THE SCIENTIFIC TEMPER (2026) Vol. 17 (1): 5570-5581

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2026.17.01.20

https://scientifictemper.com/

RESEARCH ARTICLE

# Multi-Metric Evaluation Framework for Machine Learning-Based Load Prediction in e-Governance Systems

**Sanjeev Kumar[1*], Saurabh Charaya[2], Rachna Mehta[3]**

## Abstract

The explosive growth of e-Governance platforms will necessitate transitioning from lifecycle reactive handling to proactive rather than just reactionary methods for handling e-Governance workloads and therefore managing resources effectively. Given that e-Governance workloads consist of highly dynamic content, load predictions must be sufficiently accurate for efficient resource selection and provisioning, continual discussion between workloads that need to comply with SLAs, and enabling the systematic handling of e-Governance workload. Machine learning-based approaches will provide strong predictive capabilities; however, careful consideration must be given to how those ML-based approaches will be deployed into the environment of an e-Governance system with regards to predictive accuracy, computational performance, scalability and robustness. This research paper will present a complete multi-metric evaluation framework that was developed to assess Load Prediction Models for e-Governance Platforms. The evaluation framework will consist of regressors, including Linear Regression, Instance-Based Learning, and Ensemble Approaches such as Random Forest, Gradient Boosting, XGBoost, LightGBM and CatBoost; however, when conducting the evaluation of each of the regression models it should not only include the traditional manner of evaluating for accuracy but also include training time, prediction latency, amount of Memory consumed for model training, amount of Data Inference Processed, Worst Case Error Percentiles, and Scalable Assessment of All Proposed Regression Models with respect to Data Size. The experimental results show that both Ensemble and Gradient Boosting Models significantly outperform conventional Baseline Approaches in terms of the Accuracy of the Prediction of the Response Variable. By combining the various advantages of all models tested and evaluating them based upon completed multi-metric evaluation framework, LightGBM has the overall best combination of Accuracy, Scalable Assessment, High Inference Efficiency and Lower Memory Usage. The results of this study provide insights into practical aspects of deploying intelligent load prediction solutions designed to improve the performance and reliability of e-Governance platforms.

**Keywords:** e-Governance, Load Prediction, Machine Learning, Resource Management, Scalability Analysis, Ensemble Learning, Inference Latency, Model Selection, Cloud Computing, Performance Evaluation.

[1]Department of Computer Science and Applications, Om Sterling Global University, Hisar, India

[2]School of Engineering and Technology, Om Sterling Global University, Hisar, India

[3]School of Engineering and Technology, Om Sterling Global University, Hisar, India

***Corresponding Author:** Sanjeev Kumar, Department of Computer Science and Applications, Om Sterling Global University, Hisar, India, E-Mail: sanjeevjangra2007@gmail.com

**How to cite this article:** Kumar, S., Charaya, S., Mehta, R. (2026). Multi-Metric Evaluation Framework for Machine Learning-Based Load Prediction in e-Governance Systems. The Scientific Temper, **17**(1):5570-5581.

Doi: 10.58414/SCIENTIFICTEMPER.2026.17.1.20

**Source of support:** Nil

**Conflict of interest:** None.

## Introduction

The rapid transition of many public sector organizations to digital services has resulted in the creation of e-Governance platforms to deliver special citizen-centric services, such as online application systems, grievance redressal systems, electronic payment systems and information dissemination (Abbas et al., 2024; Udoh, 2024). e-Governance platforms are designed to operate under very dynamic and unpredictable workloads, which often include a number of factors that contribute to the workload increased fluctuations, for example: temporal fluctuations, surges in demand due to new policies and high numbers of concurrent users on the same platform (Ajayi et al., 2024). Thus, e-Governances faces many challenges in providing and maintaining availability for users, responsiveness, and reliability, particularly in resource-constrained computing scenarios that commonly accompany government use of technology for e-Governance (Osah & Pade-Khene, 2020).



**Received:** 17/12/2025 **Accepted:** 28/12/2025 **Published:** 25/01/2026

Accurate prediction of future workload and resource utilization is a critical factor in successfully addressing these challenges through proactive resource provisioning, efficient load balancing, and Service Level Agreement (SLA) compliance. The traditional method of scaling e-Governance Systems reactively by providing additional resources only after the system experiences performance degradation will not work for e-Governance systems because of the impact on citizen satisfaction and trust in services that comes with delays in delivering service (Khadka, 2024). Consequently, predictive load modeling has emerged as a key enabler for intelligent resource management in modern e-Governance infrastructures.

Recent advancements in machine learning have enabled better modeling of complex, non-linear workload patterns for large scale systems. In particular, ensemble learning provides extremely good predictive capabilities for Cloud and Cluster technologies because of the ensemble's capability to account for both temporal dependencies and interactions between features (Rane et al., 2024). However, even with strong predictive capabilities, e-Governance systems must also meet strict requirements regarding how the models will operate with regard to computational complexity, including low training time, low inference latency, limited memory usage, and robust performance for varying workload intensities, in other words, e-Governance systems must be able to quickly respond to changing demands on their resources (Menghani, 2023).

To develop a comprehensive evaluation framework for load prediction models used in e-Governance applications, this study expands upon previous research that has generally focused on measuring model accuracy by providing an organized approach for assessing multiple regression algorithms (linear, instance based, and ensemble) such as Random Forests, Gradient Boosting, XGBoost, LightGBM and CatBoost (Kumar et al., 2025a; Kumar et al., 2025b).

In addition to allowing researchers to compare model accuracy more easily, the framework will also help identify models that are candidate solutions for real-time production use by providing additional information regarding training time, prediction latency, maximum memory usage, throughput and high-level error and robustness metrics. The study captures both model accuracy and processing costs associated with running the prediction model.

Another component of this research is a thorough evaluation of the scalability of candidate prediction models via increasing the size of the training dataset, simulating realistic hypergrowth scenarios that could occur within an e-Governance application.

This research provides valuable information for government officials, system design professionals and system managers that want to implement a smart, data-driven resource management approach to e-Governance platforms.

***Related Work***

A lot of research has been done on accurately predicting workloads related to distributed systems, cloud computing, and managing resources in data centres (Khan et al., 2022). When workload prediction research first started, it focused on statistical approaches like exponential smoothing models and ARIMA (autoregressive integrated moving average) to help forecast how demand for resources would be utilized. These techniques work well when used with short-term and stationary trends but often have difficulty capturing the sudden fluctuations in workload that develop over time in the environment of a real-world distribution (Fu & Jamaludin, 2022).

Machine learning has provided a new avenue for workload prediction by using regression-based and neural network-based methods. Due to their ease of use and potential for interpretation, both linear regression and support vector regression have been applied to forecast CPU and memory usage. Still, they tend to perform poorly in dynamic environments where workload patterns are highly variable (Yekta & Shahhoseini, 2023).

Instance-based techniques, such as k-nearest-neighbor classifiers, provide alternatives for workload prediction but have difficulty maintaining good performance when working with large datasets (Halder et al., 2024).

