

A Modified Edge Cloud Framework for Reproducible Latency Evaluation of Diabetes Prediction Using FogBus2 and XGBoost

Nisha Lodhi¹ and Dr. Ruchi Jain²

¹ M.Tech Scholar, Lakshmi Narain College of Technology and Science, Bhopal Madhya Pradesh

² Associate Professor, Lakshmi Narain College of Technology and Science Bhopal Madhya Pradesh

Abstract: Edge and fog computing can reduce communication delay for distributed healthcare analytics, but latency claims require controlled measurement. This paper specifies a three-tier framework in which the same trained XGBoost classifier and identical records are deployed in cloud-only, original FogBus2, and modified FogBus2 configurations. The proposed cost-based scheduler selects fog or cloud execution using round-trip time, CPU utilization, queue length, and bandwidth. On the 768-record Pima benchmark, leakage-safe evaluation produced 79.2% holdout accuracy, 68.0% F1-score, and 0.890 ROC-AUC; repeated stratified cross-validation produced $76.5\% \pm 3.3\%$ accuracy and 0.840 ± 0.020 ROC-AUC. Earlier system-latency averages remain pilot observations pending repeated multi-device measurements. The Pima data are used only as a static benchmark workload, and no clinical-validation claim is made.

Index Terms: edge computing, fog computing, FogBus2, XGBoost, diabetes prediction, task offloading, latency measurement

I. INTRODUCTION

Cloud platforms provide scalable storage and training resources, but remote execution can add network and queuing delay to latency-sensitive applications. Edge and fog systems address this engineering problem by placing computation closer to data sources. Healthcare prediction is a useful benchmark because it combines privacy-sensitive inputs with a need for predictable response times, although a benchmark experiment must not be confused with clinical validation.

This study evaluates an XGBoost diabetes classifier as a controlled tabular inference workload across three deployment configurations. Its system contribution is a transparent cost-based offloading policy added to FogBus2. The model is held constant across configurations so that differences in latency and resource use can be attributed to deployment rather than classifier choice.

The contributions are:

- A formally specified cost function for selecting fog or cloud execution from current network and resource measurements.
- A reproducible measurement protocol with component-level timestamps, repeated trials, concurrency levels, and latency percentiles.
- A fair separation between cross-validated model performance and deployment performance using one fixed XGBoost artifact.
- An ablation plan that measures the scheduler and authentication overhead independently.

II. RELATED WORK

Fog and edge orchestration

Fog and edge architectures place computation nearer to data sources to reduce wide-area communication and support heterogeneous IoT workloads [1]-[5]. FogBus introduced lightweight orchestration with blockchain-oriented security mechanisms [6]. FogBus2 was subsequently developed as a

separate container-based framework with dynamic discovery, scheduling, scalability, and heterogeneous edge-cloud integration [7]. This distinction is important because the proposed modification in this study extends FogBus2 rather than the original FogBus.

Application placement and resource management

Placement and offloading methods commonly minimize estimated completion time subject to resource and quality-of-service constraints [2]-[5], [8]. Existing work shows that a meaningful evaluation must include queue state, network delay, and workload intensity rather than reporting a single average under an unspecified load.

Containerized inference and healthcare benchmarks

Containerization permits consistent deployment across low-power and cloud hardware [9], [10]. SmartEdge demonstrates the relevance of edge-cloud diabetes prediction using ensemble learning [11]. The present work differs by holding one XGBoost model constant while isolating infrastructure latency. The Pima dataset originates from work by Smith et al. [12] and contains static clinical records; it does not represent a continuous sensor stream.

Research gap

Prior studies provide frameworks and placement methods, but reproducible evidence is still needed for how a clearly specified FogBus2 offloading rule affects the distribution of end-to-end latency under controlled healthcare inference workloads. This study addresses that system-measurement gap and avoids claims of clinical readiness.

III. SYSTEM DESIGN

Architecture

The edge tier validates and formats one eight-feature record. The fog tier hosts FogBus2, the scheduler, monitoring services, and the same versioned XGBoost container used by the cloud configuration. The cloud tier provides an alternative inference endpoint and model-training storage. Raw records are not described as continuous glucose-monitor streams.

