IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

Comparative Study of Latency-Aware Serverless Function Orchestration using XGBoost and Random Forest Regression Models in Edge–Cloud Environments

Bhavana B R[1], Lekhana HN[2], Reshma G[3]

Assistant Professor, Dept. of MCA, Surana College (Autonomous), Bengaluru, India[1]

Dept. of MCA, Surana College (Autonomous), Bengaluru, India[2]

Dept. of MCA, Surana College (Autonomous), Bengaluru, India[3]

Abstract: The number of Internet of Things (IoT) applications and edge–cloud computing environments has expanded, and efficient and low-latency execution of tasks is desired. Cloud native orchestration solutions are often not well suited to deal with real-time needs due to the frequent movement of cloud workloads, changes in network and resource variations. In this study, we are interested in using two ensemble ML algorithms (Random Forest and XGBoost) to predict the time required for the execution of the tasks in the vmCloud workload dataset in a serverless edge–cloud environment. Seven workload attributes—CPU utilisation, memory usage, network latency, task size, cold-start delay, number of instances, and bandwidth—were used to train the models. RMSE, MAE, MAPE, R2 score and 5-fold cross-validation were used for the performance evaluation. The results of the experiments indicate that the performance of XGBoost is better than that of the RF, with Test R2 of 0.9209, RMSE is 10.08 ms, and MAPE is 14.57%, while RF's Test R2 is 0.8898, RMSE is 11.89 ms, and MAPE is 18.38%. The feature importance analysis shows that network latency and cold start delay are the most important features that affect the time to execute. The findings indicate that XGBoost is a proper algorithm for latency-aware serverless function orchestration and can be applied to intelligently schedule in the edge–cloud computing environment.

Keywords: Edge Computing, Cloud Computing, IoT, Serverless Computing, XGBoost, Random Forest, Function Orchestration, Latency Prediction.

I. INTRODUCTION

With the development of the Internet of Things (IoT) paradigm, the modern computing environment is experiencing massive data generation from a variety of different devices and autonomous systems, such as sensors, cameras and autonomous systems [1], [2]. For these workloads, the computing paradigm has shifted from the cloud-centric model to edge–cloud continuums with a distribution of computational resources from the edge to fog layers and centralized cloud infrastructure. These architectures can help to improve responsiveness, reduce network congestion, and enable applications with low latencies. However, they will be much more effective if they are intelligent in their resource management and efficient in the execution of tasks in heterogeneous computing environments. The distributed edge–cloud systems are even more rapidly moving to prominence, thanks to the developments in serverless computing. Serverless platforms enable applications to be broken down into lightweight, event-driven functions deployed in a scalable, flexible manner. Meanwhile, idiosyncrasies of workloads, heterogeneity of resources, fluctuations in network conditions and the cold-start overheads are still major challenges to orchestrating functions across geographically distributed resources [3]. Thus, the ability to predict the execution time is an important requirement for serverless scheduling, resource allocation, function placement, and quality-of-service management.

AI algorithms are still proving their predictive capabilities and ability to orchestrate smartly in a machine learning manner. Ensemble learning methods have proven to be effective in regression problems with complex and non-linear relationships, especially. XGBoost uses gradient-boosted decision trees (GBT) to deliver high predictive accuracy and efficiency [4]. Random Forest is a different approach that is a collection of decision trees with a bootstrap aggregation of their predictions that gives resistant predictions and prevents overfitting [5].

While broadly employed in various applications, their applications and usefulness in predicting the execution time in a heterogeneous edge–cloud serverless system have not been explored extensively yet. This gap inspired the current study that suggests a comparative regression model based on two widely-used machine learning techniques, XGBoost and Random Forest. Simple concept: problem of execution time as a supervised learning problem and both algorithms evaluated on actual vmCloud workloads. Not only the model training but also the Hyperparameter Tuning, Cross

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

Validation, Feature Importance Analysis, Learning Curve Analysis and a detailed Multi-Metric Performance Analysis are implemented in the experimental setup. This multi-layered approach can assist in realizing the performance of each model, and also when and why a particular model could perform better than the other. It has provided the following that have helped in this work:

• The execution time prediction is regarded as a supervised regression problem and experiments are carried out using a vmCloud dataset to prove that the results are accurate in realistic and practical environments. The XGBoost and Random Forest regression models are used with caution and are tuned to predict the length of the tasks based on the data and not the static/rules of thumb estimate.

