and job start-state. The predictive models are developed using decision-tree induction techniques, and the performance of the models is evaluated using two workload logs from different HPC systems. The study aims to improve the efficiency of the systems at the job level by providing users with guiding tools to make efficient decisions when submitting jobs. The models can help users effectively manage their job submissions, ultimately enhancing the efficiency of the systems. The results show that the predictive models built from decision tree algorithms exhibit high accuracy in various test cases, although issues related to imbalance ratios between majority and minority classes are identified. Overall, the paper presents a comprehensive approach to addressing job failure prediction in the systems and evaluates the effectiveness of the proposed models.

In 2023, team [2] published a paper that highlights strategies to optimize HPC system management by predicting job failures at submit-time. The authors study job failure prediction using machine learning algorithms, combined with Natural Language Processing (NLP) tools to represent jobs. The novelty of their approach lies in working in an online fashion in a real system, as opposed to using random splits of historical data. The study is based on a dataset from the Marconi100 HPC system hosted at the HPC center CINECA in Italy.

In 2022, team [3] proposed a new job failure prediction method for supercomputers based on enhancing semantic information about job applications. Prior works rely heavily on real-time monitoring of job performance metrics, which is difficult to implement. Instead, this paper analyzes over 472K jobs on a production supercomputer to extract names, submission paths, and cluster jobs into related applications. This application-aware modeling with semantics about relationships between jobs allows effective failure prediction with lower overheads. Experiments using decision trees and random forests demonstrate significant gains - final accuracy reaches 88.16% with 95.23% specificity and 88.24% sensitivity. The job application semantic enhancement provides 5-6% better performance over baseline models using time and resources alone. Thus, this technique can generalize across users, job types, and system architectures.

In 2021, team [4] also introduced an innovative framework that leverages machine learning, utilizing novel features like job running path and retry count, which reflect the job's application type and user behavior, respectively. The study demonstrates that incorporating these features significantly enhances the prediction accuracy, outperforming existing methods by about 4% in comprehensive evaluation, achieving over 89% accuracy in identifying job failures.

In 2023, researchers in the [5] conducted an extensive analysis of scheduler logs from a production-level HPC system. They provided a comprehensive statistical and machine learning examination of job failures, benefiting from a rich dataset that allowed for in-depth feature analysis to pinpoint crucial predictors of job outcomes. The comparison across six machine learning models revealed that tree-based approaches excelled in predicting job failures, showcasing superior accuracy and computational efficiency, thereby pushing forward the capabilities in predictive modeling within HPC environments.

In the literature review, we observed several key methodologies for enhancing the reliability and efficiency of the HPC systems through job failure prediction. Each study underscores the criticality of leveraging job submission data attributes, such as job ID, user ID, and resource requests, in conjunction with advanced machine learning algorithms to anticipate failures. This collective body of work validates its approaches through the analysis of authentic datasets from HPC facilities, rigorously assessing model performance to ensure real-world

applicability and operational timeliness. However, there are also several notable challenges mentioned in the paper: balancing prediction precision against computational demands, curating optimal feature sets without resorting to overly broad or sensitive data, ensuring model adaptability across diverse HPC environments, and achieving predictions in real or near-real time to facilitate preemptive action.

By addressing a few of these challenges, our research proposes enhancements in several critical areas. We aim to develop real-time predictive analytics to offer immediate, actionable insights for preempting job failures. Furthermore, we plan to enhance model training through job profile analysis, leveraging historical data on job performances to refine prediction precision. Incorporating Explainable AI (XAI) stands as another cornerstone of our approach, aiming to demystify the prediction process and foster trust among users and administrators by clarifying the logic behind predictive outcomes.

III. Workflow Model

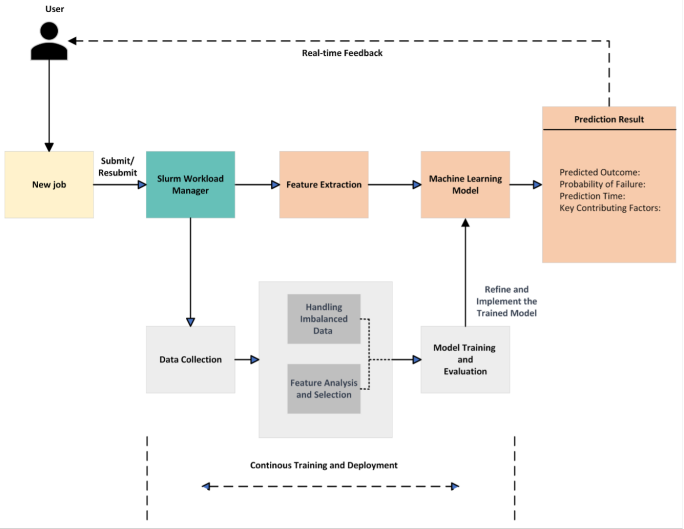

Fig. 1. Diagram showcasing the process flow for job prediction and inference.

The workflow(fig.1) for job prediction in HPC environments [6] operates in two distinct modes: real-time inference and offline training. In the real-time inference mode, users submit jobs to the Slurm Workload Manager, which triggers the predictive model to assess the likelihood of job success or failure. This immediate analysis allows users to make informed decisions, such as whether to adjust job configurations and resubmit to avoid potential failures, based on the real-time prediction results provided post-submission. In the offline mode, our model focuses on continuous improvement by regularly incorporating new data from the Slurm workload manager. This approach not only increases the model's accuracy but also ensures its adaptability to new patterns and trends in the data.

IV. Data Collection and Data Labeling

This section will provide details on how the job data was collected from the 1027-node internal HPC system using the Slurm workload manager. The preprocessing steps taken to prepare the raw data for analysis will then be described, including handling missing values through deletion or imputation, normalizing numerical features, encoding categorical variables, and extracting new features as needed. Useful visualizations and statistics on the data will also be presented. Additionally, exploratory analysis like job profile segmentation using clustering algorithms will be discussed to discern patterns and commonalities in job behaviors. This provides a foundation for the downstream predictive modeling. Finally, the methodology for selecting the most relevant input features through statistical correlation analysis and predictive model-based feature importance techniques will be outlined. This subset of features serves as input into the job failure prediction models to improve efficiency and performance.

A. Dataset

The dataset for this research was collected from a 1027-node internal HPC system running the Slurm workload manager for cluster management and job scheduling. Slurm is widely used in HPC environments and provides capabilities such as scalability, fault-tolerance, and ease of configuration.

The data encompasses over 300,000 job records spanning user jobs across various teams and workloads on the system from October 2022 to September 2023. The jobs data was gathered using Slurm's accounting interface sacct, which exposes rich information on each job across over 100 data fields. While most features were directly obtained from Slurm, some additional features were derived by processing certain raw fields:

- CPU, Memory, Nodes: Derived by splitting the AllocTRES field.
- CPUTime: Calculated using Elapsed time * CPU count.