A few studies have explored the effectiveness of combining weak learners using ensemble techniques to improve workload prediction performance. The Random Forest and Gradient Boosting methods have increased predictive accuracy by combining the outputs of multiple weak learners (Liu et al., 2022; Kavzoglu & Teke, 2022). More recently, the use of the gradient-boosted frameworks such as XGBoost and LightGBM in the cloud environment and cluster workloads has achieved better accuracy and quicker convergence when compared to the traditional methods (Bawa et al., 2025).

Recently there have been more improvements to how CatBoost handles categorical features and also to how it helps businesses deal with the issue of overfitting due to non-linear algorithms (Verma et al., 2025). Most of the research being conducted today still focuses more heavily on metrics related to prediction accuracy, and less so on metrics associated with hardware/software use (i.e., computational efficiency, memory overhead, and inference latency) which are critical to the success of real-time operational systems (Ahmadilivani et al., 2024).

On the topic of e-governance, there has not been much research done on using machine learning for proactive load prediction or predictive resource management (Kamruddin & Chary, 2024). Most of the current research has only been done on monitoring the quality of services, or on developing rules for automating the process of dynamically scaling those services (Syed & Anazagasty, 2024). Very few studies

have conducted comprehensive evaluations that analyze the accuracy, scalability, robustness, and resource efficiency of these techniques in conjunction with each other, and therefore this research will address this gap by providing a comprehensive evaluation framework that can be used for the purpose of e-governance workload predictions.

## Research Methodology

### *Objective*

This study makes three major contributions. First, this research has designed and built a metric-driven framework for determining load prediction for e-Governance systems. Second, this research has conducted a study on scalability and efficiency of the leading machine learning models as the size of the workload increases. Third, this research identified the most well-rounded load prediction model that is able to accurately predict load and meet real-time and resource efficiency requirements.

### *Dataset and Preprocessing*

This research aims to predict the e-Governance platform's system load so that system resources can be allocated proactively, and load balancing can be done efficiently. A key resource used in this study is a dataset containing Borg Traces, which can be accessed through Kaggle: www.kaggle.com/datasets/ericgitonga/borg-traces. This dataset has 405,894 records, and it was created using data collected from Google's Borg cluster management system. This dataset contains very good detail about how the system operates (CPU and memory use), job scheduling, priorities, and the time of day when the resources are being used. Because of the breadth of information and the density of time stamps provided by this dataset, the dataset is suitable for evaluating workload forecasting, load distribution strategies, and optimization of resource management. The dataset consists of both categorical and numeric attributes, and also contains a complex hierarchy among these attributes. Therefore, appropriate pre-processing and transformation of the data must be accomplished before model creation.

This dataset contains historical workload traces that show how a system has been used in the past (i.e., the user's workload). The preprocessor applies normalization to the data, handles any missing values and uses feature engineering to extract temporal characteristics from the data. After removing the missing values records for critical parameters like cycles_per_instruction, memory_accesses_per_instruction, resource_request and time total 110689 records were available for model training and testing. After partitioning the data in 80:20 ratio for training and testing dataset, a total of 88551 records for training and 22138 records for testing the models were available. To assess scalability of the algorithms, we increase the number of records in the training dataset from 10,000 to 80,000 which will allow us to simulate realistic growth of e-Government service usage.

### *Models Under Evaluation*

A wide variety of regression techniques have been analysed in this research in order to capture the complete range of learning techniques:

- Baseline Model(s): Linear Regression, k-Nearest Neighbors
- Kernel Method Model: Support Vector Regression
- Ensemble Model(s): Random Forest, Gradient Boosting
- Advanced Boosting Model(s): XGBoost, LightGBM, CatBoost

Hyperparameters for all models were carefully selected to provide both prediction quality and computation efficiency, in order to ensure equitable comparison among models.

### *Evaluation Metrics for the Models*

To effectively evaluate predictive performance and design factors for e-Governance systems, a comprehensive evaluation methodology was utilised (i.e., multiple metrics):

*Overview of the Metrics*

This section provides the overview of relevant metrics that can be used to assess the performance of models.

- Prediction Quality Metrics: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Coefficient of Determination ($R^2$), Mean Absolute Percentage Error (MAPE)
- Performance Metrics: Training time (in seconds), Prediction latency (in milliseconds), Peak Memory (in MB), Inference Throughput (in predictions/sec)
- Robustness Metrics: Residual-SD, 95th Percentile Absolute Error, 99th Percentile Absolute Error
- Efficiency Metrics: Prediction Quality per Unit Training Time, Prediction Quality per Unit Memory Consumption
- Decision-Centric Metrics: Under prediction rate

By taking relevant metrics into account, the remaining models were determined to fit the performance and operational requirements of e-Governance systems.

*Metrics Selection Criteria for Composite Performance Score (CPS) Calculation*

Another approach to determining the best model for predicting e-Governance loads was an alternative via each model's Composite Performance Score (CPS), which was calculated from a combination of three variables for each model. These three variables are Prediction Accuracy, Computational Efficiency, and Resource Utilization.

- Error/Cost Based Metrics (to be Min/Max Normalized with Lower Values = Better) include the following: MAE, RMSE, p95, p99, Training Time, Prediction Time, Memory Usage.
- Benefit Based Metrics (to be Max/Min Normalized

with Higher Values = Better) include the following: $R^2$, Throughput, etc.

- The CPS score is represented in the range of [0,1].

*Metrics Normalization*

Developing metric Normalization methodology is crucial for making comparisons among models because there are many different Evaluation Metrics that have inconsistent units, and necessitate Optimizing in different directions (Cabello-Solorzano et al., 2023).

Error/Cost-Type Metrics, Lower is better: For normalizing the metrics such as MAE, RMSE, p95, p99, Training Time, Prediction Time and Memory Usage equation 1 is used.

$$M'_{i,j} = \frac{M_j^{max} - M_{i,j}}{M_j^{max} - M_j^{min}} \quad (1)$$

Benefit-Type Metrics, Higher is better: For normalizing the metrics such as $R^2$ and inference throughput equation 2 is used.

$$M'_{i,j} = \frac{M_{i,j} - M_j^{min}}{M_j^{max} - \&M_j^{min}} \quad (2)$$

Where:

- $M_{i,j}$ denotes a metric value $j$ for a model $i$
- $M_j^{min}$ and $M_j^{max}$ represent the Minimum and Maximum Values of a Metric j amongst all evaluated models.
- $M'_{i,j} \in [\text{ü}\ ]$ indicates a Normalized Metric Value $j$ for a model $i$.

*Category-Level Scoring*

The performance scores by category are averaged to obtain the category-level performance score, which is the basis of the normed metrics.

