A Decision Tree-Based Cloud Replication Model for Enhanced Data Management

Aws I. Abueid
Faculty of Computing Studies, Arab Open University, Kuwait
a.abueid@arabou.edu.kw (corresponding author)

Received: 26 September 2025 | Revised: 12 April 2025 | Accepted: 19 April 2025

 DOI: https://doi.org/10.48084/etasr.11170

ABSTRACT

This paper introduces the Decision Tree Cloud Replication (DTCR) model, a novel Artificial Intelligence (AI)-based approach for managing data replication in cloud environments. The model is designed to enhance data availability, optimize performance, and reduce resource costs by intelligently deciding when to add or remove replicas. Unlike traditional static replication strategies, DTCR adapts dynamically to system conditions, response time targets, and tenant budget constraints. The methodology involves using supervised learning via Decision Tree (DT) algorithms trained on synthetic datasets generated using the Markov Chain Monte Carlo (MCMC) method to simulate realistic replication scenarios. Hyperparameter tuning is performed through grid search to determine optimal settings, such as maximum tree depth and minimum samples per split, whereas cross-validation ensures a reliable evaluation process. The implementation is carried out using the Weka platform, and model performance is assessed using multiple metrics, including accuracy, precision, recall, F-measure, and the Matthews Correlation Coefficient (MCC). The results indicate that the DTCR model achieves a classification accuracy of 100%, with balanced performance across both target classes, demonstrating its effectiveness in real-time decision-making for replica placement and removal. Further discussion shows that the model generalizes well to unseen data, avoids overfitting through optimal depth control, and maintains high availability with minimal overhead. This confirms the model's potential for integration into cloud scheduling policies, offering practical benefits in fault tolerance, latency reduction, and cost efficiency. The contribution of this work lies in presenting an intelligent, scalable, and cost-aware solution for replication management, which outperforms conventional approaches and addresses critical challenges in cloud-based data systems.

Keywords-Artificial Intelligence (AI); cloud environment; data management; data replication; DTCR model

I. INTRODUCTION

Due to scalability, high availability, and fast access, the cloud is now the best solution for the rising need for storage resulting from the wide-ranging and large-scale incorporation of Internet services and big data. Due to the potential for significant benefits for industry and the community, the cloud computing paradigm is well-known in the academic and industrial sectors [1]. The increased usage of cloud services and their repurposing depend heavily on resource sourcing [2]. Cloud computing enables the construction of numerous massive data centers at various locations around the globe. To ensure efficient operation and reliability across these data centers, data replication is supported as a successful strategy for fault tolerance, latency reduction, and minimized data exchange over a network. As a result, replica management has become a challenging area for suppliers.

Cloud computing is a new concept that provides services, including computing, communication, and storage resources, across a network. However, the delivery of cloud applications and services is often limited by communication resources. Replication strategies efficiently bring data closer to users, enhancing performance and availability.

Resource scheduling is a crucial task in cloud computing, as the appropriate allocation of resources to cloud workloads depends on the Quality-of-Service (QoS) requirements of cloud applications. The heterogeneity, uncertainty, and dispersion of resources in the cloud pose challenges for resource allocation, which cannot be adequately addressed with existing policies. Authors in [3] provide a comprehensive and systematic literature review of resource scheduling in cloud computing. They also provide a detailed description of resource scheduling algorithms, management approaches, types, and benefits, along with tools, scheduling aspects, and resource distribution policies. Consequently, distributed storage is in high demand to manage resources efficiently, making replication technologies essential for ensuring fault tolerance and high availability of data in cloud storage.

Dependent data replication is a popular technique that ensures availability and performance by distributing multiple copies across different locations. Effective management of storage and replication increases the possibility that at least one copy will be available in case of failure. This approach addresses several objectives, including reducing storage costs, improving fault tolerance, and shortening access latency [4]. Therefore, replication is a critical component of cloud

computing systems, as many users access their data from multiple locations with slight variations in latency.

Several studies have described data replication strategies in cloud systems, such as the Cost-effective Dynamic Replication Management (CDRM) strategy and the Performance- and Profit-oriented Data Replication Strategy, version 2 (PEPRv2) [5, 6], designed to improve data availability. Authors in [7] discussed effective replica management for improving reliability and availability in an edge-cloud computing environment and proposed a replica placement strategy based on the fast non-dominated sorting Genetic Algorithm (GA).

Therefore, this study proposes a method to reduce time and resource consumption by estimating the likelihood of different outcomes using a random variables approach, input analysis, and test suite iteration. Furthermore, attaining optimal results is aided by calibrating the relevant parameters of the proposed Decision Tree Cloud Replication (DTCR) model.

II. LITERATURE REVIEW

Authors in [8] proposed a replication management strategy that fully considers data block temperature and access response time to dynamically adjust the number of data replicas, enhancing data service quality. In this strategy, data nodes are selected for newly added replicas to achieve load balancing while considering network distance, node load, and cluster storage utilization. The delay-adaptive replica synchronization approach is employed to reduce high data write latency and improve data availability. This method updates only a subset of replicas with solid consistency, ensuring that written data blocks remain accessible and further enhancing overall data availability.

Authors in [9] proposed a dynamic, cost-conscious, and optimized data replication strategy that defines a minimum number of replicated data copies to ensure availability. They recommended an optimized and efficient portfolio approach to reduce replication costs while introducing redundancy and controlling expenditure in the data center, thus minimizing costs without hampering data availability. To implement this, the authors introduced a new EIMORM algorithm, which applies the Knapsack algorithm to optimize replication cost. This algorithm effectively reduces file service time and access latency while improving availability and load balancing.

In [10], the authors developed a novel dynamic data replication strategy using an Intelligent Water Drop (IWD) algorithm to address challenges in replication and cloud storage management. The IWD algorithm, based on swarm intelligence optimization, was used to optimize the cloud storage replication and management process. The authors compared the D2R-IWD algorithm with popular optimization techniques, including Particle Swarm Optimization (PSO) and GA. Their results demonstrated that the D2R-IWD methodology improves access efficiency, thereby enhancing cloud performance. They also noticed a nearly 40% increase in free storage capacity.