The key fields extracted into the dataset include:

- User ID: Uniquely identifies the user submitting the job.
- Job Name: Free-form name assigned to the job.
- State: Job status indicating whether the job was successfully concluded or not.
- CPU: Number of CPU cores allocated.
- Memory: Amount of memory allocated.
- Nodes: Count of nodes allocated.
- CPU Time: Total CPU time used by the job.
- Submit Time: Job submission timestamp.
- Start Time: Job start execution timestamp.
- End Time: Job completion timestamp.
- GID: Group ID of the user.

These fields were selectively chosen based on their potential influence on job outcomes and failure prediction capabilities based on research into prior work. The total feature set provides detailed resource allocation, usage, and runtime insights on each job. In the subsequent data preprocessing phase, additional engineered features are extracted from this raw set

to better expose patterns from the data. The final output is a refined, analysis-ready dataset feeding into the later machine learning modeling and evaluation for predictive analytics.

B. Labeling and Problem Framing

The raw job data from Slurm contains a variety of completion status labels across different failure modes and outcomes. After examining the distribution of these labels, we found that completed jobs constitute about 92% of the data, while the remaining labels represent specific, relatively infrequent failure cases. Some examples of the low-frequency labels include BOOT_FAIL, NODE_FAIL, OUT_OF_MEMORY, PREEMPTED, etc. With limited data samples for each individual failure type, modeling them independently would be statistically ineffective.

Therefore, we frame the job failure prediction task as a binary classification problem. The jobs are labeled as either Completed (1) or Failed (0). The failed class consolidates all failure outcomes like BOOT_FAIL, NODE_FAIL, etc., into a single category representing jobs that did not successfully complete. This simplifies the classification modeling while addressing the substantial class imbalance between successful and unsuccessful jobs. It also matches the end objective of identifying jobs likely to fail rather than diagnosing the exact failure mode, which has little added utility. The encoding of the raw status labels into this binary representation is done programmatically using a LabelEncoder. As seen in figure(fig.2), the resultant distribution has approximately 92% Completed and 8% Failed jobs - still imbalanced but significantly more manageable compared to modeling separate failure types.

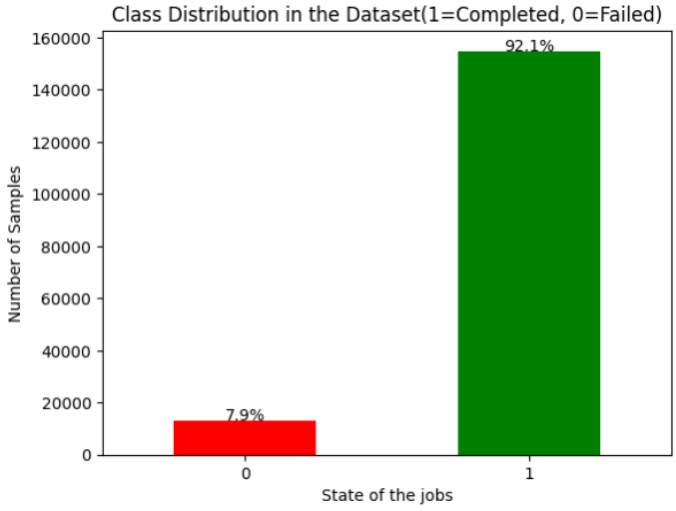

Fig. 2. The figure illustrates the imbalance present in the data label distribution.

Framing the prediction as a binary classification allows applying more effective modeling techniques. It also elegantly transforms the raw multi-class formulation with skewed labels into a cleaner problem statement focused singularly on predicting if jobs will fail or succeed.

V. Preprocessing

In this research, we undertook a comprehensive approach to refine and enhance the data collected from HPC job logs, aiming to boost the predictive accuracy of our machine learning models. The process began by cleaning the dataset, where we removed any records that were incomplete to maintain the quality of our inputs. Alongside that, we focused our attention on individual job records by filtering out job steps and omitting columns like JobName and ExitCode, which did not contribute to our predictive goals.

During the feature engineering phase, we introduced new attributes to gain further insights. We calculated the run time of jobs by measuring the interval between their start and end times, which helped us understand how long jobs were running. Similarly, we determined wait time to quantify the delay between when a job was submitted and when it started, giving us an indication of the system's queuing time. We also extracted temporal features from the job submission times, such as the day of the week and month of submission, among others. This detail helped us investigate whether the timing of job submissions could affect their outcomes. To prepare for future analyses, we excluded jobs that ran for less than two minutes, and categorized users based on their activity levels. The exclusion of jobs that ran for less than 2 minutes serves to diminish noise within the dataset, ensuring that our focus remains squarely on computational tasks that meaningfully engage HPC resources. Such jobs are more likely to exhibit discernible patterns related to failures or successes, thus enhancing the overall reliability and quality of our analysis. Furthermore, this criterion aligns with our forward-looking objective to integrate more complex and indicative metrics into our model, particularly those relevant to longer-running jobs. By concentrating on these jobs, we position our research to offer more accurate and operationally relevant predictions, catering specifically to the substantive computational activities within HPC systems.

To gain a deeper understanding of job behaviors, we applied unsupervised clustering methods like K-Means and Gaussian Mixture Models(GMM), which allowed us to group job profiles. This step added an interpretive layer to our dataset.

This detailed process of data cleaning and feature engineering transformed the raw dataset into a more informative and refined form, setting a solid foundation for the next stage of our research—applying classification models to predict the outcomes of new job submissions with greater accuracy.

VI. Exploratory Data Analysis

Under the umbrella of exploratory data analysis, performing clustering on job data emerges as a pivotal step toward unraveling the underlying patterns and groupings within HPC systems. This exercise's main purpose is to categorize jobs into distinct clusters based on their characteristics and performance metrics. This categorization aids in identifying similarities and differences among jobs, potentially uncovering factors that contribute to job success or failure.

We chose GMM for clustering because of its adaptability to clusters of varying sizes and configurations. Contrary to K-Means, which presupposes the clusters to be spherical, GMM can handle ellipsoidal shapes. This characteristic makes it a better fit for the intricate data patterns frequently observed in HPC job logs. GMM is built on the premise that data points emerge from a combination of several Gaussian distributions, each corresponding to a cluster. It computes the likelihood of each data point's association with the clusters, enabling a more nuanced classification.

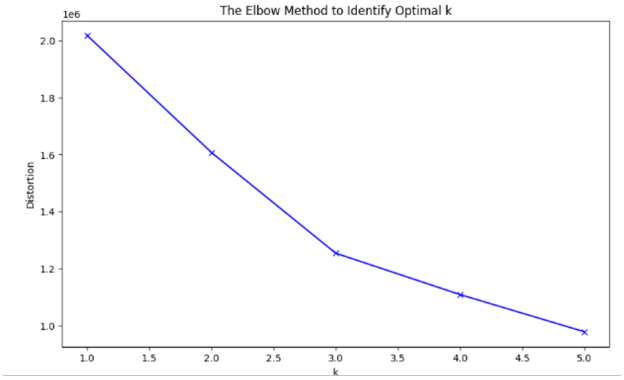

Fig. 3. Elbow plot showing the relationship between the number of clusters (K) and the distortion (within-cluster sum of squares). The elbow point, located around K=3, indicates the optimal number of clusters for the given dataset, balancing the trade-off between the number of clusters and the compactness of the clusters.