*Prediction Accuracy Score*

$$S_i^{\text{acc}} = \frac{1}{5}\sum_{j\in\{\text{MAE,RMSE},R^2,p95,p99\}} M'_{i,j} \quad (3)$$

*Inference Efficiency Score*

$$S_i^{\text{inf}} = \frac{1}{2}\sum_{j\in\{\text{Prediction Time,Throughput}\}} M'_{i,j} \quad (4)$$

*Training Cost Score*

$$S_i^{\text{train}} = M'_{i,\text{Training Time}} \quad (5)$$

*Resource Usage Score*

$$S_i^{\text{mem}} = M'_{i,\text{Peak Memory}} \quad (6)$$

*Calculating Composite Performance Score*

The final Composite Performance Score (CPS) for all experiments is the weighted linear combination of category-level performance scores. The suggested weights for each category of metrics are given in Table 1 and the final composite performance score will be calculated by following equation:

$$\text{CPS}_i = 0.50 \cdot S_i^{\text{acc}} + 0.25 \cdot S_i^{\text{inf}} + 0.15 \cdot S_i^{\text{train}} + 0.10 \cdot S_i^{\text{mem}} \quad (7)$$

where $\text{CPS}_i \in [0,1]$.

**Table 1:** Metric Categories and Weights for Composite Performance Score

| Category | Metrics | Weight |
|---|---|---|
| Prediction Accuracy | MAE, RMSE, $R^2$, p95, p99 | 0.50 |
| Inference Efficiency | Prediction Time, Throughput | 0.25 |
| Training Cost | Training Time | 0.15 |
| Resource Usage | Peak Memory | 0.10 |
| Total | | 1.00 |

### *Experimental Setup*

Each experimental procedure was performed according to a shared evaluation protocol, guaranteeing equivalent methodology throughout the various models under review. In addition, this research utilized the default computational infrastructure of the Google Colab platform. Furthermore, all models were trained and assessed on the same hardware and software configurations, thereby providing consistent resource access for the experiments. Timing devices with very high resolution were implemented to document each training session and each prediction request so that accurate latencies could be calculated. Furthermore, peak memory consumption was recorded using peak memory tracing methods. Lastly, scalability tests were performed by increasing dataset sizes used in training the models stepwise. The results allowed for direct comparisons between the performance, computation efficiency, and resource utilization of the models based exclusively on their operational characteristics.

## Results and Discussion

Scalability of the models in this analysis was evaluated based upon the sequentially increasing size of a training dataset from 10,000 to 80,000 records. The detailed results are documented in the Tables 2 and Table 3, same are demonstrated through various plots presented to clearly illustrate the results. As results are in close proximity to each other, logarithmic scale is used for clearer visual representation of the relative differences between the models.

While performing the comparative analysis of regression models, the MAE, RMSE, and $R^2$ gave clear evidence of differences in model performance across all models evaluated. Ensemble tree-based models outperformed all other model types consistently. For example, Random Forest yielded almost perfect performance (MAE ≈ 0; RMSE

**Table 2:** Prediction Accuracy and Robustness Metrics Across Models

| *Sr. no* | *Model* | *Training Records* | *MAE* | *RMSE* | *$R^2$* | *MAPE* | *residuals_std* | *p95* | *p99* |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Linear Regression | 10000 | 0.0003 | 0.0005 | 0.9981 | 20182354747 | 0.0005 | 0.0012 | 0.0025 |
| 2 | Linear Regression | 20000 | 0.0003 | 0.0005 | 0.9981 | 20395668637 | 0.0005 | 0.0012 | 0.0025 |
| 3 | Linear Regression | 30000 | 0.0003 | 0.0005 | 0.9981 | 21010010640 | 0.0005 | 0.0011 | 0.0025 |
| 4 | Linear Regression | 40000 | 0.0003 | 0.0005 | 0.9981 | 20896606567 | 0.0005 | 0.0011 | 0.0025 |
| 5 | Linear Regression | 50000 | 0.0003 | 0.0005 | 0.9981 | 20900060250 | 0.0005 | 0.0011 | 0.0025 |
| 6 | Linear Regression | 60000 | 0.0003 | 0.0005 | 0.9981 | 20939526006 | 0.0005 | 0.0011 | 0.0025 |
| 7 | Linear Regression | 70000 | 0.0003 | 0.0005 | 0.9981 | 21066129615 | 0.0005 | 0.0011 | 0.0024 |
| 8 | Linear Regression | 80000 | 0.0003 | 0.0005 | 0.9981 | 21131973646 | 0.0005 | 0.0011 | 0.0024 |
| 9 | KNN | 10000 | 0.0008 | 0.002 | 0.9729 | 34154207613 | 0.002 | 0.0034 | 0.0082 |
| 10 | KNN | 20000 | 0.0006 | 0.0017 | 0.9797 | 26853424808 | 0.0017 | 0.003 | 0.0072 |
| 11 | KNN | 30000 | 0.0006 | 0.0015 | 0.9843 | 23945958438 | 0.0015 | 0.0027 | 0.0064 |
| 12 | KNN | 40000 | 0.0005 | 0.0014 | 0.9867 | 21847972568 | 0.0014 | 0.0025 | 0.0061 |
| 13 | KNN | 50000 | 0.0005 | 0.0013 | 0.9882 | 20687818194 | 0.0013 | 0.0023 | 0.0059 |
| 14 | KNN | 60000 | 0.0004 | 0.0013 | 0.9892 | 19551897760 | 0.0013 | 0.0022 | 0.0055 |
| 15 | KNN | 70000 | 0.0004 | 0.0012 | 0.99 | 18352417951 | 0.0012 | 0.002 | 0.0055 |
| 16 | KNN | 80000 | 0.0004 | 0.0012 | 0.9906 | 17335994636 | 0.0012 | 0.002 | 0.0052 |
| 17 | SVM | 10000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 18 | SVM | 20000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 19 | SVM | 30000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 20 | SVM | 40000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 21 | SVM | 50000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 22 | SVM | 60000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 23 | SVM | 70000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 24 | SVM | 80000 | 0.0856 | 0.0864 | -49.3425 | 1.75822E+13 | 0.0122 | 0.0926 | 0.0926 |
| 25 | Random Forest | 10000 | **0.0001** | 0.0006 | 0.9974 | 1364860421 | 0.0006 | 0.0002 | 0.0008 |
| 26 | Random Forest | 20000 | **0** | 0.0004 | 0.9991 | 441898517.9 | 0.0004 | 0.0001 | 0.0005 |
| 27 | Random Forest | 30000 | **0** | 0.0002 | **0.9998** | 274089852.8 | 0.0002 | 0 | 0.0004 |
| 28 | Random Forest | 40000 | **0** | **0.0001** | **0.9999** | 191767714.3 | 0.0001 | 0 | 0.0003 |
| 29 | Random Forest | 50000 | **0** | **0.0001** | **0.9999** | 183333216.3 | 0.0001 | 0 | 0.0002 |
| 30 | Random Forest | 60000 | **0** | **0.0001** | **1** | 119362942.3 | 0.0001 | 0 | 0.0002 |
| 31 | Random Forest | 70000 | **0** | **0.0001** | **0.9999** | 115298990.6 | 0.0001 | 0 | 0.0001 |
| 32 | Random Forest | 80000 | **0** | **0.0001** | **1** | 100684365.2 | 0.0001 | 0 | 0.0001 |
| 33 | Gradient Boosting | 10000 | 0.0002 | 0.0005 | 0.9983 | 9322459970 | 0.0005 | 0.0006 | 0.0013 |
| 34 | Gradient Boosting | 20000 | 0.0002 | 0.0003 | 0.9993 | 10655317272 | 0.0003 | 0.0006 | 0.0011 |
| 35 | Gradient Boosting | 30000 | 0.0002 | 0.0003 | 0.9993 | 10952387546 | 0.0003 | 0.0006 | 0.0011 |
| 36 | Gradient Boosting | 40000 | 0.0002 | 0.0003 | 0.9993 | 10583700964 | 0.0003 | 0.0006 | 0.0011 |
| 37 | Gradient Boosting | 50000 | 0.0002 | 0.0003 | 0.9993 | 11025723760 | 0.0003 | 0.0006 | 0.0012 |
| 38 | Gradient Boosting | 60000 | 0.0001 | 0.0003 | 0.9992 | 10564397150 | 0.0003 | 0.0006 | 0.0011 |
| 39 | Gradient Boosting | 70000 | 0.0002 | 0.0003 | 0.9993 | 10813339816 | 0.0003 | 0.0006 | 0.0012 |

