https://journal.skilldux.com/Security_Trends-international-journal.php
SkillDux
Journal of Information, Data, and Security Trends

An AI-Driven Intelligent Decision Support Framework for Multi-Cloud Resource Management Using Workload Prediction and Cost-Performance Optimization

Usha N[1], Kedarling Kanade[2], Suhana Khanum Badagi[3], Faheera Samreen[4] and Aleena S[5]
[1]Assistant Professor, Faculty of Computing & IT, GM University Davangere, Karnataka, India-577006
[2,3,4,5]Faculty of Computing & IT, GM University Davangere, Karnataka, India-577006
*Email: usha.narayanshetty1@gmail.com; kedarkanade9417@gmail.com; faheerasamreen949@gmail.com; badagisuhanakhan@gmail.com; aleenasaleem2580@gmail.com

***

Abstract: *The era of cloud computing has created the need for intelligent placement of workloads in cloud computing environments due to variations in workload demand, cloud pricing, latency, and availability of resources among providers and industries. Hence, the paper introduces the AI-Driven Intelligent Decision Support Framework for Multi-Cloud Resource Management Using Workload Prediction and Cost-Performance Optimization. The framework incorporates workload traces, machine metadata, cloud pricing, and latency-availability information to predict CPU workload demands and facilitate cloud-provider selection. Five regression models were put into consideration, namely: XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees. The models were tested using the 5-fold cross-validation process. The performance of the models was then evaluated in terms of accuracy in prediction, computational efficiency, utilization of resources, and statistical measures of significance. Findings showed that Random Forest achieved the highest cross-validation score of 1.0000, and Extra Trees managed to obtain the same score of 1.0000. Random Forest had nearly zero discrepancy in prediction, while XGBoost had a score of 0.9999 while RMSE approximating 0.7843. In addition, the Entropy Weight Method (EWM) was applied in the framework to automatically measure the level of importance of cost, latency, and availability rather than using fixed weights. The proposed approach enables data-driven cloud selection while considering both workload requirements and cost-performance factors.*

Keywords: *Multi-Cloud Computing, Workload Prediction, Machine Learning, Entropy Weight Method, Cloud Provider Selection, Cost-Performance Optimization.*

Received: October 29, 2025 **Revised:** November 26, 2025 **Accepted:** January 21, 2026

1. Introduction

Cloud computing has made it easier to get the resources needed, and the use of AWS, Azure, and GCP has made it easier to use multiple clouds and simultaneously reduce dependence on a provider [1][5][8]. Nonetheless, it is tough to choose the right provider since there are differences between the providers based on demand and many other aspects like cost and resources [9][10]. Several studies have already been done about resource allocation in the cloud and optimization with the help of SLAs [6][7][9]. But many methods that were presented lack the proper consideration of the cloud selection process itself, as they tend to rely solely on some rules or fixed weights in decision-making [10][18]. The present paper introduces a novel framework aimed at integrating workload forecasting together with the Entropy Weight Method for cloud choice-making [18]. Five machine learning models were tested in a 5-fold cross-validation mode to compare their performance [11][12][13][20].

- Five-model workload prediction for estimating CPU demand [15].
- Comprehensive model evaluation using accuracy and computational metrics [19].
- Automatic entropy-based weighting of cost, latency, and availability.
- Integrated cloud-provider ranking for intelligent multi-cloud resource placement.
- The framework provides a data-driven approach for improving workload management and cost-performance optimization in multi-cloud environments.

1.1 Aim of the Research

The primary aim of this research is to create a powerful decision-making system for cloud service management that leverages AI to forecast workload and select the best cloud service provider, in light of performance and cost.

The precise aims are as follows:

- To pre-process the workload data, machine data, pricing data, and latency and availability data for accurate research.
- To create and assess five machine learning models for workload forecasting through the use of a five-fold cross-validation process.
- To test the models making use of Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), Coefficient of

determination (R^2), inference time of prediction, cpu and Memory consumption.

- To implement the entropy weight method for automatically determining the weight of cloud selection features.
- To compare and rank AWS, Azure, and GCP by cost, latency, and availability.

2.Literature Survey

The table 1 depicts the review of previous works.