To determine the optimum number of clusters, we employed the elbow method(fig.3), which is a common technique used to determine the optimal number of clusters (K) in a dataset when using clustering algorithms like K-means. It helps to balance the trade-off between the number of clusters and the within-cluster sum of squares (WCSS), also known as distortion. As shown, the elbow point is located around K=3. This suggests that choosing 3 clusters would be a good choice for the given dataset. The clustering outcomes offer a multifaceted view of job executions, highlighting patterns that were previously obscured. By discerning the nuances of job types, durations, user activities, and system states, we can effectively refine our predictive models.

	cluster	cpu mean	cpu std	mem mean	mem std	run_time mean	run_time std	wait_time mean	wait_time std
0	0	256.00	0.00	446.00	0.00	133.38	19.78	1.23	3.04
1	1	23917.03	46280.18	42406.76	81721.48	2268.03	6099.85	2431.51	8777.29
2	2	235.70	67.95	445.98	1.87	785.07	1651.60	20.71	17.52

Fig. 4. Job profiles based on resource usage, run_time and wait_time.

Fig.4 shows distinct patterns in terms of CPU and memory usage, as well as run and wait times. Cluster 0 represents a group of jobs that use minimal CPU and memory resources on average and have the lowest mean wait and run time. This cluster could be indicative of lightweight or low-priority jobs. On the other hand, cluster 1 is characterized by significantly higher resource usage, with the highest mean CPU and memory utilization. The run time mean, and standard deviation are also the highest among the clusters, suggesting these jobs are long-running, resource-intensive tasks. The wait time has a high mean and standard deviation, which could indicate these jobs may have higher priority or they require specific resources that are less available. Finally, jobs in cluster 2 have moderate CPU and memory usage, run times, and wait times when compared to the other two clusters. The standard deviation for run time is high, suggesting variability in the job lengths within this cluster.

Duration	Long	Medium	Short
cluster			
0	0	0	78983
1	3583	8571	13071
2	1727	7800	54391

Fig. 5. Job profiles based on job duration.

Fig.5 illustrates the job duration profiles of the three clusters. Cluster 0 consists entirely of short-duration jobs, suggesting simple, low-complexity tasks. Cluster 1 exhibits a diverse mix of job durations, indicating a general-purpose cluster handling various tasks. Cluster 2 primarily contains short-duration jobs, with some medium-duration tasks, sitting between the profiles of Clusters 0 and 1.

Activity	Frequent	Infrequent	New
cluster			
0	0	1	0
1	36	37	68
2	12	12	21

Fig. 6. Job profiles based on user activity.

Fig.6 shows the user activity levels across the three clusters. Cluster 0 has only one infrequent user, suggesting under utilization or specific use cases. Cluster 1 is the most active, with a balanced mix of frequent, infrequent, and new users, indicating a diverse user base. Cluster 2 has a smaller but balanced distribution of user activity, showing less activity than Cluster 1 but still a healthy mix of user engagement.

State	Completed	Failed
cluster		
0	78949	34
1	17479	7746
2	58418	5500

Fig. 7. Job profiles based on job status.

Fig.7 presents the distribution of job states ('Completed' or 'Failed') across the three clusters. Cluster 0 and cluster

2 are dominated by state 'Completed' jobs, indicating high success rates. In contrast, cluster 1 shows a more balanced distribution between the two states, with a significant number of state 'Failed' jobs, suggesting a higher occurrence of failures compared to the other clusters.

Type cluster	Balanced	Compute	Memory
0	78983	0	0
1	25005	60	160
2	63914	0	4

Fig. 8. Job profiles based on job size and type.

Fig.8 illustrates the distribution of job types (balanced, compute-intensive, or memory-intensive) across the three clusters. Cluster 0 and cluster 2 are composed almost exclusively of balanced jobs, suggesting they are optimized for standard workloads with even resource requirements. Cluster 1, while primarily handling balanced jobs, also accommodates a small number of compute-intensive and memory-intensive jobs, indicating a more diverse job mix compared to the other clusters. Based on extensive clustering analysis performed, we can glean several interesting characteristics about the clusters. The key findings can be summarized as follows:

- Cluster 0 is likely used for lightweight, low-complexity jobs that consistently succeed and are handled by users who infrequently utilize the system.
- Cluster 1 is a versatile and highly active cluster handling a range of job types and durations, used by a diverse group of users from regular to first-time users, and experiences a significant number of job failures.
- Cluster 2, while less active than Cluster 1, is utilized for a healthy mix of short and medium-duration jobs, has a high success rate, and like Cluster 0, is optimized for balanced jobs but has some memory-intensive tasks.

Next, we will look at the correlation map to understand the intricate interplay of features and how they potentially influence job outcomes.

The plot in fig.9 offers a visual exploration of the interconnectedness of user behavior, job duration, resource utilization, and temporal submission patterns, unveiling the subtle and sometimes unexpected drivers of job success or failure. Our analysis revealed that user behavior(as denoted by user_id) exhibited a negative correlation with job state (r = -0.33), suggesting that some users have distinct patterns that affect job performance. The moderate negative correlation (r = -0.25) between job duration and job state reveals an intriguing relationship between the runtime of a job and its probability of success or failure. This correlation suggests that longer job durations are associated with a higher likelihood of job failure, while shorter-running jobs tend to have a higher success rate. Temporal features like submission month and week of the year demonstrated negative correlations (r = -0.29 and r = -0.28,

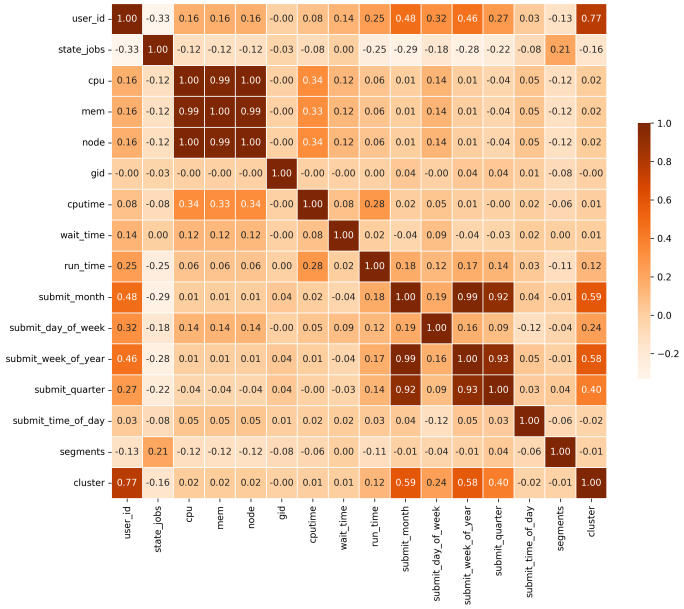

Fig. 9. A heatmap visualization of feature correlations with job state.

