Augmenting ML-based Predictive Modelling with NLP to Forecast a Job's Power Consumption

Francesco Antici
University of Bologna
Bologna, Italy

Keiji Yamamoto
RIKEN Center for Computational Science
Kobe, Japan

Jens Domke
RIKEN Center for Computational Science
Kobe, Japan

Zeynep Kiziltan
University of Bologna
Bologna, Italy

ABSTRACT

As modern High-Performance Computing (HPC) reach exascale performance, their power consumption becomes a serious threat to environmental and energy sustainability. Efficient power management in HPC systems is crucial for optimizing workload management, reducing operational costs, and promoting environmental sustainability. Accurate prediction of job power consumption plays an important role in achieving such goals. In this paper, we apply a technique combining Machine Learning (ML) algorithms with Natural Language Processing (NLP) tools to predict job power consumption. The solution is able to predict job *maximum* and *average* power consumption per node, leveraging only information which is available at the time of job submission. The prediction is performed in an *online* fashion, and we validate the approach using batch system logs extracted from Supercomputer Fugaku, hosted at the RIKEN Center for Computational Science, in Japan. The experimental evaluation shows promising results of outperforming classical technique while obtaining an R2 score of more than 0.53 for our two prediction tasks.

KEYWORDS

High-performance computing, power consumption, workload analysis, energy efficiency, data analysis, natural language processing and machine learning

ACM Reference Format:
Francesco Antici, Keiji Yamamoto, Jens Domke, and Zeynep Kiziltan. 2023. Augmenting ML-based Predictive Modelling with NLP to Forecast a Job's Power Consumption. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3624062.3624264

1 INTRODUCTION

High-performance computing (HPC) systems have emerged as pivotal infrastructure for executing complex and computationally demanding tasks across various domains. Latest HPC systems have reached exascale performance, and in the future more systems are expected to have similar characteristics [8]. Their power requirements have become a significant concern due to the steady increase of overall system power, electricity costs, and negative environmental impact from carbon emissions. Thus, developing efficient strategies for power resource management in HPC systems is needed to optimize the power consumption and performance of the system.

This challenge can be addressed at workload level, by predicting power consumption of HPC jobs prior to their execution on the system. Such information can be exploited to compute the system power consumption beforehand, enabling better workload management decisions. Prior work [2, 4, 5, 14] leverages on job power prediction models to infer power awareness in workload management strategies, such as power capping and workload scheduling. Hence, the need for accurate prediction models for job power consumption gained prominence in the HPC community.

In our previous work[1], we presented a methodology to perform online workload features prediction in an HPC system. We leveraged NLP tools to extract more meaningful job information from the textual data of the job, and we showed that the combination of ML and NLP techniques outperformed classical models in the context of job failure prediction (defined as a binary classification task). Moreover, we explained that predictive models in HPC systems are required to work in an online context, where jobs with different characteristics are continuously submitted to the system. The results of our experimental evaluation showed that it is fundamental that the models are re-trained and updated to adapt to the change of workload of the system, aiming to guarantee optimal prediction accuracy over time.

In this work, we propose the following contributions. First, we explain how to modify our original algorithm to create a pipeline for the prediction of job power consumption. The task is here defined as a regression problem, since we target the estimation of a mapping between the job workload manager data and its power consumption value. Second, we enhance our algorithm with new ML models, and we compare them to related work (i.e., traditional ML approaches which had been designed specifically for this task). Third, the prediction models are validated on a new dataset, namely the job traces extracted from the Supercomputer Fugaku. Finally, we present an online job power prediction algorithm which is able to efficiently predict two different targets, namely the maximum and average job power consumption.

SC-W 2023, November 12–17, 2023, Denver, CO, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624264

The rest of the paper is organized as follows. We first introduce the data and the models used in our experiments in Section 2. Then, we describe in Section 3 the methodology used to perform online prediction and how we apply it to the new task and the new data. Finally, we present our experimental evaluation (Section 4) and conclusions (Section 6).

2 BACKGROUND

In this section, we present the data used in this study and the models selected for the power consumption prediction tasks.

2.1 Fugaku Traces

Supercomputer Fugaku. The data used in this study is extracted from the operational log data of Fugaku, an exascale supercomputer hosted at RIKEN Center for Computational Science[1], in Japan. The system was deployed in 2020, and, at the time of writing, is ranked second in the TOP500 list[2] of the most powerful supercomputers in the world. A summary of the system characteristics of Fugaku is reported in Table 1.

System characteristic	Description
Architecture	Armv8.2-A SVE 512 bit
OS	Red Hat Enterprise Linux 8
#Nodes	158.976
#Cores	48 + 2 assistant cores (per node)
Memory	HBM2, 32 GiB (per node)
Peak Performance	≈ 537 Pflop/s (FP64)
Internal Network	Tofu D Interconnect (28 Gbps)

Table 1: Fugaku system architecture.

Fugaku relies on a proprietary operations management software[3] which includes job management components, such as job manager and job scheduler. The software supports logging functions that allow for the collection of information on the job execution after its ending.

Job data. In this work, we consider the traces collected between January and March 2022, which contain the data of 1.5 million of jobs. We do that because we consider the amount of data in the sample already suitable for our experiments.

The job data comprises features related to the job submit-time, run-time and end-time. The first category contains the information available when a job is submitted, such as *submission time, requested resources, user information and system state.* To the second category belongs all the information about the job launch, such as waiting time, execution start time, and the actual number of allocated resources. At job termination, the end-time features are collected, such as ending time, duration, outcome of the execution, and power consumption values. The full list of features of the job is reported in the GitHub repository.[4]

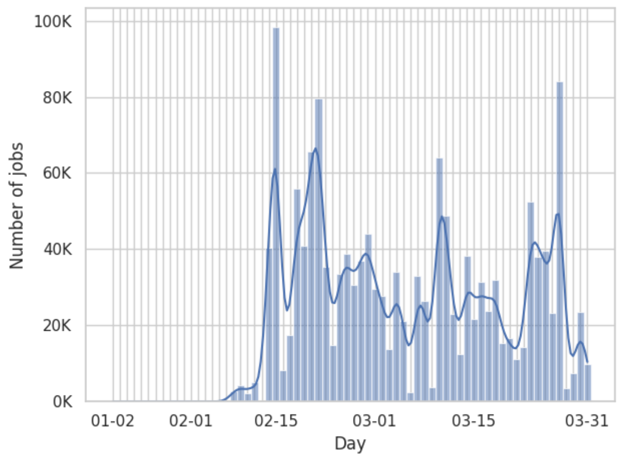

Figure 1: Distribution of job submission date.

