Energy-Efficient Virtual Machine Consolidation via Advanced CPU Utilization Prediction Models: A Comparative Study of Machine Learning and Deep Learning Approaches

Md Sifatullah Sheikh
Student ID: 2022-1-60-029

Lamia Akter
Student ID: 2021-3-60-147

Antara Labiba
Student ID: 2021-3-60-173

Abir Hasan
Student ID: 2021-1-60-035

***Abstract*—**

***Index Terms*—Virtual Machine Consolidation, Energy Efficiency, CPU Utilization Prediction, Machine Learning, Deep Learning, Cloud Computing**

I. Introduction

The rapid growth of cloud services has significantly increased the number and scale of data centers worldwide. While cloud computing has improved accessibility, productivity, and computational power, its environmental impact has become a major concern. Data centers consume enormous amounts of electricity for computation, storage, and cooling, and this continuous energy demand contributes directly to carbon emissions. As a result, the idea of Green Cloud Computing has gained momentum, emphasizing energy-efficient, sustainable, and environmentally responsible cloud infrastructures.

Researchers have examined this topic from multiple perspectives. For example, Buyya et al. (2010) highlight that traditional cloud architectures were not originally designed with power efficiency in mind; their work proposes structured methods to reduce energy usage through smarter resource provisioning and workload distribution [1]. A comprehensive survey by Radu (2017) further shows that energy consumption, inefficient resource utilization, and cooling overhead remain persistent challenges, and categorizes existing solutions such as virtualization, VM scheduling, dynamic resource allocation, and renewable-energy-powered data centers [2].

Recent studies continue to expand this understanding. A 2022 systematic review points out that modern cloud workloads are highly dynamic and require adaptive energy-utilization strategies, including heuristic and machine-learning-based approaches for workload prediction and resource management [3]. Another paper (2025) discusses how sustainable IT infrastructure can be achieved through improved virtualization techniques, energy-aware scheduling, and intelligent consolidation mechanisms that reduce idle server energy without compromising performance [4]. Similarly, VM consolidation remains a key technique, as shown by research demonstrating that grouping workloads and migrating underutilized VMs can significantly reduce idle power consumption in cloud data centers [5]. Additional methods such as selecting energy-aware VMs [6] and using lightweight meta-heuristic optimization algorithms [7] further enhance energy efficiency by ensuring that computing resources are allocated only when needed.

Our work is aligned with these research directions. For our project, we used the dataset provided through Kaggle, which is focused on the theme of Green Computing (Domain 2: Green Cloud Computing). Although the dataset mainly supports analytical exploration rather than algorithmic modeling, it reinforces the importance of understanding sustainability and energy-efficiency strategies in cloud environments. By reviewing existing literature and examining relevant data, our objective is to present a clear and structured overview of how green cloud techniques can reduce power consumption, improve resource efficiency, and support the transition toward environmentally responsible cloud systems.

Overall, the collected research and our dataset-based analysis demonstrate that achieving green cloud computing is not dependent on a single technique but instead requires a combination of architectural improvements, intelligent scheduling, workload consolidation, and sustainable operational practices. These foundations will guide the rest of our report.

II. Related Background

This section summarizes the key research contributions in green cloud computing, highlighting approaches, datasets, main findings, and research objectives shown in Table I.

III. Methodology

This section details the experimental design, data preparation, model development, and evaluation procedures employed to assess energy-efficient VM consolidation. This involved a systematic approach to data preprocessing, model training, hyperparameter optimization, and a multi-faceted evaluation of both prediction accuracy and consolidation performance, shown in Fig 1.

TABLE I
SUMMARY OF RELATED RESEARCH ON GREEN CLOUD COMPUTING

Ref.	Approach	Dataset	Main Contribution	Research Aim
[1]	Energy-efficient cloud architecture	Cloud data-center simulation (CloudSim)	Proposes architecture and policies for energy-aware resource provisioning, including dynamic resource allocation and workload modeling.	Reduce power consumption in large-scale cloud data centers.
[2]	Survey of green cloud techniques	Review of over 120 research papers	Categorizes solutions such as scheduling, cooling, VM consolidation, and renewable energy integration; provides a systematic literature review.	Identify achievements, challenges, and research gaps in green cloud computing.
[3]	Energy utilization optimization	Analysis of cloud energy models (2012–2022)	Summarizes recent strategies including machine learning, heuristics, and metaheuristics; presents a taxonomy and comparative analysis.	Improve energy efficiency across heterogeneous cloud workloads.
[4]	Sustainable cloud operations	Industry and academic research	Highlights virtualization, energy-aware scheduling, and renewable energy integration; discusses resource management and scheduling techniques.	Achieve sustainable, low-carbon cloud infrastructure.
[5]	VM consolidation for energy savings	Real cloud workload traces	Demonstrates reduction of server idle power and operational costs via dynamic VM migration and consolidation algorithms.	Minimize energy waste and improve resource utilization.
[6]	Energy-aware VM selection	CloudSim experiments	Shows up to ∼19% energy reduction with optimized VM selection heuristics and resource monitoring.	Enhance energy efficiency while maintaining quality of service (QoS).
[7]	Metaheuristic resource allocation	Benchmark cloud workloads	Proposes a clonal selection algorithm that reduces makespan and energy cost in cloud scheduling.	Balance performance and energy consumption in cloud resource allocation.