respectively) with job state, implying that job performance might be cyclically influenced by systemic or external factors. In terms of resource usage, the almost perfect correlation (r = 1.00) between CPU and memory usage underscores that resource-intensive jobs demand high levels of both CPU and memory. However, when considering the job state factor, there is a noticeable negative correlation with both CPU and memory usage (r=-0.12), implying that jobs with higher resource demands do not necessarily correspond with higher success rates. This finding challenges the common assumption that more resources directly lead to better outcomes. Instead, it suggests that the relationship between resource allocation and job success is more complex and warrants the development of advanced models for a deeper understanding of how resource dynamics affect job states in HPC environments. Additional noteworthy correlations include:

- **Temporal Submission Patterns:** The strong correlations within temporal features such as submit month and week of the year, and their relationship with quarters, reflect a potential link to organizational workflows and planning periods, which could inform system usage forecasting and capacity planning.
- **User-Cluster Allocation:** The high positive correlation(r=0.77) between user IDs and cluster assignments indicates a meaningful relationship where certain user IDs are systematically associated with specific clusters, possibly reflecting organizational strategies, user or job characteristics, or resource allocation policies.

These multifaceted correlations elucidate a complex architecture of job executions within the HPC system. Incorporating these insights into our predictive modeling will not only bolster the accuracy of job outcome forecasts but also equip system administrators with the data-driven knowledge necessary for

informed decision-making.

VII. Final List of Features for Modeling

In the process of constructing a predictive model, the selection of features is a crucial step that directly impacts the model's performance. There are certain features that our analysis suggests are indispensable:

- User ID: Reflects user-specific patterns that may influence job outcomes.
- Run Time: Indicates the job duration, which could be predictive of job success or failure.
- CPU and Memory Usage: Despite their perfect correlation suggesting redundancy, these features are key indicators of job resource intensity.
- Temporal Submission Features: Including submit_month, submit_day_of_week, submit_week_of_year, and submit_quarter, as they may capture cyclical patterns in job success rates.

Conversely, there are features whose contributions may appear marginal or redundant, yet we opt to retain them in the initial modeling phase. This comprehensive approach allows the machine learning model to evaluate each feature's impact through techniques such as feature importance. The features in question include:

- Node and GID: They might capture infrastructure-specific influences on job performance.
- CPUtime: While it may be correlated with run_time, it could offer additional granularity.

As we proceed, the model will dictate the relevance of each feature through empirical results. It is through this iterative process of model training and evaluation that we will discern whether the exclusion of certain features enhances model accuracy. By maintaining a full set of features initially, we leave no stone unturned, ensuring our model has the broadest possible basis to learn from.

VIII. Evaluation Metrics

When evaluating the performance of our job failure prediction model, it is essential to look beyond accuracy measures, especially when dealing with imbalanced class distributions. In our case, the minority class represents job failures, which are of interest. Relying solely on accuracy can be misleading, as it may overlook the model's ability to effectively identify these critical failure events.

To overcome this challenge and ensure a thorough evaluation, we employ three key metrics: precision, recall, and the F1 score. These metrics provide a more nuanced assessment of the model's capability to accurately predict job failures while considering the potential impact of false positives and false negatives.

Precision, also referred to as the positive predictive value, indicates the proportion of correctly predicted job failures (true positives) among all instances predicted as failures. It is calculated as follows:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (1)$$

Recall, also known as sensitivity or true positive rate, measures the proportion of actual job failures that are successfully identified by the model. It is calculated using the following formula:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2)$$

On one hand, a model with high precision but low recall may be too conservative, missing many actual failures. On the other hand, a model with high recall but low precision may correctly identify most failures but also generate a high number of false alarms.

To find a balance between precision and recall, we utilize the F1 score, which is the harmonic mean of these two metrics. The F1 score is calculated as follows:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

The F1 score ranges from 0 to 1, with 1 representing perfect precision and recall. By considering both metrics, the F1 score offers a comprehensive measure of the model's performance, accounting for the trade-off between false positives and false negatives.

By emphasizing these metrics, we ensure a rigorous and accurate evaluation of our model's performance, considering the challenges posed by class imbalance. This comprehensive assessment enables us to make well-informed decisions regarding model selection, hyperparameter tuning, and potential enhancements to improve the model's ability to predict job failures effectively.

IX. Advanced Prediction Modeling

In our study, we explored various supervised machine learning algorithms to tackle the job failure prediction task. Among these techniques, XGBoost emerged as the best performer in terms of accuracy and efficiency. XGBoost, an optimized gradient boosting framework, combines multiple weak learners to create a strong predictive model. We chose XGBoost for several reasons:

- **Ability to handle imbalanced datasets**: XGBoost provides hyperparameters such as scale_pos_weight that can effectively handle class imbalance by assigning higher weights to the minority class.
- **Robustness to outliers**: XGBoost's decision tree-based architecture makes it inherently robust to outliers, as it focuses on splitting criteria rather than being influenced by extreme values.
- **Efficiency and scalability**: XGBoost is known for its fast training speed and ability to handle large datasets efficiently, making it suitable for our HPC job failure prediction task.
- **Feature importance ranking**: XGBoost provides a built-in feature importance ranking, allowing us to identify the most influential features contributing to job failures.

By fusing supervised learning (XGBoost) with unsupervised learning (GMM), we aimed to enrich our data analysis and

TABLE I
COMPARISON OF MODEL PERFORMANCE

Model	Precision (0/1)	Recall (0/1)	F1 Score (0/1)	Accuracy
XGBoost	0.59/1.00	0.98/0.94	0.73/0.97	0.94
XGBoost + Random oversampling	0.06/0.91	0.31/0.58	0.10/0.71	0.56
XGBoost + SMOTE oversampling	0.03/0.91	0.10/0.76	0.05/0.72	0.70
XGBoost + Active learning	0.78/0.98	0.87/0.97	0.82/0.97	0.97

Evaluation results of different models on the test data, showing precision, recall, and F1-score for each job outcome class (0: Failed, 1: Completed). For instance, the precision for the XGBoost model with active learning on test data is 0.78 for class 0 and 0.98 for class 1.

capture underlying patterns in job characteristics. This approach allowed us to identify distinct job profiles and leverage this information to improve the accuracy of our job failure predictions.