In Figure 1, we show the distribution of the number of jobs (y-axis) submitted per day (x-axis), and its Kernel Density Estimate (KDE). The KDE (blue solid line) represents the probability density function estimation of the job submission per day, and is plotted to ease the outlining of patterns or seasonality in the distribution. However, neither seasonality nor uniformity is present in the data. This characteristic is common in HPC systems, since their loads can change for several reasons. For instance, the difference in load between weekdays and weekend[5], scheduled maintenance, or occasional dedication of large portion of the system to particular tasks.[6] Moreover, not all the jobs actually executed are kept in the final dataset, due to errors in the data collection process, missing values or wrongly formatted data.

Furthermore, we plot the distribution of the nodes allocated per job in Figure 2, along with its KDE (solid blue line). The plot shows that the majority (960K) of the jobs present in our dataset run on a single node, while very few jobs use a large portion of the system nodes. This reveals that our data is composed mainly of small-scale jobs, and it confirms that large-scale executions are an exception to the normal behaviour of the system. Nevertheless, we also have records of such large-scale jobs (the maximum of nodes allocated to a job is 152064), showing that our data is comprehensive of the different loads of the system.

For each job j, the data contains the power consumption of the different resources of the system (processor, core, node, memory) allocated to the job j during its execution. Fugaku's job manager prevents the sharing of nodes among multiple jobs. Therefore, the values of power consumption depend only on job j execution and there are no interferences caused by other jobs' execution (besides the typically shared resources: storage and network).

For the purposes of this work, we define the job power consumption as the power consumption of the job j recorded at node level. We do that because the node power consumption includes the

[1]https://www.riken.jp/en/
[2]https://www.top500.org
[3]https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article10.html#cap-03
[4]https://github.com/francescoantici/job-pcon-predictor/blob/main/documentation/job_features.md
[5]The jobs submitted the 02-15 (Tuesday) are almost 100K, while the jobs submitted the 02-12 (Saturday) and 02-13 (Sunday) are close to zero.
[6]https://www.hpci-office.jp/en/using_hpci/proposal_submission_past/r03_fugaku_full-scale

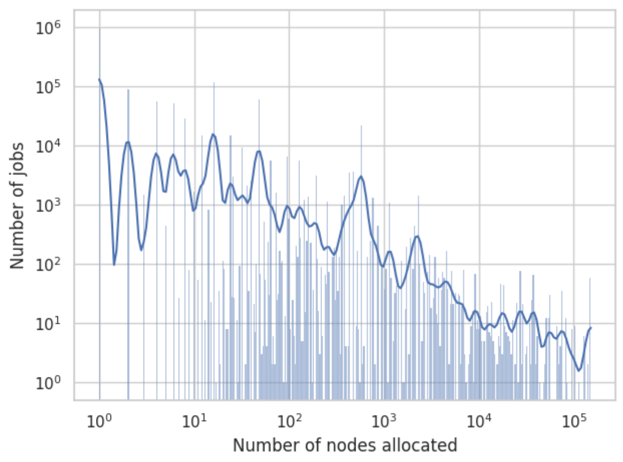

Figure 2: Distribution of the number of nodes allocated per job.

power consumption of all the other resources of the node (processors, cores and memory). In the original data, each job j information contains three node power consumption features, namely $minpcon_j$, $avgpcon_j$, and $maxpcon_j$. These are the minimum ($minpcon_j$), average ($avgpcon_j$) and maximum ($maxpcon_j$) value of the sum of the power consumption of all the nodes allocated to the job during its execution. Each of these values is a single number, measured in Watts.

2.2 Regression and Language Models

We approach the job power prediction task as a regression problem, exploiting supervised ML techniques for regression, along with a pre-trained state-of-the-art NLP model to represent jobs features as input to the model.

As for the ML algorithms, we consider widely adopted ensemble methods, such as Random Forest, XGBoost and Adaptive Boosting. Ensemble methods are particularly well suited for ML tasks, since they exploit multiple learning algorithms to obtain better predictive performance than any of the constituent learning algorithms alone[11].

Random Forest. Random Forest (RF) [6] is a popular ML algorithm used for both classification and regression tasks. It belongs to the ensemble learning family, and it is based on creating a diverse set of Decision Tree (DT) regressors by introducing randomness in each DT construction. Each DT is constructed independently using a random subset of the training data and a random subset of the input features. As discussed in [6], individual DTs typically exhibit high variance and tend to overfit, so the aim of the RF ensemble method is to remove the error by taking an average of the predictions of the single DTs.

XGBoost. XGBoost (XG) [9], is a gradient boost algorithm designed for several prediction tasks, such as classification and regression. XG relies on an ensemble of predictive models (usually DTs), with each model attempting to correct the mistakes made by the other ones. Leveraging on Gradient-based optimization, XG aims to optimize a specific loss function defined for the task. Moreover, XG includes regularization techniques to control model complexity and prevent overfitting. Due to such characteristics, XG excels in handling large-scale datasets, capturing complex interactions among features and delivering high predictive performance, as discussed in [9].

Adaptive Boosting. Adaptive Boosting (AD) [12], is a popular ML algorithm combining several prediction models to obtain better prediction performances. It is particularly effective in binary classification problems, but can also be extended to multi-class classification and regression tasks. During the training, AD computes a weight to assign to each model by analysing its prediction performance. In this phase, wrongly predicted instances are valued more than correct ones, making difficult instances more influential in subsequent iterations. The final prediction is delivered through a majority voting, weighted accordingly to the weights computed in the training phase.

Sentence BERT. Sentence-BERT (SBert) [15] is a language model that leverages pre-trained transformer-based models, typically variants of BERT (Bidirectional Encoder Representations from Transformers) [10], to generate numerical vector representations (embeddings) for sentences or short text segments. Unlike traditional BERT, which was primarily designed for word-level contextual embeddings, SBert focuses on obtaining semantically meaningful sentence embeddings. As discussed in [15], the key idea behind SBert is to fine-tune pretrained BERT models on specific tasks related to sentence similarity or sentence-level tasks. By doing so, BERT becomes capable of producing meaningful sentence embeddings that capture the meaning and context of entire sentences or pieces of text. The final representation of a string of text produced by SBert is a fixed-size 384-dimensional floating-point array, which can be used as input to train models for specific sentence-level tasks.