In [11], the authors designed models to address energy consumption and bandwidth demands posed by database access in cloud computing data centers. Furthermore, the work presented an effective energy replication strategy based on these models, resulting in an enhanced Quality of Service (QoS) and reduced communication delays. The assessment demonstrated performance and energy efficiency trade-offs through comprehensive simulations, which subsequently guided the design of future data replication solutions.

Amazon Web Services (AWS) provides several cloud storage services supporting data replication, including Amazon S3, Amazon EBS, and Amazon EFS. These services allow data replication within and across regions to ensure availability and durability. AWS also provides Application Programming Interfaces (APIs) that allow developers to access and manage these services programmatically, including the ability to replicate data [12].

Microsoft Azure is a cloud computing platform that enables the creation, delivery, and management of applications and services through a worldwide network of data centers under Microsoft's management. Azure has experienced considerable growth and acceptance by companies and organizations worldwide during the last five years. In the public cloud infrastructure market, Azure's market share climbed from 10% in 2017 to 15% in 2021 [13]. Azure has expanded services to include the Internet of Things (IoT), analytics, Machine Learning (ML), and Artificial Intelligence (AI), making it a key competitor in the industry.

Data management and classification facilitate the selection of protection methods, application of policies, and efficient access to data, saving time and reducing cost. One standard ML algorithm that can be applied in replication strategies is the Decision Tree (DT), which is well-known for data classification. Numerous researchers have regarded the DT algorithm as one of the essential algorithms for classification in various fields [14, 15]. Figure 1 illustrates the structure of a DT.

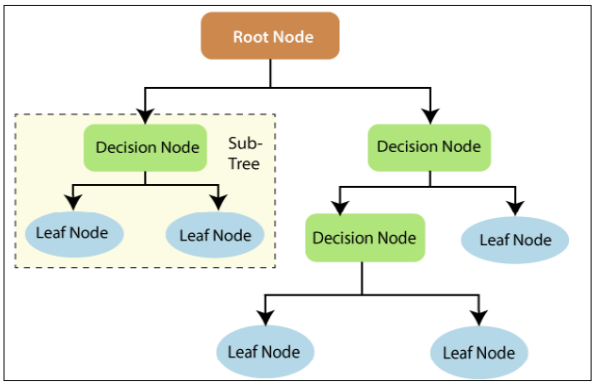

Fig. 1. Structure of a DT used for data classification.

There are several DT algorithms in the literature [6, 16-23], including Conditional Inference Trees (CTREE), Iterative Dichotomiser 3 (ID3), ID3 successor (C4.5), Chi-Squared Automatic Interaction Detection (CHAID), and Classification and Regression Trees (CART). Authors in [17] compared the various DT algorithms widely utilized in the field, and Table I presents the most popular DT algorithms.

TABLE I. COMPARISON OF POPULAR DT ALGORITHMS

Method	Algorithm		
	CART	**C4.5**	**CHAID**
Measurement employed to gather input variables	Gini index; towing criteria	Entropy information gain	Chi-square
Pruning	Pre-pruning with a single-pass algorithm	Pre-pruning with a single-pass algorithm	Chi-square test for independence during pre-pruning
Dependent variable	Categorical / continuous	Categorical / continuous	Categorical
Input variables	Categorical / continuous	Categorical / continuous	Categorical / continuous
Node split	Binary and linear combination	Multiple	Multiple

The authors in [24] proposed several ML methods for creating prediction models for data redundancy in the cloud. Among these, DTs and Neural Networks (NNs) are two of the most commonly used approaches. DTs are supervised learning algorithms that can be applied to classification or regression tasks. DTs are relatively easy to comprehend but are not always the most precise or effective approach for all tasks.

On the other hand, NNs are a ML approach inspired by the structure and operation of the human brain. They comprise several interconnected layers of "neurons" that can recognize patterns in data. Although NNs might be more complex and time-consuming to train than DTs, they are frequently more effective and accurate than DTs. In addition to DTs and NNs, other ML techniques, such as Random Forests (RFs), Support Vector Machines (SVMs), and k-Nearest Neighbors (k-NN), can be employed for classification, regression, clustering, or dimensionality reduction. The choice of the technique depends on the particular problem and dataset, as each has its own advantages and disadvantages.

Authors in [25] highlighted the importance of replication management in cloud computing and AI algorithms' role in managing replication strategies to increase efficiency and reliability while reducing resource consumption and cost. Using these algorithms to determine whether a particular query entails data redundancy is crucial, as it allows for decisions about replicas based on the current state of the data and resources.

The proposed model's strength lies in its ability to dynamically adjust replication strategies based on system conditions, response time targets, and tenant budget constraints, which significantly improves data availability and resource optimization. Furthermore, the DTCR model leverages supervised learning with DT algorithms and is trained on synthetic datasets. This model not only generalizes well to unseen data but also offers high availability with minimal overhead, making it a scalable and cost-efficient solution for cloud-based data systems.

III. THE PROPOSED MODEL

The proposed DTCR model comprises six development stages for managing replication strategies in the cloud. In the first stage, the process involves creating multiple random walks using the Markov Chain Monte Carlo (MCMC) algorithm (Algorithm 1). The mathematical concept of a random walk, a stochastic process, defines a path of successive random movements over specific mathematical spaces, such as the set of integers. The MCMC algorithm is implemented to generate multiple random walks (i.e., more iterations).

The MCMC algorithm is widely used in many fields, particularly for inference modeling. It is a powerful technique for exploring complex and high-dimensional spaces, making it well-suited for various applications, including the suggested model. One significant advantage of the MCMC algorithm is its ability to generate random samples from probability distributions that are difficult to sample directly. In the proposed model, the MCMC algorithm allows for the exploration of the parameter space to estimate the underlying data distribution accurately. By iteratively generating random walks, the MCMC algorithm converges to a stationary distribution, providing samples that approximate the desired distribution.