To address the challenge of class imbalance in our dataset, where the majority of jobs belonged to the completed class, while job failures were relatively rare, we aim to bolster our model with oversampling techniques. We explored random oversampling and SMOTE (Synthetic Minority Over-sampling Technique) but our experiments with these oversampling methods did not yield satisfactory results.

Thus, we explore more advanced oversampling strategies. Active learning is such a process which involves an iterative process that intelligently selects the most informative data points for labeling, rather than relying on random sampling. By focusing on the most informative samples, active learning allows us to build robust models with smaller datasets while improving generalization performance.

The key advantages of employing active learning in our job failure prediction task are therefore as follows:

- **Efficient use of labeling resources**: Active learning reduces the number of labeled examples required to achieve good performance by strategically selecting the most informative samples.
- **Improved generalization**: By iteratively training the model on carefully selected data points, active learning helps the model learn from diverse and representative samples, enhancing its ability to generalize well to unseen data.
- **Handling class imbalance**: Active learning can effectively address class imbalance by focusing on informative samples from both majority and minority classes, leading to a more balanced and effective learning process.
- **Adaptability to changing workloads**: Active learning allows the model to adapt to evolving workloads and system characteristics by continuously incorporating new informative samples into the training process.

In our implementation of active learning, we calculated the top N samples based on uncertainty and region density. These samples were considered the most informative and were fed into the XGBoost classifier for training. We evaluated the model's performance using a validation set and repeated this process for a fixed number of iterations. The best-performing model obtained during this iterative process was then used to make predictions on the final test set.

Our experimental results demonstrated that active learning significantly improved the precision, recall, and F1 scores of our job failure prediction model as shown in Table I. Moreover, the active learning process converged quickly, requiring only a few iterations to achieve optimal performance.

X. COMPREHENSIVE PERFORMANCE EVALUATION

After the training and validation phases, the model was evaluated on a reserved test dataset, and segregated from the entire dataset to avoid any data leakage. The application of our active learning resulted in an improvement of model performance, effectively rectifying the data imbalances present and making the model more robust. This improvement is reflected in Table I.

The table shows results of different models on the test set. Comparing the model results on train set and test set, we observe all models perform extremely well on train set and not as well on the test set. In machine learning, we try to strike a balance between the discussed scores for a particular model on both train and test sets. This ensures the model is not overfitting, or underfitting. Considering a real-world scenario, we need the model to be adaptable to changes in data. Active learning fulfils this requirement as we can see in the results table. The precision and recall metrics for both classes have shown improvement and are more closely aligned with each other when utilizing active learning. This indicates a reduction in bias, either towards a particular class or a specific type of error, as compared to scenarios without active learning.

In fig.10 the curve demonstrates the trade-off between the True Positive Rate (sensitivity) and the False Positive Rate (1-specificity) across different thresholds. The area under the curve (AUC) is 0.99, indicating that the model has a near-perfect measure of separability with a high capability to distinguish between the classes. The closer the AUC is to 1, the better the model is at predicting 'Failed' and 'Completed' classes with minimal error.

The exploration of the confidence score, in tandem with other evaluation metrics, offers a deeper dive into understanding model confidence. The fig.11 shows the distribution plot for both the classes on the test data. For the 'Completed' class, we see a concentrated cluster of high confidence scores near the upper end of the scale, signifying a strong conviction from the model about its predictions of job completion. However, for the 'Failed' class, while there is a presence of high confidence scores, the distribution suggests a broader range of

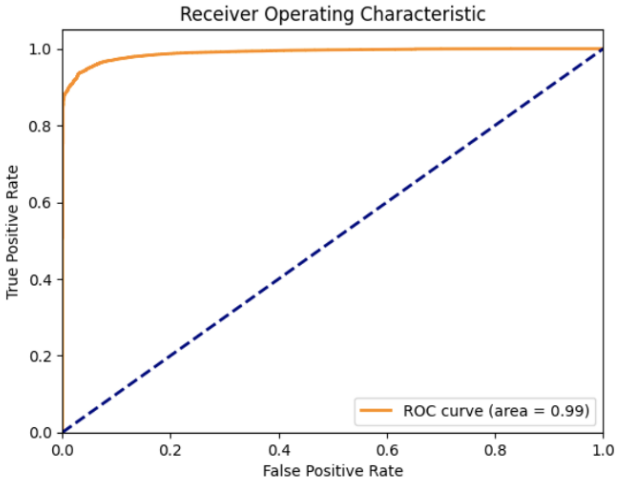

Fig. 10. ROC curve (AUC = 0.99) demonstrating the job failure prediction model's near-perfect discrimination ability between 'Failed' and 'Completed' classes.

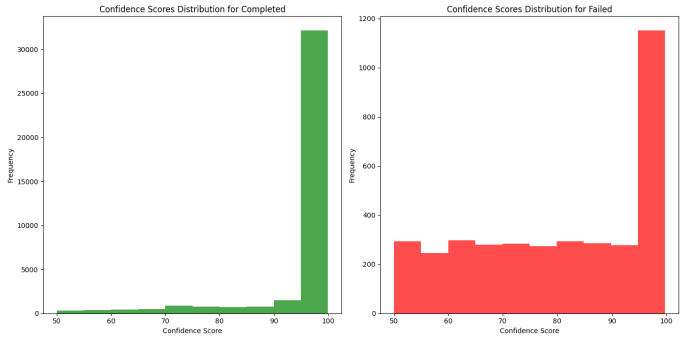

Fig. 11. Distribution plots for the 'Failed' and 'Completed' classes offering a visual examination of the model's predictions.

confidence, indicating room for improvement in the model's certainty when predicting failures.

The model in use has a preset threshold of 0.5 for its predictions. This means that predictions with a confidence score above this threshold are classified as 'Completed', and those below as 'Failed'. Adjusting this threshold can alter the model's sensitivity to predicting one class over another. The selection of an optimal threshold is case-dependent; it should be tailored to the unique needs and error tolerance of the specific application. By fine-tuning the threshold, we can strike a balance that maintains the model's accuracy and enhances the trust in its predictions, especially when it is applied to new, unseen data in real-world situations. Such calibration is crucial to ensure that the model provides actionable insights and assists in informed decision-making processes.

XI. User-Focused Interface

Our predictive modeling and data preprocessing libraries are designed to be highly versatile and easily integrated into various environments, enabling users to gain real-time insights into job performance and make data-driven decisions. The modular nature of our libraries allows for seamless integration into interactive dashboards, Jupyter notebooks for exploratory data analysis, or existing machine learning workflows for specialized tasks. In this section, we demonstrate two primary use cases:

- Functioning User Interface (UI) for Real-time Job Analytics: We showcase a user-friendly and interactive dashboard that harnesses the power of our predictive models to provide real-time visibility into job performance and predictions. The dashboard features intuitive visualizations, allowing users to monitor job progress, identify potential issues, and make informed decisions to optimize resource allocation and minimize failures. Through a typical user journey, we highlight how the dashboard empowers users to proactively manage their job submissions and improve overall system efficiency.
- Jupyter Notebook Integration for Extensibility: In addition to the dashboard, we demonstrate how our libraries can be seamlessly integrated into Jupyter notebooks, providing users with a flexible and interactive environment for conducting further exploratory data analysis (EDA). This integration showcases the extensibility of our libraries, enabling users to adapt and customize the analysis to suit their specific requirements and uncover additional insights.