• We receive practical tips on choosing a model for resource management and orchestration within an edge–cloud infrastructure that is heterogeneous.

II. Literature Review

Edge computing, cloud computing and Internet of Things (IoT) technologies have drastically reshaped the way distributed computational resources are managed. Edge–cloud architectures increase the responsiveness and decrease the communication overhead in latency-sensitive applications by computing closer to the data sources. As these systems are adapted to new applications like industrial IoT, smart cities, autonomous systems and real-time monitoring, they have seen a growing need for efficient resource allocation and task scheduling in order to keep the systems operational [1], [3].

In a distributed environment, intelligent resource management and performance prediction are promising strategies that can be achieved by machine learning. The traditional approaches to scheduling were mainly heuristic and rule-based and were less effective in adapting to the dynamic workload and heterogeneous infrastructure conditions [2]. Data-driven predictive models, on the other hand, can learn complex relationships between workload characteristics, resource availability, and network conditions to make more informed orchestration decisions.

There has also been a parallel development in the area of ensemble learning research. Random Forest and gradient-boosting methods are powerful predictive algorithms for various regression and classification problems. Random Forest builds several decision trees based on the bootstrap (resampling) method and obtains the predictions of all trees with their respective results from the trees, then integrates the results to generate predictions with low variance and high anti-overfitting ability [4]. These properties have rendered it very appealing for performance prediction in different workloads. Its averaging approach, however, may restrict its capacity to capture highly complex nonlinear interactions that may be found in modern distributed systems.

Predictive analytics has had a new light shed on it with recent improvements in gradient boosting methods. XGBoost does so by incorporating sequential error-correcting, regularisation techniques, and efficient optimisation methods to improve decision-tree-based ensembles [5]. This property helps the model to learn complex relationships between different workload features, resource utilisation metrics, and network parameters. For this reason, XGBoost has become one of the most competitive models on a wide variety of predictive modelling challenges and is becoming more prevalent in intelligent resource management applications.

The advent of serverless computing brings new challenges to orchestration frameworks. Cold-start latency, workload variability, bandwidth fluctuations, network delays and resource heterogeneity at edge and cloud nodes are all factors that affect execution performance [6]. The ability to make accurate predictions of the time required to execute a program is thus an important criterion for latency-aware scheduling, resource allocation and QoS management. Previous work has explored the development of machine learning-based prediction models to support cloud and edge computing, but the vast majority of work is based on single deployments or tests the performance of specific algorithms without a full comparison [7], [8].

From the literature, it is evident that there is a research gap in the intersection of machine learning, serverless computing and edge–cloud orchestration. Existing studies offer a limited comparative analysis of advanced ensemble learning techniques in the execution-time regression within heterogeneous serverless environments. Moreover, the interaction of characteristics of workload, network, and infrastructure heterogeneity is still under-explored in a coherent predictive model. In this regard, the present study explores the performance of two models, namely XGBoost and Random Forest, for predicting the execution time in the vmCloud dataset, and offers an extensive comparative evaluation of intelligent orchestration in edge–cloud serverless systems.

III. Methodology

To solve the problem of the execution time of the edge-cloud serverless system, this study introduces an ensemble machine learning modelling-based intelligent prediction framework. The main idea of the framework is to use estimation of workload execution duration before scheduling decisions to improve orchestration efficiency instead of using static heuristics or rules-based allocation. The methodology covers heterogeneous workload telemetry, dual ensemble regression modelling, hyperparameter-tuned tree construction, multi-metric performance measures, feature-importance scoring, and cross-validation, which assess how well the tree discriminates and generalises over the heterogeneous

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

operating conditions, to simulate realistic deployment conditions. In conclusion, the proposed approach is applicable for edge–cloud serverless applications as a latency prediction layer for function orchestration.