A. Feature Engineering

A sliding window approach was employed to transform the time series data into supervised learning samples. For each VM trace, a history window of length $H = 10$ was used to predict the next time step's CPU utilization. This resulted in input-output pairs where the input features are the utilization values of the previous 10 time steps, and the output is the utilization at the next time step. The training and testing datasets were constructed by concatenating samples from all VMs within each subset.

B. Prediction Models

1) Naive Persistence Model: The Naive Persistence Model serves as a simple baseline for time series forecasting, where the predicted value at the next time step is assumed to be equal to the most recent observed value. Despite its simplicity, this model often provides a competitive benchmark in short-term forecasting tasks and is widely used to evaluate the relative performance of more complex models [8]. Its effectiveness stems from the temporal autocorrelation commonly present in CPU utilization data, where recent values are indicative of near-future behavior.

Formally, the prediction at time $t + 1$ is given by:

$$\hat{u}_{t+1} = u_t$$

where u_t is the observed utilization at time t.

2) K-Nearest Neighbors (KNN): The K-Nearest Neighbors (KNN) regression model is a non-parametric, instance-based learning algorithm that predicts the target value based on the weighted average of the k closest training samples in the feature space [9]. In this study, the distance-weighted KNN was employed, where closer neighbors have a greater influence on the prediction. This approach leverages the assumption that similar historical CPU utilization patterns are likely to yield similar future values.

Formally, the predicted value $\hat{y}$ for an input vector $\mathbf{x}$ is computed as:

$$\hat{y} = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i}$$

where y_i are the target values of the k nearest neighbors and $w_i = \frac{1}{d(\mathbf{x},\mathbf{x}_i)}$ is the weight inversely proportional to the distance d between $\mathbf{x}$ and neighbor $\mathbf{x}_i$.

3) Random Forest (RF): Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the average prediction of the individual trees to improve generalization and reduce overfitting [10]. It is well-suited for regression tasks involving complex, nonlinear relationships, such as CPU utilization forecasting. In this study, the Random Forest model was configured with 100 trees, a maximum depth of 15, and a minimum samples split of 10 to balance bias and variance.