A. Functioning User Interface (UI) for Real-time Job Analytics

Fig.12 shows the 'Job Prediction Dashboard' that integrates raw data, real-time predictions, feature insights, and job profiles, creating a comprehensive and actionable view of job performance. Let's walk through a typical user journey to understand how the dashboard empowers users to monitor, analyze, and optimize their job submissions.

Upon opening the dashboard, users can see an overview table presenting a comprehensive list of jobs, and the predicted outcome (Complete/Fail) with a confidence score. This real-time snapshot allows users to quickly assess the current state of the system and anticipate potential job failures.

The dashboard provides two primary modes of interaction. In the overview mode, users can explore the entire landscape of running jobs, gaining a high-level perspective of the system's performance. Alternatively, the specific job search mode enables users to dive deep into the details of a particular job by searching for its unique identifier. This targeted approach allows users to focus on jobs of special interest and investigate the factors influencing their predicted outcomes. Selecting a job from the table triggers an update in the 'Insights into Feature Contributions' section, revealing a ranked list of job attributes that most significantly impact the prediction. This explainable interface sheds light on the model's decision-making process, helping users understand why a particular job might be prone to failure. The 'Job Profiles' section adds another layer of interpretability by allowing users to understand their selected job in relation to broader job archetypes. By identifying the job profile that aligns with their job's

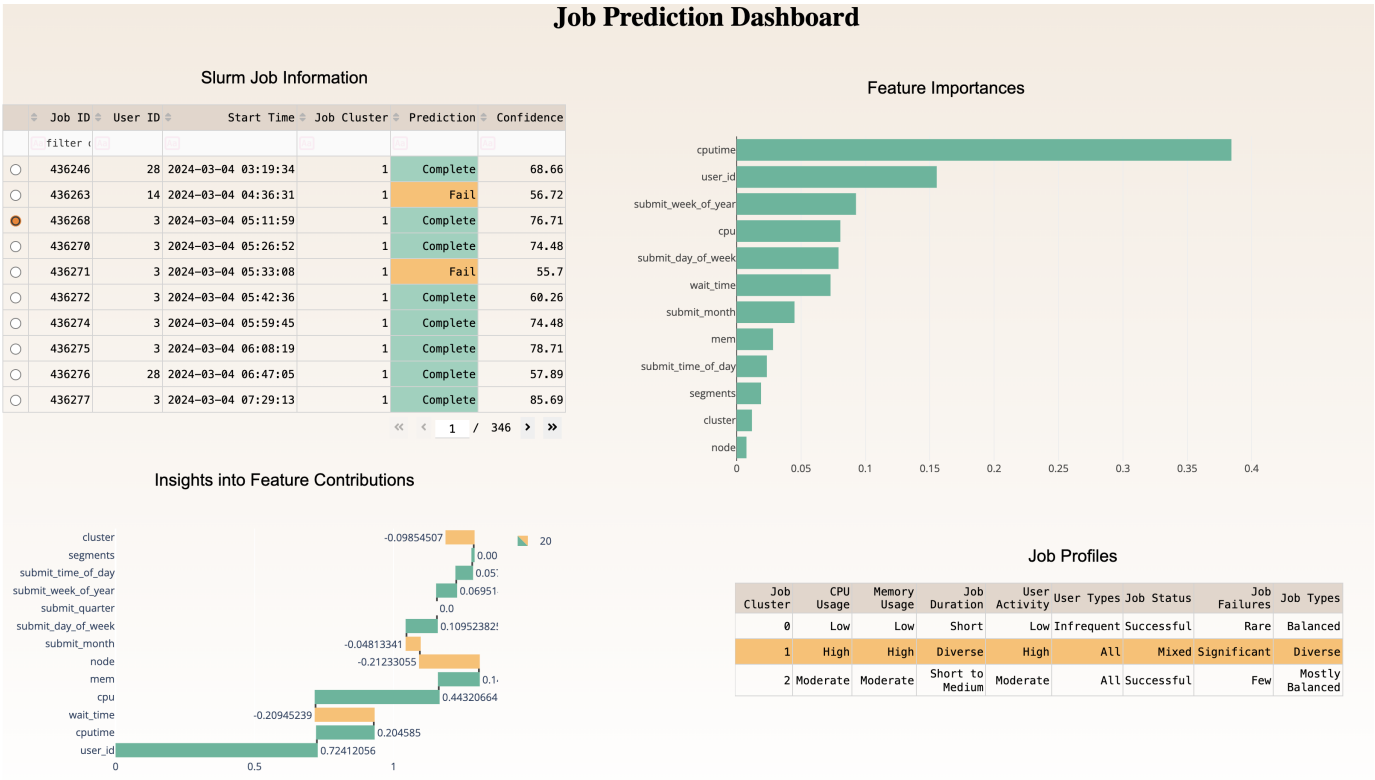

Fig. 12. Interactive dashboard showcasing real-time job failure predictions facilitating immediate analysis and decision-making.

characteristics, users can compare its attributes to the typical behavior of similar jobs, gaining insights into complexity, resource usage, and user frequency patterns.

When a job is predicted to fail with high confidence, users can leverage the insights from feature contributions, feature importance and job profiles to pinpoint the root causes. For example, a job with an unusually long wait time compared to its assigned job profile might be flagged as a potential failure risk. Armed with this knowledge, users can take proactive measures to mitigate failure risks. The real-time nature of the dashboard allows users to make informed decisions on the fly. They can choose to let the job continue running or cancel it proactively based on the prediction results and confidence scores. If a job is deemed likely to fail, users can investigate the contributing factors and make necessary adjustments, such as modifying resource requests or rectifying errors in the job script. Upon resubmission, the dashboard dynamically updates its predictions, providing instant feedback on the effectiveness of the modifications.

Over time, users can track the evolution of their job failure rates and analyze trends to identify recurring patterns. By correlating job features and job profiles with failure propensity, users can continuously optimize their job submission strategies, leading to enhanced system performance and reduced wastage of computational resources.

B. Jupyter Notebook Integration for Extensibility

In addition to the user-friendly dashboard, our predictive modeling and data preprocessing libraries seamlessly integrate with Jupyter notebooks, offering an immersive and interactive environment for EDA and collaborative insights.

The integration of our libraries with Jupyter notebooks also enhances the reproducibility and transparency of the analysis. By encapsulating the entire workflow, from data preprocessing to model training and evaluation, within a notebook, users can ensure that their results are fully reproducible. This is particularly important in the context of job failure prediction, where the ability to replicate and validate findings is crucial for building trust and confidence in the predictive models.