3 METHODOLOGY

In this section, we describe how we modify and apply our previously published methodology [1] to a new task, namely the job power consumption prediction of the jobs in the Fugaku dataset. First, we describe how we prepare the data for the prediction tasks, presenting the features used to represent the jobs and the job power consumption values used as target. Then, we present the original methodology used to perform the job feature encoding and how we apply it to the Fugaku data.

We describe the two algorithms used to evaluate the models in the original work, namely *offline* and *online*. In the first setting, the training and testing of the models are performed just once on fixed splits of data, while, in the second, the model is retrained periodically on the data of a fixed recent period, and it is tested on a future one coming close in time.

3.1 Data Preparation

Starting from the original job data presented in Section 2.1, we perform some data engineering steps to isolate the information we need to perform the prediction.

Name	Description	Type
Job type	Category of the job (Batch, Step, etc.)	String
User	User name	String
Group	Name of the group the user belongs to	String
User id	ID of the user submitting the job	Integer
Group id	Group of the user submitting the job	Integer
Frequency	Requested frequency of the processor	Int
Job name	Name of the job	String
Host name	Name of the host node	String
Priority	The priority assigned to the job	Int
#Cores Requested	The number of cores requested	Int
#Nodes Requested	The number of nodes requested	Int
Arrival time	Time of job submission	Timestamp
Memory size limit	The limit to the memory size allocated	Int
Time limit	Maximum allowed run time in minutes	Integer
Environment	The set of environment variable	String

Table 2: Job features description.

Feature selection. In order to describe the characteristics of a job in a prediction task, we need to associate it with certain features. As in [1], we can only rely on job submit-time features, since our goal is to predict the power consumption in advance. Such features are the information available when a job is submitted, hence that can be retrieved without further modification to the normal workload submission workflow. Moreover, this allows to compute a prediction before the job is queued, staged, and executed. After analysing the full set of job features available in the dataset, we filter the submit-time ones. The final list of submit-time features of the job are listed in Table 2, along with their description.

As explained in [1], in HPC production systems, users tend to submit jobs in batches containing similar experiments. Jobs submitted in the same batch are prone to have similar names, characteristics and perform similar operations. Given that the power consumption of a job depends on the computational operations it performs, jobs performing the same or similar operations will have similar power consumption. Therefore, features like the user name, job name and environment variables, might be the key to identify similar jobs, and consequently, perform accurate job power consumption prediction.

In an initial experimental phase, we evaluate models' prediction performance on a smaller sample of the data, using different subsets and combinations of the features presented in Table 2. We observed that the use of particular subsets of features to represent a job is beneficial for both prediction performance[7] and computation time, since the number of features to encode is smaller. The subset of features which yields the best predictive performance is composed of the following features, *user name, job name, # cores requested, # nodes requested* and *environment.* Therefore, we decide to use such features to represent each job in our dataset.

Job power consumption. The prediction tasks require the definition of a prediction target for the training phase of the models. In this work, we focus on the maximum and average job power consumption. As explained in Section 2.1, both this information is present in each job j data, namely in the $maxpcon_j$ and $avgpcon_j$ features. The original power consumption values range from few to millions of Watts, depending on the resources allocated to the job. This makes the prediction task very hard and the possible relative prediction error very high. In light of that, we decide to perform some data pre-processing to make the target more suitable for the regression task, as outlined hereafter.

We analyse the power traces of the nodes allocated to the jobs which run on multiple nodes. The analysis reveals that, during the job j execution, there is a small difference between the power consumption values of the different nodes allocated to the job ($nodes\_allocated_j$). Thus, for each $n \in nodes\_allocated_j$, its power consumption is well approximated by the average of the power consumption of all the nodes n in $nodes\_allocated_j$. This allows us to predict each job power consumption as if it was running on a single node, making the prediction task less error-prone. The final job power consumption values used in the prediction tasks are defined as the average of $maxpcon_j$ (p_max) and $avgpcon_j$ (p_avg) per node, as shown in Equations 1 and 2.

$$p\_max_j = \frac{maxpcon_j}{\#nodes\_allocated_j} \tag{1}$$

$$p\_avg_j = \frac{avgpcon_j}{\#nodes\_allocated_j} \tag{2}$$

In order to gain more insights on the data for the prediction phase, we plot the distribution of p_max_j and p_avg_j in Figure 3 and 4. We do that also to show that the two values provide different information, thus needing two different prediction tasks. The values range from a minimum of 24 W (for p_max_j) and 20W (for p_avg_j), to a maximum of 200 W for both power values. The average values for p_max_j and p_avg_j are, 97 W and 90 W. Nevertheless, the majority of the values are around 110 W for p_avg_j, while, for p_max_j, the values are shifted towards the 120W. We can observe that in both cases, there is a predominance of jobs in two power consumption bands, with the first being less than 50W per node, and the second being in the range 110-130 W. The values in between the two power bands are more uniformly distributed, while the jobs consuming very high amount of power ($\geq$ 130W) are in a very limited number with respect to the others.

This analysis shows that p_max_j and p_avg_j are indeed different, and predicting them defining two different tasks is necessary to obtain reliable results.

3.2 Job Power Consumption Prediction

We describe the methodology for the prediction as presented in [1], focusing on the job features encoding and on the models' training and testing algorithm.

Feature encoding. In order to compute a prediction for a job, we need to convert the job data into the correct format to feed into the regression models presented in Section 2.2. We achieve that by relying on job feature values, and we use the two different encodings proposed in [1], namely Integer encoding (INT) and SBert based (SB). In the first (INT), an integer is assigned to the values which are not numerical, i.e. *user name, job name*, and *job environment*, while setting all the missing values in the other fields to a default value of -1. In the second encoding (SB), first all the

[7]The investigation of this phenomenon is still ongoing and outside the scope of this work, we will provide a thorough study on the feature selection process in future work.

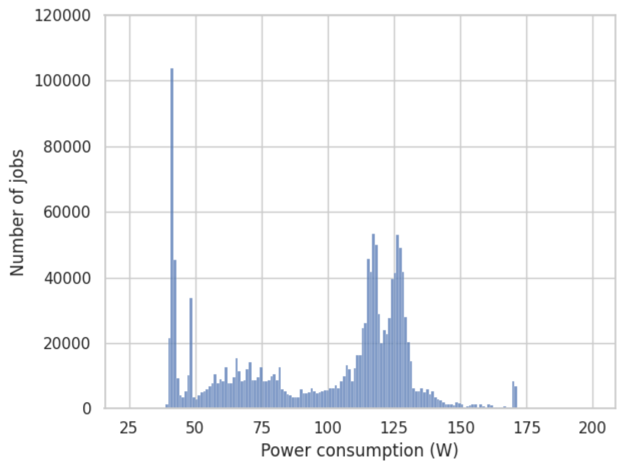

Figure 3: Distribution of p_max_j.