Furthermore, the MCMC method is effective for handling latent variables and complex dependencies by estimating posterior distributions through simulated random walks, enabling more accurate predictions. Another advantage of MCMC is its flexibility and adaptability to different modeling scenarios. It can handle models with nonlinear relationships, non-Gaussian distributions, and complex parameter spaces. This flexibility makes it well-suited for the proposed DTCR model, which likely involves intricate dependencies and nontrivial relationships between variables.

```
Algorithm 1: MCMC Algorithm
Input:  Number of iterations (N); Lower
Limit (L); Upper Limit (U) //Distribution
limits (L, U)
Output: Random walk path (series), Step S
//Time to cover the path
1: Initialize CurrentMin, CurrentMax, Step
   S, and CurrentPosition P
2: Randomly initialize Ri
3: While (iterations < N) do
4:   While (CurrentPosition P != L &&
     CurrentPosition P != U) do
5:     Generate Ri = Random.math()
6:     If (Ri < Mid(L, U)) then
         CurrentPosition P++
         If (P > CurrentMax) then
           CurrentMax = P
         End if
       Else
         CurrentPosition P--
         If (P < CurrentMin) then
           CurrentMin = P
         End if
       End if
7:     Step S++
     End while
   End while
```

The MCMC algorithm described in Algorithm 1 is crucial for creating random walks in the proposed model. The process

begins by defining variables such as N, L, and U, which specify the iteration count and boundary limits. Variables including CurrentMin, CurrentMax, Step S, and CurrentPosition P are initialized to appropriate values, and a random value R_i is also initialized. The algorithm then enters the main loop, which repeats until the number of iterations (N) is reached. Within each iteration, the algorithm enters an inner loop that continues until the CurrentPosition P is equal to either L or U, indicating the bounds of the distribution. A random value R_i is generated using the Random.math() function. If R_i is less than the midpoint between L and U (Mid(L, U)), CurrentPosition P is incremented by 1. If the updated CurrentPosition P exceeds CurrentMax, the CurrentMax value is updated accordingly. Conversely, if R_i is greater than or equal to Mid(L, U), CurrentPosition P is decremented by 1. If the updated CurrentPosition P is less than CurrentMin, the CurrentMin value is updated accordingly. The Step S counter is incremented after each iteration. The inner loop continues until CurrentPosition P reaches either L or U, and the outer loop continues until the number of iterations reaches N.

In this work, the scale used in the DTCR model is 0:100, meaning that one unit is equivalent to 100 km on the ground. The model operates on the first quadrant (x, y) of the Cartesian coordinate system (or Universal Transverse Mercator (UTM)) to avoid negative coordinates.

The following stages describe the DTCR model development:

1. Obtain and evaluate the results from all MCMC iterations to generate a dataset containing the following attributes: location, number of replica occurrences (from 9,000 iterations), total hits for each location, average replicas per occurrence, and total hits. The results can be stored using software tools such as Excel.
2. Classify the dataset based on ordered averages of replicas and total hits per site. A new feature, "target," is defined with the value "Yes" if the average replicas and total hits per site are ≥10,000, and "No" otherwise. The Weka tool, a Java-based collection of ML algorithms that runs on almost any platform, is used for classification.
3. Create the DTCR model in Weka to predict whether a location is selected (Yes) or ignored (No).
4. Training set: Approximately 60%–80% of the ordered dataset (out of 90,000 iterations) is used for training. This subset enables the DTCR model to learn and adapt its parameters through iterative optimization.
5. Validation set: 10%–20% of the data is used for hyperparameter tuning, model selection, and performance monitoring to evaluate the DT's predictive accuracy.
6. Results: The proposed model produces predictive outcomes using its DT-based framework, as detailed in the following sections.

Using the DTCR model as described, the system can make predictions or decisions based on new data to add or remove replicas, which can then be used by the scheduler policy to update replicas in the cloud environment.

IV. RESULTS

This study employed a random walk strategy to select replication locations and determine the number of hits at each location. The MCMC algorithm generated 100 locations, over 90,000 iterations. The study aimed to identify the optimal replication locations within the 0–100 range and then employ the DTCR algorithm to decide whether to add or remove replicas. This approach enables the model to be assessed for its ability to manage replication strategies in a cloud environment. The location and number of hits were arbitrarily established during the iteration, and the model recorded them as it traversed from the initial location to the domain boundary.

A. Data Generation and Evaluation Phase

The data generation and evaluation phase generates outputs for the 100 locations created by the random walk algorithm as a dataset. The outputs include: location ID (loc_id), number of repeated locations for each location over 90,000 iterations (no_rep_locations), total hits per location (T_hits), and average replicas per location (aveg_per_rep). Figures 2 and 3 illustrate the characteristics of the datasets distributed within a cloud computing environment. Figure 2 shows the replica locations and the number of appearances in 90,000 iterations, whereas Figure 3 presents the replica locations and the accumulated number of hits for the same locations.

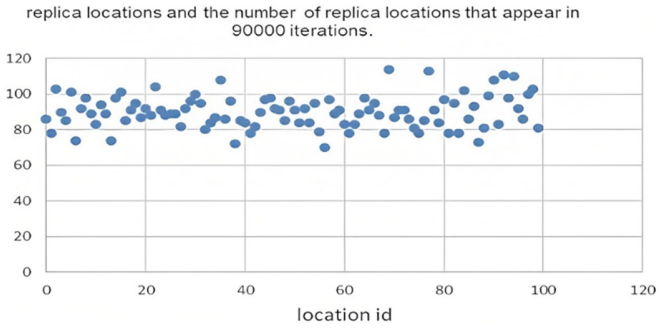

Fig. 2. Locations and frequency of appearance in the dataset.