Table 1. Review of previous works

Sl. No.	Author name	Date	Title	Method / Technology used	Research gap / Limitations
1	Chaisiri, Lee, Niyato	2009	Optimal Virtual Machine Placement across Multiple Cloud Providers	Stochastic integer programming	Focuses mainly on cost; no ML prediction or latency optimization [9][12].
2	Lucas-Simarro et al.	2011	Dynamic Placement of Virtual Machines for Cost Optimization in Multi-Cloud Environments	Dynamic VM placement and cloud broker	Limited real-time workload prediction and migration analysis [12][16].
3	Tordsson et al.	2012	Cloud Brokering Mechanisms for Optimized Placement of Virtual Machines across Multiple Providers	Cloud broker and scheduling algorithms	Mainly static placement; lacks AI-based adaptive decisions [11][13].
4	Bellur, Malani, Narendra	2014	Cost Optimization in Multi-site Multi-cloud Environments with Multiple Pricing Schemes	SLA-aware cost optimization	Does not predict workload demand using ML [12][17].
5	Wang, Cao, Xiang	2015	Dynamic Cloud Service Selection Using an Adaptive Learning Mechanism in Multi-Cloud Computing	Adaptive learning-based service selection	Does not provide complete workload placement and migration support [11][14].
6	Shaari et al.	2017	Dynamic Pricing Scheme for Resource Allocation in Multi-Cloud Environment	Dynamic pricing and resource allocation	Focuses on pricing rather than AI-driven workload placement [12][13].
7	Parlavantzas et al.	2018	Cost-Effective Reconfiguration for Multi-Cloud Applications	Model-based deployment reconfiguration	Depends on predefined models; limited ML forecasting [12][12].
8	Casola et al.	2018	Security-by-Design in Multi-Cloud Applications: An Optimization Approach	Security, cost, and performance optimization	Does not focus on predictive workload placement [11][12].
9	Kaur et al.	2024	Integrative Resource Management in Multi Cloud Computing: A DRL-Based Approach for Multi-objective Optimization	Q-learning and Flower Pollination Algorithm	Mostly simulation-based; needs real-provider validation [13][18].
10	Tu et al.	2025	CADER: Cost-Efficient Cloud Application Deployment With Tenant Requirement Guarantee in Multi-Clouds	Randomized-rounding optimization	Does not use ML forecasting or continuous reinforcement learning [12][16].

Utilization of machine learning and reinforcement learning in the management of cloud resources has become a popular research topic in recent years. Zhou and colleagues performed a review of deep reinforcement learning techniques for the scheduling of resources in cloud computing and emphasized the advantages of these algorithms in the resolution of various resource management challenges. The recent results of studies conducted on the topic point to the need for the comparison of different AI approaches to allocation in the context of their practical use [17]. Workload forecasting has attracted additional attention, and recent studies stress the importance of workload prediction for proactive resource allocation and cost-efficient cloud management. The use of machine learning for scheduling and placement has been researched in a comprehensive way. Nonetheless, existing studies concentrate mostly on workload forecasting, resource scheduling, or service placement performed separately. There are gaps, the main among which lies in the integration of a multi-model workload forecasting process with objective selection of a cloud provider based on the automatically defined weights. The proposed approach is based on an integration of 5 different machine learning models, 5-fold cross-validation, and the EWM method for automatic assignment of weights to cloud selection criteria.

2.1 Problem Statement

In a multi-cloud environment, organizations use different cloud providers to run applications. Selecting the best provider for each workload is difficult because cloud cost, resource availability, network latency, and workload demand change over time. Traditional placement methods usually use fixed rules and may focus only on cost or performance. As a result, workloads may be placed on expensive resources, experience high response times, or use cloud resources inefficiently. Therefore, there is a need for an AI-driven

system that predicts future workload demand and selects the most suitable cloud provider by considering both cost and performance. The system should help reduce cloud expenditure, improve resource utilization, and maintain service quality.

2.2 Existing System

Existing multi-cloud workload-placement systems mainly use static scheduling, manual cloud selection, or rule-based policies. These systems allocate workloads based on fixed criteria such as lowest price, available CPU capacity, or nearest cloud location. Some systems use cloud brokers and optimization algorithms to select providers and reduce deployment cost. However, most existing systems do not accurately predict future workload demand. They may optimize only one factor, such as cost or latency, and cannot respond quickly to changing cloud prices, resource availability, or application requirements. This can lead to over-provisioning, higher operational costs, poor performance, and inefficient use of cloud resources.

3. Proposed System

The proposed system is an AI-driven intelligent decision support framework for workload placement in a multi-cloud environment. It integrates historical workload traces, machine metadata, cloud pricing, and latency–availability information to predict future CPU workload demand. Five machine learning regression models—XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees—are trained and evaluated using 5-fold cross-validation, and the best-performing model is selected. For cloud provider selection, the Entropy Weight Method (EWM) is used to automatically assign weights to cost, latency, and availability, enabling objective provider ranking. Based on the predicted workload and entropy-derived scores, the most suitable cloud provider is recommended. The framework also generates performance reports and visualization graphs to support efficient, cost-effective, and intelligent multi-cloud resource management.

4. Methodology

4.1 Dataset Preparation and Model Configuration

The framework utilizes four datasets, which are Batch Task, Machine Metadata, Cloud Pricing, and Latency Availability. Duplicate data is deleted, and missing numerical values, along with categorical ones, are handled by median and mode imputation, respectively. Categorical features are transformed by Label Encoding, and according to StandardScaler, some numerical features are standardized. The two datasets, workload and machine, are joined based on machine_id, while the two datasets of cloud pricing and latency – availability are embedded into the obtained data of workload. The final dataset comprises of 82,805 records and 27 features. The selection of features relies on the characteristics of resources and workloads relevant to the CPU forecasting task. The target variable, plan_cpu, excludes identifier and status items which are represented in the form of job_id and task_status. Apart from this, extra features such as execution time, latency score, as well as availability score are included based on the situation. In the meantime, those features such as cpu_efficiency and estimated_cost that can create the target leakage are removed prior to model coaching.