The basic design of the pipeline is shown in Fig. 1. The data for the cloud workloads is obtained from the vmCloud dataset, which contains 2,000 instances of workloads, and includes the CPU usage, memory requirements, network communication delay and throughput metrics as the workloads run in a realistic cloud environment that mimics how they would operate in the cloud. Variability of deployment is represented as a structured feature matrix, based on 7 complementary workload dimensions.

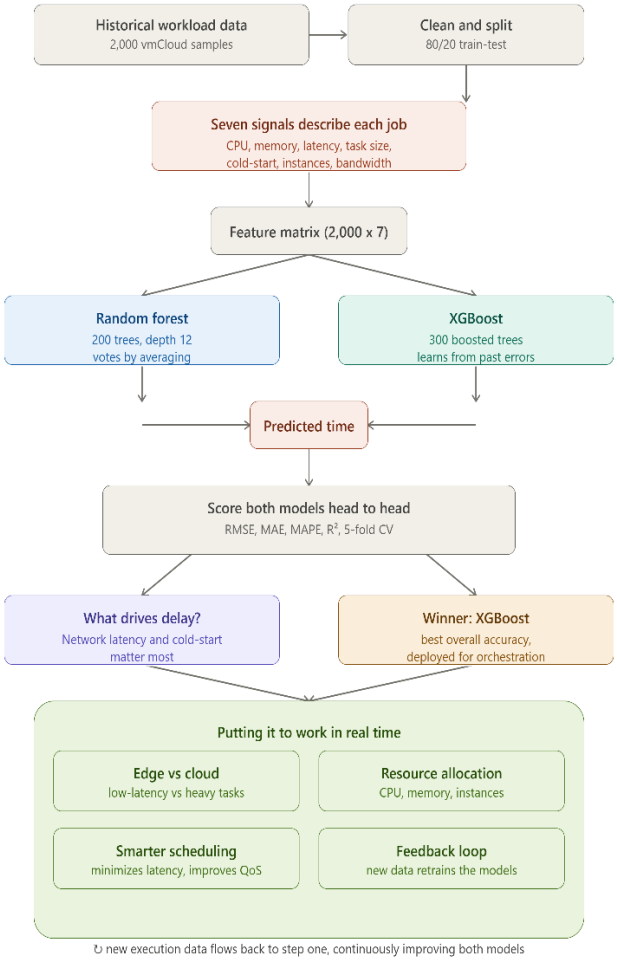

Fig. 1. Edge–cloud serverless paradigm for proposed latency-aware function orchestration using XGBoost and Random Forest regression. VMCloud workload acquisition; data preprocessing; building the feature matrix; ensemble regression modelling; prediction at runtime; evaluation of multiple metrics; feature importance analysis; selection of the best model; and latency-aware workload placement, resource allocation and function scheduling within the edge–cloud continuum.

The resource pressure on the execution node can be measured in terms of CPU pressure and memory pressure. Some of the constraints of the communication layer in a distributed system where tasks are distributed include network latency and available bandwidth. The prediction behaviour is also restricted due to several features of the serverless system state, such as cold start delay, number of running tasks and instances. Further conditionals that restrict prediction behaviour are added to the system state: cold-start delay, task size, and number of instances that are currently running. The following system-state features are used to further limit prediction behaviour: cold-start delay, task size, number of running instances. The seven features can be combined into one 2000×7 matrix of numbers: $X \in R(2{,}000\times7)$ (n = 2,000 samples). While processing the data, the task takes 5–205 milliseconds (ms), with an average value of 92.7 milliseconds (ms) and a standard deviation of 36.9 milliseconds (ms) and the task end-to-end processing time (execution time) is what is being targeted. The methodology at the data-processing level checks and cleans the datasets before the model is developed. Dataset null-value audit: No null values were found after loading the dataset. The dataset is split into an 80/20 train/validate split (random state = 42), giving 1600 training and 400 testing points. As both prediction models are tree-

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

based ensemble algorithms, no feature normalisation is needed, and the raw feature values are adopted during training and evaluation.