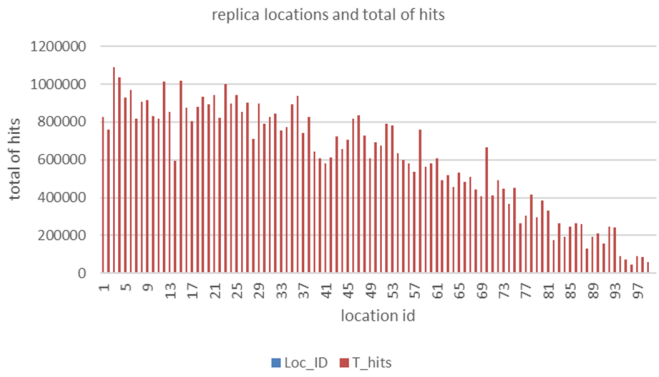

Fig. 3. Replica locations and their accumulated total hits.

B. Dataset Classification

The analysis phase uses Weka tools to classify the dataset and extract meaningful insights. The new feature, "target", was added to indicate whether the average replicas per location exceed 10,000. As shown in Figure 4, the dataset includes 89 locations with target "Yes" and 11 locations with target "No".

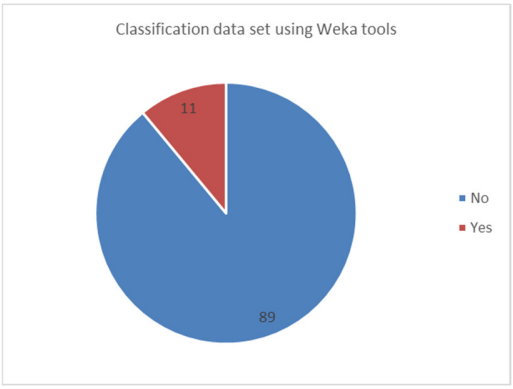

Fig. 4. Classified dataset using Weka tools.

The dataset statistics provide valuable insights into its characteristics. The sum of observations and the sum of squared deviations from the mean are essential measures of the central tendency and variability of the data, respectively. The mean of the observations serves as a representative value of the dataset, whereas the standard deviation measures the dispersion of the data. In this case, the small standard deviation of 9.1585 suggests that the observations are closely clustered around the mean value of 90. However, it is also essential to consider the skewness and kurtosis of the data, as these measures can reveal information about the shape of the distribution and the presence of outliers. These statistics can uncover data patterns and trends and guide decision-making processes.

C. Training the DTCR Model

Using the Weka tools in the training phase, the DTCR model was trained on 60% of the 90,000 iterations, and the remaining 40% was reserved for the testing phase. Figure 5 shows a snapshot of the training process, illustrating the five attributes used in the dataset (loc_id, no_rep_locations, T_hits, aveg_per_rep, and target).

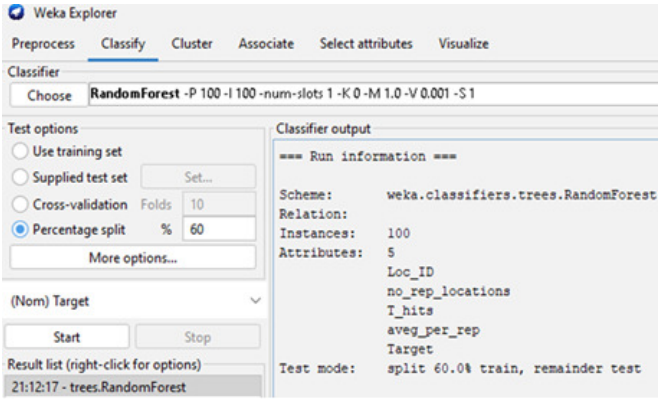

Fig. 5. Snapshot of the training process using Weka tools.

D. Testing the DTCR Model

In the test phase, using Weka tools, the DTCR model evaluated 40% of the 90,000 iterations to assess its performance. Table II compares the data split ratios used in this study with those adopted in similar works. The first row represents the current study, which used a 60%–40% training–testing data split to maximize the size of the training set while ensuring adequate testing data. The second row corresponds to the study in [26], which used a 70%–30% split to maximize the size of the training set. The third row corresponds to the study in [27], which also adopted the 70%–30% split to ensure equal representation of the classes in the training and testing sets. The fourth row corresponds to the study in [28], which used an 85%–15% split to maximize the size of the training set.

TABLE II. COMPARISON OF TRAINING AND TESTING DATA SPLITS IN DT ALGORITHMS

Study	Training data	Testing data	Reason for split selection
This study	60%	40%	To maximize the size of the training set while ensuring adequate testing data
[26]	70%	30%	To maximize the size of the training set
[27]	70%	30%	To ensure equal representation of the classes in both sets.
[28]	85%	15%	To maximize the size of the training set

E. Cross-Validation

Cross-validation is a widely used technique in ML to assess the performance and generalizability of predictive models. It provides a robust evaluation framework by partitioning the available data into multiple subsets, or folds, and iteratively trains and tests the model on different combinations of these folds. Cross-validation helps mitigate overfitting issues and provides a more reliable estimate of the model's performance on unseen data.

By applying cross-validation, the authors evaluated the DTCR model's ability to generalize and predict unseen instances while ensuring consistent and reliable results across different data subsets. The dataset was partitioned into multiple folds, each serving as both a training and testing set, allowing a comprehensive assessment of the model's performance.

The analysis revealed 99% correctly classified instances, indicating the model's ability to accurately predict the class labels for unseen data. Evaluation measures, such as accuracy, precision, recall, F-measure, and the Matthews Correlation Coefficient (MCC), further supported the robustness and accuracy of our model. The results showed high true positive rates, low false positive rates, and balanced precision and recall values for both classes. The weighted average accuracy reached 99%, confirming the algorithm's strong performance.

Further analysis revealed that the Kappa statistic was 0.9509, indicating a substantial agreement between the predicted and actual class labels. The Mean Absolute Error (MAE) was 0.01, indicating a slight average deviation between the predicted and actual values. Similarly, the Root Mean

Squared Error (RMSE) was 0.1, reflecting a low level of variability in the predicted values. The Relative Absolute Error (RAE) and Root Relative Squared Error (RRSE), which assess the average and relative deviations of the predictions, were 4.9356% and 31.9148%, respectively. These values suggest that the model's predictions were close to the true values, with moderate variability.