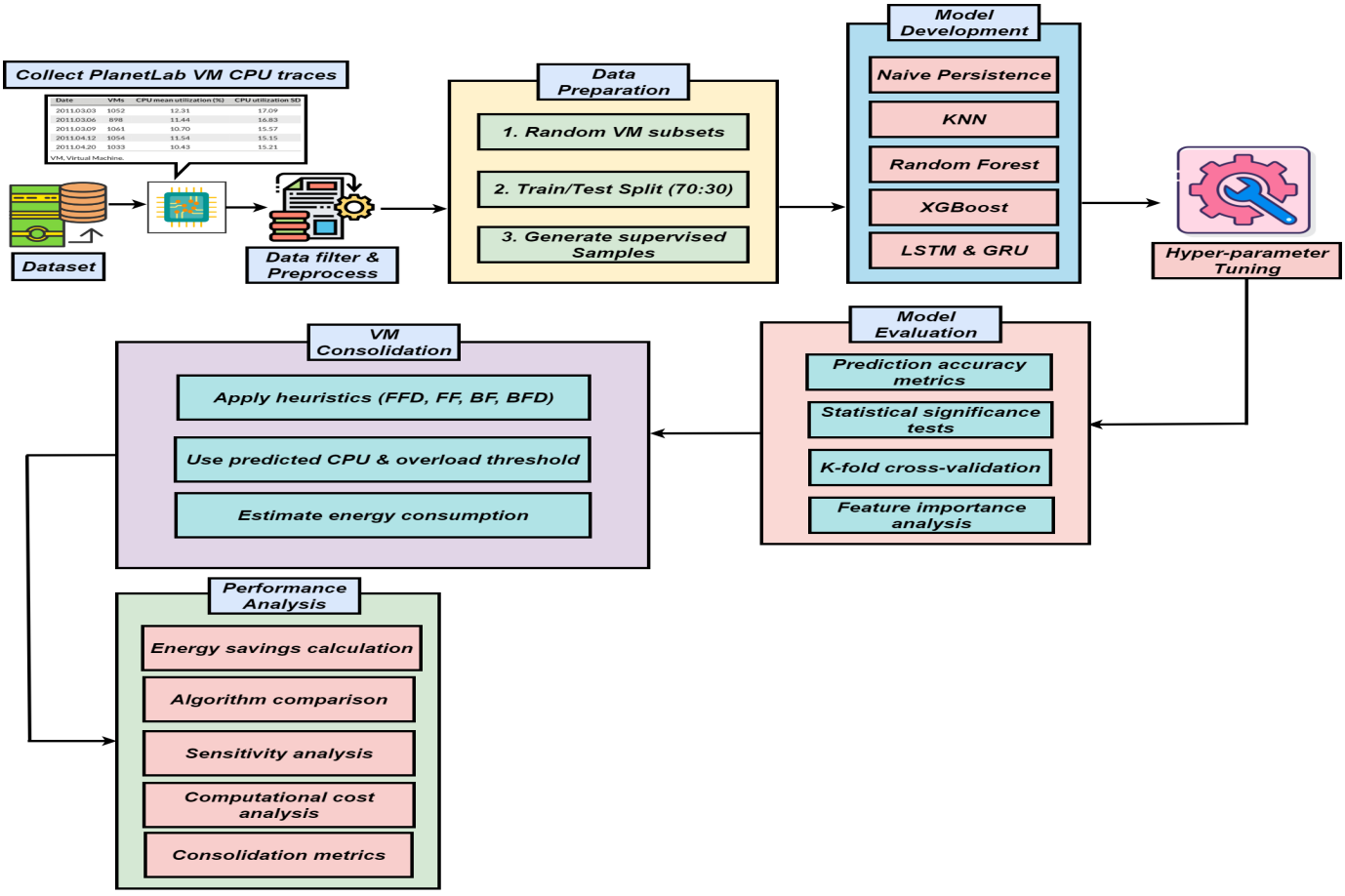

Fig. 1. Workflow Schematic Representation.

4) Extreme Gradient Boosting (XGBoost): XGBoost is a scalable and efficient implementation of gradient boosting machines that builds additive models in a forward stage-wise manner by optimizing a differentiable loss function [11]. It is known for its high predictive accuracy and ability to handle heterogeneous data. The model was employed with default parameters, including 100 estimators and a maximum tree depth of 6, with hyperparameter tuning applied to further optimize performance.

5) Deep Learning Models: LSTM and GRU: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are specialized recurrent neural network architectures designed to capture long-term dependencies in sequential data by mitigating the vanishing gradient problem [12], [13]. Both models consist of gated cells that regulate information flow, making them effective for time series forecasting tasks, such as predicting CPU utilization.

In this study, two-layer stacked LSTM and GRU networks with dropout regularization were implemented. Models were trained using the Adam optimizer with mean squared error loss, and early stopping was applied to prevent overfitting.

C. Model Training and Hyperparameter Tuning

Models were trained on the training sets of each subset. Hyperparameter tuning for KNN, RF, and XGB was performed using randomized search with time series cross-validation (three splits) to optimize mean absolute error (MAE). The search spaces included parameters such as number of neighbors for KNN, number of trees and depth for RF, and learning rate and max depth for XGB.

Deep learning architectures were explored by varying the number of units and dropout rates in LSTM and GRU layers. Models were trained for up to 50 epochs with early stopping based on validation loss.

D. Evaluation Metrics

Prediction performance was quantitatively assessed using multiple complementary metrics to capture different aspects of forecasting accuracy. These included Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Coefficient of Determination (R^2), and Mean Absolute Percentage Error (MAPE). MAE provides an interpretable average magnitude of errors, RMSE penalizes larger errors more heavily, R^2 measures the proportion of variance explained by the model, and MAPE expresses errors as relative percentages, which is useful

for understanding performance across varying utilization levels [14].

To statistically evaluate whether the improvements of advanced models over the naive persistence baseline were significant, the Wilcoxon signed-rank test was employed with a one-sided alternative hypothesis, suitable for paired, non-parametric data [15]. Additionally, Cohen's d effect size was calculated to quantify the practical significance of observed differences, providing insight into the magnitude of improvements beyond mere statistical significance [16].

E. VM Consolidation and Energy Modeling

Predicted CPU utilizations were used to perform VM consolidation via the First Fit Decreasing (FFD) bin-packing algorithm. VMs were sorted in descending order of predicted utilization and assigned to hosts without exceeding an overload threshold of 80%. The server power consumption was modeled linearly as:

$$P(u) = P_{\text{idle}} + (P_{\text{max}} - P_{\text{idle}}) \times \frac{u}{100}$$

where $P_{\text{idle}} = 162$ W and $P_{\text{max}} = 215$ W, and u is the host utilization percentage.