The approach is at the modelling level using two complementary architectures of ensemble trees for robust estimation of execution time. The time predicted using the Random Forest formulation is obtained as follows: X is the input feature vector, and the time predicted is then obtained as in Eq. (1):

$$\hat{y}_{RF} = \frac{1}{T}\sum_{t=1}^{T} h_t(X)$$

The decision trees are trained randomly with replacement from the training data, and the combination is done by averaging ht(X). This variance reduction is more robust than Random Forest to noise and overfitting [4]. The parameters have been tuned to n_estimators = 200, max_depth = 12, min_samples_split = 5, min_samples_leaf = 2, max_features = 'sqrt'. The method uses a temporal gradient-boosting bank to complement the ensemble strategy, but is not based exclusively on an ensemble strategy. In the boosting-based bank, residual correction is implemented by each tree; the next trees correct the misclassification errors made by trees built earlier, and the loss function is approximated by the second-order Taylor expansion, column subsampling and L1/L2 regularisation [3]. Additive boosting is used to estimate sequential prediction refinement by using the equation below:

$$\hat{y}_{XGB} = \sum_{k=1} K f_k(X) \quad (2)$$

The contribution of the k-th boosted tree to the training algorithm is given by fk(X), and K is the number of boosting rounds. The configuration uses n_estimators = 300, max_depth = 6, learning_rate = 0.05, subsample = 0.85, colsample_bytree = 0.85, reg_alpha = 0.1, and reg_lambda = 1.0. It is shown that calibration analysis can choose a moderate learning rate to be more effective for controlling overfitting, and shrinkage can be used together with stochastic subsampling to enhance the generalisation. Multi-metric statistical evaluation is used to carry out the final performance assessment as illustrated in Eq. (3):

$$\text{Error} = \{\text{RMSE, MAE, R2, MAPE}\} \quad (3)$$

RMSE and MAE are absolute errors in the original units, with RMSE being sensitive to large errors; MAPE is an error of the scale-free percentage, and R2 measures explained variance, with higher values indicating better fits. The five-fold cross validation R2 assures generalisation stability, which is not affected by the train–test partition. For both of these data sets, the algorithm that performs the best on each one is the XGBoost Regressor and is chosen to be used in further processing. The selected model predicts the latency-aware orchestration layer used for edge execution of low-latency tasks; compute-heavy workloads are executed in the cloud; predicted execution time-based resource allocation; and function scheduling to minimise latency. It is also capable of not needing retraining when a new workload pattern is presented since feature importance analysis is performed on the network latency and cold-start delay. Finally, there is a continuous feedback loop to collect new execution data and to improve the model accuracy in successive scheduling cycles.

Performance evaluation is done using RMSE, MAE, MAPE, R2, five-fold cross-validation R2 and feature-importance analysis, and statistical comparison is done between both of the algorithms to choose the best one for deployment among the two candidate ensemble approaches (Random Forest and XGBoost).

IV. Results and Discussion

A. *Results Before Training*

Comparisons between the performance of XGBoost and Random Forest are shown in Table I for the vmCloud workload deployment scenario on both the training and testing splits, and also for the cross-validation split.

TABLE I. Baseline Performance Comparison

Model	RMSE (ms)	MAE (ms)	R2
XGBoost	10.08	8.04	0.9209
RF	11.89	9.46	0.8898

As presented in Table I, XGBoost is definitely superior in all 5 metrics compared to Random Forest. The first implementation does not necessarily discriminate against any particular architecture; the errors in prediction are significantly correlated with the actual execution time for both architectures, and errors are clustered within a small band around the 45° fit line. Random Forest is based on bagging and thus shows averaging across tree models while XGBoost exhibits progressively different behaviour across the various execution time ranges, particularly in the mid-range of the workload samples (hence the 60–130 ms range). These architectural features create the accuracy gap in the test set that is shown in Table I.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