The precise accuracy by class provides a more granular view of the model's performance for each class. For the "No" class, the true positive rate, precision, recall, and F-measure were all above 0.989, indicating high performance in correctly classifying instances belonging to this class. Similarly, for the "Yes" class, all these measures were 1.000, indicating perfect performance in classifying instances of this class. The confusion matrix summarizes the classification results, showing the number of instances classified as "No" or "Yes" and comparing them with the actual class labels. Of the 99 instances labeled as "No," 88 were correctly classified, with only one misclassified. For the "Yes" class, all 11 instances were correctly classified.

Including these evaluation metrics and the confusion matrix in our analysis provides a comprehensive and reliable assessment of the DTCR model through cross-validation. Figure 6 demonstrates the model's evaluation results along with the corresponding confusion matrix, showing the number of instances correctly classified for each class.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

aveg_per_rep <= 9843: No (89.0)
aveg_per_rep > 9843: Yes (11.0)

Number of Leaves  :     2

Size of the tree :      3


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          99               99      %
Incorrectly Classified Instances         1                1      %
Kappa statistic                          0.9509
Mean absolute error                      0.01
Root mean squared error                  0.1
Relative absolute error                  4.9356 %
Root relative squared error             31.9148 %
Total Number of Instances              100

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
                 0.989    0.000    1.000      0.989    0.994      0.952    0.994     0.999     No
                 1.000    0.011    0.917      1.000    0.957      0.952    0.994     0.917     Yes
Weighted Avg.    0.990    0.001    0.991      0.990    0.990      0.952    0.994     0.990