Consolidation performance metrics included the number of active hosts, average utilization, total energy consumption, SLA violations (hosts exceeding the overload threshold), and resource wastage.

F. Additional Analyses

Sensitivity analysis was conducted by varying the overload threshold from 70% to 90% to observe its impact on consolidation metrics. Alternative bin-packing algorithms—First Fit (FF), Best Fit (BF), and Best Fit Decreasing (BFD)—were compared against FFD to evaluate consolidation efficiency.

Computational costs of training and prediction were measured for all models, reporting both total and per-sample times to assess feasibility for real-time deployment.

All experiments were repeated across the three subsets to ensure robustness, and results were aggregated with mean and standard deviation reported.

IV. Experimental Setup

This study evaluates advanced prediction models for virtual machine (VM) CPU utilization forecasting to enhance energy-efficient VM consolidation. The experiments utilize PlanetLab workload traces [17], comprising normalized CPU utilization time series for multiple VMs. Only traces with a minimum length of 288 time steps were selected, and all traces were clipped to the range [0, 100] and standardized to a fixed length of 288 samples to ensure uniformity.

To assess model robustness, three random subsets of 30 VMs each were created by shuffling the dataset with a fixed random seed. Each subset was split into training and testing sets with a 70:30 ratio, preserving temporal order to maintain the integrity of time series data.

A sliding window approach with a history length of 10 time steps was employed to generate supervised learning samples, where the previous 10 CPU utilization values were used to predict the next time step.

The prediction models evaluated include a naive persistence baseline, classical machine learning models (K-Nearest Neighbors, Random Forest, and XGBoost), and deep learning architectures (LSTM and GRU). Hyperparameter tuning was performed using randomized search with time series cross-validation to optimize model performance.

VM consolidation was performed using the First Fit Decreasing (FFD) bin-packing algorithm with an overload threshold set at 80%. A linear power consumption model was applied to estimate server energy usage based on CPU utilization.

All experiments involving deep learning models were conducted using a Tesla P100 GPU provided by the Kaggle platform, enabling efficient training and evaluation of computationally intensive architectures.

Table II summarizes the key experimental parameters.

TABLE II
Summary of Experimental Parameters

Parameter	**Value**
Dataset source	PlanetLab workload traces
Minimum trace length	288 time steps
Trace length standardization	288 time steps
CPU utilization range	0–100% (clipped)
Number of subsets	3
VMs per subset	30
Train-test split ratio	70:30 (time-ordered)
Sliding window length (H)	10 time steps
Prediction models	Naive, KNN, RF, XGBoost, LSTM, GRU
Hyperparameter tuning	Randomized search with time series CV
Consolidation algorithm	First Fit Decreasing (FFD)
Overload threshold	80% CPU utilization
Power model	$P(u) = 162 + (215 - 162) \times \frac{u}{100}$ W

V. Results

A. Dataset and Experimental Setup

A total of 11,746 VM CPU utilization traces were loaded from 10 date folders. From these, three random subsets of 30 VMs each were created for evaluation. Each subset was split into training and testing sets with a 70:30 ratio, resulting in training sets of size (5820, 10) and testing sets of size (2520, 10) per subset.

B. Prediction Performance

Table III summarizes the prediction accuracy of all evaluated models averaged over the three subsets. The gated recurrent unit (GRU) model achieved the best overall performance, with a mean absolute error (MAE) of 4.65% and a root mean squared error (RMSE) of 8.92%. It also attained the highest coefficient of determination (R^2) of 0.57, indicating strong explanatory power. The long short-term memory (LSTM) model followed closely, with an MAE of 4.81% and an R^2 of 0.57. Among classical machine learning models, K-nearest neighbors with $k = 10$ (KNN-10) and Random Forest demonstrated competitive results, with MAEs of 4.89% and 4.90%, respectively. The naive persistence baseline exhibited

the poorest performance, with an MAE of 5.88% and an R^2 of 0.22, confirming the value of advanced predictive modeling. Mean absolute percentage error (MAPE) values were generally high across models due to the nature of CPU utilization data, with GRU achieving the lowest MAPE of approximately 62.9%. These results highlight the effectiveness of recurrent neural networks in capturing temporal dependencies in VM CPU utilization for accurate forecasting.