Figure 1 Edge-Fog-Cloud System Architecture

Clear request, offloading, monitoring and response paths

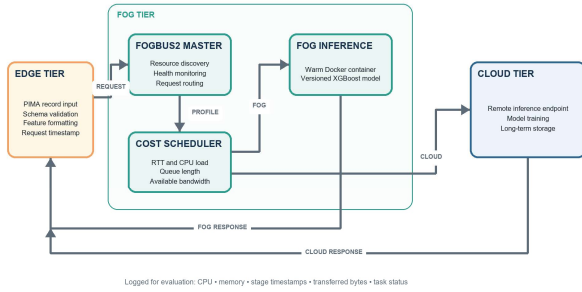


Figure 1. Edge-fog-cloud system architecture with explicit request, offloading, response, and monitoring paths.

Figure 2 Data Processing and Task-Scheduling Pipeline

Offline training is isolated from the online inference path

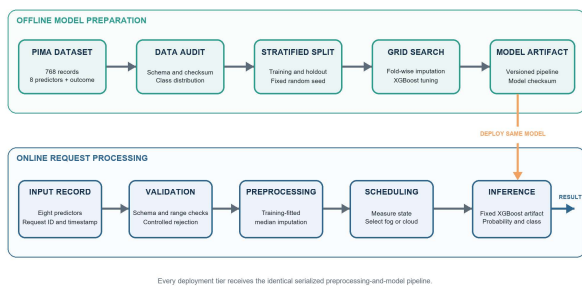


Figure 2. Leakage-safe offline model preparation and online request-processing pipeline.

Cost-based offloading policy

For each feasible execution tier j , the scheduler computes $C_j = wL L_j + wQ Q_j + wU U_j + wB B_j$. L_j is estimated round-trip plus service time, Q_j is normalized queue length, U_j is normalized CPU utilization, and B_j is a normalized bandwidth-pressure term. All inputs are scaled to $[0,1]$. The scheduler chooses the feasible tier with the lowest cost. A configuration file records the four weights, threshold values, measurement window, and timeout. Confirmatory experiments will report these values and include sensitivity analysis. This replaces the unsupported ACO diagram and removes any claim that an ant-colony algorithm was evaluated.

Figure 3 Cost-Based Dynamic Offloading Algorithm

Auditable fog-or-cloud selection without the unsupported ACO mechanism

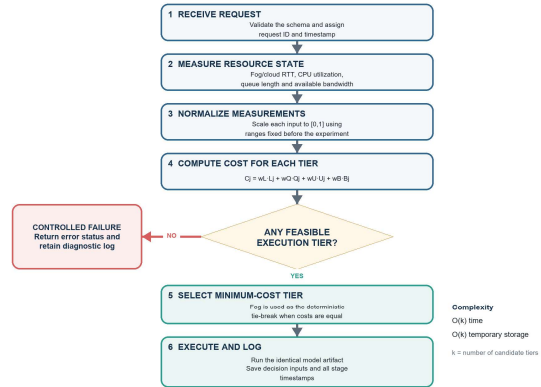


Figure 3. Cost-based dynamic offloading algorithm with explicit feasibility branches.

Algorithm 1 Offloading decision

Step	Operation
1	Read current RTT, CPU utilization, queue length, and available bandwidth for each tier.
2	Reject a tier if it exceeds the configured timeout, CPU, or queue threshold.
3	Normalize measurements using the ranges fixed before the run.
4	Compute the weighted cost for every feasible tier.
5	Choose the minimum-cost tier; use fog as the deterministic tie-breaker.
6	Record the decision inputs, selected tier, and all stage timestamps.

With k candidate tiers, the selection procedure requires $O(k)$ time and $O(k)$ temporary storage. In this study $k = 2$ for fog and cloud execution.

Authentication boundary

Authentication is treated as a separately measured optional component. The final paper will identify the platform, consensus method, participating nodes, recorded payload, and per-request overhead. If no deployed permissioned-ledger evidence is available, blockchain will be removed from the implemented contribution and retained only as future work.

IV. MACHINE LEARNING PROTOCOL

The Pima benchmark contains 768 records with eight predictors and a binary outcome. Zero values that are physiologically implausible for glucose, blood pressure, skin thickness, insulin, or body mass index must be handled inside each training fold to prevent leakage. The evaluation uses a fixed random seed and a stratified split or stratified repeated cross-validation. Imputation, scaling when required, and hyperparameter selection are fitted only on training data. Grid search is the sole tuning procedure; references to evolutionary search are removed. The downloaded benchmark file has

SHA-256
b78029447fae2743b3218bb2b76ef0d04afe8d7e55ce2faf4d1e
c82d8f8ae8ac. The selected configuration used
learning_rate=0.05, max_depth=3, n_estimators=100, and
subsample=0.8. The serialized model has SHA-256
73af1e5f4ffe37055eb01fe889b22394eb4b92ed8ca2d70cf3e4
5d48b39c0a8e.