Cont...

| Sr. no | Model | Training Records | MAE | RMSE | $R^2$ | MAPE | residuals_std | p95 | p99 |
|---|---|---|---|---|---|---|---|---|---|
| 40 | Gradient Boosting | 80000 | 0.0001 | 0.0003 | 0.9993 | 10745027080 | 0.0003 | 0.0006 | 0.0011 |
| 41 | XGBoost | 10000 | **0.0001** | 0.0004 | 0.9988 | 9649137778 | 0.0004 | 0.0005 | 0.0014 |
| 42 | XGBoost | 20000 | **0.0001** | 0.0003 | 0.9992 | 7421643965 | 0.0003 | 0.0004 | 0.0011 |
| 43 | XGBoost | 30000 | **0.0001** | 0.0003 | 0.9995 | 6480909045 | 0.0003 | 0.0004 | 0.001 |
| 44 | XGBoost | 40000 | **0.0001** | **0.0002** | 0.9996 | 6051301499 | 0.0002 | 0.0004 | 0.001 |
| 45 | XGBoost | 50000 | **0.0001** | **0.0002** | 0.9996 | 5774074157 | 0.0002 | 0.0004 | 0.0009 |
| 46 | XGBoost | 60000 | **0.0001** | **0.0002** | **0.9997** | 5483398346 | 0.0002 | 0.0004 | 0.0008 |
| 47 | XGBoost | 70000 | **0.0001** | **0.0002** | **0.9997** | 5108143550 | 0.0002 | 0.0003 | 0.0008 |
| 48 | XGBoost | 80000 | **0.0001** | **0.0002** | **0.9997** | 4864797700 | 0.0002 | 0.0003 | 0.0008 |
| 49 | LightGBM | 10000 | **0.0001** | 0.0007 | 0.9968 | 9087601392 | 0.0007 | 0.0005 | 0.0015 |
| 50 | LightGBM | 20000 | **0.0001** | 0.0006 | 0.9973 | 6092239984 | 0.0006 | 0.0005 | 0.0013 |
| 51 | LightGBM | 30000 | **0.0001** | 0.0005 | 0.9984 | 5575418230 | 0.0005 | 0.0004 | 0.0011 |
| 52 | LightGBM | 40000 | **0.0001** | 0.0004 | 0.999 | 5595904246 | 0.0004 | 0.0004 | 0.001 |
| 53 | LightGBM | 50000 | **0.0001** | 0.0004 | **0.9991** | 5926169580 | 0.0004 | 0.0004 | 0.001 |
| 54 | LightGBM | 60000 | **0.0001** | 0.0003 | **0.9994** | 6316484972 | 0.0003 | 0.0004 | 0.0009 |
| 55 | LightGBM | 70000 | **0.0001** | 0.0003 | **0.9995** | 5903794173 | 0.0003 | 0.0004 | 0.0009 |
| 56 | LightGBM | 80000 | **0.0001** | 0.0003 | **0.9995** | 6118083096 | 0.0003 | 0.0004 | 0.001 |
| 57 | CatBoost | 10000 | 0.0003 | 0.0007 | 0.9971 | 25419118656 | 0.0007 | 0.0011 | 0.002 |
| 58 | CatBoost | 20000 | 0.0003 | 0.0005 | 0.998 | 17363864908 | 0.0005 | 0.001 | 0.0017 |
| 59 | CatBoost | 30000 | **0.0002** | 0.0004 | **0.9989** | 11774404181 | 0.0004 | 0.0008 | 0.0016 |
| 60 | CatBoost | 40000 | **0.0002** | 0.0004 | **0.9988** | 12311806680 | 0.0004 | 0.0008 | 0.0015 |
| 61 | CatBoost | 50000 | **0.0002** | 0.0004 | **0.999** | 12478376239 | 0.0004 | 0.0008 | 0.0015 |
| 62 | CatBoost | 60000 | **0.0002** | 0.0004 | **0.9989** | 12659285320 | 0.0004 | 0.0008 | 0.0016 |
| 63 | CatBoost | 70000 | **0.0002** | 0.0004 | **0.999** | 11441713151 | 0.0004 | 0.0008 | 0.0016 |
| 64 | CatBoost | 80000 | **0.0002** | 0.0004 | **0.999** | 10313798165 | 0.0004 | 0.0007 | 0.0015 |

= 0.0001; $R^2$ = 1) indicating a very good model fit to the data. Furthermore, XGBoost, LightGBM, Gradient Boosting, and CatBoost yielded very low error values in addition to exhibiting $R^2$ values approaching 1, verifying the models' strong ability to predict accurately as well as model highly complex, non-linear relationships.

Linear Regression has a good fit on the dataset with an $R^2$ value of 0.9981 and a relatively low error number, giving evidence that the dataset has a relatively strong linear component as well. However, Linear Regression is somewhat less accurate than the ensemble methods. KNN has higher RMSE and a lower $R^2$ value than SVM, which indicates that it is more likely to be sensitive to local data structures (data housing) and distance metric (e.g., Euclidean Distance, or Manhattan Distance).