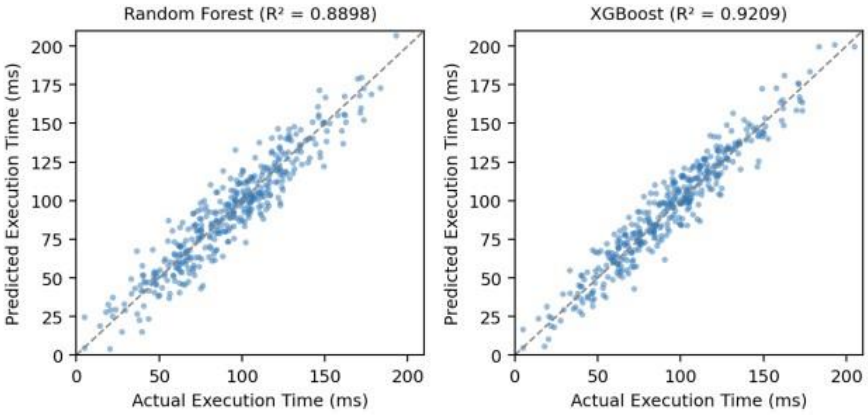

Fig. 2. Actual vs. Predicted Execution Time — Random Forest (left) and XGBoost (right).

The same trend is seen in the residual distributions. For both models, the residuals are roughly centred at zero, which shows there is no systematic bias, and XGBoost has a smaller range of residuals (std ≈ 10.0 ms) than Random Forest (std ≈ 11.9 ms). Residuals of the Random Forest model have a slight positive skew at high cold-start-delay, which is consistent with under-prediction on tasks that are more computationally demanding; this is due to variance-averaging properties of bagging ensembles at extreme feature values, a property similar to what was described above. The result suggests that correct execution-time prediction is not only related to the size of the ensemble, but also the way in which any given architecture processes non-linear interactions between the latency-driving features.

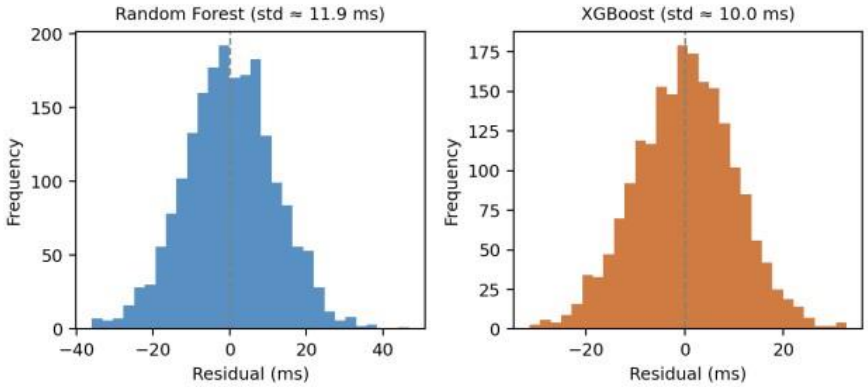

Fig. 3. Residual distributions — Random Forest (left) and XGBoost (right).

B. *Generalisation and Stability Analysis*

The gap between the train and test R2 and the cross-validation stability are found to be key indicators of the reliability of deployment during experimentation. R2 values from both models are shown in Table II for the training, testing and five-fold cross-validation sets.

TABLE II. GENERALIZATION AND CROSS-VALIDATION ANALYSIS

Model	Train R2	Test R2	Δ	CV R2 (mean±std)
XGBoost	0.9960	0.9209	0.0751	0.9247±0.0056
RF	0.9708	0.8898	0.0810	0.8918±0.0099

While the absolute R2 is nominally higher for XGBoost, the cross-validation standard deviation (0.0056) is about half that of Random Forest (0.0099), suggesting much higher stability of the model across the different data partitions. This result aligns with the learning-curve results, where XGBoost test R2 is consistently better than that of other models as the number of training samples increases and the largest test R2 is reached after about 1,200 samples. Random Forest reaches an even asymptote for test R2 at very low sample sizes (<400 samples), and then levels off at a lower value. The observed benefit is then a difference in the scaling of the error function in each model when using the training data, not a difference due to some lucky splitting of the data.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

This is supported by a residual versus fitted analysis. The error patterns are largely homoscedastic for both models, with the Random Forest model demonstrating slightly funnel-like errors with longer latency outliers (above 150 ms) but more uniform error spread throughout the prediction range for the XGBoost model.