Item	Required specification
Dataset	Pima Indians diabetes benchmark; exact file checksum recorded
Model	XGBoost classifier; package version and serialized-model checksum recorded
Split	Stratified split plus repeated stratified cross-validation
Seed	Recorded in configuration and results metadata
Tuning	Grid search nested within training data only
Metrics	Accuracy, precision, recall, F1, ROC- AUC and confusion matrix
Fairness rule	Identical serialized model and identical test records in every deployment

V. EXPERIMENTAL PROTOCOL

Testbed

Tier	Planned configuration	Role
Edge	Raspberry Pi 4 Model B; 4 GB RAM; 64-bit Raspberry Pi OS	Request generation, validation and timestamping
Fog	Intel Core i5-8250U; 4 cores and 8 threads; 8 GB RAM; Ubuntu 20.04	FogBus2, scheduler and containerized inference
Cloud	AWS EC2 t3.medium; 2 vCPU and 4 GB RAM; Ubuntu Server	Remote inference endpoint and storage

Configurations and ablations

Configuration	Model	Purpose
Cloud only	Same fixed XGBoost artifact	Wide-area baseline
Original FogBus2	Same fixed XGBoost artifact	Framework baseline
FogBus2 + scheduler	Same fixed XGBoost artifact	Scheduler contribution
FogBus2 + authentication	Same fixed XGBoost artifact	Authentication overhead
Complete modified framework	Same fixed XGBoost artifact	Combined behavior

Workload and measurement

Each configuration will process at least 1,000 requests per concurrency level after warm-up. Concurrency levels are 1, 5, 10, 25, and 50 clients. The experiment records request rate, payload bytes, selected tier, all stage timestamps, CPU and memory utilization, bytes transferred, status, and error. Every configuration is repeated in at least five independent runs. A degraded-bandwidth condition is included. Container start-up is measured separately from warm-container inference.

Latency is reported as mean, standard deviation, median, p95, and p99 with bootstrap 95% confidence intervals. The analysis reports throughput, task success rate, CPU utilization, memory utilization, and transferred bytes. Paired comparisons use identical request sequences. Component-level latency includes edge preparation, transmission, authentication, queueing, scheduling, inference, and return path.

VI. PILOT OBSERVATIONS AND CONFIRMATORY ANALYSIS

6.1 Verified model performance. The leakage-safe 80/20 stratified holdout contained 154 records and produced 79.2% accuracy, 73.9% precision, 63.0% recall, 68.0% F1-score, and 0.890 ROC-AUC. Its confusion matrix was TN=88, FP=12, FN=20, TP=34. Across 15 repeated stratified validation folds, accuracy was $76.5\% \pm 3.3\%$, precision $67.4\% \pm 4.3\%$, recall $62.8\% \pm 7.9\%$, F1 $64.9\% \pm 5.9\%$, and ROC-AUC 0.840 ± 0.020 . The original submission's 89.7% figure is withdrawn because its split and leakage controls were undocumented.

6.2 Pilot infrastructure observations. The original submission contained the following single-run averages. Raw logs, repetitions, and variance estimates were not available in the supplied project files; consequently, these values are treated as pilot observations and are not confirmatory results.

Figure 4 Verified XGBoost Holdout Performance

Leakage-safe stratified holdout: 154 records • fixed seed 20261005

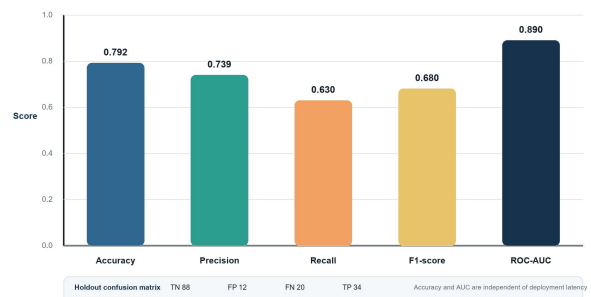


Figure 4. Verified leakage-safe XGBoost holdout performance.