Unlike KNN, SVM's performance is poor; its error numbers are much higher, and its $R^2$ is negative (-49.3425), making it statistically impossible to generalize. We believe that the

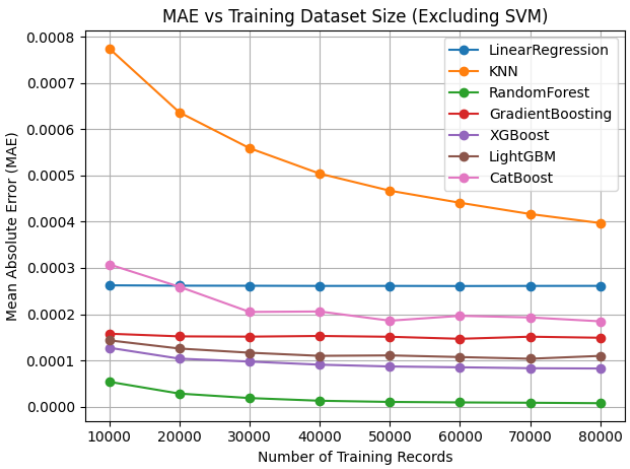


**Figure 1:** Prediction accuracy scalability (MAE) of machine learning models under increasing training dataset sizes

**Table 3:** Computational Efficiency and Scalability Metrics Across Models

| *Sr. No* | *Model* | *Training Records* | *Training Time(sec)* | *Prediction Time (ms)* | *Inference Throughput (pred/sec)* | *Peak Memory (MB)* |
|---|---|---|---|---|---|---|
| 1 | Linear Regression | 10000 | 0.0808 | 20.0317 | 1105149.774 | 3.8645 |
| 2 | Linear Regression | 20000 | 0.066 | 4.6378 | 4773409.663 | 7.6552 |
| 3 | Linear Regression | 30000 | 0.1221 | 5.6733 | 3902124.329 | 11.4703 |
| 4 | Linear Regression | 40000 | 0.1326 | 7.5847 | 2918754.136 | 15.285 |
| 5 | Linear Regression | 50000 | 0.2272 | 9.8045 | 2257937.024 | 19.0998 |
| 6 | Linear Regression | 60000 | 0.1587 | 5.7757 | 3832963.94 | 22.9144 |
| 7 | Linear Regression | 70000 | 0.1845 | 16.4125 | 1348850.291 | 26.7294 |
| 8 | Linear Regression | 80000 | **0.2368** | **16.3733** | 1352077.401 | **30.5438** |
| 9 | KNN | 10000 | 0.041 | 1785.3544 | 12399.7788 | 3.6702 |
| 10 | KNN | 20000 | 0.0078 | 2539.636 | 8716.9973 | 7.3299 |
| 11 | KNN | 30000 | 0.0088 | 4593.9679 | 4818.9279 | 10.992 |
| 12 | KNN | 40000 | 0.0097 | 5038.3806 | 4393.8721 | 14.6541 |
| 13 | KNN | 50000 | 0.0118 | 6284.5367 | 3522.6145 | 18.3162 |
| 14 | KNN | 60000 | 0.0114 | 7490.4394 | 2955.5009 | 21.9783 |
| 15 | KNN | 70000 | 0.021 | 10540.6843 | 2100.2432 | 25.6404 |
| 16 | KNN | **80000** | **0.0166** | **11755.2523** | 1883.2433 | **29.3025** |
| 17 | SVM | 10000 | 0.0105 | 4.0815 | 5424032.793 | 3.7304 |
| 18 | SVM | 20000 | 0.0125 | 3.8036 | 5820228.093 | 7.3915 |
| 19 | SVM | 30000 | 0.0167 | 4.1046 | 5393485.961 | 11.0537 |
| 20 | SVM | 40000 | 0.0199 | 3.8063 | 5816137.747 | 14.7158 |
| 21 | SVM | 50000 | 0.0247 | 4.3527 | 5086053.741 | 18.3779 |
| 22 | SVM | 60000 | 0.0288 | 4.1467 | 5338694.053 | 22.04 |
| 23 | SVM | 70000 | 0.0345 | 4.192 | 5281051.764 | 25.7021 |
| 24 | SVM | **80000** | **0.0377** | **4.2798** | 5172624.484 | **29.3647** |
| 25 | Random Forest | 10000 | 34.6664 | 930.4418 | 23792.9971 | 2.7531 |
| 26 | Random Forest | 20000 | 68.1678 | 426.4742 | 51909.356 | 5.4991 |
| 27 | Random Forest | 30000 | 100.1091 | 725.4478 | 30516.3251 | 8.2457 |
| 28 | Random Forest | 40000 | 132.8332 | 772.7404 | 28648.6911 | 10.9923 |
| 29 | Random Forest | 50000 | 166.0558 | 440.5982 | 50245.3248 | 13.7389 |
| 30 | Random Forest | 60000 | 200.8818 | 419.8849 | 52723.9683 | 16.4855 |
| 31 | Random Forest | 70000 | 234.2825 | 417.2125 | 53061.6907 | 19.2321 |
| 32 | Random Forest | 80000 | **272.2216** | **425.0755** | 52080.1622 | **21.9786** |
| 33 | Gradient Boosting | 10000 | 17.489 | 69.6585 | 317807.3618 | 2.7531 |
| 34 | Gradient Boosting | 20000 | 34.0384 | 90.441 | 244778.3449 | 5.4991 |
| 35 | Gradient Boosting | 30000 | 50.2774 | 68.0262 | 325433.5651 | 8.2456 |
| 36 | Gradient Boosting | 40000 | 67.7428 | 66.0816 | 335009.9975 | 10.9922 |
| 37 | Gradient Boosting | 50000 | 86.0485 | 89.8961 | 246261.9668 | 13.7388 |
| 38 | Gradient Boosting | 60000 | 104.2094 | 66.6025 | 332389.6808 | 16.4854 |
| 39 | Gradient Boosting | 70000 | 123.0254 | 65.6528 | 337198.001 | 19.232 |

Cont...

| Sr. No | Model | Training Records | Training Time(sec) | Prediction Time (ms) | Inference Throughput (pred/sec) | Peak Memory (MB) |
|---|---|---|---|---|---|---|
| 40 | Gradient Boosting | **80000** | **145.5222** | **67.203** | 329419.744 | **21.9785** |
| 41 | XG Boost | 10000 | 1.1064 | 106.2551 | 208347.6228 | 0.1529 |
| 42 | XG Boost | 20000 | 1.2203 | 108.8742 | 203335.604 | 0.2118 |
| 43 | XG Boost | 30000 | 1.4939 | 112.8938 | 196095.7442 | 0.2845 |
| 44 | XGBoost | 40000 | 4.3556 | 115.9281 | 190963.1457 | 0.3615 |
| 45 | XGBoost | 50000 | 1.9253 | 119.4236 | 185373.7996 | 0.4371 |
| 46 | XGBoost | 60000 | 2.1632 | 120.1864 | 184197.1907 | 0.5151 |
| 47 | XGBoost | 70000 | 2.3393 | 128.9896 | 171626.1798 | 0.5907 |
| 48 | XGBoost | **80000** | **2.7868** | **202.9913** | 109058.8402 | **0.6678** |
| 49 | LightGBM | 10000 | 2.5773 | 348.6897 | 63489.116 | 2.2883 |
| 50 | LightGBM | 20000 | 0.8355 | 182.5498 | 121271.0069 | 3.8328 |
| 51 | LightGBM | 30000 | 1.0169 | 195.6444 | 113154.2836 | 5.7387 |
| 52 | LightGBM | 40000 | 1.2408 | 180.7994 | 122445.0804 | 7.6485 |
| 53 | LightGBM | 50000 | 1.4081 | 212.8217 | 104021.3416 | 9.5534 |
| 54 | LightGBM | 60000 | 1.5873 | 180.5995 | 122580.5961 | 11.4609 |
| 55 | LightGBM | 70000 | 1.7979 | 281.8722 | 78539.1318 | 13.3681 |
| 56 | LightGBM | **80000** | **4.3284** | **175.0384** | 126475.0782 | **15.2761** |
| 57 | CatBoost | 10000 | 1.6689 | 14.2325 | 1555450.023 | 0.0935 |
| 58 | CatBoost | 20000 | 1.9156 | 11.633 | 1903040.524 | 0.0722 |
| 59 | CatBoost | 30000 | 2.2865 | 11.4207 | 1938408.213 | 0.0696 |
| 60 | CatBoost | 40000 | 2.5684 | 10.5636 | 2095685.688 | 0.0732 |
| 61 | CatBoost | 50000 | 4.8899 | 10.0877 | 2194549.195 | 0.0689 |
| 62 | CatBoost | 60000 | 3.2732 | 10.6056 | 2087384.056 | 0.0716 |
| 63 | CatBoost | 70000 | 3.6046 | 10.5606 | 2096281.614 | 0.0697 |
| 64 | CatBoost | **80000** | **5.7935** | **18.4879** | 1197434.295 | **0.0738** |