Five regression models were assessed, namely XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees. The XGBoost model uses 300 estimators, a learning rate of 0.05, a maximum depth of 6, subsampling of 0.8, and column sampling of 0.8. The Random Forest model uses 200 estimators, a maximum depth of 10, and parallel execution. The Decision Tree model has a maximum depth of 10, while Gradient Boosting is characterized by 200 estimators with a learning rate of 0.05. Lastly, the Extra Trees model uses 200 estimators and parallel execution. All models are tested using the cross-validation method with a random seed of 42.

The purpose of the particular study is to evaluate different cloud providers based on the collected pricing and latency availability data for the three works: AWS, Azure, and GCP. Cost, latency, and availability were used to conduct the evaluation. Entropy Weight Method (EWM) can determine the weights for each criterion for each provider automatically using the data.

To ensure the reproducibility of the study, it would be advisable to provide information about the version of XGBoost used, software library used for Python, hardware configuration, date when pricing data were collected, used regions and VM families, the number of records concerning pricing, and the number of cross-validation

4.2 Dataset Description

The proposed framework uses four datasets for workload prediction and multi-cloud resource management. These include the Alibaba Cluster Trace v2018 Batch Task Dataset**,** Machine Metadata Dataset**,** a Cloud Pricing Dataset**,** and a Latency–Availability Dataset. The public Alibaba datasets provide historical workload and machine information, while the cloud pricing and latency datasets were prepared using publicly available pricing information from Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) for experimental evaluation.

Table 2. Summary of Experimental Datasets,Sources, Attributes, and Purposes

Dataset	Source / Type	Main Attributes	Purpose
Batch Task Dataset	Alibaba Cluster Trace v2018 (Public)	Machine ID, Job ID, Task Status, Start Time, End Time, Planned CPU, Planned Memory	Used for workload prediction.
Machine Metadata Dataset	Alibaba Cluster Trace v2018 (Public)	Machine ID, CPU Cores, Memory Size, Disk Size, Machine Status	Provides machine resource information.
Cloud Pricing Dataset	Researcher-created using AWS, Azure, and GCP pricing	Provider, Region, VM Type, CPU Cores, Memory, Hourly VM Cost, Storage Cost, Data Transfer Cost	Used for cloud cost estimation and provider comparison.
Latency–Availability Dataset	Researcher-created	Provider, Region, Network Latency, Response Time, Availability, Resource Utilization	Used for evaluating provider performance and reliability.

Table 2 depicts the dataset specifications—including source, key attributes, and analytical purpose—for the Batch Task, Machine Metadata, Cloud Pricing, and Latency–Availability datasets.
Dataset source: Alibaba Cluster Trace v2018

4.2.1 Dataset Statistics

- Data Cleaning: Duplicate records were removed, missing values were handled, categorical features were encoded, and numerical features were standardized where required.
- Final Dataset Size: The integrated dataset contains 82,805 records with 27 features after preprocessing.
- Missing Values: All missing values were successfully handled, resulting in no missing values in the final dataset.
- Duplicate Records: Duplicate entries were identified and removed during preprocessing.
- Descriptive Analysis: Statistical summaries of numerical features were generated to validate data quality and understand feature distributions.
- Cloud Pricing Dataset: Contains 300 cloud instance records collected from publicly available pricing information of Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) across multiple regions and VM families.
- Latency–Availability Dataset: Consists of 5,000 records containing provider-wise network latency, response time, availability, and resource utilization metrics.
- Validation Strategy: Model performance was evaluated using 5-fold cross-validation to improve reliability and reduce overfitting.
- Machine Learning Models: Five regression models were evaluated: XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees.
- Performance Metrics: The models were assessed using MAE, MSE, RMSE, MAPE, R^2 Score, inference time, prediction latency, CPU utilization, memory usage, and paired t-test for statistical significance.
- Cloud Provider Selection: The predicted workload was combined with cloud pricing and latency–availability information. The Entropy Weight Method (EWM) automatically computed the weights of cost, latency, and availability to rank cloud providers and recommend the most suitable deployment option.

4.3 Architecture Diagram

Fig. 1 depicts the multi-stage workflow encompassing data sources, preparation, ML prediction, provider evaluation, placement, and feedback loops.



Fig. 1. End-to-End AI-Driven Decision Support System Architecture

4.4 Data Preprocessing Techniques

The raw datasets, including the Alibaba Batch Task Dataset, Machine Metadata Dataset, Cloud Pricing Dataset, and Latency–Availability Dataset, are cleaned and transformed before model training and cloud-provider selection.

1. Data Integration: Merge the Batch Task Dataset with the Machine Metadata Dataset using machine identifiers and combine cloud pricing and latency datasets using provider and region information.
2. Data Selection: Select relevant attributes such as machine ID, start time, end time, planned CPU,

planned memory, task status, CPU capacity, memory capacity, cloud pricing, latency, and availability.