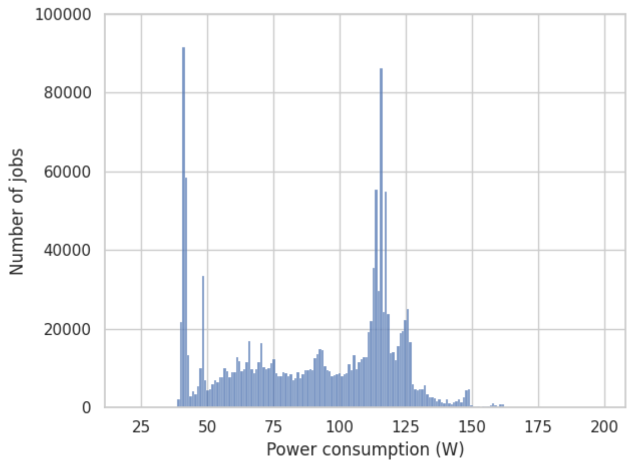

Figure 4: Distribution of p_avg_j.

feature values are concatenated into a comma-separated string, e.g. *user1, job1, 1, 1, env1*. Then the string is encoded with SBert, obtaining a 384-dimensional floating-point array.

Following the hypothesis presented in [1], we believe that with SBert we can extract more fine-grained insights about job features expressed in natural language (e.g. *user and job name*). This is because SBert is designed to represent sequences, with semantically similar contents, with similar encodings. As we discussed in Section 3.1, jobs with similar names and users could belong to the same submission batch running similar operations, therefore, such features could reveal important patterns on the nature of the job and its workload. This is hard to recognize with the INT encoding, since similar natural language values will be mapped to different integer values, while they would have similar representation in SB, due to semantic similarity.

Models training and testing. As explained in [1], it is not realistic to do inference on a job by learning from the data of the future jobs submitted at a later time. This is because in a real system, job data can be evaluated and collected correctly only after the job ends and the data collection process finishes successfully. Thus, whenever a job is submitted, we can use only the data of the jobs which are already finished to perform the learning phase.

For the training and testing of the models, we use the same settings presented in [1], namely *offline* and *online*. In both settings, the data are ordered chronologically based on their submission time, thus ensuring that the training data always comes chronologically before the test set one.

The first setting is the *offline*, where the job data are considered as a whole, training the model once on one portion of it, and testing it using the data of the other portion in chronological order. To do this, the jobs are split into two, using the first split preceding in time as the training set, and the other as the test set.

The second setting is defined as *online*. As mentioned earlier, this is more suitable to our context, since the job data is treated as live and streaming in time. The model is re-trained periodically every ω days on the data of the last α days. It is then tested it on the job data of the future ω days. To guarantee soundness of the setting, we use the time information provided by the *end_time* of the jobs to ensure that all the jobs in the training split end before the submission of the jobs in the test set. We consider as the first training set all the jobs that were submitted in the first α days and not finished after the date of the first test set. Starting from the submission time of the first job not present in the first training set, we divide the data in batches in chronological order, where each batch contains the jobs submitted in the next ω days. We then iterate over each batch, considering it as a new test set. At every iteration, the training set is updated with the data of the last α days and the models are retrained.

As we discussed in Section 3.1, the workload of an HPC system is usually submitted in batches having similar characteristics, therefore the workload of an HPC submitted close in time can have similar characteristics, while those may change completely at a later time. Our experimental results in Section 4 confirm that the retraining of the model on data close in time (*online* setting) significantly improves the prediction performance with respect to a single training over more timely distant data (*offline* setting).

The same set of experiments is performed for predicting both the p_max_j and p_avg_j value of a job. From here on, we refer to the prediction of the p_max_j and p_avg_j as the *maximum* and *average* task.

4 EXPERIMENTAL STUDY

In this section, we present the experimental setting and the results of the tests conducted for the study.

4.1 Experimental Setting

We run our experiments on a machine equipped with two AMD EPYC 7302 CPUs with 64 cores and 512 GB of RAM.

The RF and AD algorithms are implemented with the *scikit-learn*[8] Python library, while the XG implementation is retrieved

[8]https://scikit-learn.org/stable/

from the *xgboost*[9] library. The sequence encoder model is provided by the *sentence transformers* library[10], while the weights for SBERT are pulled from huggingface.[11] We use the pre-trained model *all-MiniLM-L6-v2*[12], since it is the best trade-off between prediction performance and speed [15]. All the models are instantiated with the default setting provided by the libraries. The implementation and the details regarding the Python version and its packages are available in a GitHub repository.[13]

Job power prediction. We evaluate the models for the prediction of job power consumption on several metrics typically used in regression tasks, namely Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Coefficient of Determination (R2).

For the *offline* setting, the parameters are set as in [1], thus we take the first 70% of the data as the training set and the remaining 30% as the test set. Concerning the *online*, we experiment with three different values for the α, namely 15, 30 and 60. We observe that, for the Fugaku data, the best prediction performance is obtained with $\alpha = 60$. Therefore, differently from our previous work, we fix the training interval α to 60 (30 in the past work) days, while we keep $\omega = 1$ day. The training/testing step of the models is performed only if the data splits defined by α and ω contain more than 1 element each, otherwise, we move on to the following splits. We are able to compute 16 iteration over the data in the *online* settings, corresponding to the job data submitted between the 16^{th} and the 31^{st} of March 2022. Therefore, the results reported in Table 5 and 6 are the average of the results of the 16 tests.

For the evaluation phase, we distinguish between the job feature encodings (INT and SB) and the supervised algorithms (AD, XG, RF). Each regression algorithm is evaluated using the two feature encodings, and are compared with two simple baselines predicting constant values, namely c_max and c_avg. These baselines always predict the same value, ignoring the input feature values (thus we don't need to distinguish between INT and SB encoding since the input features are not considered for the prediction). The c_max always predicts the maximum value among the power consumption of the jobs in the training set, while the c_avg always predicts the average one.

System power prediction. Furthermore, after testing the models at job level, we want to evaluate the prediction performances at system level. We do that for two main reasons, (i) to see if our models are capable of reconstructing the system power state accurately, thus providing a tool which is able to predict the power consumption of a whole system by considering only the job submitted, and (ii) because the prediction error on a single job is either an overestimate or an underestimate of the actual power consumption, so we believe that such errors might cancel out each other at system level, given the large amount of jobs running concurrently.