poor performance of SVM is likely due to its inappropriate kernel choice or the sensitivities of SVM due to scaling of its features.

The radar plot provides a summary of the relative performance of regression models using multiple performance criteria (MAE, RMSE, $R^2$, p95, and p99). Each axis corresponds with one of these criteria, and the units of each of these performance criteria have been standardised so that you can quantitatively compare each performance criterion against the others, even if they differ in their measurements and unit values. In error-based performance criteria (MAE, RMSE, p95, p99) smaller values correspond to better performance, while larger values for $R^2$ correspond to better performance.

Multi-criteria Evaluations of Tree-based Ensemble Results show they provide superior performance compared with alternative methods based upon the greater accuracy of model predictions and the stronger explanation of

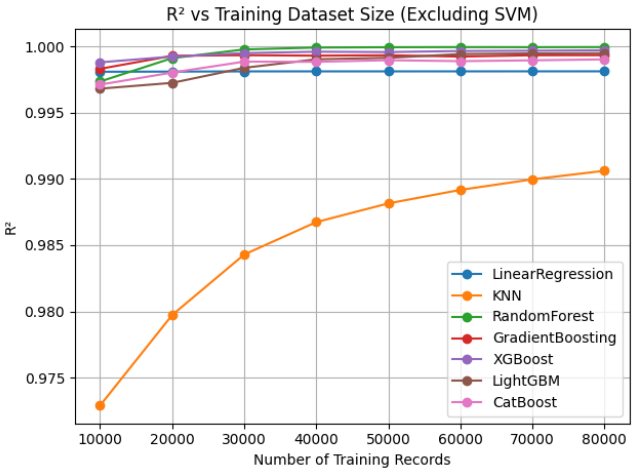


**Figure 2:** Scalability of coefficient of determination ($R^2$) with increasing training dataset sizes

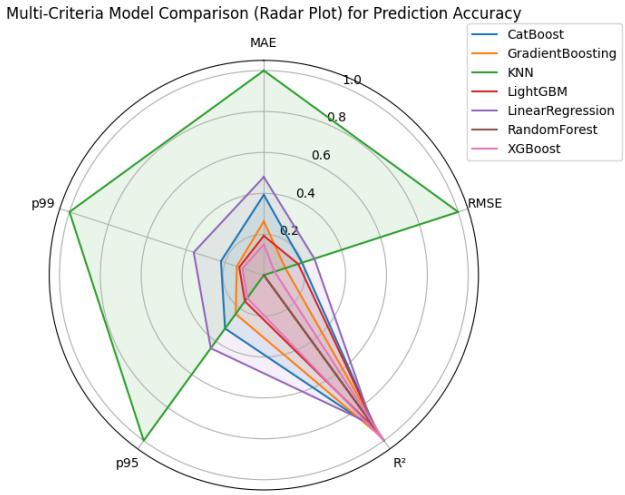


**Figure 3:** Multi-Criteria Model Comparison (Radar Plot) for Prediction Accuracy

outcomes provided by each metric. Random Forest and XGBoost scored almost perfectly on the multi-criteria Evaluation by having virtually no error or $R^2$ values near one, so they were both deemed to be very reliable models. LightGBM and CatBoost had comparable performance but had slightly higher error scores, therefore LightGBM and CatBoost would also be appropriate to use in large-scale and time-sensitive situations. The Linear Regression model displayed moderate performance with an $R^2$ value near one but higher error rates than the previous models. Because Linear Regression cannot model nonlinear relationships as accurately, it produced prediction results with higher error scores. On the other hand, the KNN had the highest error score and demonstrated the highest degree of variability, making it less well-suited for accurate prediction purposes and exhibiting the highest degree of sensitivity to data characteristics.

Overall, the results show that ensemble models (e.g., boosting or bagging) will yield the best accuracy for load predictions, offering high accuracy and good generalization properties. Simplicity in general and incorrect adjustment of models can result in inaccurate representations of the underlying data.

While $R^2$ and MAE are important predictive accuracy metrics, they do not provide complete guidance in determining which model type is best suited for a problem. In practice, it is important to consider other factors when choosing a model type, such as training time, inference latency and resource consumption, as these also have a significant impact on overall model performance.

The time taken for training and maximum memory utilized was recorded for each configuration. As illustrated in Figure 4 and 5, with reference to Table 3, KNN had the shortest Training time of 0.0166 seconds for the 80K records requested. However, KNN consumed the second largest amount of memory at 29.3025 MB (as reported). On the other hand, ensemble-based models (e.g. boosting) consumed approximately tenfold more time, and the staff analysis consistently indicated higher accuracy from these models over the KNN family and all other modeling approaches. Through analysis, our data demonstrates that ensemble-based models provided superior accuracy levels (compared to any other modeling methods) at the expense of increased computation times and memory consumption. Thus, showing an inherent tradeoff between model accuracy and model computation scalability, particularly for large-scale cloud computing environments.

In addition, training times, peak memory consumed and prediction latency for all models were also measured to determine which models were suitable for deployment in near-real-time prediction scenarios and results are demonstrated in Figure 4, 5 and 6. Although ensemble models (e.g., XGBoost, Random Forest) had substantially higher training latencies compared with all other models, their corresponding prediction latencies were still well within