=== Confusion Matrix ===

  a  b   <-- classified as
 88  1 |  a = No
  0 11 |  b = Yes
```

Fig. 6. Cross-validation results and confusion matrix using Weka tools.

F. Hyperparameter Tuning

This study employed the DecisionTreeClassifier from Weka to construct the DT model. To address the limitations associated with fixed data splitting, cross-validation was adopted to partition the dataset into multiple subsets, allowing for a comprehensive assessment of the model's performance across different splits. Furthermore, a grid search was conducted using Weka tools to explore various combinations of hyperparameters and identify the optimal configuration. The results in Figure 7 illustrate the best hyperparameters discovered.

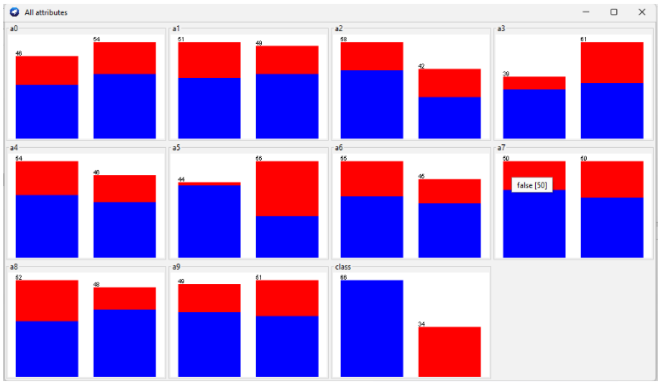

Fig. 7. Snapshot of the hyperparameter tuning process using Weka tools.

The optimal model incorporates settings such as the maximum tree depth (e.g., 85, 74, 90) and the minimum number of samples required for a split (e.g., 99, 3, 87) for the "no_rep_locations" configuration. These hyperparameters were applied to the DTRC model to evaluate its accuracy on the test set. In addition, setting the maximum depth plays a crucial role in controlling the tree's complexity and avoiding overfitting, thereby enhancing the model's generalizability.

G. DTCR Evaluation by Class

Analyzing a DT algorithm's performance by class shows how well the DTCR model performed for each class, pinpointing any class imbalances or classes for which the model has trouble making accurate predictions. Table III presents the detailed accuracy by class for the DTCR model, including TP rate, FP rate, Precision, Recall, and weighted averages:

- TP rate (sensitivity / recall): Measures the model's effectiveness in correctly identifying true positive instances. It is calculated as the ratio of true positives (TP) to the total number of true positives and false negatives (FN), as shown in (1) [29]:

$$\text{TPR} = \frac{\text{TP}}{\text{TP+FN}} \quad (1)$$

  High recall indicates that the model correctly identifies most positive samples, whereas low recall indicates the model is likely to miss some positive samples.

- FP rate: Measures how well a DT algorithm correctly identifies negative instances. It is calculated as the ratio of false positives (FP), i.e., incorrectly classified negatives, to the total number of actual negative instances, as shown in (2) [29]:

$$\text{FPR} = \frac{\text{FP}}{\text{FP+TN}} \quad (2)$$

- Precision: Measures a DT algorithm's ability to classify positive instances correctly. It is calculated as the number

of true positives divided by the total number of positive instances predicted by the DTCR model, as shown in (3) [29]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (3)$$

Precision indicates how many of all the positive predictions made by the DTCR model were correct. High precision means the model is less likely to predict a positive class when the sample is negative. In contrast, low precision means the model is more likely to predict a positive class when the sample is negative. It is important to note that precision should be combined with TPR (or recall) to evaluate the performance of a DTCR model. High precision and low recall or low precision and high recall indicate that the model is biased toward a particular classification error. The results demonstrate that the DTCR model is not biased toward any particular classification error.

- F-measure (F1-score): Measures a DT algorithm's performance by combining precision and recall and it is calculated as their harmonic mean, as shown in (4) [29]:

$$\text{F} - \text{measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision+Recall}} \quad (4)$$

- MCC: Measures a DT algorithm's performance by combining TP, TN, FP, and FN into a single value ranging from -1 to 1, where 1 means a perfect prediction, zero means no better than a random prediction, and -1 means a completely incorrect prediction. It is calculated as in (5) [29]:

$$\text{MCC} = \frac{\text{TP} * \text{TN} - \text{FP} * \text{FN}}{\sqrt{(\text{TP+FP}) * (\text{TP+FN}) * (\text{TN+FP}) * (\text{TN+FN})}} \quad (5)$$

The results demonstrate that DTCR model achieves the optimal MCC value of 1.0, representing a perfect prediction.

- Weighted average: Computes average performance accounting for class imbalance and it is calculated as in (6) [29]:

$$\text{Weighted avg} = \frac{\sum(\text{Probability of each class} * \text{Accuracy of each class})}{\text{Total number of classes}} \quad (6)$$

TABLE III. DETAILED ACCURACY BY CLASS FOR THE DTCR MODEL

Class	TP rate	FP rate	Precision	Recall	MCC
No	1.000	0.000	1.000	1.000	1.000
Yes	1.000	0.000	1.000	1.000	1.000
Weighted avg.	1.000	0.000	1.000	1.000	1.000

This evaluation demonstrates that the DTCR model correctly classified all instances, with no misclassifications, achieving high robustness and reliability for both classes.

H. Confusion Matrix and Decision Tree Visualization

A confusion matrix evaluates the performance of a DT algorithm in a classification problem. It summarizes the number of TP, TN, FP, and FN predictions. In a DT algorithm, the confusion matrix depends heavily on the splitting criteria and the dataset. Table IV presents the detailed confusion matrix for the DTCR model, using the 40% of the dataset reserved for testing. The classification labels used are "Yes" and "No." The results were generated using the Weka software tools.

TABLE IV. DTCR CONFUSION MATRIX

Actual \ Predicted	Yes	No
Yes	33	0
No	0	7

To provide a visual representation of the model's performance, the confusion matrix was plotted using the Seaborn (SNS) library in Python. Figure 8 visually displays the confusion matrix, allowing for a quick assessment of the model's strengths and weaknesses. The figure highlights the correct predictions (TP, TN) and the misclassifications (FP, FN), enabling informed decisions to improve the model's overall performance.

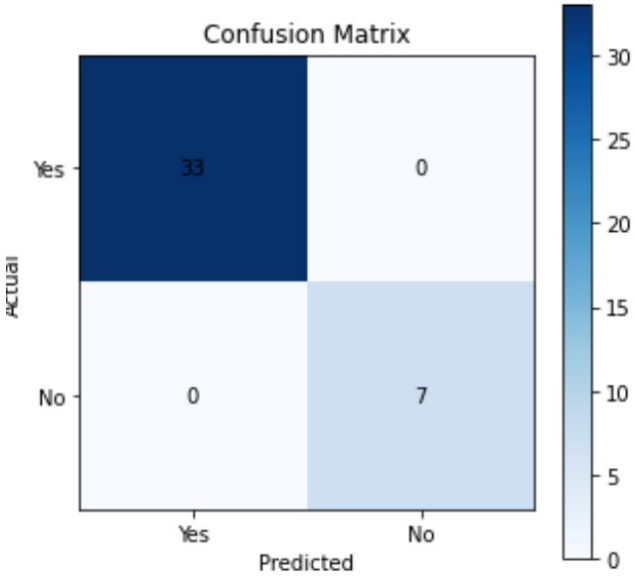

Fig. 8. Visual representation of the DTCR confusion matrix.

Visualizing the DT can help in understanding the decision-making process of the algorithm and can also help in determining any potential problems or biases in the tree structure. Figure 9 shows the DT visualization for the DTCR model using Weka tools.

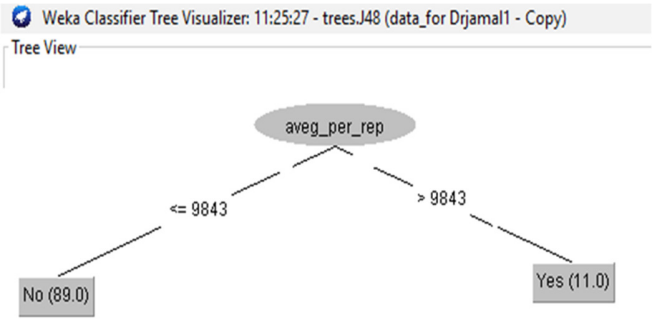

Fig. 9. Decision tree visualization of the DTCR model using Weka tools.

I. Comparative Analysis and Benchmarking

In this section, the proposed DTCR model is compared with three existing approaches to cloud replication management [30–32]. The comparison highlights key characteristics and methodological differences among these studies, as summarized in Table V.

TABLE V. COMPARATIVE ANALYSIS OF REPLICATION STRATEGIES

Approach / study	Description
Prefetching-aware Data Replication (PDR) / [30]	Determines data file correlation using file access history and prefetches popular files; includes a replica replacement strategy based on a fuzzy inference system.