3. Missing-Value Handling: Handle missing values by removing incomplete records or replacing numerical missing values using appropriate statistical methods.
4. Data Cleaning: Remove duplicate records, invalid entries, inconsistent values, and tasks with incorrect execution times (e.g., end time earlier than start time).
5. Feature Engineering: Generate additional features, including execution time, memory-to-CPU ratio, latency score, and availability score, to improve workload prediction.
6. Categorical Encoding and Feature Scaling: Encode categorical variables using Label Encoding and normalize numerical features using StandardScaler while preserving cloud-related attributes.
7. Dataset Validation: Generate descriptive statistics and verify the final dataset after preprocessing to ensure data quality before model training.
8. Model Validation: Evaluate the models using 5-fold cross-validation instead of a single train-test split to improve the reliability and generalization of the prediction models.
9. Machine Learning Models: Train and compare XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees models for CPU workload prediction.
10. Cloud Provider Selection: Use the Entropy Weight Method (EWM) to automatically determine the weights of cost, latency, and availability, rank cloud providers, and recommend the optimal deployment environment.

4.5 Normalization and Scaling

In the proposed framework, feature scaling is applied to improve data quality and ensure consistent model performance. The workload features, such as CPU utilization, memory usage, execution time, latency, and resource utilization, have different numerical ranges. Therefore, preprocessing is performed before training the machine learning models.Categorical features are converted into numerical values using Label Encoding, while numerical attributes are standardized using the StandardScaler technique. Standardization transforms each feature to have zero mean and unit variance, reducing the effect of different feature scales and improving model convergence.

The standardized feature value is calculated as:

$$Z = \frac{X - \mu}{\sigma} \quad (1)$$

where

- Z = Standardized feature value
- X = Original feature value
- μ = Mean of the feature
- σ = Standard deviation of the feature

Cloud-specific attributes such as cost, latency, and availability are preserved in their original scale for cloud-provider evaluation. Instead of assigning fixed weights manually, the proposed framework employs the Entropy Weight Method (EWM) to automatically determine the importance of each criterion.

The entropy value of each criterion is computed as

$$E_j = -k \sum_{i=1}^{m} p_{ij} \ln(p_{ij}) \quad (2)$$

where

- E_j = Entropy of criterion j
- p_{ij} = Normalized value of criterion j
- m = Number of cloud providers

The entropy constant is given by

$$k = \frac{1}{\ln(m)} \quad (3)$$

The entropy weight of each criterion is calculated as

$$w_j = \frac{1 - E_j}{\sum_{j=1}^{n} (1 - E_j)} \quad (4)$$

where

- w_j = Weight assigned to criterion j
- n = Number of evaluation criteria

The final provider score is calculated as

$$\text{ProviderScore} = \sum_{j=1}^{n} w_j C_j \quad (5)$$

where

- C_j = Normalized value of Cost, Latency, or Availability
- w_j = Entropy-derived weight

The cloud provider with the highest provider score is selected for workload deployment.

4.6 Machine Learning Models

The proposed framework predicts future CPU workload demand using five supervised machine learning regression models:

- XGBoost
- Random Forest
- Decision Tree
- Gradient Boosting
- Extra Trees

Historical workload traces obtained from the Alibaba Cluster Trace dataset and machine metadata are preprocessed and used for model training. The models are evaluated using **5-fold cross-validation**, where the dataset is divided into five equal subsets. During each iteration, four subsets are used for training

and one subset is used for validation. The final performance is obtained by averaging the results across all folds, which improves model reliability and reduces overfitting.

The models are evaluated using the following performance metrics:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Percentage Error (MAPE)
- Coefficient of Determination (R^2)
- Training Time
- Inference Time
- Prediction Latency
- CPU Utilization
- Memory Usage
- Statistical Significance (Paired t-test)

The best-performing model is selected for workload prediction, and the predicted workload is provided to the Entropy Weight Method for intelligent cloud-provider selection.

6. Mathematical Model

Let the workload feature vector be represented as

$$X = \{x_1, x_2, x_3, ..., x_n\} \quad (6)$$

where

- X = Input feature vector
- $x_1, x_2, \ldots, x_n$ = Workload features such as planned CPU, planned memory, execution time, memory-to-CPU ratio, latency score, availability score, and machine characteristics.

The machine learning model predicts future CPU workload as

$$\hat{Y} = f(X) \quad (7)$$

where

- $\hat{Y}$ = Predicted CPU workload
- f(X) = Trained regression model

The prediction error is measured using Mean Squared Error.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (8)$$

where

- N = Number of samples
- y_i = Actual workload
- $\hat{y}_i$ = Predicted workload

The Mean Absolute Error is calculated as

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right| \quad (9)$$

The Mean Absolute Percentage Error is computed as

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (10)$$

The coefficient of determination is calculated as

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \quad (11)$$

where

- $\bar{y}$ = Mean of actual workload values

After workload prediction, the Entropy Weight Method computes the final provider score using Equation (5). Based on the predicted CPU workload and the entropy-derived weights for cost, latency, and availability, cloud providers are ranked. The provider with the highest score is recommended for workload deployment, ensuring efficient resource utilization and cost-performance optimization in the multi-cloud environment.