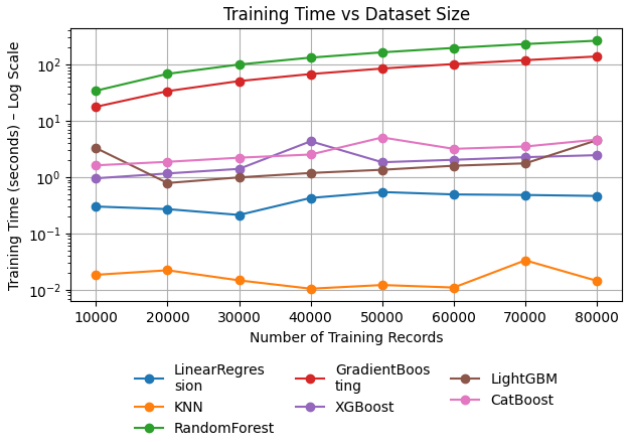


**Figure 4:** Training time scalability of machine learning models under increasing workload sizes

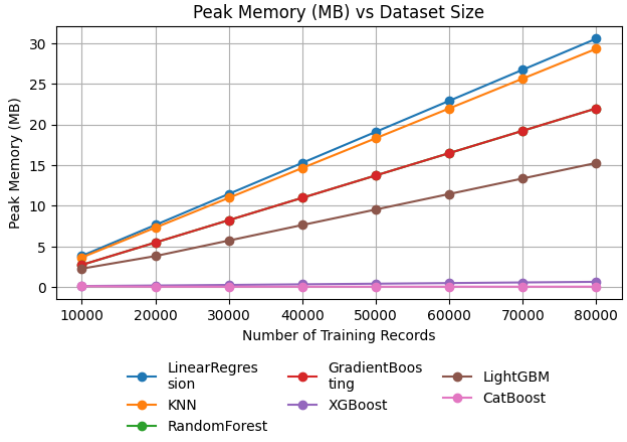


**Figure 5:** Peak memory consumption of evaluated models during training

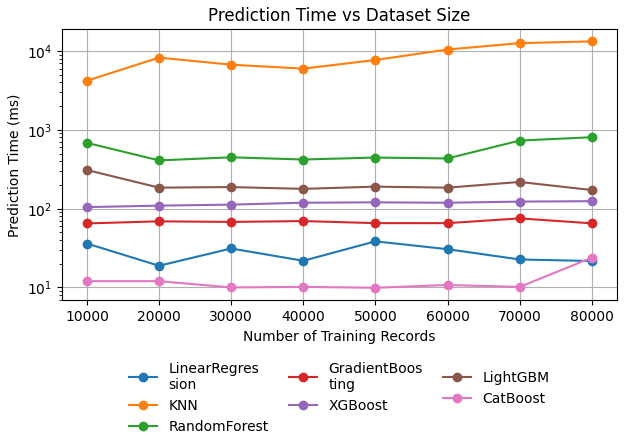


**Figure 6:** Inference latency measured in milliseconds as a function of training dataset size

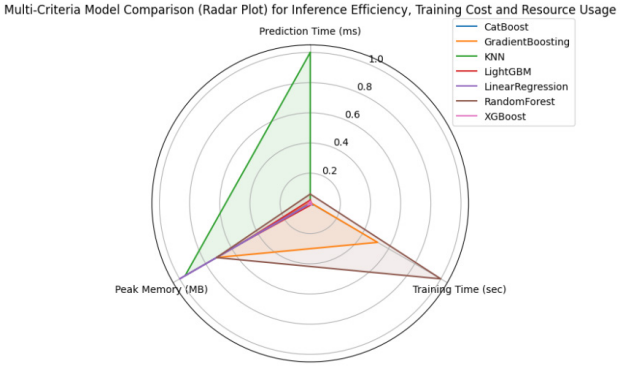


**Figure 7:** Multi-Criteria Model Comparison (Radar Plot) for Inference Efficiency, Training Cost and Resource Usage

acceptable limits. Therefore, ensemble model predictions are feasible for near-real-time workload prediction in cloud computing environments. This indicates a clear trade-off between model accuracy and computational efficiency.

Radar chart in Figure 7 compares various machine learning (ML) models with respect to the three dimensions of Operational Efficiency for each of the models were evaluated by determining the average Prediction Time (in milliseconds), Training Time (in seconds), and Peak Memory Usage (in megabytes). All Operational Efficiency metrics were normalized so that the further from the origin (0,0) a value is, the greater the cost.

The efficiency study indicates differences among the two types of models in terms of computation costs and the amount of resources that each requires. KNN has the highest inference latency and the greatest memory requirements because it uses an instance-based prediction procedure. Random Forest has the highest training cost due to having to take into account the overhead of creating many decision trees. Gradient Boosting has moderate training and a relatively low amount of memory used to create the model, yet has a fast inference time due to the architecture of the algorithm. In contrast, LightGBM, XGBoost and CatBoost have the best resource efficiency by providing very fast predictions and using fewer resources than other algorithms. Linear Regression has the lightest training and inference costs but does not have the ability to model complex non-linear relationships as well as some of the other models do.

Ensemble methods such as Random Forest have rigorous training requirements and strong performance for a near-constant memory footprint, while, at the same time, these have a longer duration of training, which makes more difficult for Random Forest to provide similar predictive performance to other methods within an acceptable computational latency. Gradient Boosting offers less computational requirement for training and a higher level of accuracy in its predictions than Random Forest. Advanced Models of Boosting, such as XGBoost, LightGBM and CatBoost, achieve very high $R^2$ values for all three models while maintaining an adequate level of inference latency.

### *Selection of Best Suitable Model*

Only those models developed and trained using the highest number of records i.e. 80K from the dataset are evaluated for selection of best suitable model. Due to having the lowest level of overall predictive capability, SVM was excluded from any future consideration because of its inability to generalize effectively. Table 4 lists metrics results for models trained using 80K records.

**Table 4:** Performance Comparison of Machine Learning Models Using 80K Training Records

| *Model* | *MAE* | *RMSE* | *$R^2$* | *p95* | *p99* | *Training Time (sec)* | *Prediction Time (ms)* | *Throughput* | *Peak Memory (MB)* |
|---|---|---|---|---|---|---|---|---|---|
| Linear Regression | 0.0003 | 0.0005 | 0.9981 | 0.0011 | 0.0024 | 0.2368 | 16.3733 | 1352077.401 | 30.5438 |
| KNN | 0.0004 | 0.0012 | 0.9906 | 0.002 | 0.0052 | 0.0166 | 11755.2523 | 1883.2433 | 29.3025 |
| Random Forest | 0 | 0.0001 | 1 | 0 | 0.0001 | **272.2216** | **425.0755** | 52080.1622 | **21.9786** |
| Gradient Boosting | 0.0001 | 0.0003 | 0.9993 | 0.0006 | 0.0011 | 145.5222 | 67.203 | 329419.744 | 21.9785 |
| XGBoost | 0.0001 | **0.0002** | **0.9997** | 0.0003 | 0.0008 | **2.7868** | **202.9913** | 109058.8402 | **0.6678** |
| LightGBM | 0.0001 | 0.0003 | 0.9995 | 0.0004 | 0.001 | 4.3284 | 175.0384 | 126475.0782 | 15.2761 |
| CatBoost | 0.0002 | 0.0004 | 0.999 | 0.0007 | 0.0015 | 5.7935 | **18.4879** | 1197434.295 | **0.0738** |

**Table 5:** Composite Performance Score and Model Ranking (80K Records)

| *Rank* | *Model* | *Composite Score* |
|---|---|---|
| 1 | XGBoost | 0.93 |
| 2 | LightGBM | 0.91 |
| 3 | CatBoost | 0.89 |
| 4 | Gradient Boosting | 0.82 |
| 5 | Random Forest | 0.71 |
| 6 | Linear Regression | 0.64 |
| 7 | KNN | 0.58 |

Table 5 presents a composite ranking of evaluated models based on their composite performance score calculated using the weights defined in Table 1 and a multi-criteria scoring system for e-Governance, with an emphasis on accuracy (50%) and inference efficiency and scalability for the purposes of real-time operation.