We estimate the real system global power consumption for each hour ($psys_h$) of each day of the *online* testing phase. We start by computing the power consumption of the system for a single minute, by summing the power consumption of all the jobs $j \in J_{h,m}$ running concurrently on the system between the minutes m and $m+1$ of the hour h. The hourly power consumption of the system is then computed by taking the average of its power consumption for each minute of the hour, as shown in Equation 3 and 4.

$$psys\_avg_h = \frac{\sum_{m=1}^{60} \sum_{j \in J_{h,m}} p\_avg_j}{60} \tag{3}$$

$$psys\_max_h = \frac{\sum_{m=1}^{60} \sum_{j \in J_{h,m}} p\_max_j}{60} \tag{4}$$

The same methodology is used to compute the predicted system power, by replacing the true job power consumption values with the predicted ones.

4.2 Results

After computing all the experiments, we analyse the results obtained. First, we present the power prediction performance of the models in the *offline* and *online* setting. Finally, we show how well our models perform at system level.

Offline job power prediction. Tables 3 and 4 report the results of the *offline* experiments. These show that the SB+XG model is the best-performing model for both the *maximum* and *average* task, obtaining a RMSE of 29.6 and 25.8, respectively. It is noticeable how the use of the SB encoding improves the performances of all the models. For instance, on the RF model, the R2 score goes from -0.15 (INT+RF) to 0.23 (SB+RF), confirming that a meaningful representation of textual features improves the prediction performance. The two baselines employed for the prediction tasks obtain poor results in both the *average* and *maximum* tasks. While the const_max baseline obtains the worst results for the prediction task, the const_avg baseline performs similarly to models trained with the INT encoding. These results highlight the difficulty of the prediction task and the importance of leveraging on a trained model to perform the prediction.

Online job power prediction. In Tables 5 and 6 we show the prediction performance of the models in the *online* setting. Coherently with the *offline* setting, the use of the SB encoding improves all the models performances significantly. For instance, the RMSE score of the INT+RF model is almost twice the one of SB+RF for both the *maximum* and *average* task. Moreover, the R2 score of all the models increases significantly with the SB encoding, going from values lower than zero to greater than 0.50 (INT+XG and SB+XG). In this setting, the models obtaining better results are the SB+XG and SB+RF. In the *maximum* task, both the models obtain the same MAPE of 27%, but the SB+XG reaches slightly better results in terms of MAE (18.70 vs 18.88), MSE (631.63 vs 655.32), RMSE (25.13 vs 25.59) and R2 (0.57 vs 0.55). Conversely, on the *average* task, the trend is reversed. Indeed, the SB+RF model obtain better results in terms of MAE (16.80 vs 17.14) and MSE (557.65 vs 557.81), while the MAPE (0.26), RMSE (23.61) and R2 (0.53) scores are equivalent.

The comparison of the results of the *online* and *offline* setting outlines that retraining the models is beneficial for prediction performance. The most noticeable enhancement is obtained in terms of the R2 score. Indeed, the score increases by 0.16 (*maximum* task) and

[9]https://xgboost.readthedocs.io/en/stable/index.html
[10]https://www.sbert.net
[11]https://huggingface.co
[12]https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2
[13]https://github.com/francescoantici/job-pcon-predictor

Model	MAE (W)	MAPE (W)	MSE (W)	RMSE(W)	R2
INT+AD	33.43	0.53	1372.86	37.05	0.05
INT+XG	35.00	0.55	1643.22	40.05	-0.14
INT+RF	39.64	0.59	2029.66	45.05	-0.41
SB+AD	33.41	0.52	1337.30	36.56	0.07
SB+XG	**22.70**	**0.37**	**844.84**	**29.06**	**0.41**
SB+RF	28.15	0.50	1368.74	37.00	0.05
c_max	110.42	1.80	13632.96	116.76	-8.46
c_avg	33.93	0.59	1564.92	39.55	-0.09

Table 3: Results in the offline setting for the *maximum* power consumption task. For MAE, MAPE, MSE and RMSE metrics the lower, the better, while for R2 score the higher, the better. Best results are highlighted in bold.

Model	MAE (W)	MAPE (W)	MSE (W)	RMSE (W)	R2
INT+AD	33.37	0.52	1346.27	36.69	0.08
INT+XG	37.75	0.41	2207.05	46.97	-0.50
INT+RF	43.01	0.57	2464.04	49.63	-0.68
SB+AD	28.74	0.44	1008.74	31.76	0.31
SB+XG	**18.70**	**0.27**	**631.63**	**25.13**	**0.57**
SB+RF	18.88	**0.27**	655.32	25.59	0.55
c_max	110.36	1.81	13647.47	116.82	-8.29
c_avg	34.21	0.59	1573.37	39.66	-0.07

Table 5: Results in the online setting for the *maximum* power consumption task. For MAE, MAPE, MSE and RMSE metrics the lower, the better, while for R2 score the higher, the better. Best results are highlighted in bold.

Model	MAE (W)	MAPE (W)	MSE (W)	RMSE (W)	R2
INT+AD	30.30	0.52	1161.94	34.1	0.0
INT+XG	25.16	0.38	886.54	29.77	0.24
INT+RF	31.76	0.44	1346.77	36.70	-0.15
SB+AD	29.72	0.51	1112.15	33.35	0.05
SB+XG	**19.90**	**0.32**	**666.05**	**25.80**	**0.43**
SB+RF	23.47	0.42	895.98	29.93	0.23
c_max	116.58	1.95	14758.07	121.48	-11.65
c_avg	31.18	0.55	1288.17	35.90	-0.10

Table 4: Results in the offline setting for the *average* power consumption task. For MAE, MAPE, MSE and RMSE metrics the lower, the better, while for R2 score the higher, the better. Best results are highlighted in bold.

Model	MAE (W)	MAPE (W)	MSE (W)	RMSE (W)	R2
INT+AD	30.29	0.51	1129.74	33.61	0.05
INT+XG	33.15	0.42	1519.56	38.98	-0.28
INT+RF	33.48	0.39	1667.66	40.83	-0.41
SB+AD	27.37	0.45	925.28	30.41	0.22
SB+XG	17.14	**0.26**	557.81	**23.61**	**0.53**
SB+RF	**16.80**	**0.26**	**557.65**	**23.61**	**0.53**
c_max	116.50	1.96	14756.10	121.47	-11.47
c_avg	31.22	0.55	1281.47	35.79	-0.08

Table 6: Results in the online setting for the *average* power consumption task. For MAE, MAPE, MSE and RMSE metrics the lower, the better, while for R2 score the higher, the better. Best results are highlighted in bold.