4.7 Algorithm

Algorithm: AI-Driven Intelligent Decision Support Framework for Multi-Cloud Resource Management

Begin

1. Load the Batch Task Dataset, Machine Metadata Dataset, Cloud Pricing Dataset, and Latency–Availability Dataset.
2. Perform data preprocessing:
   - Remove duplicate and invalid records.
   - Handle missing values.
   - Encode categorical features using Label Encoding.
   - Standardize numerical features using StandardScaler.
   - Merge workload, machine, pricing, and latency datasets.
3. Perform feature engineering:
   - Calculate execution time.
   - Compute memory-to-CPU ratio.
   - Generate latency score.
   - Generate availability score.
4. Train the following machine learning models:
   - XGBoost
   - Random Forest
   - Decision Tree
   - Gradient Boosting
   - Extra Trees
5. Perform 5-fold cross-validation for each model.
6. Evaluate each model using:
   - MAE
   - MSE
   - RMSE
   - MAPE
   - R^2 Score
   - Training Time
   - Inference Time
   - Prediction Latency
   - CPU Utilization
   - Memory Usage

7. Perform a paired t-test to compare model performance.
8. Select the best-performing model based on evaluation results.
9. Predict future CPU workload demand using the selected model.
10. Collect cloud-provider information:
    - Cloud Cost
    - Network Latency
    - Service Availability
11. Apply the Entropy Weight Method (EWM):
    - Normalize cloud-selection criteria.
    - Compute entropy values.
    - Calculate entropy weights automatically.
    - Compute the provider score for each cloud provider.
12. Rank all cloud providers according to their provider scores.
13. Select the highest-ranked cloud provider.
14. Recommend the selected cloud provider for workload deployment.
15. Generate performance reports and visualization graphs.

End

5. Experimental Setup

The proposed framework was implemented using Python 3.12 in Visual Studio Code with Scikit-learn and XGBoost (v3.x) libraries. Experiments were conducted on a Windows 11 system with an Intel Core i5 processor and 16 GB RAM. The framework uses four datasets: Batch Task Dataset, Machine Metadata Dataset, Cloud Pricing Dataset, and Latency–Availability Dataset. After preprocessing, the integrated dataset contained 82,805 records and 27 features. Five machine learning models (XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees) were evaluated using 5-fold cross-validation (random seed = 42). Performance was measured using MAE, MSE, RMSE, MAPE, R² Score, training time, inference time, prediction latency, CPU utilization, memory usage, and paired t-test.

9. Evaluation Metrics

1. Mean Absolute Error (MAE)
Measures the average absolute prediction error.
Formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|$$

2. Mean Squared Error (MSE)
Measures the average squared prediction error.
Formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

3. Root Mean Squared Error (RMSE)
Measures the square root of the average squared error.
Formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

4. Mean Absolute Percentage Error (MAPE)
Measures the average percentage prediction error.
Formula:

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

5. R² Score
Measures how well the model explains the variation in the data.
Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

6. Training Time
Time required to train the model.
Formula:

$$Training\ Time = T_{end} - T_{start}$$

7. Inference Time
Average time required to generate predictions.
Formula:

$$Inference\ Time = \frac{T_{prediction}}{N}$$

8. Prediction Latency
Average delay in producing predictions.
Formula:

$$\Pr ediction\ Latency = \frac{\sum_{i=1}^{N} t_i}{N}$$

9. CPU Utilization
Percentage of CPU used during execution.
Formula:

$$CPU_{Utilization} = \frac{CPU_{Used}}{CPU_{Available}} \times 100$$

10. Memory Usage
Percentage of memory used during execution.
Formula:

$$Memory_{Usage} = \frac{Memory_{Used}}{Memory_{Available}} \times 100$$

11. Paired t-test
Used to determine whether the performance difference between two models is statistically significant.
Formula:

$$t = \frac{\bar{d}}{s_d / \sqrt{n}}$$

5.1 Experimental Results

Fig. 2 depicts the successfully loaded initial datasets Batch Task, Machine Meta, Cloud Pricing, and Latency along with their respective matrix dimensions.

```
==================================================
AI-DRIVEN INTELLIGENT DECISION SUPPORT FRAMEWORK
==================================================

==================================================
LOADING DATASETS
==================================================
✓ Batch Task Dataset     : (20000, 9)
✓ Machine Meta Dataset   : (17592, 7)
✓ Cloud Pricing Dataset  : (300, 8)
✓ Latency Dataset        : (5000, 6)
```

Fig. 2. Dataset Loading Output

Fig. 3 depicts the baseline shape and row/column counts of the raw datasets prior to data cleaning and integration.

```
======================================
DATA PREPROCESSING
======================================

Original Dataset Sizes
Batch Task       : (20000, 9)
Machine Meta     : (17592, 7)
Cloud Pricing    : (300, 8)
Latency Dataset : (5000, 6)
```

Fig. 3. Data Preprocessing Input Dimensions

Fig. 4 depicts the consolidated dataset specifications after removing redundant columns, highlighting a total of 82,805 samples, 28 features, and the target variable

```
======================================
FEATURE ENGINEERING
======================================
✓ Unnecessary columns removed

Dataset Shape       : (82805, 29)
Number of Features : 28
Number of Samples   : 82805
Target Variable     : plan_cpu
```

