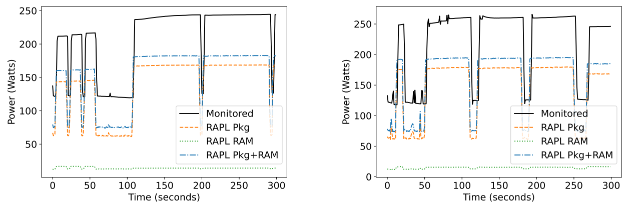
Table 3: Illustration of our hyper-parameter tuning using Grid Search. Tuned values are highlighted in bold.

XGBoost	SVR	Decision Tree	Random Forest	Neural Network
learning_rate: [0.01, **0.03**, 0.1, 0.5] n_estimators: [100, 200, 300, **900**] max_depth: [3, 5, **6**] min_child_weight: [1, **3**, 5] subsample: [**0.6**, 0.8, 1.0] colsample_bytree: [**0.6**, 0.8, 1.0]	C: [0.1, **1**, 10, 100] kernel: [linear, **rbf**, poly] gamma: [**scale**, auto, 0.01, 0.1, 1] epsilon: [0.1, 0.2, **0.5**, 1.0]	max_depth: [None, 5, 10, 15, **20**] min_samples_split: [**2**, 5, 10] min_samples_leaf: [1, **2**, 4] max_features: [auto, sqrt, **log2**] max_leaf_nodes: [None, **10**, 20, 30] min_impurity_decrease: [**0.0**, 0.1, 0.2]	n_estimators: [100, 200, **300**, 500] max_depth: [None, 5, 10, **20**, 30] min_samples_split: [**2**, 5, 10] min_samples_leaf: [1, **2**, 4] max_features: ['auto', **'sqrt'**, 'log2'] bootstrap: [**True**, False] max_leaf_nodes: [None, 10, 20, 50]	activation: [**ReLU**, ELU, tanh] solver: [adam, **sgd**] learning_rate_init: [0.001, 0.01, **0.1**, 1]

B APPENDIX: EMPIRICAL DATA SHOWING THE DIFFERENCE BETWEEN FULL-SERVER MONITORED POWER AND RAPL POWER VALUES



(a) Sysbench (b) Memcached

Figure 6: Power values reported by Intel RAPL [9] and the full-server power meter when running (a) Sysbench, and (b) Memcached.

While RAPL power values can serve as ground truth for CPU and/or DRAM power consumption, they are not accurate indicators of full-server power, as shown in Figure 6. We empirically show this shortcoming with Sysbench and Memcached workloads. In general, across workloads, we found that the sum of CPU package and DRAM power values for RAPL is 30–50% lower than full-server monitored power values. Further, within a workload execution, the ratio of full-server to RAPL power values varies significantly, by as much as 1.1–1.7× for the workloads we experimented with. This is to be expected as RAPL does not include power consumed by, for example, the disks, motherboard, network, or non-integrated GPU(s).

C APPENDIX: HYPER-PARAMETER TUNING

All ML models we experimented with were first tuned via Grid Search to identify the best hyper-parameter values. Table 3 shows the hyper-parameter tuning details for five ML models (XGBoost, SVR, DT, RF, and NN), along with the best values chosen. For XGBoost, parameters such as learning rate, number of estimators, and maximum depth were adjusted to find a balance between model complexity and accuracy. SVR tuning focused on the regularization parameter C, kernel type, kernel coefficient (gamma), and epsilon (for epsilon-SVR model). For kernel type, our experiments showed that the RBF (Radial Basis Function) kernel had the best prediction accuracy. RF and DT had similar tuning to prevent overfitting while maintaining model depth. For RF, we also considered the number of estimators (300 in our case) for better accuracy. For NN, we used the Multi-layer Perceptron regressor from scikit-learn with ReLU activation and three hidden layers; the three hidden layers had size of 512, 16, and 16 neurons. For LR, we experimented with and without intercept. Including intercept, as mentioned earlier, gave better results. We also tried Ridge regression as an alternative to LR and Lasso, but its prediction accuracy was worse (5–6% higher MAPE than LR and Lasso), so we do not include it in our evaluation. For regularization, since Ridge regression did not work well, we had the alpha hyper-parameter set to 0.1 for Lasso (L1 regularization).