The following figures depict a user journey for interacting with a job failure prediction system. The workflow consists of five main steps, guiding users through the process of loading data, preprocessing it, applying a pretrained model, and obtaining insights into specific jobs.

As shown in fig.13 and fig.14, the user begins by loading the dataset required for prediction. Next, they prepare the data using preprocessing techniques to ensure its suitability for the machine learning model. The user then loads the pretrained model, which has been developed and optimized for job failure prediction. With the model ready, the user can initiate the prediction process on the new data, identifying potential job failures(fig.15). The final step, 'Get insights into your job', allows users to dive deeper into the factors influencing the

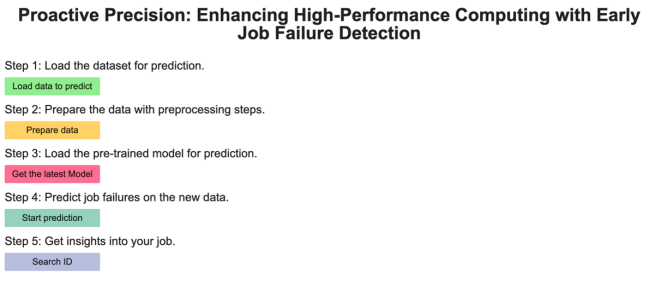

Fig. 13. An interactive workflow in Jupyter notebook environment consists of five main steps, guiding users through the process of loading data, preprocessing it, applying a pretrained model, and obtaining insights into specific jobs.

	job_id	job_name	user_id	start_time	end_time	state_jobs	exit_code	gid	submit	cputime	cpu	mem	node
0	436241	lu	160150495	2024-03-04 02:41:35	2024-03-04 02:42:35	COMPLETED	0:0	12790	2024-03-04T02:41:35	30720	512	892.0	2
1	436242	lu	160150495	2024-03-04 02:55:37	2024-03-04 02:55:55	COMPLETED	0:0	12790	2024-03-04T02:55:36	9216	512	892.0	2
2	436243	lu	160150495	2024-03-04 03:03:49	2024-03-04 03:04:07	COMPLETED	0:0	12790	2024-03-04T03:03:49	9216	512	892.0	2
3	436244	lu	160150495	2024-03-04 03:11:34	2024-03-04 03:12:22	COMPLETED	0:0	12790	2024-03-04T03:11:33	24576	512	892.0	2
4	436245	poisson	160150495	2024-03-04 03:14:02	2024-03-04 03:14:50	COMPLETED	0:0	12790	2024-03-04T03:14:01	24576	512	892.0	2

Fig. 14. This figure demonstrates the ability to explore job data prior to executing the prediction model, allowing for data exploration within the Jupyter notebook.

Step 3: Load the pre-trained model for prediction.

Get the latest Model

```
Model loaded.
```

Step 4: Predict job failures on the new data.

Start prediction

	Job ID	User ID	Start Time	Prediction	Confidence
0	436246	28	2024-03-04 03:19:34	Complete	68.660000
1	436263	14	2024-03-04 04:36:31	Fail	56.720000
2	436268	3	2024-03-04 05:11:59	Complete	76.710000
3	436270	3	2024-03-04 05:26:52	Complete	74.480000
4	436271	3	2024-03-04 05:33:08	Fail	55.700000
5	436272	3	2024-03-04 05:42:36	Complete	60.260000
6	436274	3	2024-03-04 05:59:45	Complete	74.480000
7	436275	3	2024-03-04 06:08:19	Complete	78.710000
8	436276	28	2024-03-04 06:47:05	Complete	57.890000
9	436277	3	2024-03-04 07:29:13	Complete	85.690000

```
Prediction done.
```

Fig. 15. This figure illustrates the model's identification of probable job failures within the dataset.

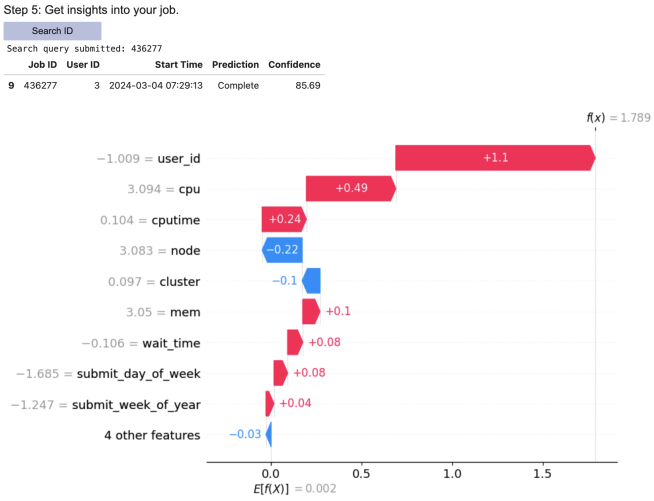

Fig. 16. This SHAP plot clarifies how individual features contribute to the model's reasoning behind each job's likelihood of failure or success, detailing the factors influencing its predictions.

prediction for a specific job. Fig.16 demonstrates this step-in detail. By entering a job ID, such as 436277, the user can retrieve comprehensive insights about that job. Furthermore, the system presents a SHAP (Shapley Additive Explanations) plot, often referred to as a waterfall plot, which provides valuable insights into the prediction for a particular instance, highlighting the key contributing factors that influence the prediction result. The SHAP plot visualizes the positive and negative impacts of each feature on the prediction, allowing users to understand which job characteristics are driving the model's decision.

The high SHAP value for the 'user_id' feature, combined with its top ranking in the feature importance list(fig.12), strongly suggests that the user's historical job performance and behavior have a substantial impact on job success. Moreover, the high SHAP values for 'CPU' and 'CPU time' indicate that for this particular job, the amount of CPU allocated and the duration of CPU time taken are key contributing factors to its successful outcome as predicted by the model. These features have a significant positive impact, suggesting that they are crucial in the model's consideration of what influences success for this job instance. Similarly, the negative value for 'node' and 'cluster' suggests that the number of nodes allocated and job profile characteristics of this jobs have a minor positive impact on job success. Gathering these insights, along with an understanding of feature importance and the actual values these features hold, can yield valuable revelations. Such comprehensive knowledge enables the initiation of proactive measures. These measures can improve the launch process of jobs, effectively helping to avoid potential failures and promoting a higher success rate in future job executions.

XII. Future Work

Building upon the current success of our job failure prediction model and user interface, we have identified several key areas for future research and development to further enhance the capabilities and impact of our system.

Firstly, we plan to explore and incorporate advanced machine learning techniques and algorithms to improve the accuracy and efficiency of our predictive model. This may include investigating deep learning architectures and transfer learning approaches to leverage knowledge from related domains. By staying at the forefront of machine learning research, we aim to continuously refine our model's performance and adapt to the evolving needs of HPC workloads.