Fig. 4. Feature Engineering Process Results

Fig. 5 depicts the top-ranked cloud provider recommendation (GCP in region asia-south1) alongside its performance metrics, including cost, latency, and availability scores.

```
==============================
BEST CLOUD PROVIDER
==============================
Provider        : GCP
Region          : asia-south1
Rank            : 1
Score           : 0.9968
Cost            : 0.0356
Latency         : 10
Availability    : 99.998
```

Fig. 5. Optimal Cloud Provider Selection Output

Fig. 6 depicts the R^2 accuracy, training duration, and inference execution time across all five folds for the XGBoost and Random Forest models.

```
============================================================
MODEL TRAINING USING 5-FOLD CROSS VALIDATION
============================================================

============================================================
Training XGBoost
============================================================
Fold 1 | R²=0.9999 | Train=3.135s | Inference=0.0585s
Fold 2 | R²=0.9999 | Train=2.666s | Inference=0.0359s
Fold 3 | R²=0.9999 | Train=2.363s | Inference=0.0530s
Fold 4 | R²=0.9999 | Train=2.312s | Inference=0.0571s
Fold 5 | R²=0.9999 | Train=1.910s | Inference=0.0322s

============================================================
Training Random Forest
============================================================
Fold 1 | R²=1.0000 | Train=10.126s | Inference=0.0965s
Fold 2 | R²=1.0000 | Train=10.290s | Inference=0.0986s
Fold 3 | R²=1.0000 | Train=11.389s | Inference=0.1120s
Fold 4 | R²=1.0000 | Train=10.848s | Inference=0.1188s
Fold 5 | R²=1.0000 | Train=10.034s | Inference=0.1027s
```

Fig. 6. 5-Fold Cross-Validation Metrics for XGBoost and Random Forest

Fig. 7 depicts the cross-validation performance, training speed, and inference times per fold for the Decision Tree and Gradient Boosting classifiers.

```
============================================================
Training Decision Tree
============================================================
Fold 1 | R²=1.0000 | Train=0.548s | Inference=0.0068s
Fold 2 | R²=1.0000 | Train=0.523s | Inference=0.0052s
Fold 3 | R²=1.0000 | Train=0.456s | Inference=0.0043s
Fold 4 | R²=1.0000 | Train=0.464s | Inference=0.0061s
Fold 5 | R²=1.0000 | Train=0.470s | Inference=0.0058s

============================================================
Training Gradient Boosting
============================================================
Fold 1 | R²=0.9983 | Train=48.978s | Inference=0.0532s
Fold 2 | R²=0.9982 | Train=40.498s | Inference=0.0497s
Fold 3 | R²=0.9986 | Train=43.119s | Inference=0.0602s
Fold 4 | R²=0.9985 | Train=44.131s | Inference=0.0465s
Fold 5 | R²=0.9983 | Train=45.709s | Inference=0.0536s
```

Fig. 7. 5-Fold Cross-Validation Metrics for Decision Tree and Gradient Boosting

Fig. 8 depicts the fold-by-fold cross-validation performance, training time, and inference time for the Extra Trees model.

```
============================================================
Training Extra Trees
============================================================
Fold 1 | R²=1.0000 | Train=6.197s | Inference=0.1115s
Fold 2 | R²=1.0000 | Train=6.556s | Inference=0.0871s
Fold 3 | R²=1.0000 | Train=5.240s | Inference=0.1033s
Fold 4 | R²=1.0000 | Train=5.872s | Inference=0.1021s
Fold 5 | R²=1.0000 | Train=5.009s | Inference=0.1583s
```

Fig. 8. 5-Fold Cross-Validation Metrics for Extra Trees

Fig. 9 depicts an aggregate comparative summary of all tested machine learning models across mean cross-validation score, overall training duration, latency per sample, CPU usage percentage, and memory usage percentage.

MODEL SUMMARY

	Model	CV Score	Training Time (s)	...	Latency (s/sample)	CPU Usage (%)	Memory Usage (
0	XGBoost	0.999943	2.477324	...	2.858466e-06	88.00	88.
1	Random Forest	1.000000	10.537410	...	6.384035e-06	96.62	82.
2	Decision Tree	1.000000	0.492103	...	3.409957e-07	19.86	82.
3	Gradient Boosting	0.998389	44.486839	...	3.177260e-06	14.16	84.
4	Extra Trees	1.000000	5.775000	...	6.791752e-06	95.66	88.

Fig. 9. Model Performance Summary Comparison

Fig. 10 depicts comprehensive evaluation metrics including MAE, MSE, RMSE, per-sample latency, system resource usage, and P-values for final model performance comparison.