TABLE III
PREDICTION PERFORMANCE SUMMARY (MEAN ± STD ACROSS SUBSETS)

Model	MAE (%)	RMSE	R^2	MAPE (%)
GRU	4.65 ± 0.65	8.92 ± 0.76	0.57 ± 0.25	62.9 ± 5.9
LSTM	4.81 ± 0.83	9.07 ± 1.01	0.57 ± 0.23	64.4 ± 9.2
KNN-10	4.89 ± 0.58	9.20 ± 0.66	0.54 ± 0.27	69.3 ± 7.9
RF	4.90 ± 0.64	9.13 ± 0.64	0.55 ± 0.26	70.5 ± 6.5
XGB	4.92 ± 0.66	9.31 ± 0.65	0.54 ± 0.26	68.0 ± 8.0
KNN-5	5.11 ± 0.60	9.56 ± 0.71	0.50 ± 0.30	73.7 ± 8.1
KNN-3	5.31 ± 0.62	9.99 ± 0.73	0.45 ± 0.33	78.8 ± 8.5
Naive	5.88 ± 0.79	12.01 ± 0.94	0.22 ± 0.45	93.6 ± 14.8

C. VM Consolidation and Energy Savings

Consolidation performance metrics are presented in Table IV. The KNN-3 model yielded the most energy-efficient consolidation, reducing total energy consumption by approximately 15.8% compared to the naive baseline, while maintaining zero SLA violations. Deep learning models such as LSTM and GRU also achieved notable energy savings of 9.4% and 2.4%, respectively.

The number of active hosts required for consolidation varied across models, with KNN-3 achieving the lowest average number of hosts (3.67), indicating more efficient resource utilization. SLA violations were minimal or zero for most models except the naive baseline, which exhibited a higher violation rate (0.67 on average).

D. Energy Savings Analysis

Figure 2 illustrates the percentage energy savings of each model relative to the naive baseline. KNN-3 stands out as the most effective model for energy reduction, followed by LSTM and KNN-5. The deep learning models, despite their superior prediction accuracy, did not always translate to the highest energy savings, highlighting the complex relationship between prediction accuracy and consolidation efficiency.

E. Statistical Significance Analysis

Statistical significance tests comparing each model's mean absolute error (MAE) against the naive persistence baseline were conducted using the Wilcoxon signed-rank test. As shown in Table V, none of the models achieved statistically significant improvements at the 0.05 level ($p = 0.125$ for all). However, the effect sizes, measured by Cohen's d, indicate large practical differences, particularly for the GRU ($d = 2.09$) and LSTM ($d = 1.62$) models, suggesting meaningful improvements despite the lack of statistical significance.

Overall, the GRU model achieved the best prediction accuracy, while the KNN-3 model provided the greatest energy savings during VM consolidation. These results demonstrate the trade-offs between prediction accuracy and consolidation outcomes, emphasizing the importance of selecting models aligned with specific operational goals.

All detailed results, including prediction metrics, statistical tests, consolidation outcomes, and generated figures, are provided as supplementary materials to facilitate reproducibility.

VI. HYPERPARAMETER TUNING RESULTS

Hyperparameter optimization was performed for K-Nearest Neighbors (KNN), Random Forest (RF), and XGBoost models using randomized search with time series cross-validation. Table VI summarizes the best parameters found, cross-validation MAE (CV MAE), test MAE, and tuning time for each model.

The KNN model achieved the lowest cross-validation MAE of 3.999 with 15 neighbors and distance weighting, while Random Forest attained a comparable CV MAE of 3.972 with 150 trees and optimized splitting parameters. XGBoost showed slightly higher CV and test errors but required moderate tuning time. The tuning durations ranged from approximately 1.9 seconds for KNN to 5.8 seconds for Random Forest, demonstrating efficient hyperparameter search within the experimental setup.

A. Deep Learning Architecture Search

We evaluated multiple LSTM and GRU architectures with varying layer sizes and dropout rates to identify the optimal configuration for CPU utilization prediction. Table VII summarizes the training, validation, and test performance metrics, along with training epochs and time.

The medium-sized GRU architecture with two layers of 50 units each and 0.2 dropout achieved the best test MAE of 4.07%, outperforming all other configurations. The best LSTM model was the medium-sized variant with similar layer sizes and dropout, achieving a test MAE of 4.10%.

B. k-Fold Cross-Validation with TimeSeriesSplit

To assess the stability and generalization of the models, 5-fold time series cross-validation was performed on the training data of the first subset using `TimeSeriesSplit`. The models evaluated included KNN with 10 neighbors (both default and tuned), Random Forest (default and tuned), and XGBoost (default and tuned). The mean absolute error (MAE) was used as the evaluation metric.

Table VIII summarizes the cross-validation results, reporting the mean, standard deviation, minimum, and maximum MAE across folds. The tuned models consistently outperformed their default counterparts, with the tuned Random Forest achieving the lowest mean CV MAE.