To further enhance the predictive capabilities of our model, expanding the range and diversity of data sources is crucial. We intend to integrate data from system logs, network performance metrics, and application-specific telemetry to provide a more comprehensive view of the factors influencing job failures. By incorporating these additional data points, we can capture subtle patterns and dependencies that may not be apparent from job characteristics alone, enabling more accurate and nuanced predictions.

As HPC systems continue to grow and become more complex, scalability and adaptability become critical considerations. We plan to conduct extensive testing and evaluation of our model's performance across a range of HPC system scales, from small research clusters to large-scale supercomputers. By analyzing the model's behavior and resource consumption under different workload conditions, we can identify potential bottlenecks and optimize its performance for seamless integration into diverse HPC infrastructures.

In addition to the technical advancements, we also recognize the importance of continuously improving the user experience. While our current user interface provides a solid foundation for interacting with the job failure prediction system, we aim to gather feedback from the user community to identify areas for enhancement. This may include incorporating customizable dashboards, advanced filtering options, and personalized recommendations. By considering user preferences and domain-specific requirements, we can create a more intuitive and tailored user experience that helps users to effectively leverage the insights provided by our system.

By focusing on these key areas of advanced techniques, additional data sources, scalability, and user interface enhancements, we aim to drive the future development of our job failure prediction system. Through continuous innovation, we strive to empower HPC users with proactive insights and tools to optimize their workflows, minimize job failures, and maximize the efficiency of their computing resources.

XIII. Conclusion

In conclusion, our research aimed to address the critical challenge of early job failure detection in HPC systems. By leveraging the Slurm workload manager and a comprehensive dataset from a 1027-node HPC system, we developed a predictive model capable of predicting job failures within two minutes of submission. Our methodology involved rigorous data preprocessing, feature engineering, and the application of both supervised and unsupervised learning techniques. We addressed the class imbalance problem prevalent in HPC job data by employing active learning and oversampling strategies, significantly enhancing the model's predictive performance. The evaluation of our model using precision, recall, and F1 score metrics demonstrated its high effectiveness in accurately identifying potential job failures. The integration of our predictive framework into Jupyter notebook environment and an interactive dashboard for real-time insights ensures accessibility and ease of use for HPC practitioners. The insights generated by our model enable proactive decision-making, resource optimization, and improved system efficiency. Our work lays the foundation for future advancements in HPC job management, including the incorporation of additional data sources and the development of intelligent job submission recommendation systems. By empowering HPC stakeholders with predictive insights, our research contributes to the overall goal of minimizing downtime, maximizing resource utilization, and enhancing the productivity of HPC systems.

XIV. Acknowledgments

We would like to express our sincere gratitude to the Jupyter community for their invaluable contributions and resources that have greatly facilitated our research. The Jupyter notebook environment has been an essential tool in our work, providing a seamless and interactive platform for data exploration, analysis, and model development.

We also extend our appreciation to the Dash Plotly community for their exceptional work in creating a powerful and intuitive framework for building interactive dashboards. The extensive documentation and examples provided by the Dash Plotly community have been crucial in our development of the user interface and visualization components of our job failure prediction system. Their dedication to creating a user-friendly and feature-rich library has greatly enhanced the usability and visual appeal of our dashboard.

Additionally, we would like to acknowledge the Slurm Workload Manager community for their significant contributions to the field of high-performance computing. The Slurm Workload Manager has been a critical component in our research, providing a robust and scalable framework for job scheduling and resource management.

Furthermore, we would like to express our gratitude to the Python community for their extensive contributions to the field of data science and machine learning. The ideation and implementation of our job failure prediction system have been primarily carried out using the Python programming language. The rich ecosystem of libraries, frameworks, and tools provided by the Python community has been crucial in our development process. From data manipulation and preprocessing to model training and evaluation, Python has been an integral part of our workflow. We are thankful for the dedicated efforts of the Python community in creating and maintaining these valuable resources.

We are particularly grateful to the scikit-learn community for their role in the development of our predictive models. Scikit-learn has offered us a comprehensive suite of simple and efficient tools for data mining and data analysis, which are accessible to everybody and reusable in various contexts. The library's consistent and easy-to-use interface has significantly streamlined our machine learning pipeline, from model selection and training to evaluation.

Moreover, we would like to recognize the open-source community as a whole for their tireless efforts in developing and maintaining the numerous libraries, frameworks, and tools that have been integral to our research. The availability of these resources has accelerated our progress and enabled us to focus on the core aspects of our work.

Lastly, we would like to thank our colleagues and collaborators for their valuable insights, feedback, and support throughout this research endeavor. Their contributions have been essential in shaping and refining our approach to job failure prediction in high-performance computing environments.

Without the collective efforts and resources provided by the Jupyter community, Dash Plotly community, scikit-learn, Slurm Workload Manager community, and our colleagues, this work would not have been possible. We deeply appreciate the guidance and help received throughout the development and writing phases of this project.

REFERENCES

[1] Anupong Banjongkan, Watthana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop, "A Study of Job Failure Prediction at Job Submit-State and Job Start-State in High-Performance Computing System: Using Decision Tree Algorithms," *Journal of Advances in Information Technology*, Vol. 12, No. 2, pp. 84-92, May 2021. doi: 10.12720/jait.12.2.84-92

[2] Francesco Antici, Andrea Borghesi, and Zeynep Kiziltan, "Online Job Failure Prediction in an HPC System," *Journal of Advances in Information Technology*, Vol. 12, No. 2, pp. 84-92, May 2021. doi: 10.12720/jait.12.2.84-92

[3] Haotong Zhang, Gang Xian, Wenxiang Yang, and Jie Yu, "A Study of Job Failure Prediction on Supercomputers with Application Semantic Enhancement," *Journal of Computing Science and Engineering*, Vol. 16, No. 4, pp. 222-232, December 2022. doi: 10.5626/JCSE.2022.16.4.222

[4] Longfang Zhou, Xiaorong Zhang, Wenxiang Yang, Yongguo Han, Fang Wang, Yadong Wu, and Jie Yu, "Predicting Job Failure on Supercomputers with Job Path and User Behavior," *Journal of Advances in Information Technology*, Vol. 12, No. 2, pp. 84-92, May 2021. doi: 10.12720/jait.12.2.84-92

[5] Ju-Won Park, Xin Huang, and Chul-Ho Lee, "Analyzing and predicting job failures from HPC system log," *Journal of Supercomputing*, Vol. 80, pp. 435-462, 2024. doi: 10.1007/s11227-023-05482-y

[6] Tanash M, Dunn B, Andresen D, Hsu W, Yang H, Okanlawon A. Improving HPC System Performance by Predicting Job Resources via Supervised Machine Learning. PEARC19 (2019). 2019 Jul;2019:69. doi: 10.1145/3332186.3333041. Epub 2019 Jul 28. PMID: 35308798; PMCID: PMC8932944.