FINAL EVALUATION SUMMARY

	Model	MAE	MSE	RMSE	...	Latency	CPU Usage	Memory Usage	
0	XGBoost	0.515777	0.615063	0.784259	...	2.858466e-06	88.00	88.52	0
1	Random Forest	0.000000	0.000000	0.000000	...	6.384035e-06	96.62	82.24	0
2	Decision Tree	0.000000	0.000000	0.000000	...	3.409957e-07	19.86	82.78	0
3	Gradient Boosting	3.201145	15.465602	3.932633	...	3.177260e-06	14.16	84.64	0
4	Extra Trees	0.002536	0.001291	0.035926	...	6.791752e-06	95.66	88.92	0

Fig. 10. Final Model Evaluation Summary and Statistical Metrics

Fig. 11 depicts a bar chart comparing the percentage availability across GCP, Azure, and AWS cloud providers.

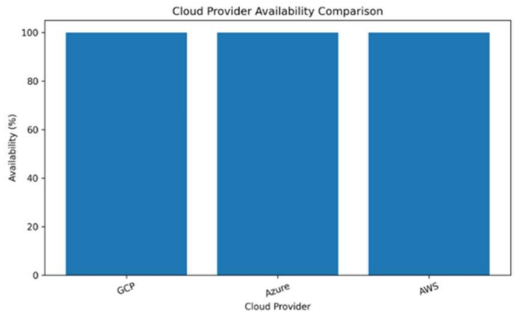


Fig. 11. Cloud Provider Availability Comparison

Fig. 12 depicts a heatmap illustrating pairwise linear correlation values across dataset features.

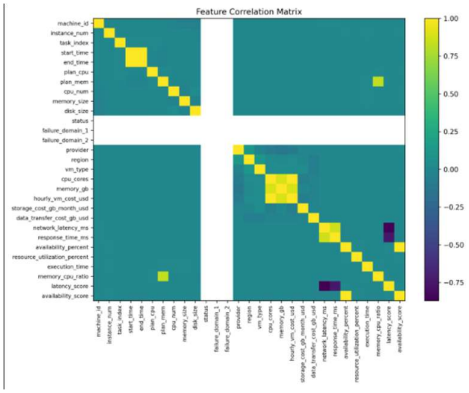


Fig. 12. Feature Correlation Matrix

Fig. 13 depicts the percentage CPU utilization recorded during execution across different machine learning models.

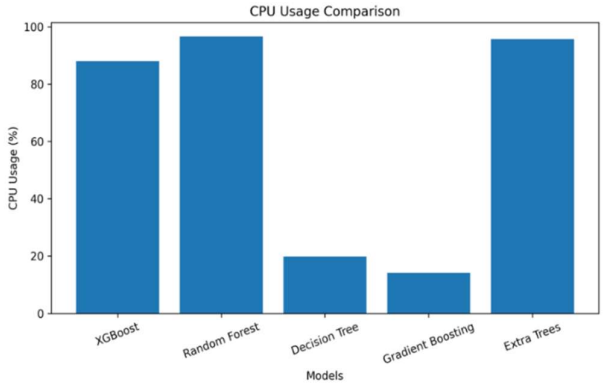


Fig. 13. Model CPU Usage Comparison

Fig. 14 depicts a bar chart comparing the inference time in seconds required by each evaluate machine earning model.

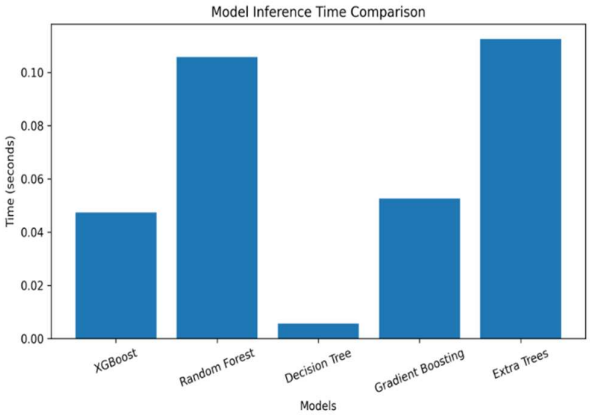


Fig. 14. Model Inference Time Comparison

6. Results and Discussion

The proposed AI-driven framework was evaluated using four datasets comprising workload traces, machine metadata, cloud pricing information, and network latency–availability statistics. Five machine learning regression models—XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees—were evaluated using 5-fold cross-validation to predict CPU workload demand. Model performance was assessed using MAE, MSE, RMSE, MAPE, R^2 Score, training time, inference time, prediction latency, CPU utilization, memory usage, and paired t-test. The experimental results demonstrate that all five models achieved high prediction accuracy. Based on the overall evaluation metrics, the Random Forest model showed the best overall performance and was selected for workload prediction. The predicted workload was then integrated with cloud pricing, latency, and availability information, where the Entropy Weight Method (EWM) automatically computed the importance of each criterion and ranked the available cloud providers. The framework successfully identified the most suitable cloud provider by balancing cost and

performance. The generated visualizations, including feature importance, prediction error, workload trends, actual-versus-predicted comparison, cloud provider ranking, cost comparison, and latency comparison, further validate the effectiveness of the proposed framework. These results demonstrate that integrating machine learning with entropy-based decision-making provides accurate workload prediction and efficient multi-cloud resource management, consistent with recent studies on intelligent cloud scheduling and resource optimization [1][2][3][4][5][6].