GA-based approach / [31]	Employs a GA as the fundamental optimization method for the allocation of data replicas.
BAT algorithm-based approach / [32]	Uses a BAT algorithm for data replica optimization.
DTCR model (proposed)	Utilizes DTs for replication management, focusing on determining whether to add or remove replicas based on tenant budget while improving availability and performance.

Following the comparative analysis, several relevant studies were also reviewed:

- Authors in [33] highlighted the rise of credit card fraud due to increased electronic transactions and proposed a novel method using Generative Adversarial Networks (GANs) to tackle challenges associated with small and imbalanced datasets. Real-world data were used, and their model achieved high accuracy (above 85%) and precision (above 95%) in fraud detection. The proposed method reportedly outperformed previous studies.
- Authors in [34] compared traditional fault detection methods with ML algorithms such as k-NN, DT, and SVM, highlighting the superior accuracy of ML approaches. Using Wavelet Decomposition (WD) for feature extraction, they achieved high fault detection and classification accuracy.
- Authors in [35] emphasized the importance of classifying Myocardial Infarction (MI) complications to predict life-threatening outcomes. They compared Multilayer Perceptron (MLP), Naïve Bayes (NB), and DT using real-world hospital data. The results showed that MLP performs best with the full dataset, whereas DT performs better after data reduction.

V. DISCUSSION, LIMITATIONS, AND FUTURE WORK

Following the results, the DTCR model demonstrates excellent performance in replication management. Using the confusion matrix and metrics such as TP rate, FP rate, precision, recall, F-measure, and MCC (Table III), the model achieves perfect predictions with values of 1.0 for all metrics.

The DTCR model is explicitly designed for cloud replication management, with the primary goal of ensuring high data availability in a cloud environment. Using a DT, the DTCR model identifies the optimal replication strategy that minimizes replication costs while maintaining data availability. In addition, the DTCR effectively reduces processing time and resource consumption during the replication process. This capability is particularly beneficial for organizations relying on cloud computing, guaranteeing data availability regardless of technical failures or network outages. The DTCR model, as an AI-based solution, demonstrates adaptability and effective decision-making for replication management.

Despite the promising results, some limitations need to be addressed in future work. This study acknowledges these limitations and highlights them for further investigation and improvement:

- Extension to multi-cloud environments: Future work should extend the proposed model to consider multi-cloud environments enabling replication strategies that consider workload distribution, network latency, and server performance in multi-provider settings.
- Handling errors and failures: While the current study assumes error-free input data and replication without network or server failures, future work should explore more robust replication strategies for these scenarios. Developing mechanisms to handle errors and failures will enhance the reliability and resilience of the replication process.
- Prototype MCMC implementation: The DTCR model presented in this study is still in its prototype phase. Future work should concentrate on realizing automated test iteration strategies based on MCMC for cloud applications. Additionally, incorporating new features and datasets and refining the model's accuracy and performance will be crucial for future enhancements.
- Integration of DTs with Deep Learning (DL) models: Exploring the integration of DTs with DL models can leverage the strengths of both approaches. While DTs offer interpretability and explainability, DL models provide enhanced predictive capabilities for large-scale, high-dimensional datasets. Combining these two domains can lead to improved accuracy, performance, and the ability to capture complex patterns in data, thus enabling more informed replication management decisions.
- Exploration of alternative AI techniques: Future research should investigate the effectiveness of other AI techniques, such as NNs and GAs, for replication management in cloud environments. Additionally, blockchain technology could be investigated for secure and transparent replication strategies.

VI. CONCLUSION

This study proposed a replica management strategy to meet availability and performance requirements in cloud environments. A Decision Tree (DT) algorithm was implemented within a novel model called the Decision Tree Cloud Replication (DTCR) model. The DTCR model employs Artificial Intelligence (AI) techniques to manage replicas and determines whether replicas should be added, deleted, or merged.

The results demonstrate that the proposed DTCR-based replication strategy effectively meets availability and performance objectives. The DT algorithm provides a reliable and adaptive solution for replication management in cloud environments. Future work may involve refining the DTCR model and validating its performance in real-world scenarios. Overall, this study contributes significantly to cloud computing by providing an efficient approach for managing replicas in large-scale systems.

CONFLICTS OF INTEREST

The author declares no conflict of interest.

REFERENCES

[1] M. De Donno, K. Tange, and N. Dragoni, "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog," *IEEE Access*, vol. 7, pp. 150936–150948, 2019, https://doi.org/10.1109/ACCESS.2019.2947652.

[2] A. Sunyaev, "Cloud Computing," in *Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies*, Cham, Switzerland: Springer Nature, 2024, pp. 165–209, https://doi.org/10.1007/978-3-031-61014-1_6.

[3] S. Singh and I. Chana, "A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, Jun. 2016, https://doi.org/10.1007/s10723-015-9359-2.

[4] L. Chen, M. Qiu, J. Song, Z. Xiong, and H. Hassan, "E2FS: an elastic storage system for cloud computing," *The Journal of Supercomputing*, vol. 74, no. 3, pp. 1045–1060, Mar. 2018, https://doi.org/10.1007/s11227-016-1827-3.

[5] P. Kumari and P. Kaur, "A survey of fault tolerance in cloud computing," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 10, pp. 1159–1176, Dec. 2021, https://doi.org/10.1016/j.jksuci.2018.09.021.

[6] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, M. Masdari, and H. Shakarami, "Data replication schemes in cloud computing: a survey," *Cluster Computing*, vol. 24, no. 3, pp. 2545–2579, Sep. 2021, https://doi.org/10.1007/s10586-021-03283-7.