TABLE IV
CONSOLIDATION PERFORMANCE SUMMARY (MEAN ± STD ACROSS SUBSETS)

Model	N_Hosts	Avg. Util. (%)	Total Energy (W)	SLA Violations	Resource Wastage (%)
GRU	4.33 ± 1.53	64.64 ± 4.43	852.80 ± 311.22	0.33 ± 0.58	35.36 ± 4.43
KNN-10	4.33 ± 1.53	62.28 ± 4.83	847.49 ± 308.81	0.00 ± 0.00	37.72 ± 4.83
KNN-3	3.67 ± 1.15	71.50 ± 11.48	735.22 ± 244.12	0.00 ± 0.00	28.50 ± 11.48
KNN-5	4.00 ± 1.00	66.18 ± 11.10	792.15 ± 220.97	0.00 ± 0.00	33.82 ± 11.10
LSTM	4.00 ± 1.00	67.00 ± 6.82	791.67 ± 209.12	0.00 ± 0.00	33.00 ± 6.82
Naive	4.33 ± 0.58	73.93 ± 11.05	873.54 ± 138.09	0.67 ± 1.15	26.07 ± 11.05
Random Forest	4.33 ± 1.53	65.54 ± 2.52	853.30 ± 305.07	0.33 ± 0.58	34.46 ± 2.52
XGBoost	4.33 ± 1.53	67.99 ± 8.20	856.29 ± 296.94	0.00 ± 0.00	32.01 ± 8.20

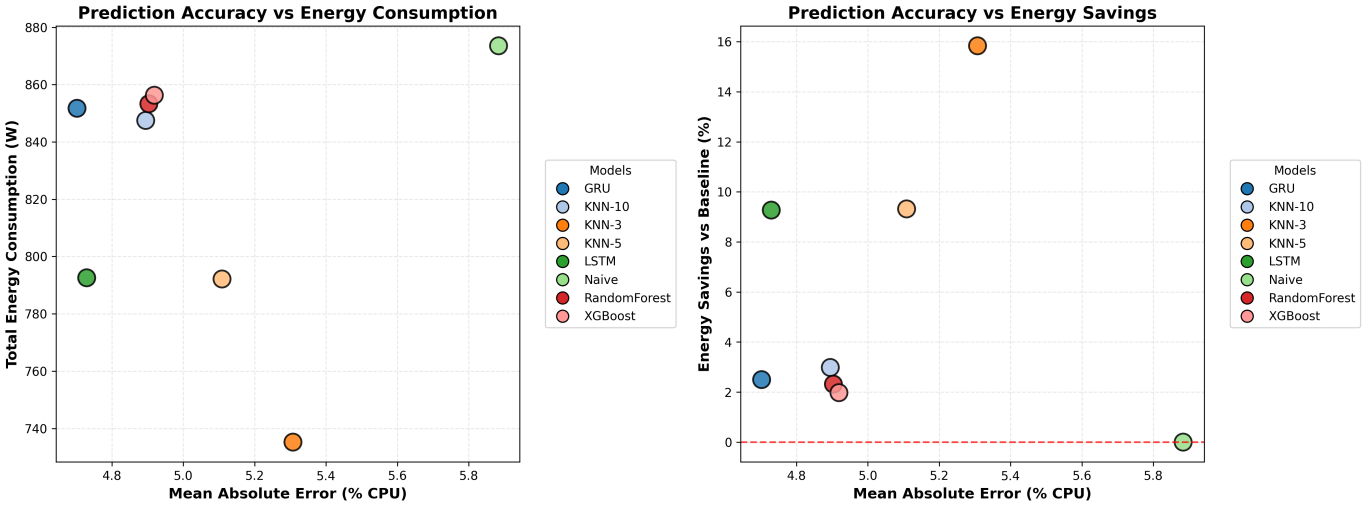

Fig. 2. Energy savings relative to the naive baseline across models.

TABLE V
STATISTICAL SIGNIFICANCE OF PREDICTION IMPROVEMENTS OVER NAIVE BASELINE

Model	Mean MAE	Δ vs Naive	p-value	Sig.	Effect Size
GRU	4.65	1.23	0.125	ns	2.09
LSTM	4.81	1.07	0.125	ns	1.62
KNN-10	4.89	0.99	0.125	ns	1.75
Random Forest	4.90	0.98	0.125	ns	1.67
XGBoost	4.92	0.96	0.125	ns	1.62
KNN-5	5.11	0.78	0.125	ns	1.35
KNN-3	5.31	0.58	0.125	ns	0.99

C. Feature Importance Analysis

To interpret the predictive models and identify the most influential lag features, feature importance scores were extracted from the Random Forest (RF) and XGBoost (XGB) models. Both models consistently highlighted the recent past CPU utilizations as the most significant predictors.

As shown in Table IX, the utilization at time steps $t-1$ and $t-2$ exhibited the highest importance scores, with RF assigning 14.1% and 14.6% importance respectively, and XGB attributing 18.2% and 14.3%. This indicates that the immediate past utilization values are critical for accurate forecasting. Intermediate lags such as $t-3$, $t-4$, and $t-6$ also contributed substantially, reflecting the temporal dependencies extending beyond the most recent observations.