0.10 (*average* task) points. Based on the definition of the R2 metric, this represents a significant improvement in the model capability of capturing characteristics and variation of the target values.

We plot Figures 5, 7, 6 and 8 to visualize how much the models are able to capture the variability of the target values. We create the figures performing the following steps. First, we group all the job data belonging to all the test splits together in a list; then we sort them in ascending order by their actual power consumption values. We further split the sorted job data in 30[14] batches, with each batch identifying a range of true power consumption values. Finally, for each batch, we show the true (solid blue line) and the predicted (whisker) power consumption values of the jobs belonging to it. This allows us to understand how the model performs on the different ranges of job power consumption values.

We plot the results of the RF and XG models, both with the INT and SB encodings, to check that the improvement in the results presented in Tables 5 and 6 actually corresponds to a better reconstruction of the target values. As expected, the distribution of the ranges is better approximated by the models with the SB encoding, in both the *maximum* and *average* tasks. The models with the INT encoding (Figures 5 and 6) tend to always predict the same values regardless of the target, while the SB encoding (Figures 7 and 8) seems to make the models more flexible and adaptable to the different distributions. Even though the prediction performance of the models is improved with the SB encoding, there is still evidence of power values that the models struggle in predicting[15]. For instance, the extreme values of the ranges, both for the *maximum* and *average* task. For the case of the jobs consuming low power, the absolute prediction error is quite small, given the low power values. Concerning the jobs consuming very high levels of power, we can observe by the power distribution plots in Figures 3 and 4 that such values are the least numerous in the dataset. Therefore, since their occurrence is very rare, it is very hard for the models to learn patterns to predict their values.

System power prediction. Figure 9 and 10 present the system-level evaluation of the best models for the *maximum* and *average* task in the online setting, namely SB+XG and SB+RF. The period of time represented in the plots is the testing period of the *online* setting, which concerns the jobs submitted between the 16^{th} and the 31^{st} of March 2022 (Section 4.1). We plot the figures to show how well the total true power consumption of the jobs (blue line) is approximated by the predictions of our models (orange line). The results confirm the hypothesis formulated in 4.1, i.e. that the prediction performance is improved at system level (MAPE of around

[14]The number of splits is decided after an empirical evaluation to guarantee readability of the plots.

[15]In Figure 8 there is an outlier in the predictions in correspondence to the true power values between 115 and 120 Watts. The reasons for this are still under investigation.

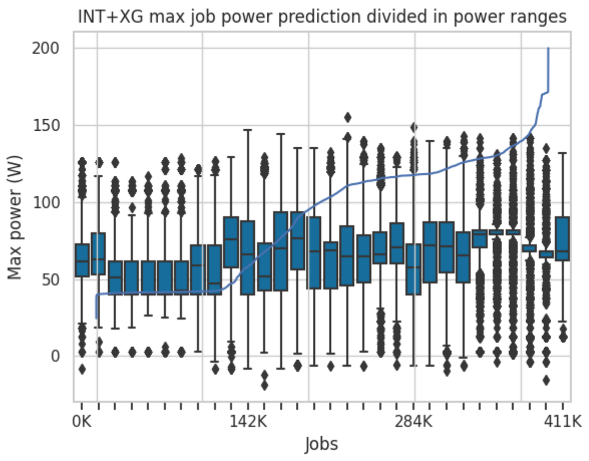

Figure 5: Job true and the predicted power for the *maximum* target with the INT+XG model. The x-axis represents the number of jobs in the test set.

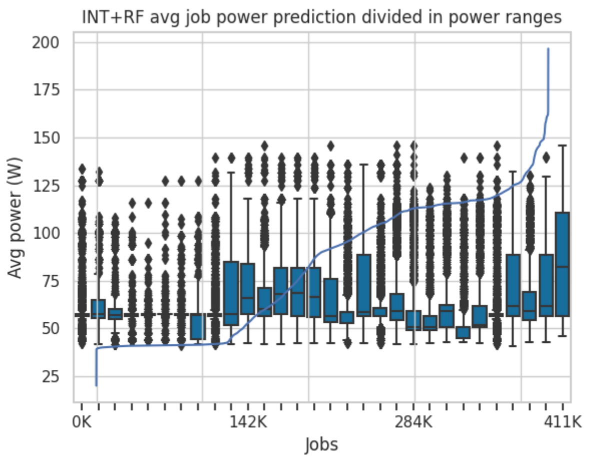

Figure 6: Job true and the predicted power for the *average* target with the INT+RF model. The x-axis represents the number of jobs in the test set.

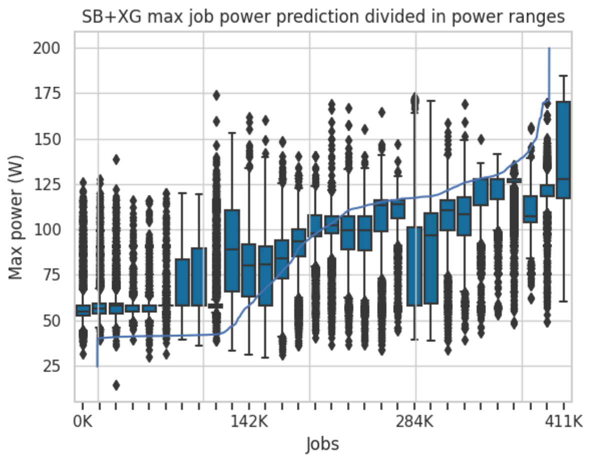

Figure 7: Job true and the predicted power for the *maximum* target with the SB+XG model. The x-axis represents the number of jobs in the test set.

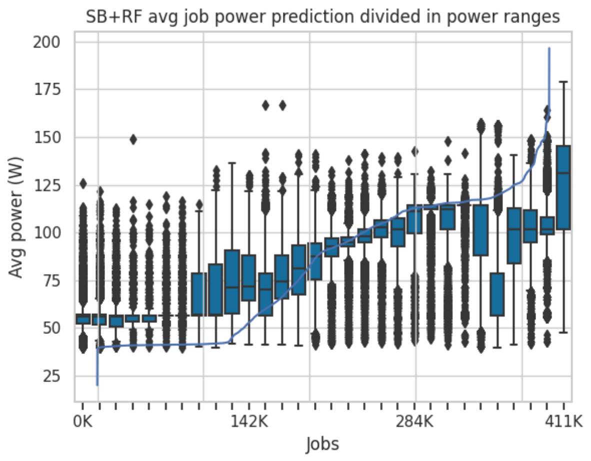

Figure 8: Job true and the predicted power for the *average* target with the SB+RF model. The x-axis represents the number of jobs in the test set.