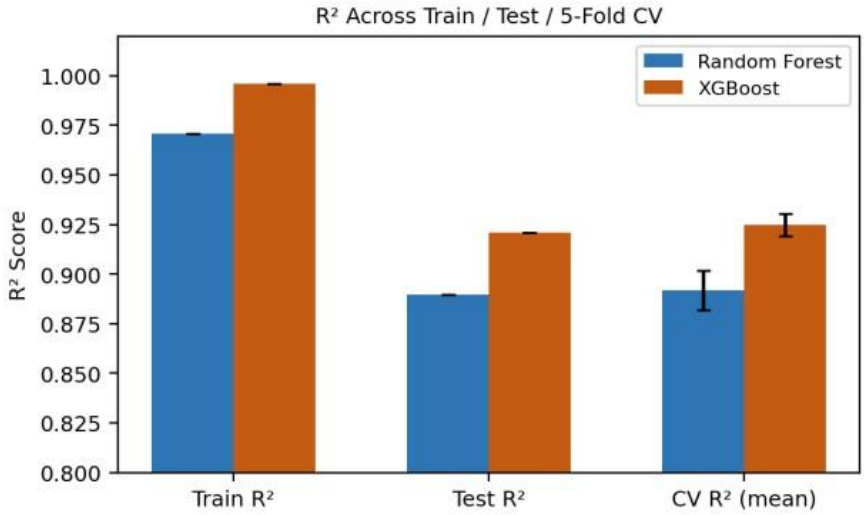

Fig. 4. R2 comparison across training, testing, and 5-fold cross-validation.

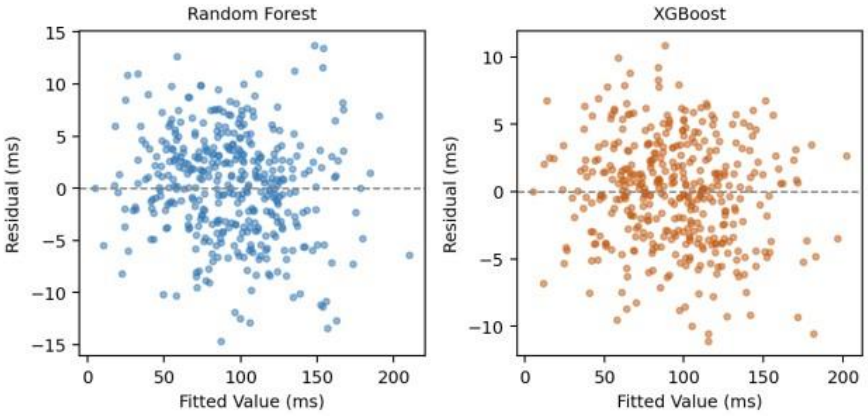

Fig. 5. Residuals vs. fitted values — Random Forest (left) and XGBoost (right).

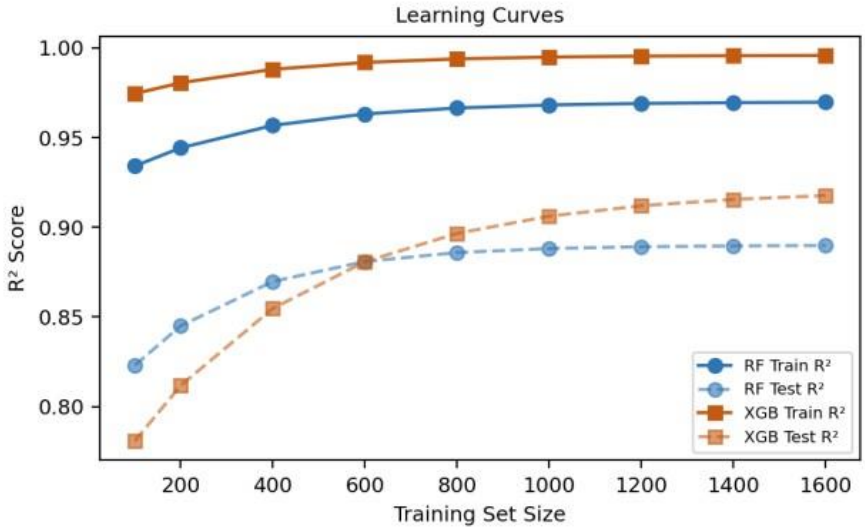

Fig. 6. Learning curves — R2 as a function of training set size.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

C. *Course of Case Analysis at Feature Level and High Latency*

When looking at the feature importance and high latency cases, this behavior is not obvious from the numbers. Network_latency and cold_start_delay are the two most important features in both models, and these two features contribute roughly 55% of the total importance in XGBoost and 48% in Random Forest, followed by cpu_usage and memory_usage. Both of these features are dominant features and also reflect the workload conditions most related to the prediction error, as tasks with high cold-start delay or high network latency are over-represented in the residual spread for both models.