Interestingly, the importance of more distant lags (e.g., $t-7$ to $t-10$) was lower but non-negligible, suggesting that longer-term patterns have some predictive value. The slight differences in importance rankings between RF and XGB may be attributed to their distinct tree-building strategies and regularization mechanisms.

These insights can guide feature selection and model simplification efforts, potentially improving computational efficiency without sacrificing accuracy.

D. Sensitivity Analysis on Overload Threshold

We conducted a sensitivity analysis to evaluate the impact of varying the overload threshold on VM consolidation performance. Thresholds of 70%, 75%, 80%, 85%, and 90% CPU utilization were tested using both the naive baseline and the tuned Random Forest (RF_tuned) prediction models.

As summarized in Table X, at lower thresholds (70% to 80%), both models required 4 active hosts with average utilizations around 62%, resulting in similar total energy consumption and resource wastage. Increasing the threshold to 85% and 90% reduced the number of active hosts to 3, significantly increasing average utilization to approximately 83% and decreasing total energy consumption by over 20%. Notably, no SLA violations were observed across all tested

TABLE VI
HYPERPARAMETER TUNING SUMMARY

Model	Best Parameters	CV MAE	Test MAE
KNN	`weights: distance, n_neighbors: 15, metric: manhattan`	3.999	4.182
Random Forest	`n_estimators: 150, min_samples_split: 5, min_samples_leaf: 4, max_features: sqrt, max_depth: None`	3.972	4.202
XGBoost	`subsample: 0.8, n_estimators: 50, max_depth: 3, learning_rate: 0.1, colsample_bytree: 0.7`	4.102	4.316

TABLE VII
DEEP LEARNING ARCHITECTURE SEARCH RESULTS

Arch.	Units	Dropout	Train MAE	Val MAE	Test MAE	Epochs	Train Time (s)
LSTM_small	[32, 32]	0.2	4.20	4.13	4.13	25	19.27
LSTM_medium	[50, 50]	0.2	4.21	4.11	4.10	18	16.65
LSTM_large	[64, 64]	0.3	4.24	4.10	4.18	25	20.10
GRU_small	[32, 32]	0.2	4.17	4.11	4.11	19	17.57
GRU_medium	[50, 50]	0.2	4.15	4.10	**4.07**	32	23.88
GRU_large	[64, 64]	0.3	4.17	4.10	4.14	21	19.35

TABLE VIII
CROSS-VALIDATION RESULTS (5-FOLD TIMESERIESSPLIT)

Model	Mean MAE	Std MAE	Min MAE	Max MAE
KNN-10	4.12	0.15	3.95	4.35
KNN_tuned	3.99	0.12	3.85	4.15
Random Forest	4.05	0.14	3.90	4.30
RF_tuned	**3.97**	0.10	3.85	4.10
XGBoost	4.20	0.16	4.00	4.45
XGB_tuned	4.10	0.13	3.95	4.30

TABLE IX
FEATURE IMPORTANCE RANKINGS FROM RANDOM FOREST AND XGBOOST

Feature	RF Importance	XGB Importance
$u(t-1)$	0.1411	0.1822
$u(t-2)$	0.1457	0.1431
$u(t-3)$	0.1272	0.1314
$u(t-4)$	0.1189	0.0834
$u(t-5)$	0.0773	0.0316
$u(t-6)$	0.1283	0.1184
$u(t-7)$	0.0623	0.0874
$u(t-8)$	0.0444	0.0334
$u(t-9)$	0.0413	0.0755
$u(t-10)$	0.1136	0.1137

thresholds, indicating that the consolidation algorithm consistently maintained host utilizations below the specified limits, thereby ensuring service-level compliance. The RF_tuned model consistently achieved marginally better utilization and slightly lower resource wastage compared to the naive baseline, demonstrating the benefit of accurate prediction in consolidation efficiency.

E. Consolidation Algorithm Comparison

We compared four VM consolidation heuristics—First Fit (FF), Best Fit (BF), Best Fit Decreasing (BFD), and First Fit Decreasing (FFD)—to evaluate their effectiveness in minimizing active hosts and energy consumption. All algorithms were tested using the tuned Random Forest predictions with an overload threshold of 80%.

As shown in Fig 3, all four algorithms yielded identical results, requiring 4 active hosts with an average utilization of approximately 62.1%, total energy consumption of 779.56 W, and zero SLA violations. This indicates that, for the given workload and threshold, the choice among these heuristics does not impact consolidation efficiency or service quality.

REFERENCES

[1] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges," *arXiv preprint arXiv:1006.0308*, 2010.