5% and a R2 > 0.97 for both tasks) with respect to the job level. This can be caused by several factors. For instance, the predicted job power consumption prediction might be either an overestimate, or an underestimate of the actual values. These can balance each other out when summing the power consumption of the concurrent jobs, obtaining a better approximation of the overall power consumption.

In this scenario, it is important to specify that the reconstruction is an estimate of the systems' power consumption, which does not match with the actual power consumption of Fugaku. This is because we don't have access to other power consumption sources of the system (e.g. idle node power) and there are some missing information in our data which prevents us to reconstruct the exact load of the system through time, as explained in Section 2.1. Hence, instead of seeing the Fugaku-typical constant load of about 19MW[16], we are seeing an increasing curve. Nevertheless, this plot shows that for the available data, our prediction model nearly matches the actual load of the system in Spring of 2022.

Finally, in Figure 11 and 12 we report the average training time for all the models employed in the experiments with the INT and SB encoding. In each heatmap we show the average, maximum and minimum training time of the model throughout the 16 days of the testing of the *online* setting. We do that to investigate if the training time has a very high variability through time, which would represent a non-predictable behaviour of our algorithm. The figures show that our models' training time does not present a significant

[16]https://status.fugaku.r-ccs.riken.jp/

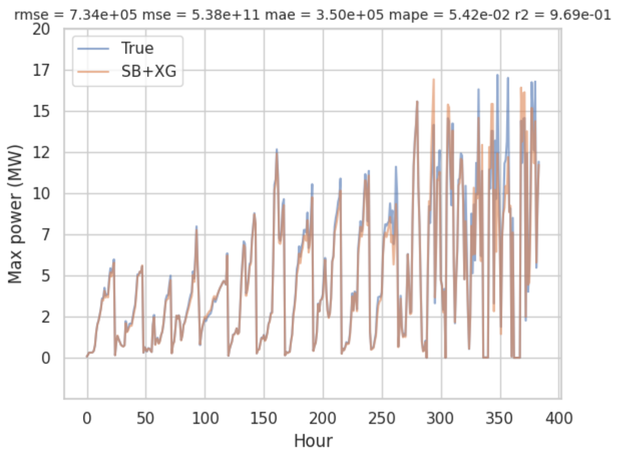

Figure 9: System true and predicted power for the *maximum* task with the SB+XG model.

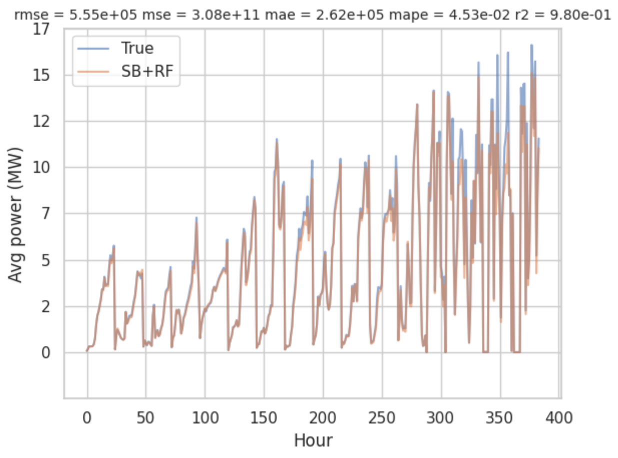

Figure 10: System true and predicted power for the *average* task with the SB+RF model.

dissimilarity between the minimum, average and maximum value. The difference is justified by the variability of the number of job data in the training splits, which range from a minimum of 1038542 jobs to a maximum of 1481037 (mean value of 1261219).

For the case of the INT encoding, the training time is almost negligible for the RF and XG models, while it is around 2 minutes for the AD.[17] As introduced in Section 3, the SB encoding maps the input to a 384 dimensional vector, with respect to the 5 dimensional of the INT representation. This step introduces a significant complexity to the model, since the feature space of the SB encoding is almost 80 times bigger than the INT one. RF relies on single features values correlation with the target, so its computation time is heavily influenced by the dimensionality of the input data (training time more than 60 times bigger in the case of SB encoding). The AD model is the one obtaining worse results in terms of training time for the INT encoding (90 seconds), while it performs better than RF with the SB encoding($\sim$ 7000 seconds less in every case). The XG model is the most robust in terms of training time, since despite the high-dimensionality of the input with the SB encoding, it is able to perform the training in around 3 minutes for the worst case scenario, making it easily deployable to a real system.

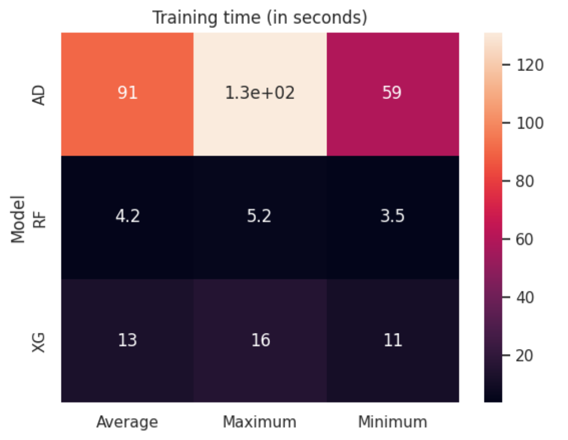

Figure 11: Models' training time with the INT encoding.

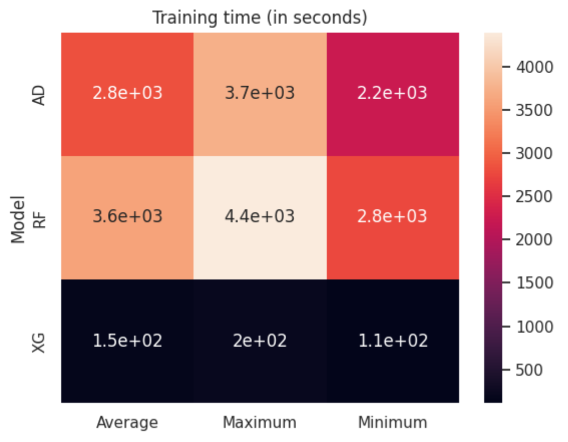

Figure 12: Models' training time with the SB encoding.

5 RELATED WORK

In the past, several works have explored techniques to estimate power consumption prediction of jobs executed on large-scale systems.