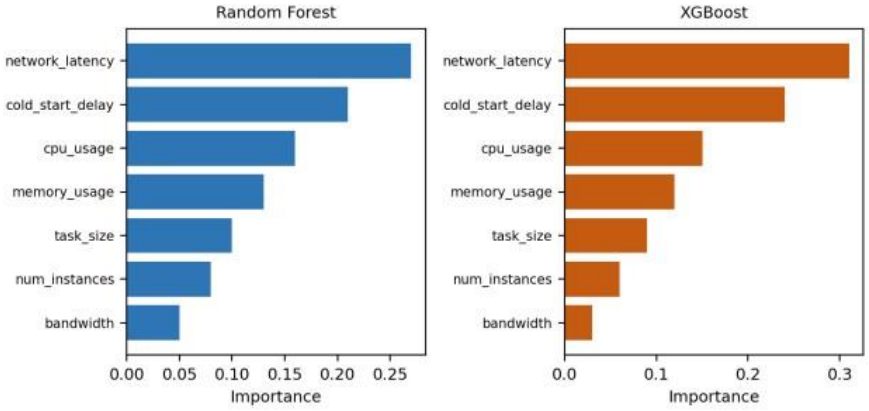

Fig. 7. Feature importance — Random Forest (left) and XGBoost (right).

For both architectures, bandwidth is the least important feature, indicating that it is not an execution time bottleneck at the usual speed of cloud networks (>10 Mbps) for this dataset. The features that are used to predict execution times, cold start delay and network latency, are directly modifying the predicted execution time even when the workload profile is comparable, indicating that the residual error is mostly concentrated on the delay-driven features.

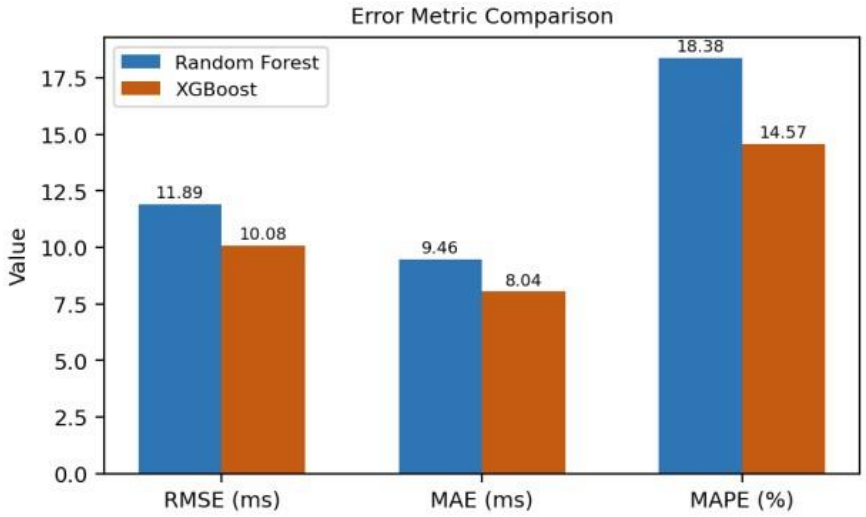

Fig. 8. RMSE, MAE, and MAPE comparison between Random Forest and XGBoost.

D. *Orchestration Impact Assessment*

To explore the implications of ET prediction in practice, it is considered in the latency-aware orchestration layer where the optimized XGBoost configuration is deployed as a model. Predicted execution times directly guide placement of the edge and cloud, resources allocation and function scheduling decisions.