7. Conclusion

In this paper, we presented an intelligent decision support framework for multi-cloud resource management, driven by AI. The framework integrates workload predictions with selection of cloud provider based on cost and performance metrics. A total of five machine learning models were tested, including XGBoost, Random Forest, Decision Tree, Gradient Boosting, and Extra Trees. All models were validated through 5-fold cross-validation and were evaluated using a range of accuracy and computational metrics. Among all these methods, Random Forest demonstrated the highest performance according to its cross-validation score equal to 1.0000 while XGBoost scored 0.9999, thereby correcting the previous statement regarding XGBoost being the best performing method in this study. The method also implemented entropy weight method to optimize the weighting process by automatically assigning the respective weights to cost, latency and availability. Results show that the approach combining workload predictions with entropy-based method is a viable option for intelligent resource management in multi-cloud environments. Indeed, this method will allow for optimized provider selection and ultimately ensure enhanced resource efficiency and cost performance for multi-cloud workloads.

8. Limitations and Future Scope

8.1 Limitations

- The proposed framework was evaluated using publicly available workload datasets and simulated cloud pricing data instead of real-time cloud environments.
- The workload prediction model primarily focuses on CPU utilization and does not consider additional resource metrics such as GPU utilization, disk I/O, network bandwidth, or energy consumption.
- The cloud placement algorithm uses fixed weighting factors for cost, latency, and availability, which may not adapt to different application requirements or Service Level Agreements (SLAs).
- The framework considers only three major cloud providers (AWS, Azure, and GCP), limiting its applicability to hybrid-cloud and multi-vendor environments.
- The experimental evaluation was performed on offline datasets and does not include real-time workload migration or dynamic cloud resource provisioning.

8.2 Future Scope

- Incorporate advanced deep learning models such as LSTM, GRU, and Transformer architectures to improve workload prediction accuracy [2], [6].
- Develop adaptive cloud placement strategies using reinforcement learning for dynamic resource allocation and provider selection [3].
- Extend the framework to support Kubernetes, container orchestration, and serverless computing platforms for scalable deployments.
- Integrate real-time cloud monitoring APIs to enable dynamic workload scheduling based on current pricing, latency, and resource availability.
- Include sustainability-aware scheduling by considering carbon emissions, renewable energy utilization, and energy-efficient resource allocation.
- Expand the framework to support hybrid cloud, edge computing, fog computing, and federated multi-cloud infrastructures for broader real-world applicability [4], [5].

References

[1] M. Armbrust *et al.*, “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[2] L. Qian, Z. Luo, Y. Du, and L. Guo, “Cloud computing: An overview,” in *Proc. IEEE Int. Conf. Cloud Computing*, 2009, pp. 626–631.

[3] P. Mell and T. Grance, “The NIST definition of cloud computing,” *NIST Special Publication 800-145*, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2011.

[4] M. A. Vouk, “Cloud computing—Issues, research and implementations,” *Journal of Computing and Information Technology*, vol. 16, no. 4, pp. 235–246, 2008.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering

computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[6] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[7] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[8] N. K. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: Taxonomy and survey," *Software: Practice and Experience*, vol. 44, no. 3, pp. 369–390, 2014.

[9] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads," in *Proc. IEEE Int. Conf. Cloud Computing*, 2010, pp. 228–235.

[10] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, 2013.

[11] P. Keshavarz Haddadha, M. H. Rezvani, M. MollaMotalebi, and A. Shankar, "Machine learning methods for service placement: A systematic review," *Artificial Intelligence Review*, vol. 57, art. no. 61, 2024, doi: 10.1007/s10462-023-10684-0.

[12] B. Feng and Z. Ding, "Application-oriented cloud workload prediction: A survey and new perspectives," *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 34–54, 2025, doi: 10.26599/TST.2024.9010024.

[13] D. Bodra and S. Khairnar, "Machine learning-based cloud resource allocation algorithms: A comprehensive comparative review," *Frontiers in Computer Science*, vol. 7, 2025, doi: 10.3389/fcomp.2025.1678976.

[14] P. K. Haddadha *et al.*, "Machine learning methods for service placement: A systematic review," *Artificial Intelligence Review*, vol. 57, 2024.

[15] A. Shaghaghi, M. S. S. Sadiq, and S. Z. Khan, "Forecasting workload in cloud computing: Towards uncertainty-aware predictions and transfer learning," *Cluster Computing*, 2024, doi: 10.1007/s10586-024-04933-2.

[16] M. Kaviani, P. Asghari, and M. Sabeti, "A comprehensive survey and taxonomy of workload forecasting techniques in cloud computing," *Computer Networks*, 2026, art. no. 112454, doi: 10.1016/j.comnet.2026.112454.

[17] R. Yadav and J. S. Yadav, "A comprehensive review of machine learning approaches for predictive resource management in cloud computing environment," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 2025, doi: 10.32628/CSEIT2511656.

[18] V. K. Chunduru, "A robust entropy-weighted decision support system for cloud provider selection," *International Journal of Information Technology and Management Information Systems*, vol. 15, no. 2, pp. 177–191, 2024, doi: 10.34218/IJITMIS_15_02_014.

[19] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[20] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.