[17]Differently from the other two models, the scikit-learn API for the AdaBoostRegressor algorithm currently does not support the distribution of the training operations on multiple processors

Prior work, such as [7, 17], explored the use of workload manager information to perform power-related prediction on the execution of the job. In their work, differently from our approach, the job data is not limited to submission time, making the online prediction non-feasible.

In [3], the authors propose to predict job average power consumption per node by using an RF model trained on historical data, without relying on an online algorithm.

In [16] it is proposed an online algorithm to predict average job power consumption per node based on exponential smoothing of similar past jobs power consumption. This approach, though, relies only on categorical features of the jobs (*user id*, *group id*, *# tasks per node*), which are very general and less informative on the nature of the job, as shown by the results obtained in this work when using a language model to encode textual features.

Differently from those works, we propose to use a prediction algorithm which is both online and relies on NLP techniques to extract more meaningful insights from the job data. We apply the methodology proposed in [1], but to a new task and new dataset. Differently from [1], where the task is a binary classification problem, namely the job failure prediction, we apply the methodology to a regression task, i.e. the job power consumption prediction. Moreover, we validate the approach using different models and on different data with respect to [1].

Furthermore, differently from all the cited works on power prediction, we perform the prediction of average and maximum job power consumption values. Both values can be exploited for different power-aware scheduling strategies, such as power capping.

For instance, in [5] the authors propose a constraint-programming based dispatching strategy exploiting job power consumption to perform power-capping. In [14] the maximum power consumption caused by the execution of the job to the system is used to make informed decisions about the scheduling of the jobs, while works like [2, 4] rely on the average job power consumption to perform the same task. Moreover, the prediction of the maximum job power consumption could be used to address the power supply demand from the supercomputing center to the electricity company, as shown in [13], aiming to estimate in advance the power load required by the system.

6 CONCLUSIONS AND FUTURE WORK

In this work, we explored the application of a methodology for online workload features prediction, aiming to perform *maximum* and *average* job power consumption in an HPC system. We proposed a similar methodology in a past work, addressing the job failure prediction as a binary classification task. However, here we modified the methodology to be suitable to predict job power consumption, which we defined as a regression problem, and evaluated the methodology's applicability to Supercomputer Fugaku and its typical job mix.

As in [1], our experimental results confirmed that (i) the retraining of the model is more suited also for the prediction of job power consumption, and (ii) the use of an NLP-based encoding to represent job features improved the prediction performance of all the models employed in the evaluation phase. We outperformed classical techniques, obtaining a R2 score of 0.57 for the *maximum* task and 0.53 for the *average* one.

We described how to approximate the system's global power consumption by using the running job power consumption. Our models are able to accurately reconstruct the global power consumption of the system, obtaining a R2 score greater than 0.96 for both the *maximum* and *average* task. Moreover, we showed that the XG model has the potential to be integrated into an existing job manager pipeline of a large-scale system, since it provides accurate power predictions, while requiring short training time.

In future work, we want to study continuous learning techniques and investigate different retraining strategies. We plan to test our approach on different data from other systems, to evaluate its generality. Furthermore, we aim at integrating the models into a scheduling pipeline to exploit the output of the model for power-aware decision-making.

REFERENCES

[1] Francesco Antici, Andrea Borghesi, and Zeynep Kiziltan. 2023. Online Job Failure Prediction in an HPC System. In *Euro-Par 2023: Parallel Processing Workshops: Euro-Par 2023 International Workshops, Limassol, Cyprus, August 28–September 1, 2023, Revised Selected Papers*. Springer Nature.

[2] Deva Bodas, Justin Song, Murali Rajappa, and Andy Hoffman. 2014. Simple power-aware scheduler to limit power consumption by hpc system within a budget. In *2014 Energy Efficient Supercomputing Workshop*. IEEE, 21–30.

[3] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. 2016. Predictive modeling for job power consumption in HPC systems. In *High Performance Computing: 31st International Conference, ISC High Performance 2016, Frankfurt, Germany, June 19-23, 2016, Proceedings*. Springer, 181–199.

[4] Andrea Borghesi, Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. 2018. Scheduling-based power capping in high performance computing systems. *Sustainable Computing: Informatics and Systems* 19 (2018), 1–13.

[5] Andrea Borghesi, Francesca Collina, Michele Lombardi, Michela Milano, and Luca Benini. 2015. Power capping in high performance computing systems. In *Principles and Practice of Constraint Programming: 21st International Conference, CP 2015, Cork, Ireland, August 31–September 4, 2015, Proceedings 21*. Springer, 524–540.

[6] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[7] Bruce Bugbee, Caleb Phillips, Hilary Egan, Ryan Elmore, Kenny Gruchalla, and Avi Purkayastha. 2017. Prediction and characterization of application power use in a high-performance computing environment. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 10, 3 (2017), 155–165.

[8] Casas Guix et al. Carpenter, Paul Matthew. 2022. ETP4HPC's SRA 5 strategic research agenda for High-Performance Computing in Europe 2022: European HPC research priorities 2023-2027. (2022).

[9] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186.

[11] Manuel Fernández-Delgado, Manisha Sanjay Sirsat, Eva Cernadas, Sadi Alawadi, Senén Barro, and Manuel Febrero-Bande. 2019. An extensive experimental survey of regression methods. *Neural Networks* 111 (2019), 11–34.

[12] Yoav Freund, Robert Schapire, and Naoki Abe. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence* 14, 771-780 (1999), 1612.

[13] Hyunsoo Kim, Jiseok Jeong, and Changwan Kim. 2022. Daily Peak-Electricity-Demand Forecasting Based on Residual Long Short-Term Network. *Mathematics* 10, 23 (2022), 4486.

[14] Dineshkumar Rajagopal, Daniele Tafani, Yiannis Georgiou, David Glesser, and Michael Ott. 2017. A novel approach for job scheduling optimizations under power cap for arm and intel hpc systems. In *2017 IEEE 24th International Conference on High Performance Computing (HiPC)*. IEEE, 142–151.

[15] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[16] Théo Saillant, Jean-Christophe Weill, and Mathilde Mougeot. 2020. Predicting job power consumption based on rjms submission data in hpc systems. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35*. Springer, 63–82.

[17] Keiji Yamamoto, Yuichi Tsujita, and Atsuya Uno. 2018. Classifying jobs and predicting applications in HPC systems. In *High Performance Computing: 33rd International Conference, ISC High Performance 2018, Frankfurt, Germany, June 24-28, 2018, Proceedings 33*. Springer, 81–99.