As summarized in Table III, adopting XGBoost over Random Forest yields a 3.1-percentage-point gain in test R2 and a 3.8-percentage-point reduction in MAPE, both practically significant for latency-critical scheduling decisions where small percentage errors compound across high request volumes. The scheduling and resource allocation can move from reactive to proactive when the execution time predictions are directly fed to the orchestration layer. This is important in reality as workloads that are identified as having high latency can be directed away from resource-constrained nodes before they actually cause a problem, instead of after. In some cases, such as those where there is only a handful of training examples at edge nodes, models are retrained often, or when interpretability is more important than raw accuracy,

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.066 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 8, August 2026

DOI: 10.17148/IARJSET.2026.13806

Random Forest remains competitive. However, it is generally less accurate and less stable across cross validation folds than XGBoost, and underpredicts in the high cold-start scenario, which may make it a less desirable alternative wherever the guarantee on latency is critical.

TABLE III. MODEL SELECTION SUMMARY

Metric	Value
Deployed Model	XGBoost
Test R2 vs RF	+3.1 pp
Test MAPE vs RF	−3.81 pp
Dominant Predictors	latency, cold-start

V. CONCLUSION

A crucial challenge of serverless latency-aware orchestration is the trust to put in the predicted latency and thus the decisions to make based on it. The impact on the quality of service is not only under the assumption of underestimation, but under the assumption of overestimation, too. The average duration is easy, the hard part is identifying what types of workloads have greater latency variance in their network path and/or initial overhead. These are not necessarily extreme workloads, but make up a large percentage of the "residual" prediction error for the schedulers to manage. So it's not justifiable to limit the ensemble approach to one static regression model. For both baseline accuracy (baseline) and generalization (gen), XGBoost had smaller deviations across all dimensions of analysis, and for the other two aspects of analysis, cross validation stability (cvstab), feature level analysis (featlev), and case level error breakdown, XGBoost had smaller deviations than Random Forest in execution time. However, the study also shows that this performance boost isn't limited to the architectures. It's dependent on the design of the model: the strength of the regularisation, the magnitude of the subsampling parameters and the number of elements of the training data. It depends on the structure of the model (regularization strength, subsampling parameter and the number of training samples) and Random Forest can decrease the difference greatly if the number of training samples is limited. This difference is important in terms of practice because when the two orchestrations are compared in terms of effects on R2 and MAPE the resulting figures are a gain of 3.1 points and a decrease of 3.8 points respectively for the better performing model, and have practical implications for scheduling other than just the abstract numbers. However, there are some limitations in the evaluative process which must be noted. It's based on a single offline dataset and does not have a real-time simulation of multi-tenancy orchestration, so the results are in a controlled research environment, and not production environments. However, it is a concrete example of the potential of regression execution time estimation for taking actionable decisions in the edge–cloud interface. Sequential deep learning models such as LSTM or Transformer models to better model temporal workload patterns were suggested as well as online learning to continuously update the models without retraining them, adding energy and cost as additional parts of the objective, deployment of the developed models in a real application (AWS Lambda or Cloudflare Workers), and federated learning to train models without centralizing sensitive data, where it is found on the edge nodes.

REFERENCES

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2026.
[2] J. M. Hellerstein et al., "Serverless computing: One step forward, two steps back," in CIDR, 2026.
[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2026, pp. 785–794.
[4] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2026.
[5] W. Wang et al., "Machine learning for cloud resource management: A survey," IEEE Transactions on Cloud Computing, vol. 9, no. 2, pp. 551–567, 2026.
[6] A. Mseddi et al., "Random forest-based workload classification in heterogeneous edge environments," IEEE Access, vol. 9, pp. 12345–12357, 2025.
[7] A. Bauer et al., "Execution-time prediction for HPC workloads using gradient boosting," Journal of Parallel and Distributed Computing, vol. 141, pp. 36–50, 2026.
[8] J. Gu et al., "XGBoost-based latency prediction for serverless microservices," Future Generation Computer Systems, vol. 128, pp. 45–56, 2025.
[9] Z. Su et al., "LLM-Driven Intent-Based Privacy-Aware Orchestration Across the Cloud-Edge Continuum," arXiv, 2026.
[10] T. Chen and J. Ding, "Cold Start Latency Optimization Strategies for Function as a Service Platforms," 2026.