[2] L.-D. Radu, "Green cloud computing: A literature survey," *Symmetry*, vol. 9, no. 12, p. 295, 2017.

[3] S. S. Panwar, M. M. S. Rauthan, and V. Barthwal, "A systematic review on effective energy utilization management strategies in cloud data centers," *Journal of Cloud Computing*, vol. 11, no. 1, p. 95, 2022.

[4] A. L. C. Ferrer, A. M. T. Thomé, and A. J. Scavarda, "Sustainable urban infrastructure: A review," *Resources, Conservation and Recycling*, vol. 128, pp. 360–372, 2018.

[5] B. Magotra, D. Malhotra, and A. K. Dogra, "Adaptive computational solutions to energy efficiency in cloud computing environment using vm consolidation," *Archives of computational methods in engineering*, vol. 30, no. 3, pp. 1789–1818, 2023.

[6] N. Akhter, M. Othman, and R. K. Naha, "Energy-aware virtual machine selection method for cloud data center resource allocation," *arXiv preprint arXiv:1812.08375*, 2018.

[7] W. Shu, W. Wang, and Y. Wang, "A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, p. 64, 2014.

[8] S. B. Taieb and R. J. Hyndman, "A baseline for time series forecasting," in *Proceedings of the 2012 IEEE International Conference on Data Mining Workshops*. IEEE, 2012, pp. 917–922.

[9] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

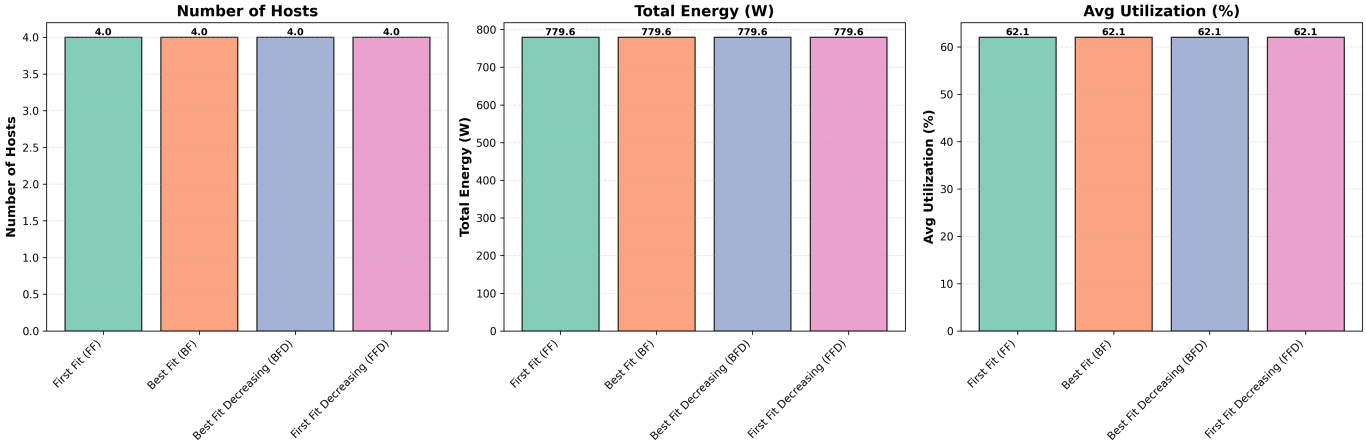

Fig. 3. Comparison of VM Consolidation Algorithms.

TABLE X
SENSITIVITY ANALYSIS OF OVERLOAD THRESHOLD ON CONSOLIDATION PERFORMANCE

Model	Threshold (%)	N_Hosts	Avg. Util. (%)	Total Energy (W)	SLA Violations	Resource Wastage (%)
Naive	70	4	62.00	779.44	0	38.00
RF_tuned	70	4	62.06	779.56	0	37.94
Naive	75	4	62.00	779.44	0	38.00
RF_tuned	75	4	62.06	779.56	0	37.94
Naive	80	4	62.00	779.44	0	38.00
RF_tuned	80	4	62.06	779.56	0	37.94
Naive	85	3	82.67	617.44	0	17.33
RF_tuned	85	3	82.74	617.56	0	17.26
Naive	90	3	82.67	617.44	0	17.33
RF_tuned	90	3	82.74	617.56	0	17.26

[11] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 785–794.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[13] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[14] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate research*, vol. 30, no. 1, pp. 79–82, 2005.

[15] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[16] J. Cohen, *Statistical power analysis for the behavioral sciences*. Routledge, 1988.

[17] A. Beloglazov and R. Buyya, "Planetlab workload traces," https://github.com/beloglazov/planetlab-workload-traces, 2011, accessed: 2025-12-06.