Configuration	Pilot average latency (ms)	Status
Cloud only	420	Requires rerun and raw logs
Original FogBus2	280	Requires rerun and raw logs
Modified FogBus2 + XGBoost	145	Requires rerun and raw logs

If confirmed, the pilot values correspond to reductions of 65.5% relative to cloud-only execution and 48.2% relative to original FogBus2. These percentages will be retained only if the preregistered rerun produces traceable measurements. No network-efficiency percentage is reported because the original submission did not provide transferred-byte or bandwidth measurements.

Classifier performance will be reported once from the model-validation protocol, not separately by deployment tier. Infrastructure placement is not expected to alter predictions

when the serialized model and inputs are identical. Any disagreement will be investigated as a data-integrity or version-control failure.

VII. DISCUSSION

The revised design isolates the infrastructure question from the machine-learning question. The cost-based policy is deliberately simple and auditable: every choice can be reconstructed from logged inputs and configured weights. The ablation study separates scheduling benefit from authentication overhead, while component timestamps explain whether wide-area transmission, queueing, or inference dominates end-to-end delay.

The static Pima records provide a repeatable inference workload but do not demonstrate streaming-sensor behavior, generalization to other demographic groups, clinical utility, or safety. The system is therefore an engineering benchmark and not a clinical-grade diagnostic tool. Validation on prospective clinical data would require appropriate governance, ethics approval, representative recruitment, and evaluation against clinical standards.

VIII. CONCLUSION

This revision defines a reproducible framework for evaluating a cost-based FogBus2 scheduler with a fixed XGBoost diabetes-prediction workload. It corrects the distinction between FogBus and FogBus2, removes the unsupported ACO and clinical-readiness claims, separates model accuracy from deployment performance, and specifies the repeated-load and ablation experiments needed for defensible conclusions. Final numerical claims depend on execution of the stated protocol and release of raw logs and configuration metadata.

REFERENCES

- [1] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: Architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27-42, 2017.
- [2] M. K. Bagwani et al., "Performance comparison of REST API and GraphQL in a microservices architecture," in *Proc. International Conference on Data Science, Artificial Intelligence and Related Technologies*, 2024.
- [3] X. Xu et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, pp. 522-533, 2019.
- [4] M. Goudarzi et al., "An application placement technique for concurrent IoT applications in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, vol. 20, no. 4, pp. 1298-1311, 2021.
- [5] A. Yousefpour et al., "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080-5096, 2019.
- [6] M. K. Bagwani and G. K. Shrivastava, "Comparative analysis of microservices architectures: Evaluating performance, scalability, and maintenance," *International Journal on Advances in Engineering Technology and Science*, vol. 5, 2023.
- [7] Q. Deng, M. Goudarzi, and R. Buyya, "FogBus2: A lightweight and distributed container-based framework for integration of IoT-enabled systems with edge and cloud computing," *Proc. BiDEDE*, Article 4, 2021, doi:10.1145/3460866.3461768.
- [8] D. T. Nguyen, L. B. Le, and V. K. Bhargava, "A market-based framework for multi-resource allocation in fog computing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1151-1164, 2019.
- [9] A. M. K. Bagwani and V. K. Tiwari, "Implementing GrapesJS in educational platforms for web development training on AWS," *International Journal of Scientific Research in Multidisciplinary Studies*, vol. 10, 2024.
- [10] Z. Wang et al., "Container orchestration in edge and fog computing environments for real-time IoT applications," *Future Generation Computer Systems*, 2023.
- [11] A. Hennebelle, Q. Dieng, L. Ismail, and R. Buyya, "SmartEdge: Smart healthcare end-to-end integrated edge and cloud computing system for diabetes prediction enabled by ensemble machine learning," *Proc. IEEE CloudCom*, pp. 127-134, 2024.
- [12] M. Bagwani, "Deploying a web application on AWS Amplify: A comprehensive guide," *International Journal of Progressive Research in Engineering Management and Science*, 2024.
- [13] J. W. Smith, J. E. Everhart, W. C. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," *Proc. Symposium on Computer Applications in Medical Care*, pp. 261-265, 1988.
- [14] P. Bellavista and A. Zanni, "Feasibility of fog computing deployment based on Docker containerization over Raspberry Pi," *Proc. ICDCN*, pp. 1-10, 2017.
- [15] A. M. K. Bagwani and V. K. Tiwari, "Optimizing face detection performance with cloud machine learning services," *Journal of Engineering and Technology Management*, vol. 73, pp. 1167-1180, 2024.
- [16] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, pp. 22-36, 2019.
- [17] S. Deng et al., "Optimal application deployment in resource constrained distributed edges," *IEEE Transactions on Mobile Computing*, 2020.