[7] G. K. F. Tso and K. K. W. Yau, "Predicting electricity energy consumption: A comparison of regression analysis, decision tree and neural networks," *Energy*, vol. 32, no. 9, pp. 1761–1768, Sep. 2007, https://doi.org/10.1016/j.energy.2006.11.010.

[8] C. Li, M. Song, M. Zhang, and Y. Luo, "Effective replica management for improving reliability and availability in edge-cloud computing environment," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 107–128, Sep. 2020, https://doi.org/10.1016/j.jpdc.2020.04.012.

[9] E. B. Edwin, P. Umamaheswari, and M. R. Thanka, "An efficient and improved multi-objective optimized replication management with dynamic and cost aware strategies in cloud computing data center," *Cluster Computing*, vol. 22, no. 5, pp. 11119–11128, Sep. 2019, https://doi.org/10.1007/s10586-017-1313-6.

[10] S. Nannai John and T. T. Mirnalinee, "A novel dynamic data replication strategy to improve access efficiency of cloud storage," *Information Systems and e-Business Management*, vol. 18, no. 3, pp. 405–426, Sep. 2020, https://doi.org/10.1007/s10257-019-00422-x.

[11] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, "Energy-efficient data replication in cloud computing datacenters," *Cluster Computing*, vol. 18, no. 1, pp. 385–402, Mar. 2015, https://doi.org/10.1007/s10586-014-0404-x.

[12] R. Bagai, "Comparative Analysis of AWS Model Deployment Services," *International Journal of Computer Trends and Technology*, vol. 72, no. 5, pp. 102–110, May 2024, https://doi.org/10.14445/22312803/IJCTT-V72I5P113.

[13] P. Dutta and P. Dutta, "Comparative Study of Cloud Services Offered by Amazon, Microsoft & Google," *International Journal of Trend in Scientific Research and Development*, vol. 3, no. 3, pp. 981–985, Apr. 2019, https://doi.org/10.31142/ijtsrd23170.

[14] N. Sharma and S. Sagar, "Diabetes Prediction Using Machine Learning Algorithms," in *2024 1st International Conference on Advances in Computing, Communication and Networking*, Greater Noida, India, 2024, pp. 449–457, https://doi.org/10.1109/ICAC2N63387.2024.10894772.

[15] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 1, pp. 20–28, Mar. 2021, https://doi.org/10.38094/jastt20165.

[16] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, "Models for efficient data replication in cloud computing datacenters," in *2015 IEEE International Conference on Communications*, London, UK, 2015, pp. 6056–6061, https://doi.org/10.1109/ICC.2015.7249287.

[17] Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry*, vol. 27, no. 2, pp. 130–135, Apr. 2015, https://doi.org/10.11919/j.issn.1002-0829.215044.

[18] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers - a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 4, pp. 476–487, Nov. 2005, https://doi.org/10.1109/TSMCC.2004.843247.

[19] W.-Y. Loh, "Fifty Years of Classification and Regression Trees," *International Statistical Review*, vol. 82, no. 3, pp. 329–348, Dec. 2014, https://doi.org/10.1111/insr.12016.

[20] C.-L. Lin and C.-L. Fan, "Evaluation of CART, CHAID, and QUEST algorithms: a case study of construction defects in Taiwan," *Journal of Asian Architecture and Building Engineering*, vol. 18, no. 6, pp. 539–553, Nov. 2019, https://doi.org/10.1080/13467581.2019.1696203.

[21] S. R. Jiao, J. Song, and B. Liu, "A Review of Decision Tree Classification Algorithms for Continuous Variables," *Journal of Physics: Conference Series*, vol. 1651, no. 1, Nov. 2020, Art. no. 012083, https://doi.org/10.1088/1742-6596/1651/1/012083.

[22] Priyanka and D. Kumar, "Decision tree classifier: a detailed survey," *International Journal of Information and Decision Sciences*, vol. 12, no. 3, pp. 246–269, Jan. 2020, https://doi.org/10.1504/IJIDS.2020.108141.

[23] S. Singh and M. Giri, "Comparative Study Id3, Cart And C4.5 Decision Tree Algorithm: A Survey," *International Journal of Advanced Information Science and Technology*, vol. 3, no. 7, pp. 47–52, Jul. 2014, https://doi.org/10.15693/ijaist/2014.v3i7.47-52.

[24] C. E. Brodley and P. E. Utgoff, "Multivariate decision trees," *Machine Learning*, vol. 19, no. 1, pp. 45–77, Apr. 1995, https://doi.org/10.1007/BF00994660.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, https://doi.org/10.1038/nature14539.

[26] A. Hannan and J. Anmala, "Classification and Prediction of Fecal Coliform in Stream Waters Using Decision Trees (DTs) for Upper Green River Watershed, Kentucky, USA," *Water*, vol. 13, no. 19, Oct. 2021, Art. no. 2790, https://doi.org/10.3390/w13192790.

[27] A. C. M. da Silveira, Á. Sobrinho, L. D. da Silva, E. de B. Costa, M. E. Pinheiro, and A. Perkusich, "Exploring Early Prediction of Chronic Kidney Disease Using Machine Learning Algorithms for Small and Imbalanced Datasets," *Applied Sciences*, vol. 12, no. 7, Apr. 2022, Art. no. 3673, https://doi.org/10.3390/app12073673.

[28] D. Phiri, M. Simwanda, V. Nyirenda, Y. Murayama, and M. Ranagalage, "Decision Tree Algorithms for Developing Rulesets for Object-Based Land Cover Classification," *ISPRS International Journal of Geo-Information*, vol. 9, no. 5, May 2020, Art. no. 329, https://doi.org/10.3390/ijgi9050329.

[29] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Tree-Based Methods," in *An Introduction to Statistical Learning: with Applications in R*, 2nd ed., New York, NY, USA: Springer US, 2021, pp. 327–365, https://doi.org/10.1007/978-1-0716-1418-1_8.

[30] N. Mansouri and M. M. Javidi, "A new Prefetching-aware Data Replication to decrease access latency in cloud environment," *Journal of Systems and Software*, vol. 144, pp. 197–215, Oct. 2018, https://doi.org/10.1016/j.jss.2018.05.027.

[31] L. Cui, J. Zhang, L. Yue, Y. Shi, H. Li, and D. Yuan, "A Genetic Algorithm Based Data Replica Placement Strategy for Scientific Applications in Clouds," *IEEE Transactions on Services Computing*, vol. 11, no. 4, pp. 727–739, Jul. 2018, https://doi.org/10.1109/TSC.2015.2481421.

[32] A. I. A. Eid, W. Awang, Mzarina, and A. Zakaria, "Replication Strategies based on Markov Chain Monte Carlo and Optimization on Cloud Applications," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 3, pp. 517–534, Feb. 2020.

[33] B. Alshawi, "Utilizing GANs for Credit Card Fraud Detection: A Comparison of Supervised Learning Algorithms," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12264–12270, Dec. 2023, https://doi.org/10.48084/etasr.6434.

[34] B. K. Ponukumati, P. Sinha, M. K. Maharana, A. V. P. Kumar, and A. Karthik, "An Intelligent Fault Detection and Classification Scheme for Distribution Lines Using Machine Learning," *Engineering, Technology & Applied Science Research*, vol. 12, no. 4, pp. 8972–8977, Aug. 2022, https://doi.org/10.48084/etasr.5107.

[35] A. Satty, M. M. Y. Salih, A. A. Hassaballa, E. A. E. Gumma, A. Abdallah, and G. S. M. Khamis, "Comparative Analysis of Machine Learning Algorithms for Investigating Myocardial Infarction Complications," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 12775–12779, Feb. 2024, https://doi.org/10.48084/etasr.6691.