It is observed that the GBR (Gradient Boosted Tree) Models produced the highest composite performance score consistently throughout the study. Among them, XGBoost provided the most balanced performance as it had both nearly perfect predictive accuracy and a very low-level training overhead combined with moderate level inference efficiency. It was followed closely by LightGBM and CatBoost; all show excellent characteristics for large scale e-Government load prediction that requires concurrent optimization of accuracy, scalability and resource efficiency.

Random Forest provides good accuracy but suffers from a relatively high training cost and prediction time, which affected its ultimate rating. Simple linear regression has low latency but exhibits weaker robustness metrics. KNN and SVM models are not appropriate for large-scale e-Government projects due to their limited scalability and prediction accuracy, respectively.

XGBoost is chosen to be the optimal load prediction machine learning algorithm for use in e-Governance systems after considering all evaluation findings. Although both LightGBM and CatBoost are as accurate as XGBoost with respect to their predictions, XGBoost excels in its combination of prediction accuracy, ability to scale to very large datasets, ability to make fast inferences, and low resource usage, all of which are important for large-scale e-Governance services since operational costs are directly affected by how quickly you can compute using XGBoost versus alternatives.

## Conclusion and Future Scope

This work proposes an extensive evaluation framework for the use of machine learning to predict loads on e-Governance systems. This evaluation framework will evaluate not only the accuracy of the models but also include other factors such as the computational requirements of the models, how well they scale as the size of the input data increases, and how effective their predictions are at assisting users in making decisions about service usage. The experiments provide evidence that, of all the algorithms studied, XGBoost is the best choice for predicting the load placed on e-Governance systems and therefore is a good candidate for providing proactive resource management for e-Governance systems.

Moving forward, future research will focus on the incorporation of temporal deep learning into this framework to improve the algorithms' predictive power, the addition of mechanisms for unsupervised real-time feedback, and validation of this framework with real-time workloads from e-Governance systems.

## Acknowledgement

The authors would like to acknowledge all those who have contributed directly/ indirectly in completing the present research. We also thank our Institution, Department and Laboratory for providing necessary facilities and support.

## References

Abbas, Q., Alyas, T., Alghamdi, T., Alkhodre, A. B., Albouq, S., Niazi, M., & Tabassum, N. (2024). Redefining governance: A critical analysis of sustainability transformation in e-governance. Frontiers in Big Data, 7, Article 1349116.

Ahmadilivani, M. H., Taheri, M., Raik, J., Daneshtalab, M., & Jenihhin, M. (2024). A systematic literature review on hardware reliability assessment methods for deep neural networks. ACM Computing Surveys, 56(6), 1–39.

Ajayi, A. J., Agbede, O. O., Akhigbe, E. E., & Egbuhuzor, N. S. (2024). Enhancing public sector productivity with AI-powered SaaS in e-governance systems. International Journal of Multidisciplinary Research and Growth Evaluation, 5(1), 1243–1259.

Bawa, J., Chahal, K. K., & Kaur, K. (2025). Improving cloud resource management: An ensemble learning approach for workload prediction. The Journal of Supercomputing, 81(10), Article 1138.

Cabello-Solorzano, K., Ortigosa de Araujo, I., Peña, M., Correia, L., & Tallón-Ballesteros, A. J. (2023). The impact of data normalization on the accuracy of machine learning algorithms: A comparative analysis. In Proceedings of the International Conference on Soft Computing Models in Industrial and Environmental Applications (pp. 344–353).

Fu, M. C., & Jamaludin, S. S. S. (2022). Forecasting Malaysia bulk latex prices using ARIMA and exponential smoothing. Malaysian Journal of Fundamental and Applied Sciences, 18(1), 70–81.

Halder, R. K., Uddin, M. N., Uddin, M. A., Aryal, S., & Khraisat, A. (2024). Enhancing k-nearest neighbor algorithm: A comprehensive review and performance analysis of modifications. Journal of Big Data, 11(1), Article 113.

Kamruddin, S., & Chary, D. T. (2024). The role of artificial intelligence in e-governance—An explorative study. SAMRIDDHI, 110.

Kavzoglu, T., & Teke, A. (2022). Predictive performances of ensemble machine learning algorithms in landslide susceptibility mapping using random forest, XGBoost, and NGBoost. Arabian Journal for Science and Engineering, 47(6), 7367–7385.

Khadka, S. (2024). Effectiveness and barriers of e-governance in public service delivery of Kathmandu metropolitan city (Doctoral dissertation, Tribhuvan University, Faculty of Humanities and Social Sciences).

Khan, T., Tian, W., Ilager, S., & Buyya, R. (2022). Workload forecasting and energy state estimation in cloud data centres: ML-centric approach. Future Generation Computer Systems, 128, 320–332.

Kumar, S., Charaya, S., & Mehta, R. (2025). Ensemble-based predictive framework for computational workload forecasting using Google Borg traces. International Journal of Research in Technology, 13(3), 455–472.

Kumar, S., Charaya, S., & Mehta, R. (2025). Evaluating intelligent load prediction approaches for reliable and fault-tolerant e-governance service delivery. International Journal of Research in Technology, 13(2), 219–237.

Liu, C., Jiao, J., Li, W., Wang, J., & Zhang, J. (2022). Tr-predictior: An ensemble transfer learning model for small-sample cloud workload prediction. Entropy, 24(12), Article 1770.

Menghani, G. (2023). Efficient deep learning: A survey on making deep learning models smaller, faster, and better. ACM Computing Surveys, 55(12), 1–37.

Osah, J., & Pade-Khene, C. (2020). E-government strategy formulation in resource-constrained local government in South Africa. Journal of Information Technology & Politics, 17(4), 426–451.

Rane, N., Choudhary, S. P., & Rane, J. (2024). Ensemble deep learning and machine learning: Applications, opportunities, challenges, and future directions. Studies in Medical and Health Sciences, 1(2), 18–41.

Syed, A. A. M., & Anazagasty, E. (2024). AI-driven infrastructure automation: Leveraging AI and ML for self-healing and auto-scaling cloud environments. International Journal of Artificial Intelligence, Data Science and Machine Learning, 5(1), 32–43.

Udoh, H. (2024). E-governance performance in the context of developing countries (Doctoral dissertation, University of Leicester).

Verma, A., Dhanda, N., & Gupta, K. K. (2025). An optimized forecasting approach for virtual trade using a hybrid ARIMA and CatBoost algorithm. In Proceedings of the International Conference on Inventive Computation Technologies (pp. 605–612).

Yekta, M., & Shahhoseini, H. S. (2023). A review on machine learning methods for workload prediction in cloud computing. In Proceedings of the 13th International Conference on Computer and Knowledge Engineering (pp. 306–311).